MATH501 Modelling and Analytics for Data Science Coursework

10480219, 10593795, 10595599

May 16, 2018

UNIVERSITY OF PLYMOUTH

FACULTY OF SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING, ELECTRONICS and MATHEMATICS

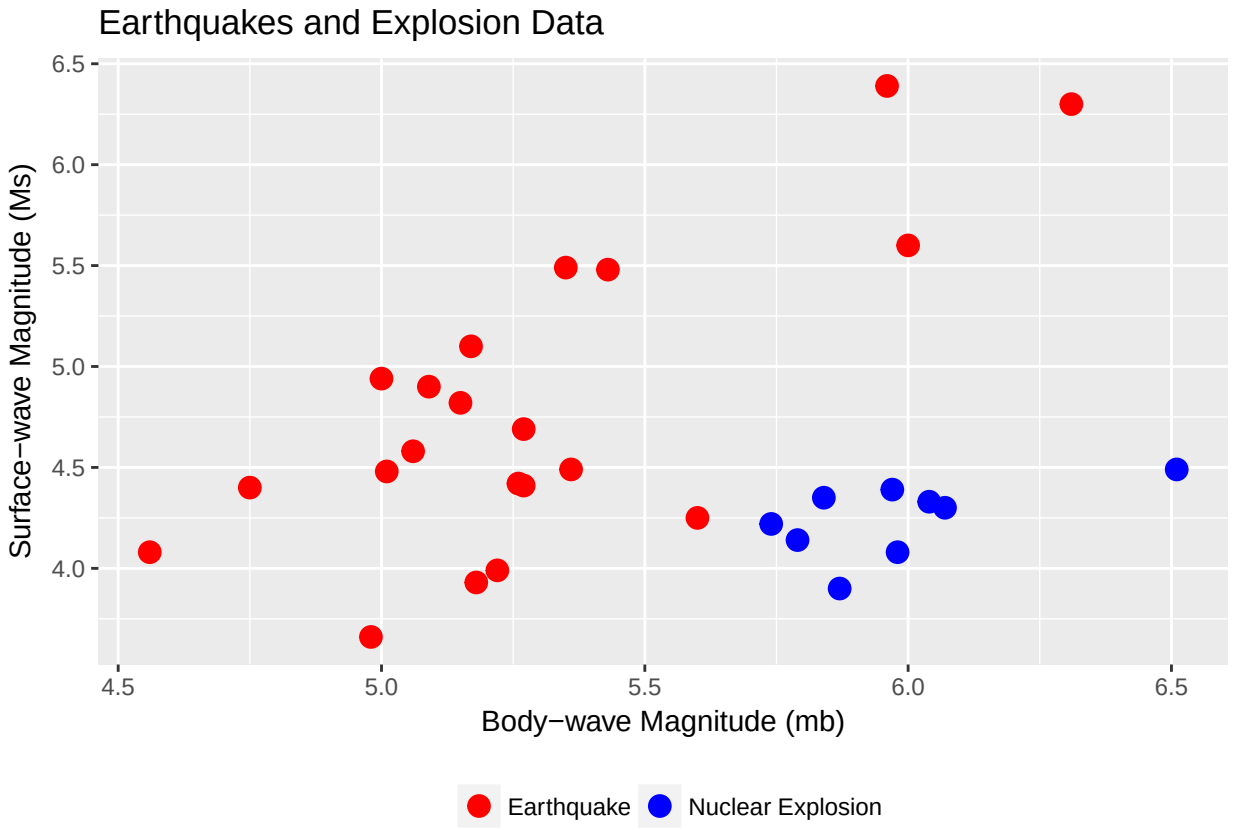
Contents

1 Machine Learning Task

1.1 General Information

The given dataset is named **earthquakes.txt**, which contains information about seismological events. There are three variables in this dataset:
– *popn* : which distinguishes between whether or not each event is an **earthquake** or a **nuclear explosion** . This will be the factor for the classification.
– *body* : which contains **body wave magnitude** (m_b). This is the magnitude of the wave that travels throught the interior of the earth.
– *surface* : which contains **surface wave magnitude** (Ms). This is the magnitude of the wave that travels along the Earth's surface.

The dataset contains 30 observations, 21 of which are confirmed as earthquakes and 9 are nuclear explosions. Graphed below is a visual representation of the data classified with different colours. Earthquakes are in red and nuclear explosions are in blue. Below is the scatter plot of the data.



- We build the colours and grid here to help us build the classification rules on the scatterplots for the KNN and QDA Analysis

```
# Here we produce a vector and give a default colour blue and
# then change the colour for the earthquake observation to red
def.col <- rep("blue",30) # vector of colours
def.col[popn == "equake"] <- "red" # red colour for the earthquake variable

# We define a grid of points that covers the entire range of the
# data clasifier on this grid in order to use it for QDA and KNN
len <- 40
xp <- seq(4,8,length = len)
yp <- seq(0,10,length = len)
xygrid <- expand.grid(body = xp , surface = yp)
```

1.2 QDA Discrimanant Analysis

The Quadratic Discriminant Analysis is a parametric method that assumes a quadratic rule can be used to classify the dataset. To calibrate this, a curve is drawn that separates the grid, which can be used to discriminate between different events.

- Firstly the QDA model fitted using the **MASS** library and the results are as follows:

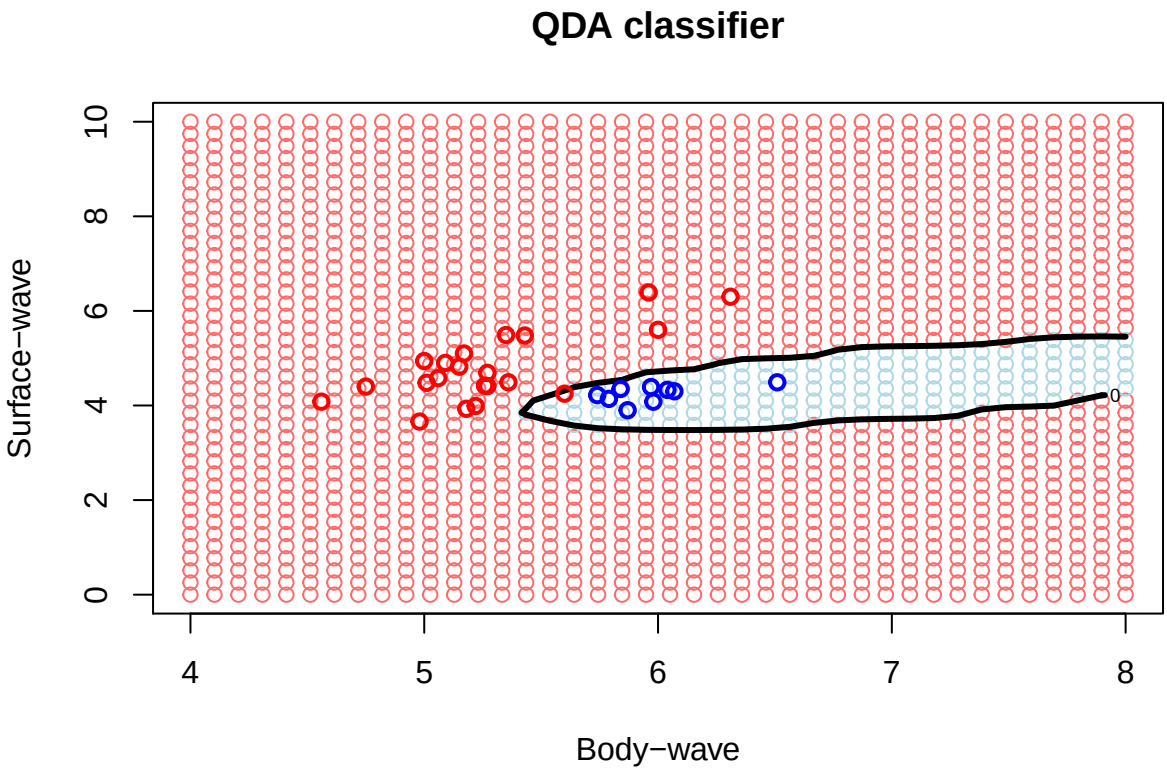
```
def.qda <- qda(popn ~ body + surface , data = earthquake)
def.qda
```

```
# Call:
# qda(popn ~ body + surface, data = earthquake)
#
# Prior probabilities of groups:
#  equake explosn
#     0.7     0.3
#
# Group means:
#             body  surface
# equake  5.284762 4.780952
# explosn 5.978889 4.244444
```

- The next step is to use the predict function in order to calculate the posterior and the class for the classification:

```
# Classify the point from the grid using the predict funcion
grid.qda <- predict(def.qda , xygrid)

# Preparing the colour of the grid to be plotted
col1 <- rep("lightblue",len*len)
for (i in 1:(len*len)) if(grid.qda$class[i] == 'equake') col1[i] <- "indianred1"

# Preparing the boundary point subracted from the earthquake from explosion posterior
zp <- grid.qda$post[ ,1] - grid.qda$post[ ,2]
```

- The classification rule can be visualised in the plot below:



- The table called below shows the incorrectly classified observations using the QDA classifier. As per the plot, there is one incorrectly classified earthquake observation

```
#           popn
# qda.class equake explosn
#   equake      20       0
#   explosn      1       9
```

```
# [1] 0.03333333
```

1.3 KNN Analysis (K-nearest neighbours)

The K-Nearest Neighbours is a non-parametric method that allows for a rule to be inducted out of the dataset based on a defined number of neighbours (k) for each point in the plane to have.

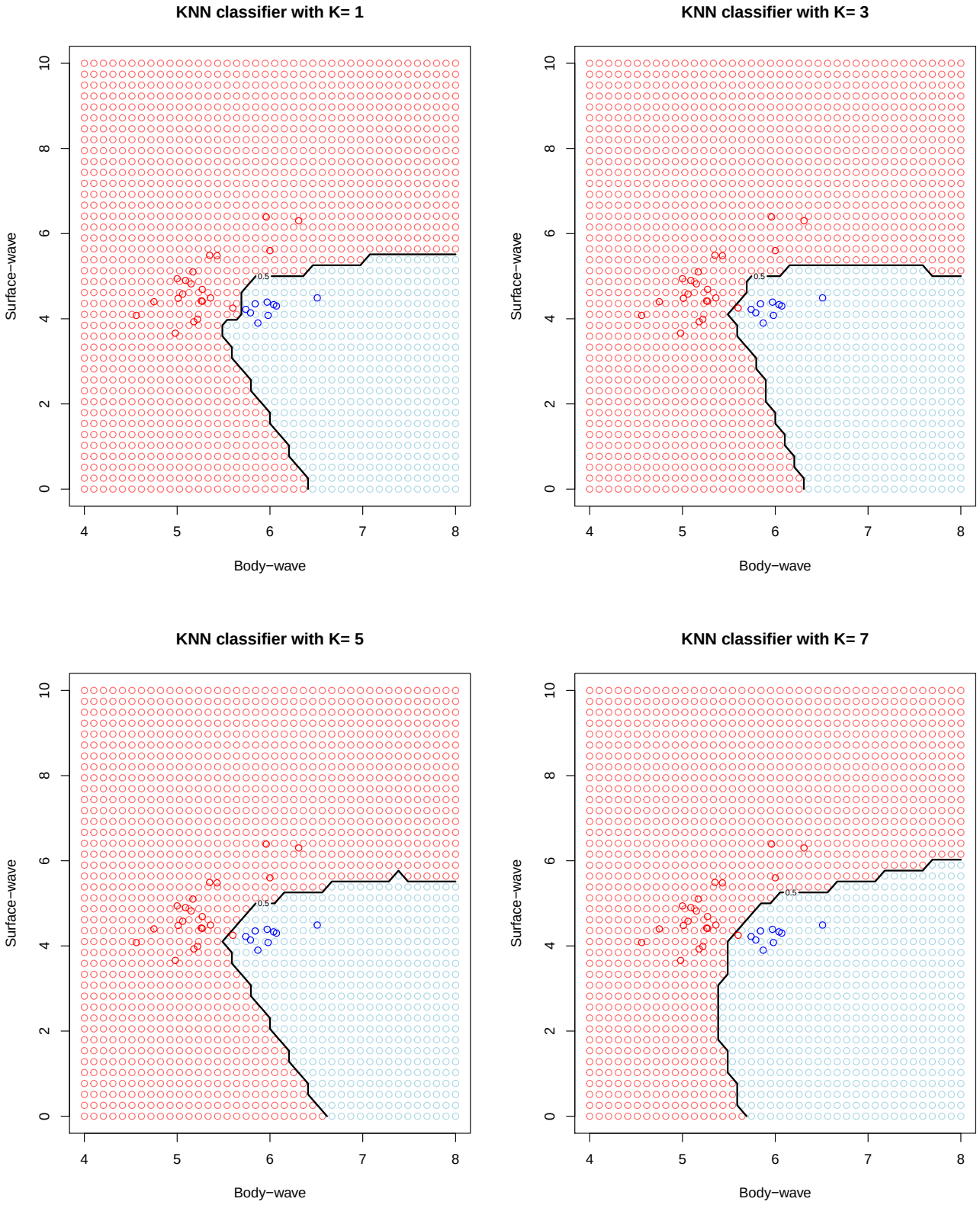
- Here we build the training matrix of the body and surface predictors in order to use it for the function below.

```
train.X <- cbind(body , surface)
train.X
```

- We create a function that automaticaly build the KNN plot based on the selected K, which is the only variable of the function. We use the **knn** function from the **MASS** library to do this as illustrated below:

```
knn.plot <- function(k){
# Create the function which uses only one variable K
#
# Fitting the KNN model using the knn function from the MASS
# library and k equal to k variable
def.knn <- knn( train = train.X, # Matrix containing predictors
                                 #for training data
                test = train.X,# Matrix containing the data which
                               # we wish to make predictions with
                cl = popn, # This vector contains the factor(labels)
                           # of the training observations
                k = k)
#The next step, is to build the grid colouring for the specific k from the
#function and plot the results:

# We represent the two classes as 1 (popn = earthquake) and 0
# (popn = explotion) for plotting the class boundary
cl <- rep(0, 30); cl[popn == 'equake'] = 1; cl <- as.factor(cl)
# We classify the points in the grid using the KNN function
grid.knn <- knn( train = train.X, # Matrix of containing predictors
                                  # for training data
                 test = xygrid, # Using the grid that we have
                                #produced above
                 cl = cl, # as the vector contains the factor we created above
                 k = k)
# We prepare the vector of colours that we use for the plot
col1 <- rep("lightblue", len*len)
for (i in 1:(len*len)) if(grid.knn[i] == "1") col1[i] <- "indianred1"
#
# Create the plot for each value of k
plot <- plot(xygrid, col = col1, main = paste("KNN classifier with K=",(k)),
             xlab = "Body-wave",ylab = "Surface-wave")
boundry<-contour(xp, yp, matrix(grid.knn, len), levels = 0.5,
                 add = TRUE, lwd = 2)
points <-points(body,surface, col = def.col)
#
# Return the plot and the def.col variable in order to build the
# table of the classifed observation
return(list(plot,boundry,points))
}
```

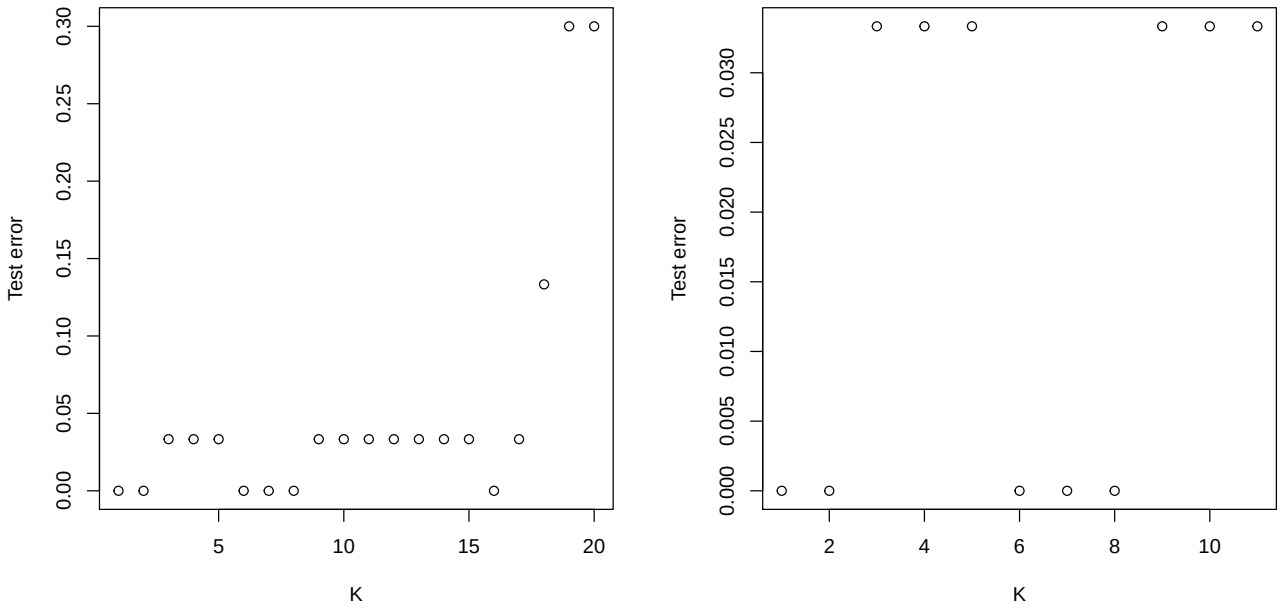
- We create plots for K = 1, 3, 5, 7 to predict **popn** the scatter plot **body** against **surface** using the **KNN_plot** function.



- We use the following function to calculate the errors based on the knn rules and then run a for loop to load all the possble errors

```
# Build a function that applies the knn rule and calculatea the
#training error from the table of classifiers
test.training.error <- function(k){
  def.knn <- knn( train = train.X, test = train.X, cl = popn, k = k)
  tab <- table(def.knn, popn)
  error <- (tab[1,2] + tab[2,1]) / 30
  return(error)
}
# Creating a variable to hold the errors and then using the
# function to calculate the errors
knn.training.errors <- rep(0,29)
for (i in 1:29) knn.training.errors[i] <- test.training.error(k = i)
# Build an empty error vector
knn.training.errors <- rep(0,30)
# Run the function in a for loop to load all the possible errors
for (i in 1:length(knn.training.errors))
  knn.training.errors[i] <- test.training.error(k = i)
```

- We plot the errors using in order to see visualize the training errors:



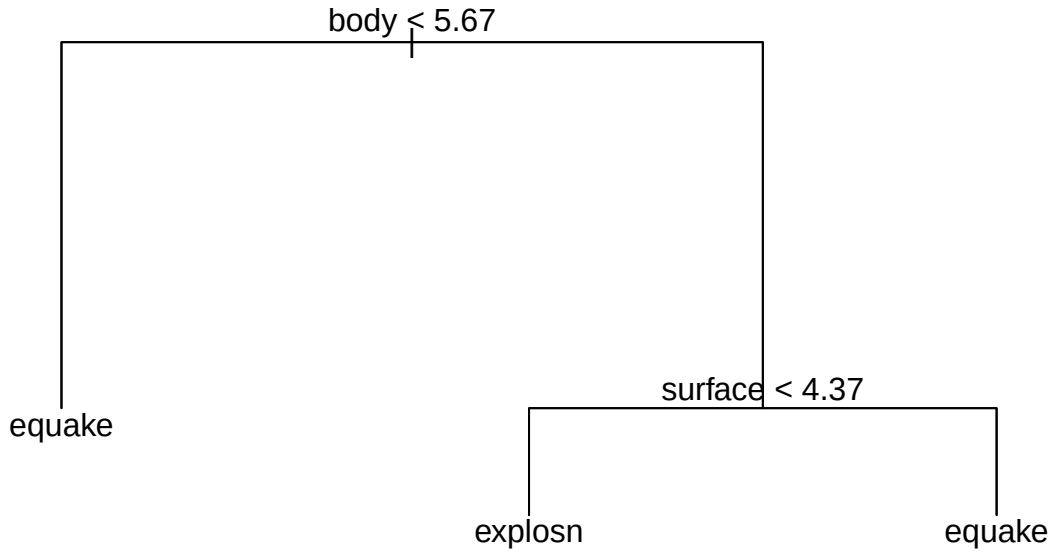
The graph shows us a visualisation of the overall training errors for classification rule value k from 1-20 on the left plot and on right plot 1 to 11.

1.4 Decision Trees

- Firstly, using the **tree** library the model is fitted for out dataset

```
tree.fit <- tree(popn ~ body + surface , data = earthquake)
```

- Then, we visualise the classification with the fitted classification tree illustrated below:



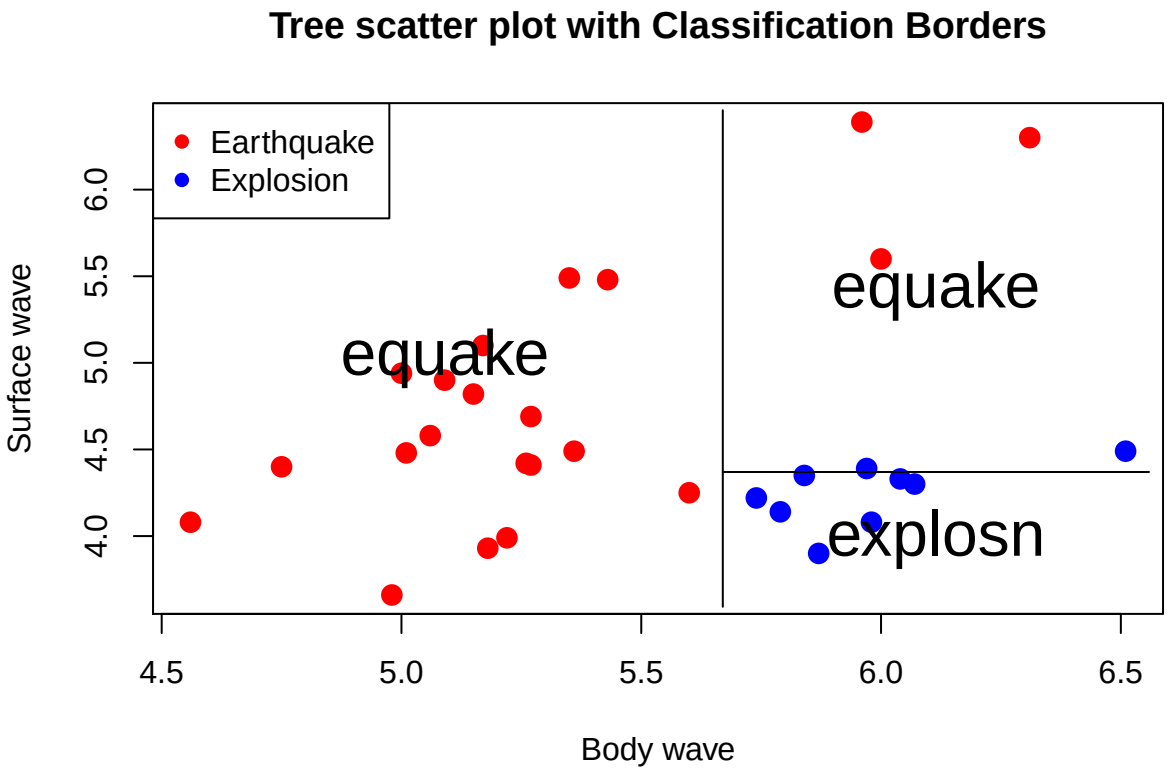
- Next we use the summary function to show the training error for the classification

```
#
# Classification tree:
# tree(formula = popn ~ body + surface, data = earthquake)
# Number of terminal nodes:  3
# Residual mean deviance:  0.2493 = 6.73 / 27
# Misclassification error rate: 0.06667 = 2 / 30
```

```
#          popn
# tree.pred equake explosn
#   equake      21       2
#   explosn      0       7
```

```
# [1] 0.06666667
```

- Finally, we produce the result of the classification tree on a scatter plot



Confusion matrix reveals that two explosions are classified incorrectly by the classification tree. and the training error was 0.0666667 which is not extremely high. From the scatter plot we see a clear classification rule and we may be able to modify the explosions border in order to have better results. For now we can say that the model fits well

1.5 Leave-one out Cross Validation (LOOCV)

Cross validation is a technique used in order to verify and test the errors by systematically testing the dataset, leaving one observation out each time.

1.5.1 Cross Validation for QDA

```
n <- nrow(earthquake)
# Here we build predictor array with 30 values where all the values are equake
cv.predictions.qda <- rep("equake", n)
# We use a for loop to just remove one row from the data frame at a time and
# We hold the data for the class in the cv.predictions.qda
for(i in 1:n){
  fit.qda <- qda(popn ~ . , data = earthquake[-i, ])
  hold <- predict(fit.qda ,newdata = earthquake[i,] , type = "class")
  cv.predictions.qda[i] <- hold$class
```

```
}
# We build a confusion table in order to check the classification rule
tab.cv.qda <- table(cv.predictions.qda, popn)
tab.cv.qda
```

```
#                    popn
# cv.predictions.qda equake explosn
#                  1     20       0
#                  2      1       9
```

```
# We calculate and display the cross validation error
cv.qda.error = (tab.cv.qda[1,2] + tab.cv.qda[2,1]) / sum(tab.cv.qda)
cv.qda.error
```

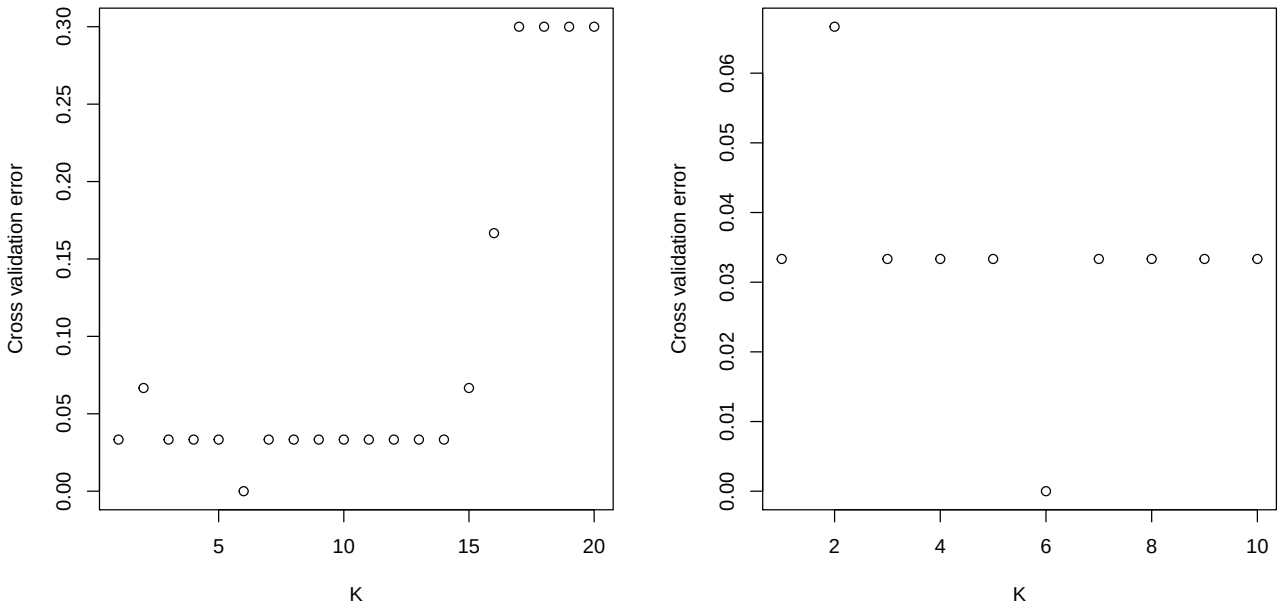
```
# [1] 0.03333333
```

1.5.2 Cross Validation for KNN

Use **knn.cv** function from **class** library which automatically crossvalidates for each selected k. For that reason we build the **cv.training.error** function to calculate and store the cross validation error.

```
cv.training.error <- function(k){
  # This function return the cross validation error for each selected k

  # Use knn.cv function from class library which does this automatically
  # cross validation for each selected k
  knn.cv.fit <- knn.cv(train.X , popn , k = i, l = 0, prob = TRUE)
  tab.knn <- table(knn.cv.fit,popn)
  cv.knn.error = (tab.knn[1,2] + tab.knn[2,1]) / sum(tab.knn)
  return(cv.knn.error)
}
# We produce an array which hold for each k the cross validation error
cv.knn.store <- rep(0,29)
set.seed(800)
for(i in 1:29) cv.knn.store[i] <- cv.training.error(i)
```

- Visualising the results of the cross validation errors:



1.5.3 Cross Validation for Classification Trees

```
# We buld the cv.prediction.tree to save the values
cv.predictions.tree <- rep('equake', nrow(earthquake))
# We build the for loop to calculate the classification using tree function
for(i in 1:n) {
  tree.fit <- tree(popn ~ ., data = earthquake[-i, ])
  cv.predictions.tree[i] <- predict(tree.fit, newdata = earthquake[i,], type = "class")
}
# We build a confusion table in order to check the classification rule
tab.cv.tree <- table(cv.predictions.tree, popn)
tab.cv.tree
```

```
#                    popn
# cv.predictions.tree equake explosn
#                   1     17       2
#                   2      4       7
```

```
# Represent cross validation error
cv.tree.error = (tab.cv.tree[1,2] + tab.cv.tree[2,1]) / sum(tab.cv.tree)
cv.tree.error
```

```
# [1] 0.2
```

1.6 Conclusions

Method Type	Test Error	CV Error
QDA	0.0333333	0.0333333
Tree	0.0666667	0.2
KNN(k=1)	0	0.0333333
KNN(k=3)	0.0333333	0.0333333
KNN(k=5)	0.0333333	0.0333333
KNN(k=6)	0	0
KNN(k=7)	0	0.0333333

We can see that the difference between the test errors and the cross validation errors. Because the test error is an unbiased indicator of the quality of the classifier and the test set is too small we can not say that is a very good measurment of the error. For that reason we use LOOCV error which is more accurate than the test error because it is based on n-1 data points. Based on these two errors we can see that the best fitted models are QDA and KNN which gives the same CV error while the tree give a larger error. Based on these two types of errors we conclude that the KNN analysis it was going to be more accurate ,if we use k = 1, 7, for this data set. On the other hand, we have two different types of classification - parametric and non-parametric. Parametric classification is usually less complex and more efficient, requiring less training data and can even work well if the fit to the data is not perfect. Because of this, it is more suitable for small, simple datasets such as this one. However, the limited complexity means that the method is more suited to simpler problems such as this one.

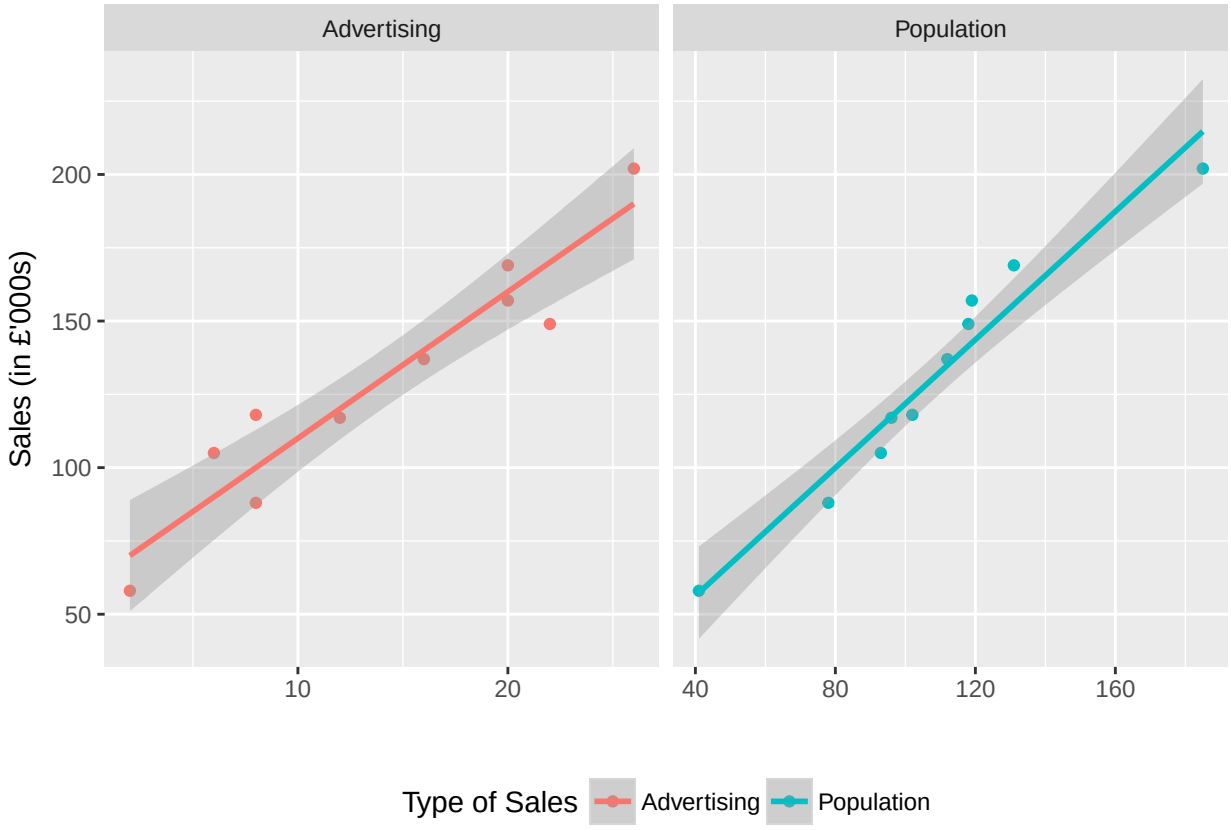
By contrast, non-parametric methods are far more flexible and offer better power and performance, which produced the lowest test errors in the KNN analysis performed. However, non-parametric methods require more training data to have safe assumptions and there is a risk of overfitting and can be slower as there are more training parameters. For all these reasons and by comparing the test and training erros, we believe that the best fitted classification model is the parametric Quadratic Discriminant Analysis for this dataset.

2 Bayesian Statistics Task

2.1 First Sub-Task

The given data named **Shops_Data.csv**, which contains information adout a retail chain which has shops in ten cities. In this data are four different columns:
– *City* : which contains diferent **Number** of the city as a label.
– *Sales* : which contains data on the **Annual Sales (units £'000s)**.
– *Advertising* : which contains **advertising expenditure (units £'000s)**.
– *Population* : which contains **City Population (units '000s)**.

2.1.1 To begin with, we build a scatter plot containing the sales as the dependent variable against advertising and population as our independent variables



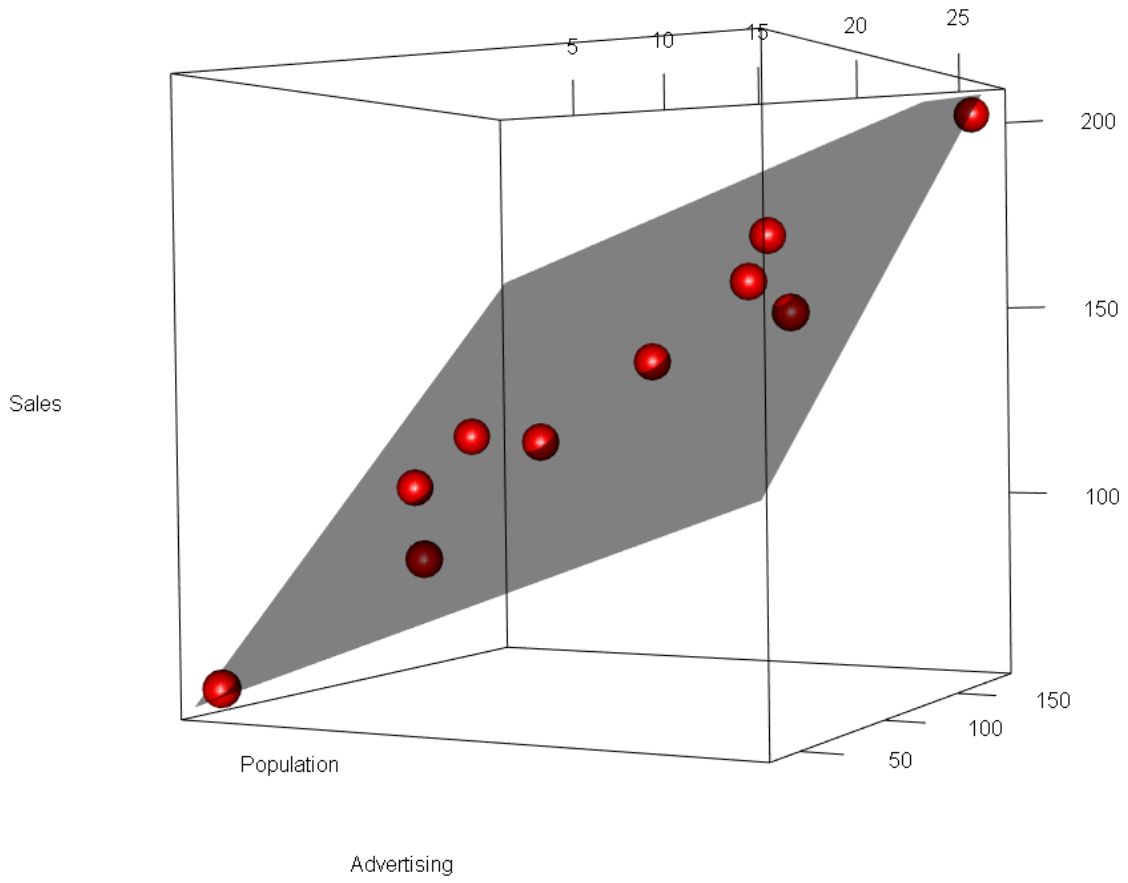


Figure 1: 3D model wirth the fitted model

2.1.2 To have a better visualisation of the model we fitting this in a 3D with the coresponding plane which can explain a visual interpretation of the parameter β_1

The parameter β_1 is the regression slope for advertising. What it tells us is by what magnitude our dependent variable, sales, should increase by if we were to increase advertising by one unit. From the fitted model below, we know this value of β_1 to be 2.0949. So if the firm were to increase its advertising spending by £1,000 then they would expect to see an increase of sales of £2,094.90

2.1.3 We fit this model in the frequentist framework and report β_0, β_1, and β_2 and perform a hypothesis test on whether advertising has an effect on sales

- Table of the coefficients and the R^2 value.

```
sum_represent$coefficients
```

```
#               Estimate Std. Error  t value    Pr(>|t|)
# (Intercept) 26.1392699  8.0014276 3.266826 0.013734355
# Advertising  2.0948617  0.6337659 3.305419 0.013023308
# Population   0.6933271  0.1351707 5.129269 0.001354747
```

```
sum_represent$r.squared
```

```
# [1] 0.9795594
```

- Table of the 95% Confidence intervals for β_0 and β_1.

```
confint(m)
```

```
#                 2.5 %    97.5 %
# (Intercept) 7.2189003 45.059640
# Advertising 0.5962434  3.593480
# Population  0.3736991  1.012955
```

The frequentist model run from the code above gives us the values of the coefficients. For β_0, the intercept, we have a value of 26.1392699. β_1 has the value of 2.0948617, so as above, for every increase in the level of advertising by £1,000, sales increase by £2,094.90. β_2 has the value of 0.6933271, so for every increase in population of 1,000, sales increase by £693.30. The R^2 coefficient of determination is 0.9795594, which means that 97.96% of the variability in sales is explained the variables in the model. Because this is close to 1, we can say that is a well fitted model.

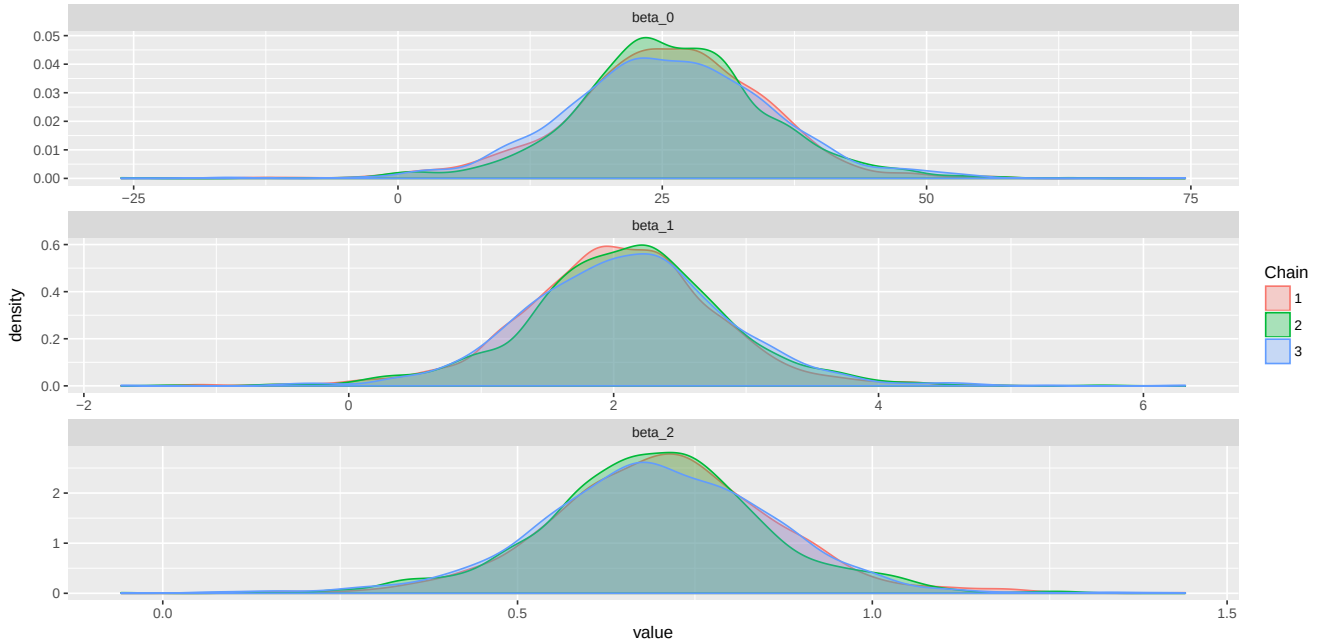
The significance level given is 95% which corresponds to our willingness to make a type I error, which is our alpha of 0.05. We fix the type 1 error probability here, and the test fixes the type 2 error which is the power of the test. For β_1, the p-value is 0.0130233 which is less than our value of alpha as 0.05. As this value is less than our alpha, we can conclude that the slope is signficant. The 95% confidence interval for β_1 is the range of values for which there is a 95% probability the true value of the slope will lie within, which for advertising is 0.5962434 to 3.5934799.

2.1.4 We write jags/BUGS code to perform the above model using the Bayesian framework

$$Sales_i = \beta_0 + \beta_1 Advertising_i + \beta_2 Population_i + \epsilon_i,\ i = 1, \ldots, n \text{ where n} = 10$$
$$\epsilon_i \sim N(0, precision \quad \tau) \text{ independently}$$
$$\beta_0 \sim N(0, precision \quad 0.0001)$$
$$\beta_1 \sim N(0, precision \quad 0.0001)$$
$$\beta_2 \sim N(0, precision \quad 0.0001)$$
$$\tau \sim Gamma(shape = 0.001, rate = 0.001)$$
$$\text{standard deviation} \quad \sigma = \frac{1}{\tau}$$

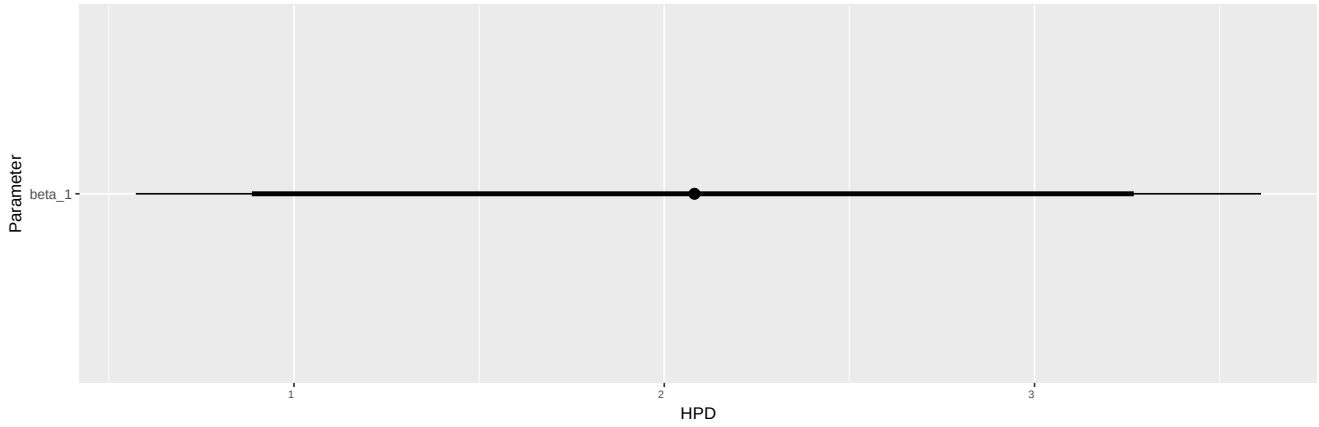
- We create a function named **Bayesian regression model** to run the simulation using the **R2jags** library. After that we use **ggmcmc** to convert the the fitted model and prepare a **ggs density** plot for β family.

```
Bayesian_regression_model <- function(){
  #
  # Data model or likelihood part
  #
  for(i in 1:n){ # n is the sample size (number of data points)
    #
    y[i] ~ dnorm(mu[i], tau) # Parametrised by the precision tau = 1 / sigma^2
    mu[i] = beta_0 + beta_1 * x1[i] + beta_2 * x2[i]
    #
  }
  #
  # Priors
  beta_0 ~ dnorm(0.0, 1.0E-4) # Prior on beta_0 is normal with low precision
  beta_1 ~ dnorm(0.0, 1.0E-4) # Prior on beta_1 is normal with low precision
  beta_2 ~ dnorm(0.0, 1.0E-4) # Prior on beta_2 is normal with low precision
  tau ~ dgamma(1.0E-3, 1.0E-3) # Prior on tau is gamma with small shape and rate paramete
  #
  # Definition of sigma: it's completely determined by tau
  #
  sigma <- 1.0 / sqrt(tau)
}
```



The posterior probability density functions illustrated are similar in each chain for each parameter. We do not see significant changes in the mean in each chain for each beta. When we run our priors, we take a very low precision in order to simulate our model with great variance. This allows the tails to be amplified lengthways in order to better understand the model. This way, predictions can be the most accurate.

2.1.5 We now calculate the 95% credible interval for β_1 and illustrate the interval numercal and with caterpillar plot:



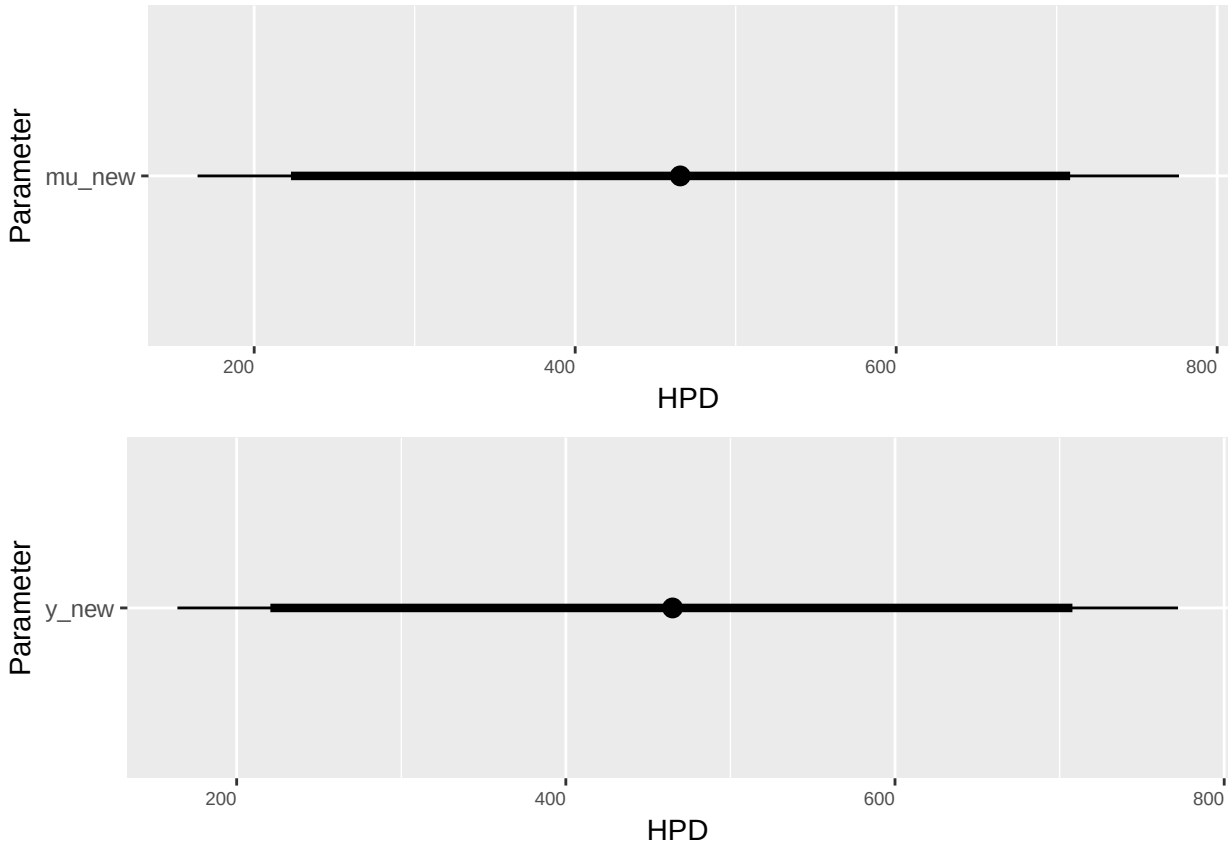

```
#       mean         sd       2.5%      97.5%       Rhat
# 2.0787843 0.7483015 0.5728565 3.6118876 1.0021611
```

The credible interval gives a 'natural' probability of 95% that the value for β_1 lies within the given interval. By contrast, the confidence interval states that the value of β_1 lies within the interval for 95% of trials. Recall the confidence interval calculated above is 0.5962434 to 3.5934799. The credible interval for β_1 shown above is 0.572856542 to 3.61188762, which is somewhat wider than for the frequentist analysis. We know that the Bayesian framework tends to give us a credible interval smaller than the frequentist confidence intervals. However, this is

not the case in this model, which could be because there may be other variables that the model has not captured that could give us more accurate results under the Bayesian framework.

2.1.6 Using the Bayesian model, we can make a predicted credible interval for a city of 200,000 people with a £30,000 advertising budget in a similar city. We include all of the above code with two extra lines at the end to make our predictions and we simulate the model again with the jags function.

```
Bayesian_regression_model_with_prediction <- function(){
  #
  # Data model or likelihood part
  #
  for(i in 1:n){ # n is the sample size (number of data points)
    #
    y[i] ~ dnorm(mu[i], tau) # Parametrized by the precision tau = 1 / sigma^2
    mu[i] = beta_0 + beta_1 * x1[i] + beta_2 * x2[i]
    #
  }
  #
  # Priors
  #
  beta_0 ~ dnorm(0.0, 1.0E-4) # Prior on beta_0 is normal with low precision
  beta_1 ~ dnorm(0.0, 1.0E-4) # Prior on beta_1 is normal with low precision
  beta_2 ~ dnorm(0.0, 1.0E-4) # Prior on beta_2 is normal with low precision
  tau ~ dgamma(1.0E-3, 1.0E-3) # Prior on tau is gamma with small shape and rate paramete
  #
  # Definition of sigma: it's completely determined by tau
  #
  sigma <- 1.0 / sqrt(tau)
  #
  # We add x new to give us the power to make the prediction
  #
  mu_new <- beta_0 + beta_1 * x1_new + beta_2 * x2_new # Value of mu at x_new
  #
  y_new ~ dnorm(mu_new, tau) # New value of y at x_new
  #
}
```



Numeric representation of the β_1, β_2, y_{new}, mu_{new} credible intervals

```
#                mean           sd        2.5%      97.5%
# beta_1    2.0920981    0.7465706   0.5946030   3.630134
# beta_2    0.6938528    0.1584543   0.3656824   1.011303
# y_new   465.3707337 150.0512457 164.0099583 771.957498
# mu_new  465.2959684 149.9489415 164.7239786 776.201649
```

These intervals, applicable assuming ceteris paribus, or all other factors remaining constant between the two cities, predict that the sales volume in this new city will be approximately 465.3707337. We are 95% confident that the sales will be between [lower bound] and [upper bound]. We can conclude that advertising and population have positive effect on sales revenue. Using this model, with the city size and advertising budget, there is a good prospect for return on investment. Sales are more sensistive to population than advertising as seen by the coefficients, and for that reason there is a margin to spend less on advertising with no significant difference in results. We would also recommend being aware of the scope of the recommendatons of this model, as there may be other variables that have not been measured or catpured in the model that may affect sales revenue. We suggest before taking this decision that the manager should check the composition/demographics/preferences of the local population to take into account factors that are not captured in the model.

2.2 Second Subtask

Twenty people were interviewed in Plymouth and asked whether they thought the UK economy would be stronger at the end of the year. The responses, either 'Yes' or 'No' and the respondent's age are given in the datset to be analysed.

Response	Age (years)
Yes	38
Yes	42
No	56
No	68
No	40
Yes	41
No	74
Yes	37
Yes	50
Yes	40
Yes	33
Yes	42
No	54
No	48
No	60
Yes	35
Yes	37
Yes	43
Yes	44
No	65

- We are loading the above data from the table in **R**.

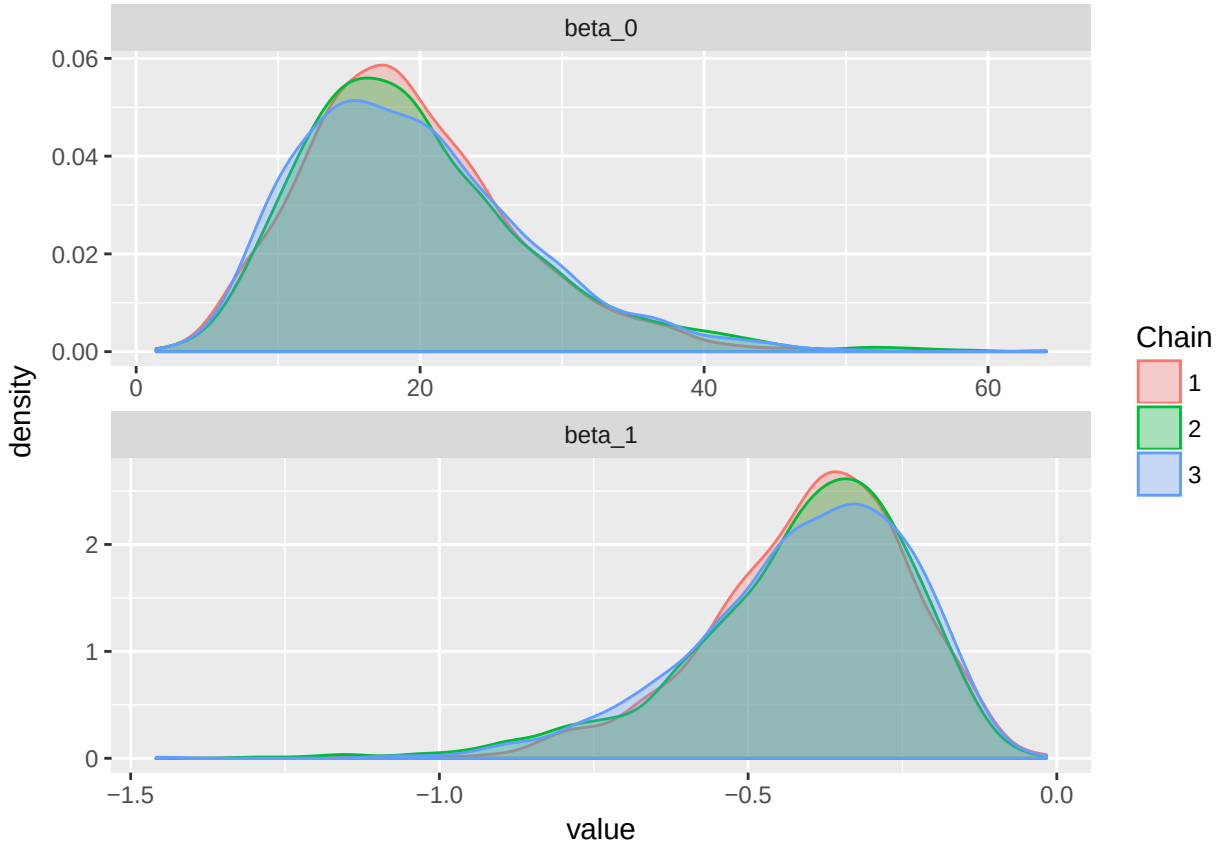
```
# Insert the data
responce <- c("Yes","Yes","No","No","No","Yes","No",
              "Yes","Yes","Yes","Yes","Yes","No","No",
              "No","Yes","Yes","Yes","Yes","No")
age <- c(38,42,56,68,40,41,74,37,50,40,33,42,54,
         48,60,35,37,43,44,65)


# Create a numeric vector containing the 0 if responce is NO and
# 1 if response is Yes
responce_prob <- rep(1,20)
for(i in 1:20) if(responce[i] == "No")responce_prob[i]<-0

# Create a data frame for the data
survey <- data.frame(responce,age,responce_prob)
```

2.2.1 We are write BUGS code for the logistic regression shown below:

$$\log \frac{p_i}{1-p_i} = \eta_i$$
$$\eta_i = \beta_0 + \beta_1 \ Age_i$$
$$\beta_0 \sim N(0, precision \quad 0.0001$$
$$\beta_1 \sim N(0, precision \quad 0.0001)$$
$$\text{where} \quad Response_i \sim dbin(p_i, 1)$$

```
Bayesian_binary_logistic_model <- function(){
  #
  # Define the likelihood part of the model
  #
  for(i in 1:n_obs){
    #
    y[i] ~ dbin(p[i],1)
    #
    # Link function
    #
    # logit(p) in BUGS give log(p / (1 - p))
    #
    # Linear predictor
    #
    logit(p[i]) <- eta[i]
    #
    eta[i] <- beta_0 + beta_1 * x[i]
  }
  #
  # Specify the prior distributions on the unknown parameters
  #
  beta_0 ~ dnorm(0.0, 1.0E-4) # Prior on beta_0 is normal with low precision
  beta_1 ~ dnorm(0.0, 1.0E-4) # Prior on beta_1 is normal with low precision
}
```

- We create density plots **ggs_density** using library **ggmcmc** for β_0 and β_1.

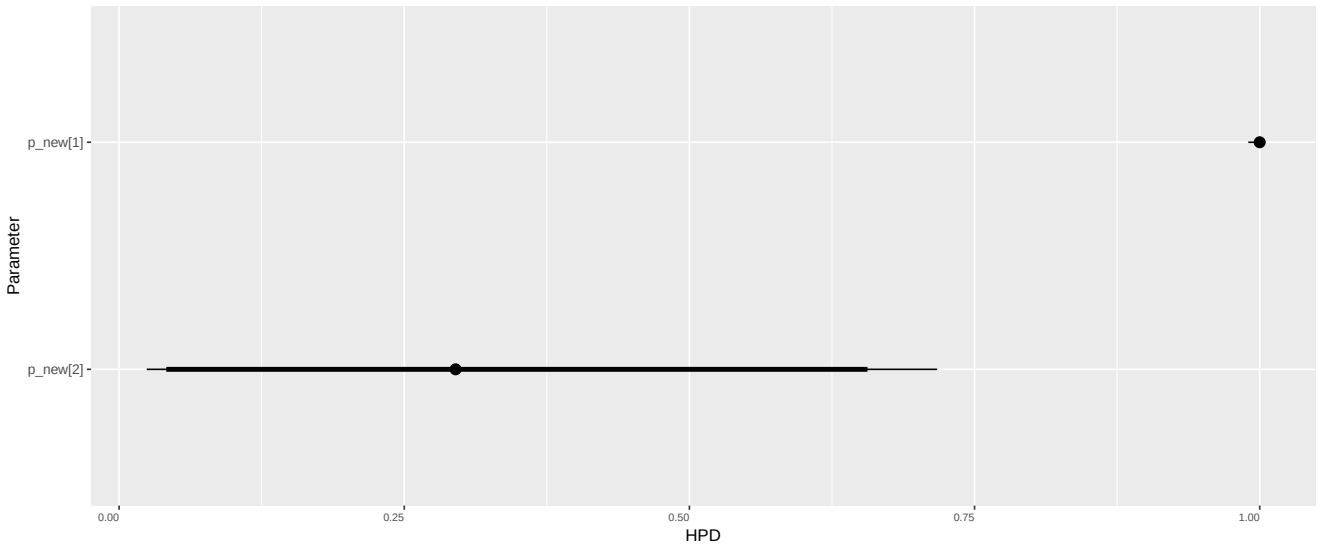


The posterior distributions are roughly the same for each chain. β_0 is our intercept and β_1 is clearly shown to have negative values, which implies that an increase in age leads to a decrease in probability that the answer about the state of the uK economy will be optimistic.

2.2.2 We now modify our code to provide the posterior medians of and 95% credible intervals for the probability that a person aged 18 and a person aged 50 would respond 'Yes':

```
Bayesian_binary_logistic_model_predict <- function(){
  #
  # Define the likelihood part of the model
  #
  for(i in 1:n_obs){
    #
    y[i] ~ dbin(p[i], 1 )
    #
    # Link function
    #
    # logit(p) in BUGS give log(p / (1 - p))
    #
    # Linear predictor
    #
    logit(p[i]) <- eta[i]
    #
    eta[i] <- beta_0 + beta_1 * x[i]
  }
  #
  # Specify the prior distributions on the unknown parameters
  #
  beta_0 ~ dnorm(0.0, 1.0E-4) # Prior on beta_0 is normal with low precision
  beta_1 ~ dnorm(0.0, 1.0E-4) # Prior on beta_1 is normal with low precision
  #
  # Evaluate the linear predictor at the new value of Dose, called x_new
  #
  # we use a for loop in order to predict more than one propabilities
    for(i in 1:length(x_new)){
    eta_new[i] <- beta_0 + beta_1 * x_new[i]
    #
    # Convert to the probability scale
    #
    p_new[i] <- exp(eta_new[i]) / (1 + exp(eta_new[i]))
    }
}
```

- We represent the credible intervals for the p_{1new},p_{2new} using **ggs_caterpillar** function and a numeric representation:




```
#                mean          sd       2.5%     97.5%
# p_new[1] 0.9989850 0.005076263 0.99000968 1.0000000
# p_new[2] 0.3153035 0.194172694 0.02423799 0.7171676
```

From the results we can clearly see that younger people are generally more optimistic for the future and are more likely to respond 'Yes; in the survey concerning the UK economy. On the other hand, older people seem to be more pessimistic and suspicious about the future so are more likely to repond no to the same question. The analysis was done over the course of the question clearly shows this trend as the probabilities increase with the age. The Bayisian prediction we made here reveals that 99% of people aged 18 are going to respond "Yes" to this question with a very small standard deviation while people age 50 are more likely to respond "No". However, the standard deviation for older people seems to be much greater than for younger people which means that this is different may be explained by other parameters when we are considering about olders people thoughts.

2.2.3 Suggestion and recommndations about future survey and why older people are generally more or less pessimistic than younger people about what will happen to the economy in the next 12 months.

Following our conclusion from the last question we need to include some other variables in our dataset to ensure that we cover other factors, such as the education level of the respondents, which may affect the results. Another possible factor could be the income or social group of the respondents, again which may influence their optimism. Essentially, we need more *quality characteristics* of the population we are studying in order to make more accurate analysis.

In addition, the use of a closed, binary question on the questionnaire may not reveal the extent to which individuals agree or disagree with a statement. This could be rectified by the use of a likert scale, which allows for a greater number of categories. Alternatively, an open question about what the participant thinks about the economy could be asked, with their answer noted and analysed using sentiment analysis to identify positive and negative words. One of the reasons why older people may be more sceptical about the economy may be because of their

experience with past and possibly because of differences in their political leanings.

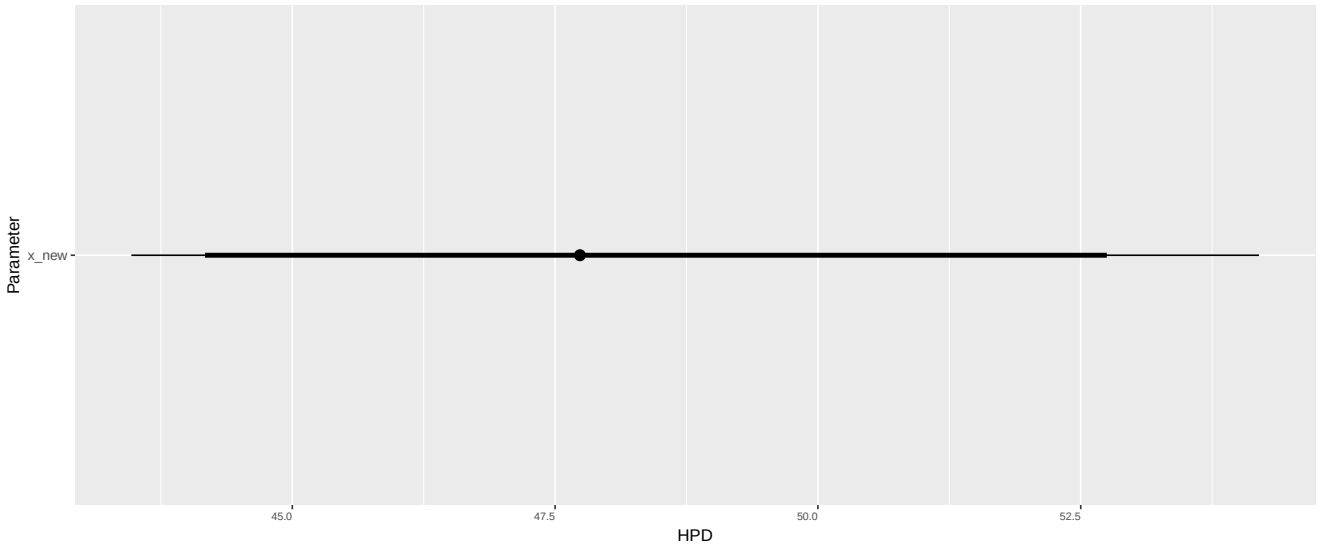
2.2.4 We modify our code to provide the posterior median of a 95% credible interval for the age at which:

$$p = Pr(Response = 1) = \frac{1}{2}$$

```
Bayesian_binary_logistic_model_predict_xnew <- function(){
  #
  # Define the likelihood part of the model
  #
  for(i in 1:n_obs){
    #
    y[i] ~ dbin(p[i], 1)
    #
    # Link function
    #
    # logit(p) in BUGS give log(p / (1 - p))
    #
    # Linear predictor
    #
    logit(p[i]) <- eta[i]
    #
    eta[i]<- beta_0 + beta_1 * x[i]
  }
  #
  # Specify the prior distributions on the unknown parameters
  #
  beta_0 ~ dnorm(0.0, 1.0E-4) # Prior on beta_0 is normal with low precision
  beta_1 ~ dnorm(0.0, 1.0E-4) # Prior on beta_1 is normal with low precision
  #
  # Evaluate the linear predictor at the new value of Dose, called x_new
  #
  eta_new <- logit(p_new)
  x_new <- (eta_new - beta_0) / beta_1

}
```

- I order to input initial values to β_0, and β_1 we create a function name **beta_inits** and we load in the **jags** function:

```
# we include the function beta_inits in order to give the initial values for the
# slope and intecept
beta_inits <- function(){
  list("beta_0"=0.0001,"beta_1"=0.0001)
}
```



We can see that the age with proportion up to 50% of answers are close to the mean of the age of our data set. This result is consistent with the limitations discussed above and we could not make many explicit assumptions about the data. We observed very big credible intervals when age is lower than 46 years old, the credible interval length is lower when the age is greater than 46 years old, which implies there is generally more consensus amongst older people, We also found that the thick intervals are greater when $x < 46$ and lower when $x > 46$. Because we have slightly more observations for younger people this might be a reason for the grater intervals for the older people, as we know larger sample sizes are associated with more confidence in our findings.

3 References

R Core Team (2018). _R: A Language and Environment for Statistical Computing_. R Foundation for Statistical Computing. Vienna, Austria. <URL: https://www.R-project.org/>.

Ripley, B. (2018). _tree: Classification and Regression Trees_. R package version 1.0-39. <URL: https://CRAN.R-project.org/package=tree>.

Wickham, H. and L. Henry (2018). _tidyr: Easily Tidy Data with 'spread()' and 'gather()' Functions_. R package version 0.8.0. <URL: https://CRAN.R-project.org/package=tidyr>.

Xie, Y. (2018). _knitr: A General-Purpose Package for Dynamic Report Generation in R_. R package version 1.20.. <URL: https://CRAN.R-project.org/package=knitr>.

Auguie, B. (2017). _gridExtra: Miscellaneous Functions for "Grid" Graphics_. R package version 2.3. <URL: https://CRAN.R-project.org/package=gridExtra>.

Boettiger, C. (2017). _knitcitations: Citations for 'Knitr' Markdown Files_. R package version 1.0.8. <URL: https://CRAN.R-project.org/package=knitcitations>.

MW, M. (2017). _RefManageR: Import and Manage BibTeX and BibLaTeX References in R._ The Journal of Open Source Software. DOI: 10.21105/joss.00338. <URL: http://doi.org/10.21105/joss.00338>.

Wickham, H. (2017). _scales: Scale Functions for Visualization_. R package version 0.5.0. <URL: https://CRAN.R-project.org/package=scales>.

Wickham, H, R. Francois, L. Henry, et al. (2017). _dplyr: A Grammar of Data Manipulation_. R package version 0.7.4. <URL: https://CRAN.R-project.org/package=dplyr>.

Wickham, H, J. Hester and R. Francois (2017). _readr: Read Rectangular Text Data_. R package version 1.1.1. <URL: https://CRAN.R-project.org/package=readr>.

Fernández-i-Mar\'in, X. (2016). "ggmcmc: Analysis of MCMC Samples and Bayesian Inference". In: _Journal of Statistical Software_ 70.9, pp. 1-20. DOI: 10.18637/jss.v070.i09.

Plummer, M. (2016). _rjags: Bayesian Graphical Models using MCMC_. R package version 4-6. <URL:

https://CRAN.R-project.org/package=rjags>.

Su, Y. and M. Yajima (2015). _R2jags: Using R to Run 'JAGS'_. R package version 0.5-7. <URL: https://CRAN.R-project.org/package=R2jags>.

Wickham, H. (2009). _ggplot2: Elegant Graphics for Data Analysis_. Springer-Verlag New York. ISBN: 978-0-387-98140-6. <URL: http://ggplot2.org>.

Venables, W. N. and B. D. Ripley (2002). _Modern Applied Statistics with S_. Fourth. ISBN 0-387-95457-0. New York: Springer. <URL: http://www.stats.ox.ac.uk/pub/MASS4>.