Modelling and Analytics for Data Science

Paul Muriithi

2022-07-29

The following libraries are required.

```
### REQUIRED LIBRARIES ####
library(CalvinBayes)
library(R2jags)
library(readxl)
library(knitr)
library(dplyr)
library(magrittr)
library(ggplot2)
library(MASS)
library(caret)
library(tree)
library(ISLR)
library(e1071)
#library(tidyverse)
```

Load the Data

The *cushings.xlsx* dataset consists of 4 variables ($Patient, Pn, THC$, and $Type$) and 150 observations as confirmed by the function *dim*(). The data is a collection of patients' with either adenoma, bilateral hyperplasia, or carcinoma syndrome based on urinary excretion rates(mg/24hr) of Pregnanetriol (Pn) and Tetrahydrocortisone(THC) steroid metabolites.

```
# import the data
cushings_df <- read_excel("cushings.xlsx", 1)
# Convert characters to factor
cushings_df <- cushings_df %>%
  dplyr::select(Pn, THC, Type) %>%
  mutate(across(where(is.character), factor))

# confirm shape of the dataset
dim(cushings_df)
```

```
## [1] 150   3
```

```
# View the first 7 rows
kable(head(cushings_df,7), caption = "The First 7 Observations")
```

Table 1: The First 7 Observations

Pn	THC	Type
8.67	2.34	a
1.30	3.00	a

Pn	THC	Type
0.10	1.90	a
0.04	5.16	a
1.11	4.10	a
0.40	1.90	a
1.00	8.33	b

3.1 Machine Learning Task

Machine Learning part(a)

Figure 1 shows the relationship between the two urinary excretion rates, Pn and THC, grouped by gender.

```
#### SCATTER PLOTS  ####
# Scatter plot for Pn against THC
p <- ggplot(cushings_df, aes(x=THC, y=Pn, color=Type, shape=Type)) +
  geom_point()+
  # add title and name the x and y-axis
  labs(title="Pn Vs. THC",
       x="THC(mg/24h)", y = "Pn(mg/24h)")+
  theme(plot.title = element_text(hjust = 0.5))
p + theme_bw()
```

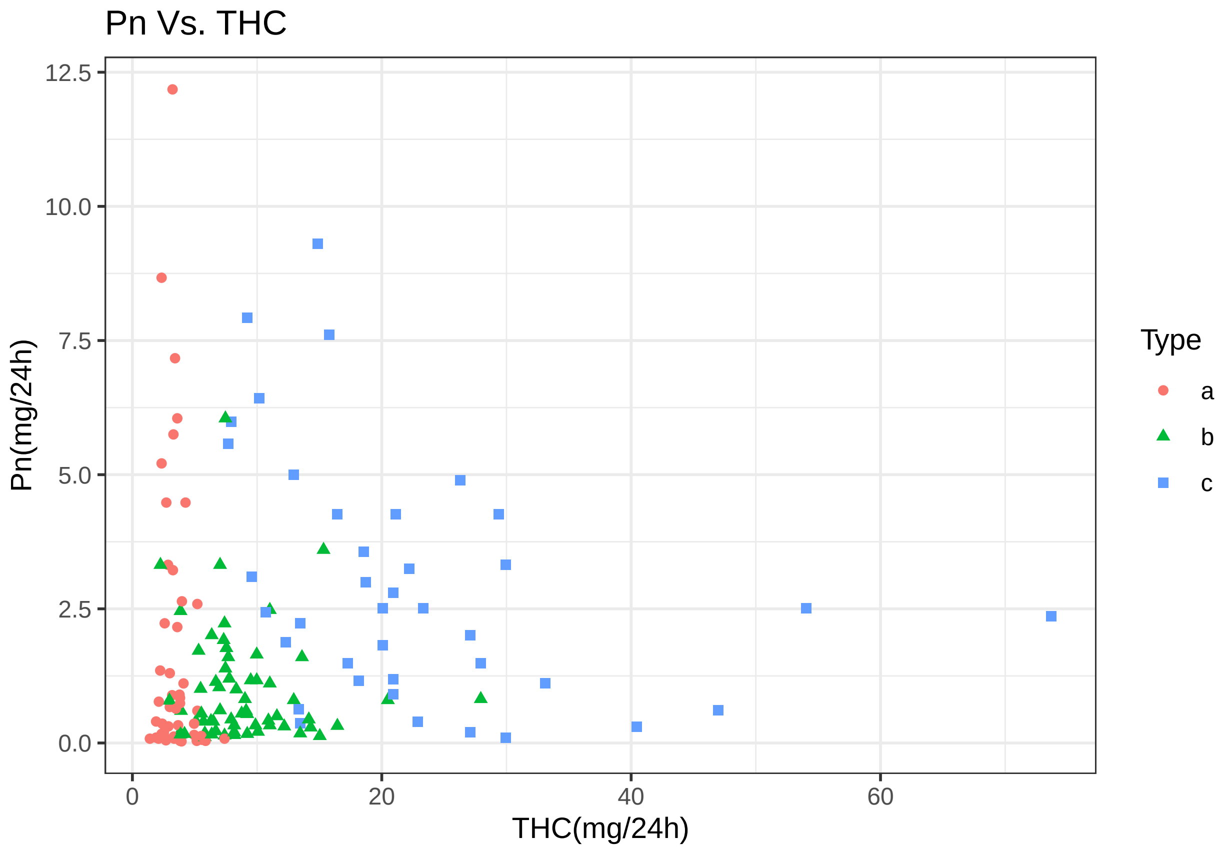


Figure 1: Scatter plot of Pregnanetriol against Tetrahydrocortisone by by type of syndrome.

From the scatter plot above, the relationship appear to be funnel-shaped indicating that the points tends to be widely spread as the two urinary excretion rates increases. This implies that there was a reporting bias during data collection or presence of missing relevant studies. It also imply the absence of symmetry, hence we can arrive at a reasonable conjecture that the data is not normally distributed, non-linearity, and the heterogeneity of variances. Additionally, even though there is a likelihood that this scenario may have occurred by chance given that our data is not "large enough", a greater probability could support that it was as a result of sub-group effects. The shape of the scatter plot therefore becomes distorted due to the presence of these sub-group effects.

To remove this asymmetrical shape (skewness), the data needs to be transformed to near normal in order to achieve a homogeneity of variances. One way of achieving this objective is through log-transformation. By so doing, the patterns in the data becomes more sensible and more interpretable, after reducing the skewness. The log-transformation of independent variables (Pn and THC) has been conduced in the chunk below, and the resulting data is visualized in Figure 2.

```r
# Log-transformation
cushings <- cushings_df
cushings$Pn <- log(cushings$Pn)
cushings$THC <- log(cushings$THC)
cushings <- cushings %>%
  rename(lPn = Pn, lTHC = THC)
kable(head(cushings,3), caption = "Log-transformed data")
```

Table 2: Log-transformed data

lPn	lTHC	Type
2.1598688	0.8501509	a
0.2623643	1.0986123	a
-2.3025851	0.6418539	a

```r
# Scatter plot of log trasformed data
p1 <- ggplot(cushings, aes(x=lTHC, y=lPn, color=Type, shape=Type)) +
  geom_point()+
  # add title and name the x and y-axis
  labs(title="lPn Vs. lTHC",
       x="THC(mg/24h)", y = "Pn(mg/24h)")+
  theme(plot.title = element_text(hjust = 0.5))+
  theme_bw()
p1
```

Part(b): Split the data into Training and Test sets

The chunk below generates a randomly ordered cushings data frame in a predefined sequence determined by the *set.seed*() function. Setting up the *set.seed*() functions is done to ensure the reproducibility such that a repeated analysis obtains identical the results. The *runif*(150) function creates a list 150 random values to match the 150 records in the cushings data set, which are then sorted position-wise by the *order*() function. It is by these positions that we in turn use to select rows which are then stored in a new data frame called *Cushings_rand*. The new data is then split into 130 observations as training set and 20 records as testing set.

```r
#### DATA PARTITIONING ####
set.seed(15)
# Generate random values
Cushings_rand <- cushings[order(runif(150)), ]
```

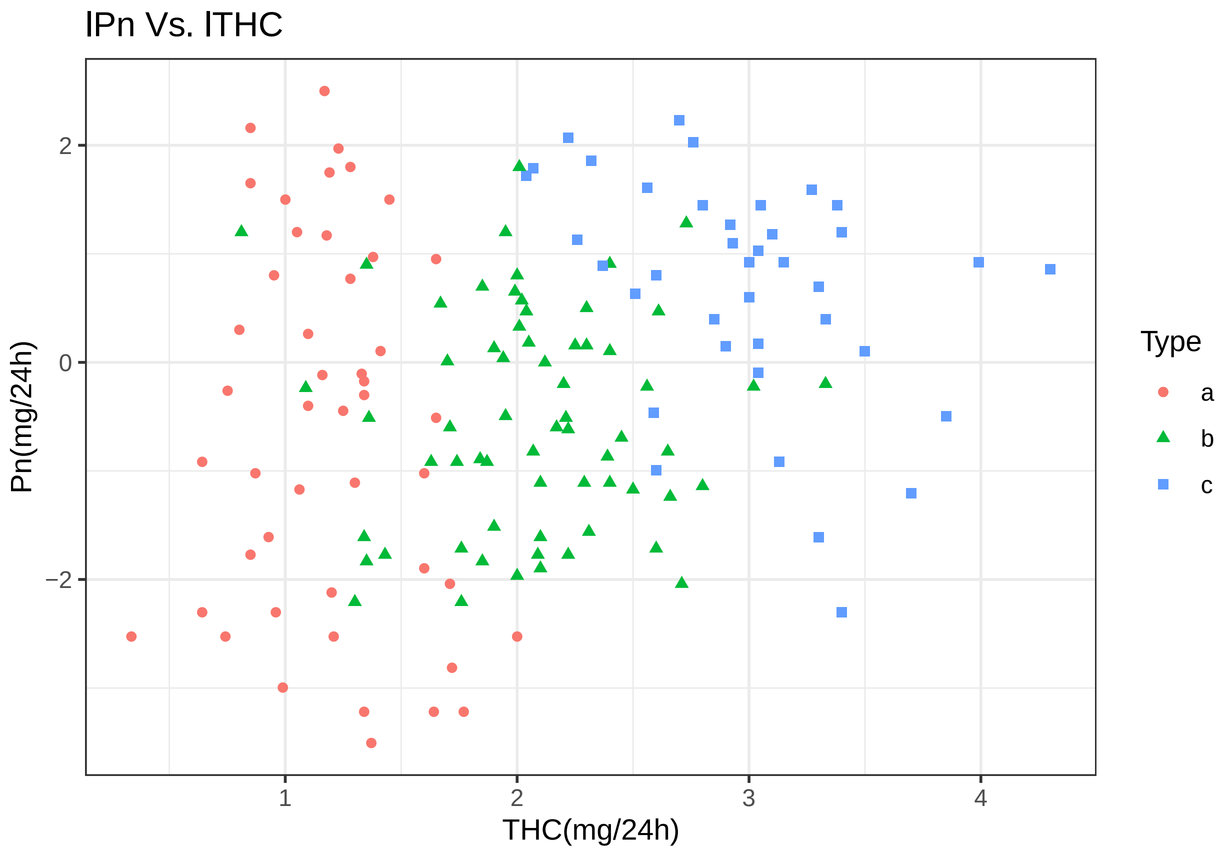


Figure 2: A scatter Plot of Pn as a function of THC grouped by the Type of syndrome after log transformation

```r
# Split the data into training and testing sets
training <- Cushings_rand[1:130, ]
testing <- Cushings_rand[131:150, ]
dim(training)
```

```
## [1] 130   3
```

```r
dim(testing)
```

```
## [1] 20  3
```

Part(c): Linear Discriminant Analysis

A *Linear Discriminant Analysis* is a technique used to reduce dimentionality in a dataset while retaining important information. The algorithm starts by finding a linear combinations of predictor variables, also called linear discriminant, to find the best possible separation between groups in the independent variable (Type).

This algorithm is based under the assumption that the independent variables follows a Gaussian distribution, and that the data in non-skewed absent outliers. Another assumption is the group-specific means and equality of variances. In this analysis we assume that these assumptions holds, given that the log-transformation of predictors has already been done.

The aim of performing a Linear Discriminant Analysis in this case is to determine to the best linear combination that Pn and THC that provides the best possible separation between the three types of syndrome.

```r
#### LINEAR DISCRIMINANT ANALYSIS, LDA ####
# Fit LDA model
LDA <- lda(Type ~ lPn+lTHC, data=training)
LDA
```

```
## Call:
## lda(Type ~ lPn + lTHC, data = training)
##
## Prior probabilities of groups:
##         a         b         c
## 0.3076923 0.4384615 0.2538462
##
## Group means:
##          lPn     lTHC
## a -0.4011809 1.252914
## b -0.4828142 2.079860
## c  0.5785799 3.060015
##
## Coefficients of linear discriminants:
##            LD1        LD2
## lPn  0.2677451  0.7439484
## lTHC 2.3018818 -0.4135992
##
## Proportion of trace:
##    LD1    LD2
## 0.9861 0.0139
```

The LDA model has determined the group means and, for each patient, computed the corresponding probabilities of falling under the three different categories of syndrome. There are 2 predictor variables and three classes $G = 3$ in the target variable (Type). This means that we can only have a maximum number of discriminant functions equals to $G - 1 = 2$, separating the three classes based on the two excretion rates.

The following elements constitute our LDA output:

- Prior probabilities of groups: These are the already existing proportions of the three classes in the training set. In this case, we have 31% of the patients having adenoma, 44% and 25% with bilateral hyperplasia and carcinoma respectively.
- Group Means: These represent the averages of urinary excretion rates within each type of syndrome. The *lda*() output above suggests that Type (c) syndrome *carcinoma* is the greatest cause of both Tetrahydrocortisone(THC) and Pregnanetriol(Pn) urinary excretion rates (Group means equals 0.5785799 and 3.060015 respectively).
- Coefficients of linear discriminants: These are the linear combinations of the 2 explanatory variables, Pn and THC that forms the LDA functions.
- The two models are: $0.2677(lPn) + 2.3019(lTHC)$ and $0.7439(lPn) - 0.4136(lTHC)$
- Proportion of trace: This is the proportion achived by each functions during the separation. For the cushings data st, we get the percentage proportions as 98.61% LDA1 and 1.39% LDA2.

```
# LDA Model Predictions
pred <- LDA %>% predict(training) #training
predicted <- LDA %>% predict(testing) #testing

# Visualize the results
LDA.dat <- cbind(training, predict(LDA)$x)
p2 <- ggplot(LDA.dat, aes(LD1, LD2)) +
  geom_point(aes(color = Type)) +
  geom_text(aes(label = Type), hjust = - 0.5) +
  labs(title="lPn Vs. lTHC",
       x="THC(mg/24h)", y = "Pn(mg/24h)")+
  theme(plot.title = element_text(hjust = 0.5))+
  theme_bw()
p2
```

Although not perfect enough, we can see that the first discriminant function(LDA1: x-axis) has separated the three classes while the second function fails(y-axis).

```
# Training Error
confMat_training <- table(list(predicted=pred$class, observed=training$Type))
# Test Error
confMat_testing <- table(list(predicted=predicted$class, observed=testing$Type))
confusionMatrix(confMat_training)
```

```
## Confusion Matrix and Statistics
##
##          observed
## predicted  a  b  c
##         a 35  7  0
##         b  5 46  6
##         c  0  4 27
##
## Overall Statistics
##
##                Accuracy : 0.8308
##                  95% CI : (0.7551, 0.8908)
##     No Information Rate : 0.4385
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

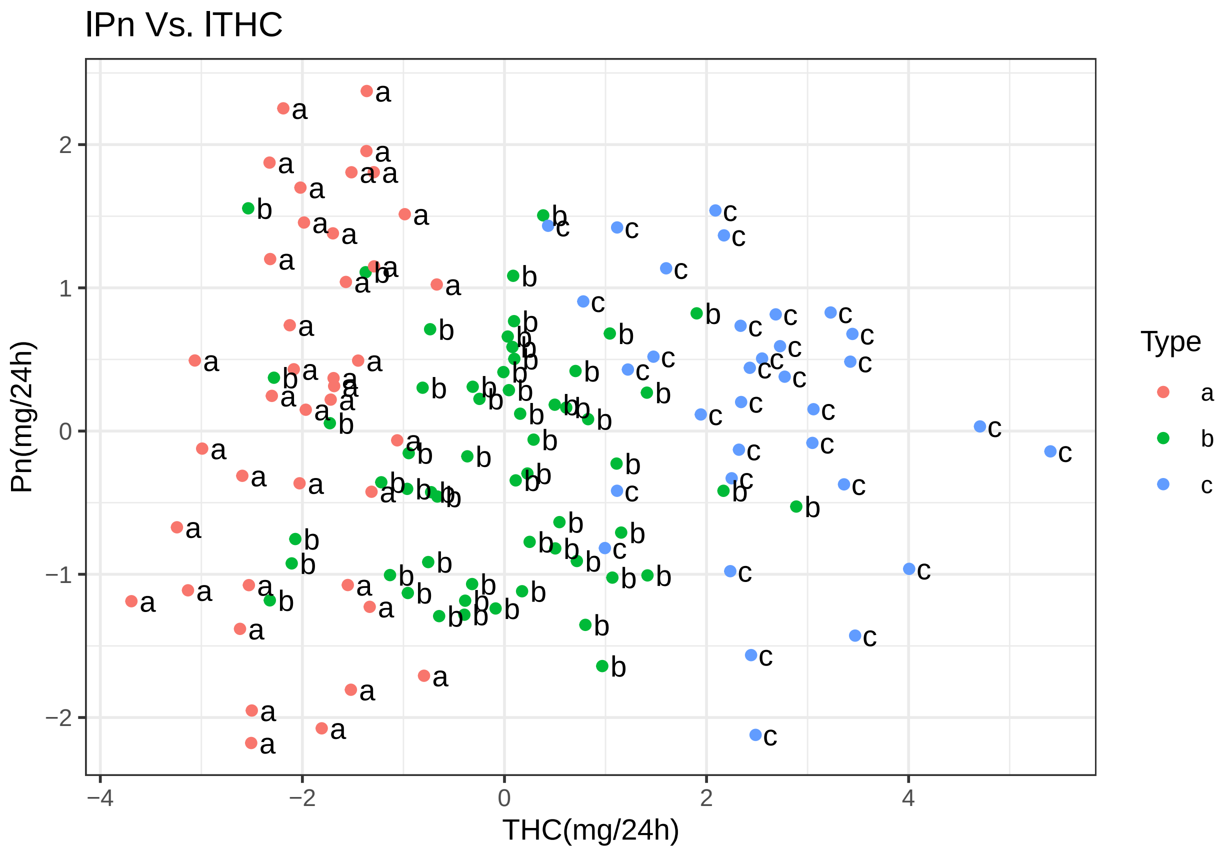


Figure 3: LDA output Grouped by Type

```
##                    Kappa : 0.7388
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                       Class: a Class: b Class: c
## Sensitivity             0.8750   0.8070   0.8182
## Specificity             0.9222   0.8493   0.9588
## Pos Pred Value          0.8333   0.8070   0.8710
## Neg Pred Value          0.9432   0.8493   0.9394
## Prevalence              0.3077   0.4385   0.2538
## Detection Rate          0.2692   0.3538   0.2077
## Detection Prevalence    0.3231   0.4385   0.2385
## Balanced Accuracy       0.8986   0.8282   0.8885
```

The LDA model has 83% accuracy and 0.17 error rate on training set having predicted $35 + 46 + 27$ out of 130 correctly, and 25% error rate on testing set with 15/20 correct predictions.

```r
confusionMatrix(confMat_testing)
```

```
## Confusion Matrix and Statistics
##
##           observed
## predicted a b c
##         a 6 1 0
##         b 1 6 3
##         c 0 0 3
##
## Overall Statistics
##
##                Accuracy : 0.75
##                  95% CI : (0.509, 0.9134)
##     No Information Rate : 0.35
##     P-Value [Acc > NIR] : 0.0003106
##
##                   Kappa : 0.6198
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                       Class: a Class: b Class: c
## Sensitivity             0.8571   0.8571   0.5000
## Specificity             0.9231   0.6923   1.0000
## Pos Pred Value          0.8571   0.6000   1.0000
## Neg Pred Value          0.9231   0.9000   0.8235
## Prevalence              0.3500   0.3500   0.3000
## Detection Rate          0.3000   0.3000   0.1500
## Detection Prevalence    0.3500   0.5000   0.1500
## Balanced Accuracy       0.8901   0.7747   0.7500
```

Part(d): Classification Tree

Decision trees are advantageous in that they are easy to make predictions fast and easy to interpret. In this part, the goal is to predict which type of patients having to either of the the threes types of syndromes based on different levels of urinary excretion rates. The *tree*() function is used to create the classification tree and *plot*() for visualizing the resulting model.

```
#### CLASSIFICATION TREE ####
tree_cushings <- tree(Type ~ lPn+lTHC, training)

# display the tree
plot(tree_cushings)
text(tree_cushings,pretty=0)
```

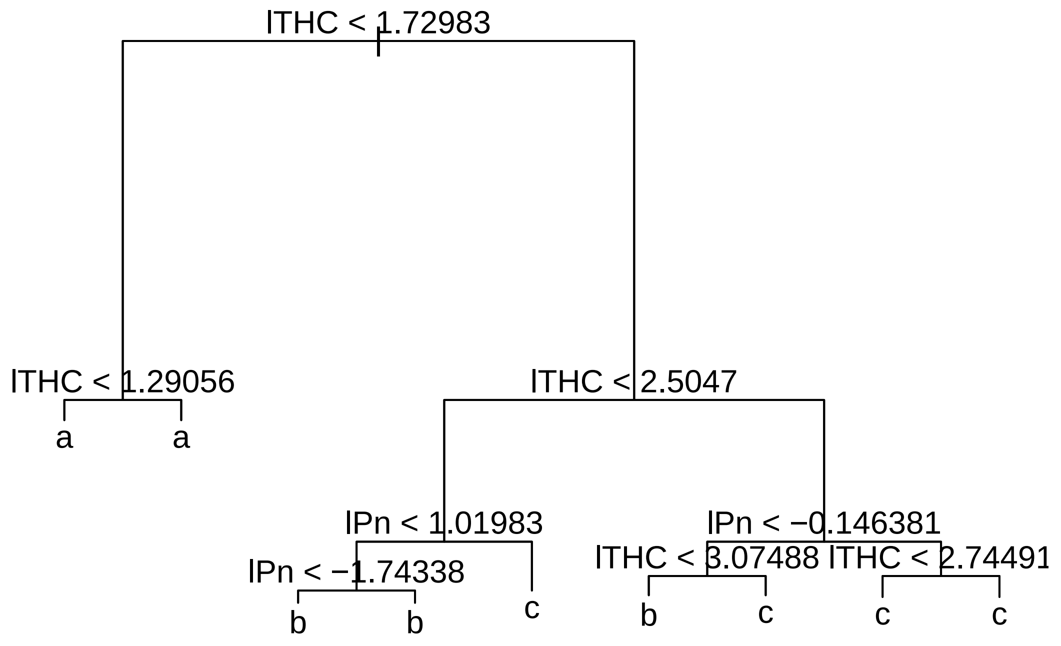


Figure 4: Classification tree Showing Type of Syndrome at each Termina Node and Features at the Interval Nodes.

Our classification tree can be summarized by using *summary*(). From the summary output, the tree has 9 terminal nodes with a misclassification error rate of 0.1462 having incorrectly classified 19 observations.

```
# print the summary of the classification tree
summary(tree_cushings)
```

```
##
## Classification tree:
## tree(formula = Type ~ lPn + lTHC, data = training)
## Number of terminal nodes:  9
## Residual mean deviance:  0.6721 = 81.33 / 121
## Misclassification error rate: 0.1462 = 19 / 130
```

The classification tree above has classified all Type (a) syndrome patients as having lTHC less than 1.7289. All Type (b) patients with lTHC less than 2.5047 have $lPn < 1.0198$ and most of the Type (c) patients are having $lTHC > 2.5047$. The summary of the original tree lists the number of terminal nodes and misclassification error rate. In this tree, we have 14.62% being misclassified.

Another advatage of using classification trees is the ability to provide optimal complexity level of the tree, which is complexity pruning of the cost, to improve the performance and ease the interpretation of the results. The *cv.tree*() helps to do this by performing cross-validation. *Prune.misclass* argument ensures that our cost function, misclassification error rate, is used to guide the process of pruning and cross-validation.

The cv.tree() output suggests that the a with 5 tree terminal nodes as the classification tree with the lowest error rate, with 30 cross-validation errors. The results above can make more sense when visualized as shown in the figure 5 below.

```r
# determine optimal tree size
set.seed(123)
optmal <- cv.tree(tree_cushings,FUN = prune.misclass)
# plot of cross-validation error rate vs tree size
plot(optmal$size,optmal$dev, xlab = "Tree size", ylab = "error rate",type='b')
```

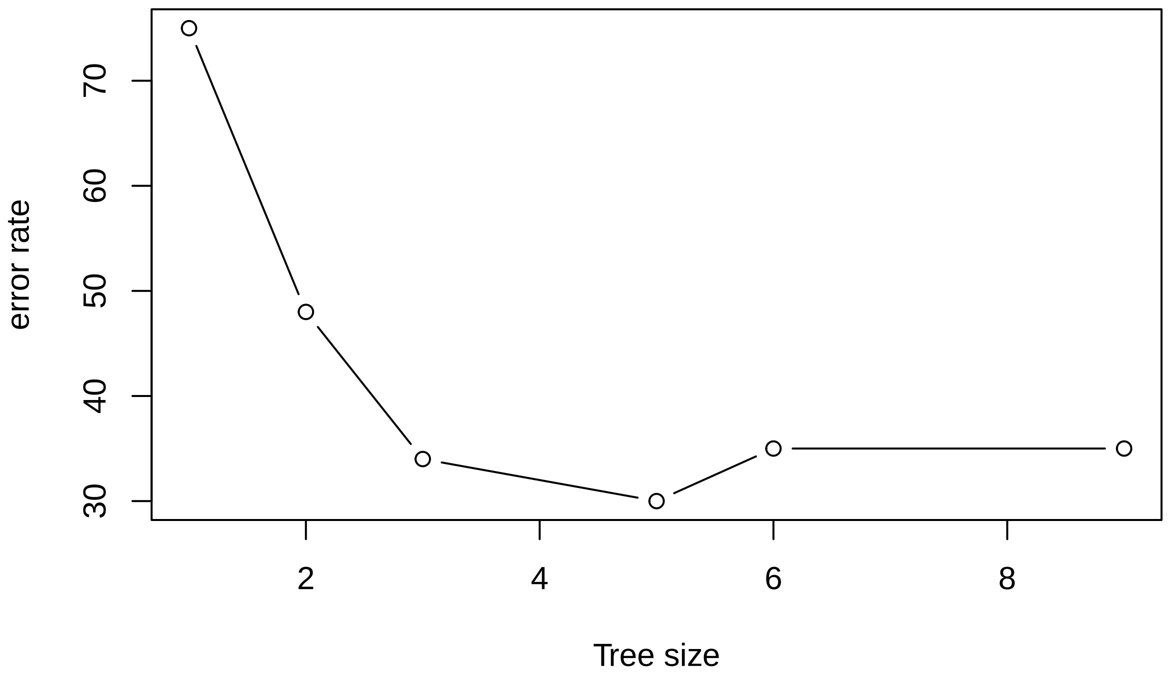


Figure 5: A graph of cross-validation error rate against tree size size

The final model in Figure 6 shows that patients having Type (a) syndrome are never misclassified where they all have $log(THC) < 1.7298$, while Types (b) and (c) with $log(Pn) < -0.1464$ tends to be misclassified.

```r
# visualize the results
classification_tree = prune.misclass(tree_cushings, best=5)
plot(classification_tree)
text(classification_tree,pretty=0)
```

The error rates from the final model's results indicates that 15.38% patients have been misclassified as either Type (b) or Type (c).

```r
## training error rates
#summary(classification_tree)
```

Evaluating the model using the test set of the data, 6 Type a patients were classified as type b, 1 type b was classified as type a and 3 as type c, and all the 3 type c patients were correctly classified by the model. This leads to a 25% error rate.

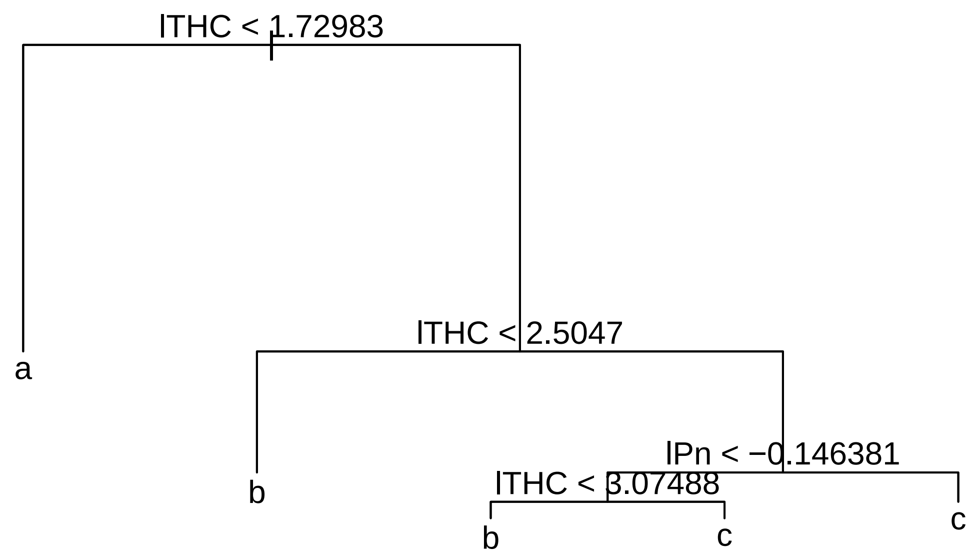


Figure 6: Final Cassification tree with 5 Terminal Nodes

```
# testing error rate
preds = predict(classification_tree, newdata = testing, type = "class")
conf_Mtrx = table(preds, testing$Type)#confusion matrix
conf_Mtrx
```

```
##
## preds a b c
##     a 6 1 0
##     b 1 6 3
##     c 0 0 3
```

```
accuracy = round(mean(preds == testing$Type)*100, 2)
print(paste0("Accuracy: ", accuracy, "%"))
```

```
## [1] "Accuracy: 75%"
```

```
print(paste0("Error rate: ", 100 - accuracy, "%"))
```

```
## [1] "Error rate: 25%"
```

Part(e): Linear Support Vector Machine (SVM)

```
#### LINEAR SUPPORT VECTOR MACHINE ####
svm_classifier = svm(formula = Type ~ .,
                 data = training,
                 type = 'C-classification',
                 kernel = 'linear')

svm_classifier
```

```
##
## Call:
## svm(formula = Type ~ ., data = training, type = "C-classification",
##     kernel = "linear")
##
```

```
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  69
```

```
## Model predictions and accuracy
# training accuracy
pred_train <- predict(svm_classifier, training)
conf_M = table(pred_train, training$Type)#confusion matrix
acc = round(mean(pred_train == training$Type)*100, 2)
print(paste0("Accuracy: ", acc, "%"))
```

```
## [1] "Accuracy: 83.85%"
```

```
# testing accuracy
pred_test <- predict(svm_classifier, testing)
conf_Mtrx = table(pred_test, testing$Type)#confusion matrix
accrcy = round(mean(pred_test == testing$Type)*100, 2)
print(paste0("Accuracy: ", accrcy, "%"))
```

```
## [1] "Accuracy: 75%"
```

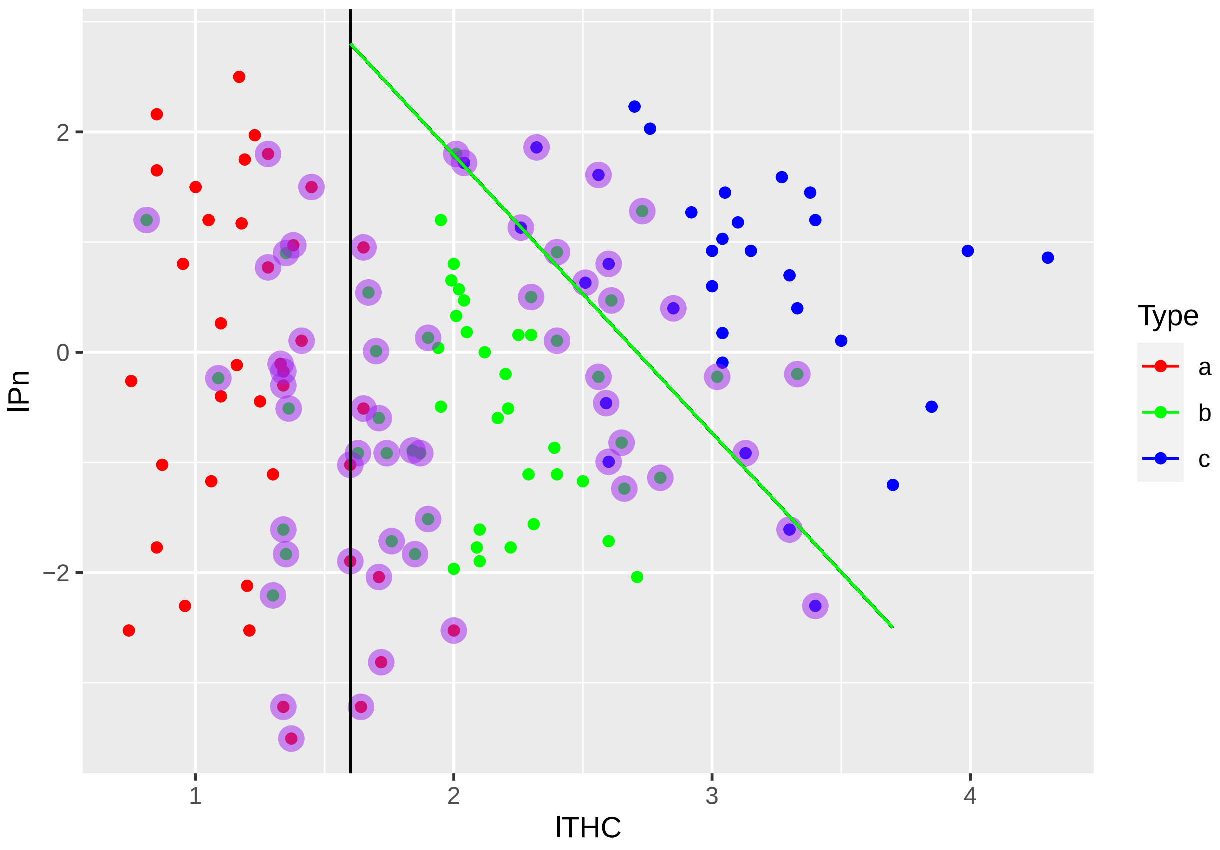
The Linear SVM model has 69 number of support vectors and when tested using the training and testing set of the data, it result in an 83.85% and 75% accuracy respectively.

Tuning the model yields a model with an optimal cost parameter value as 4.

```
## Model tuning
set.seed(1)
tuned <- tune(svm, Type~., data = training,
              ranges = list(gamma = 2^(-1:1),cost = 2^(2:5)),
              type = "C-classification", kernel = "linear")

# optimal cost
tuned$best.parameters$cost
```

```
## [1] 4
```

```
# create scatter plot of training set
scatter_plot <- ggplot(data = training,aes(x = lTHC, y = lPn, color = Type)) +
  geom_point() + scale_color_manual(values = c("red", "green", "blue"))

# adding a layer marking out the support vectors best model
layered_plt <- scatter_plot +
  geom_point(data = training[tuned$best.model$index, ], aes(x = lTHC, y = lPn),
                             color = "purple",fill = "white",size = 4, alpha = 0.5) +
  geom_vline(xintercept = 1.6) +
  geom_segment(aes(x = 1.6, y = 2.8, xend = 3.7, yend = -2.5))

#display the layered plot
layered_plt
```



```r
# training error rates
pred_train <- predict(tuned$best.model, training)
acc = round(mean(pred_train == training$Type)*100, 2)
print(paste0("Training Error rate: ", 100-acc, "%"))
```

```
## [1] "Training Error rate: 16.15%"
```

```r
# testing accuracy
pred_test <- predict(tuned$best.model, testing)
accrcy = round(mean(pred_test == testing$Type)*100, 2)
print(paste0("Testing Error rate: ", 100-accrcy, "%"))
```

```
## [1] "Testing Error rate: 20%"
```

The linear SVM model fitted on the cushings dataset with optical cost equals 4 has resulted in a final model with 16.15% and 20% error rates on training and testing sets respectively.

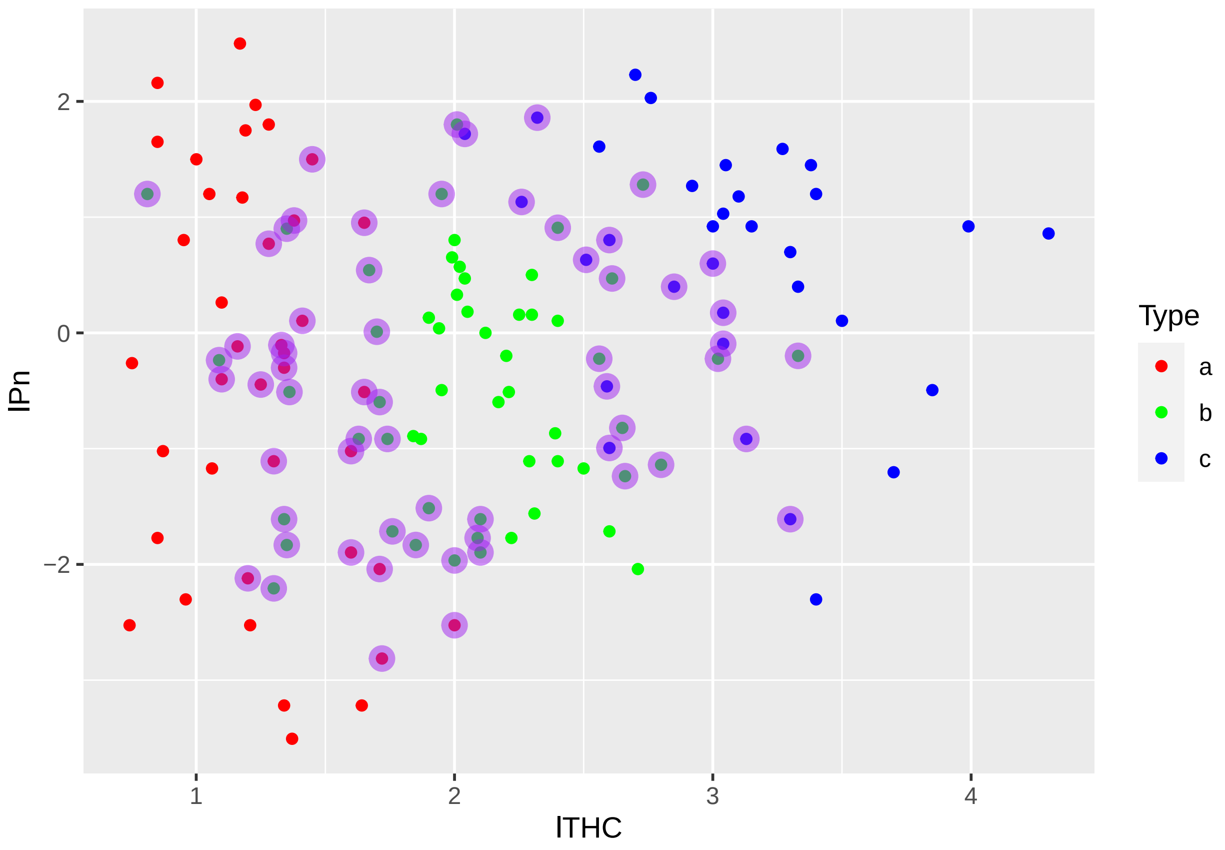
Part(f): Polynomial Support Vector Machine

```r
#### POLYNOMIAL SUPPORT VECTOR MACHINE ####
svm_poly = svm(formula = Type ~ .,
               data = training,
               type = 'C-classification',
               kernel = 'polynomial',
               degree = 2)

svm_poly
```

```
##
```

```
## Call:
## svm(formula = Type ~ ., data = training, type = "C-classification",
##     kernel = "polynomial", degree = 2)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  2
##      coef.0:  0
##
## Number of Support Vectors:  93
```

```r
## Model tuning
set.seed(1)
tuned1 <- tune(svm, Type~., data = training,
              ranges = list(gamma = 2^(-1:1),
                            cost = 2^(2:4)),
              type = "C-classification",
              kernel = "polynomial")

# optimal cost
#tuned1$best.parameters$cost

# create scatter plot of training set
scatter_plot <- ggplot(data = training,
                       aes(x = lTHC, y = lPn, color = Type)) +
  geom_point() +
  scale_color_manual(values = c("red", "green", "blue"))

# adding a layer marking out the support vectors best model
layered <- scatter_plot +
  geom_point(data = training[tuned1$best.model$index, ],
                            aes(x = lTHC, y = lPn),
                            color = "purple",
                            fill = "white",
                            size = 4,
                            alpha = 0.5)

#display the layered plot
layered
```



```
# training error rates
pred_train <- predict(tuned1$best.model, training)
acc = round(mean(pred_train == training$Type)*100, 2)
print(paste0("Error rate: ", 100-acc, "%"))
```

```
## [1] "Error rate: 19.23%"
```

```
# testing accuracy
pred_test <- predict(tuned1$best.model, testing)
accrcy = round(mean(pred_test == testing$Type)*100, 2)
print(paste0("Error rate: ", 100-accrcy, "%"))
```

```
## [1] "Error rate: 10%"
```

Part(g): Comparing models

The chunk below creates a data frame of the training and testing error rates for the models (c) to (f). Based the percentage error rates, the polynomial SVM has the best performance on the testing set. Therefore, I would recommend the polynomial support vector machine given that it makes more accurate predictions on new data.

```
comparison <- data.frame(Model = c("LDA", "C_Tree", "L_SVM", "P_SVM"),
                         TrainingError = c(17, 15, 16, 19),
                         TestingError = c(26, 25, 20, 10))
kable(comparison, caption = "Comparing error rates for training and testing error rates")
```

Table 3: Comparing error rates for training and testing error rates

Model	TrainingError	TestingError
LDA	17	26
C_Tree	15	25
L_SVM	16	20
P_SVM	19	10

3.2. Frequentist Statistics

Part(a) logistic regression

Logistic regression is a suitable predictive analysis when the dependent variable is binary. The Overdrawn variable in the *Incomes* data set is our target, {"Yes", "No"}, which we intend to predict given *Income* and *Age* as the predictor variables.

```
# Load the incomes data
incomes <- read.csv("Incomes.csv", stringsAsFactors = T)
head(incomes)
```

```
##   Overdrawn Income Age
## 1       Yes   1280  47
## 2        No   1495  54
## 3        No   1615  58
## 4       Yes    720  24
## 5       Yes    486  21
## 6        No    978  32
```

A logistic regression model is created using *glm*() function in *stats* package in R.

```
# Logistic Regression
model <- glm(Overdrawn ~., data = incomes, family = binomial)
summary(model)
```

```
##
## Call:
## glm(formula = Overdrawn ~ ., family = binomial, data = incomes)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1028  -0.4340  -0.1506   0.3112   2.2836
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  6.230368   3.238216   1.924   0.0544 .
## Income      -0.004161   0.002382  -1.747   0.0807 .
## Age         -0.073082   0.050825  -1.438   0.1505
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 24.435  on 19  degrees of freedom
## Residual deviance: 11.827  on 17  degrees of freedom
## AIC: 17.827
```

```
##
## Number of Fisher Scoring iterations: 7
```

The coefficients of Income and Age are -0.004 and -0.0731 respectively, and the intercept is 6.2303. These values are in log-odds units, which gives us the relationship between our target variable and the predictors.

The logistic equation can therefore be written as;

$$\eta = log(\frac{p}{1-p}) = \frac{exp(p)}{1+exp(p)}$$

$$\eta = \frac{exp(6.2304 - 0.0042(Income) - 0.0731(Age))}{1+exp(6.2304 - 0.0042(Income) - 0.0731(Age))}$$

which reduces to

$$= \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

- where β_0 in the intercept,
- β_1 is the coefficient of Income, and
- β_2 is the coefficient of Age.
- $x_1 = Income$ and $x_2 = Age$

Substituting the intercept and coefficient of Income and Age into the equation we obtain the final model as defined by the equation

$$\frac{exp(p)}{1+exp(p)} = 6.23 - 0.004x_1 - 0.0731x_2$$

- p is the probability of going overdrawn, $p = POverdrawn = Yes$
- $Income$ - we expect the log-odds of overdrawn to decrease by 0.004 for every 1 pound increase, when age is held constant.
- Age - The log-odds of overdrawn decreases by 0.0731 for every 1 year increase in the client's age when the account was opened, when Incomes are held constant.

This implies that the odds of overdrawn is $1 - e^{-0.004} = 0.004$ when income is increased by 1 pound, and the odds when age is increased by 1 year is $1 - e^{-0.0731} = 0.0705$. Therefore every 1 pound increase in income decreases the odds of a customer going overdrawn by 0.4% when age is fixed, and odds decreases by 7.05% for every 1 year increase in customer's age when the income is held constant, when the account is opened.

```r
# confidence intervals of the odds
Confidence_Interval <- 1 - exp(confint(model, c("Income","Age")))
Confidence_Interval
```

```
##              2.5 %        97.5 %
## Income 0.01077894  0.0006284561
## Age    0.19073662 -0.0116096265
```

The 95% confidence intervals for the odds of $Income$ and Age are $[0.0108, 0.0006]$ and $[0.1907, -0.0116]$. This means that we can be 95% confident that the true odds of a newly opened account going overdrawn given the initial net monthly income and age of the client lies within these ranges.

```r
# create a new data frame
newdata <- data.frame(Income = c(500, 2000, 842), Age = c(20,50,35))
# Probabilities
probs <- model %>% predict(newdata, type = "response")
probs
```

```
##           1           2           3
## 0.936320883 0.003184571 0.542081048
```

Part (b): Frequentist Statistics

There is a relationship between age and income of client and their chance of their account to become overdrawn. Particularly the odds become even higher when the newly opened account belongs to a youth. considering a youth a potential customer aged between 18 and 35 years, there is greater than 50% odds increase that such a client's account will be more likely to go overdrawn.

3.3: Bayesian Statistics with Frequentist Analysis

Part(a)

```r
# P/E ratio data
Japan <- c(15,21,18,25,31,23,64,66,33,68)
US <- c(69,24,24,43,22,14,21,14,21,38,65,14)
n1 = length(Japan)
n2 = length(US)
mu1 = mean(Japan)
mu2 = mean(US)
sigma1 = var(Japan)
sigma2 = var(US)

# Visualize the data using density plots
x1 <- data.frame(Japan)
x2 <- data.frame(US)
Japanese <- ggplot(x1, aes(x = Japan)) +
  geom_density(lwd = 1, colour = 4,
               fill = 4, alpha = 0.25)

Us <- ggplot(x2, aes(x = US)) +
  geom_density(lwd = 1, colour = 2,
               fill = 2, alpha = 0.25)
gridExtra::grid.arrange(Japanese,Us, ncol=2)
```

The two density plots shows that the Price/Earning ratios for both Japanese companies and US companies are left skewed hence the mean ratios are less than the median. The ratios range between 10 and 70, where in most of the companies the P/E ratios are below 50 for both countries.

Part(b): Frequentist Hypothesis

In this part, according to the suggested model, we are assuming that the variances of the two samples are equal (Pooled sample t-test). We have two sets of companies: Japanese companies and US companies. Therefore we need to conduct a pooled t-test which assumes that the variances of the two population are equal, regardless of whether they are or they are not equal.

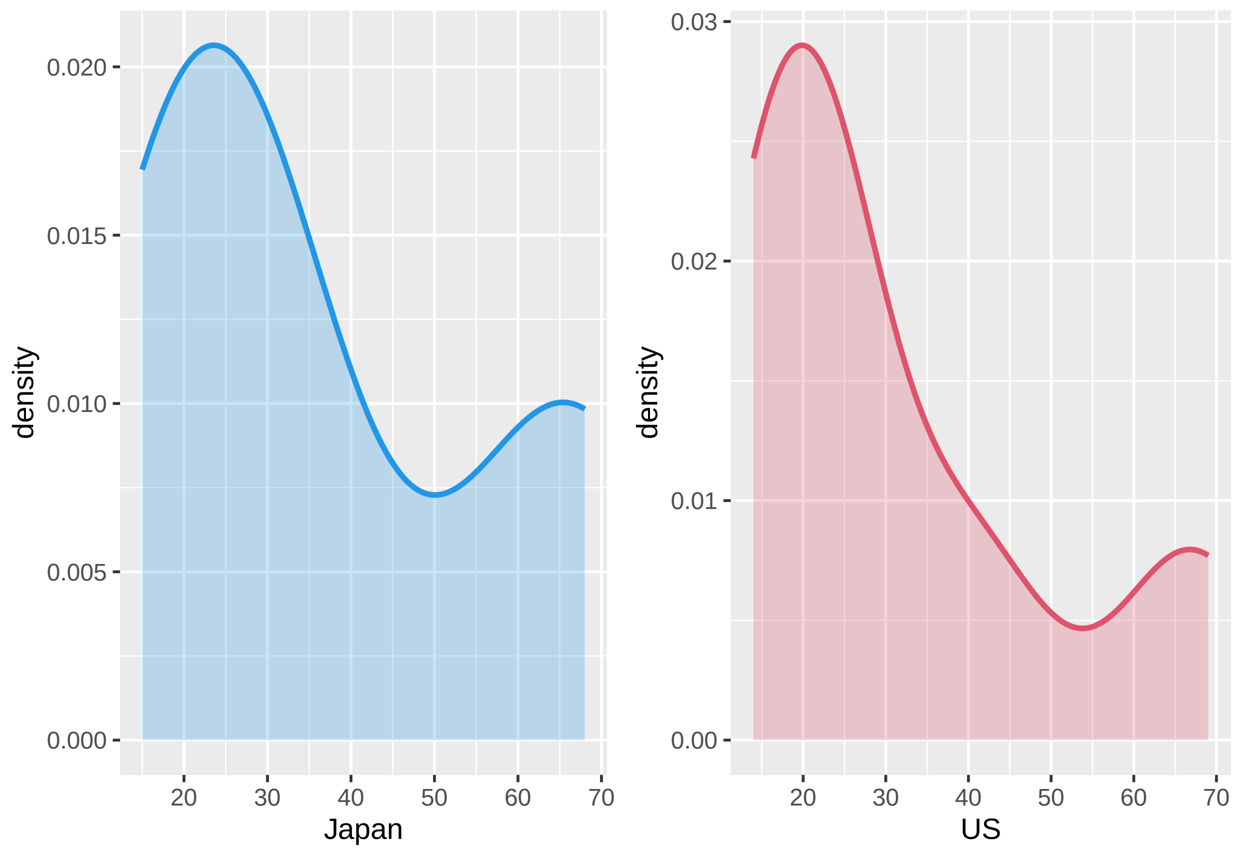


Figure 7: Density plot showing the distribution of P/E ratios.

$$Y_{1i} \sim N(\mu_1, \sigma^2), i = 1, ..., n_1$$

$$Y_{2i} \sim N(\mu_2, \sigma^2), i = 1, ..., n_1$$

- Null Hypothesis: The average price/earnings (P/E) ratio for the two sets of companies are equal.

$$H_0 : \mu_1 - \mu_2 = 0$$

- Alternative Hypothesis: The average price/earnings (P/E) ratio for the two sets of companies are not equal

$$H_a : \mu_1 - \mu_2 \neq 0$$

- Significance level: $\alpha = 0.05$
- Decision rule: If the obtained test statistic is less that the critical value, reject the null hypothesis H_0.

The pooled test statistic is calculated using the formula;

$$S_p^2 = \frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2} = (\frac{n_1 - 1}{n_1 + n_2 - 2})S_1^2 + (\frac{n_2 - 1}{n_1 + n_2 - 2})S_2^2$$

```
# pooled variance
pooled_var <- (sigma1*(n1-1) + sigma2*(n2-1))/(n1+n2-2)
pooled_var
```

```
## [1] 402.9325
```

$$= \frac{10 - 1}{10 + 12 - 2}(446.71) + \frac{12 - 1}{10 + 12 - 2}(367.11) = 402.93$$

This pooled test statistic follows a student's t distribution with $n1 + n2 - 2 = 20$ degree of freedom. That is;

$$t = \frac{\bar{x_1} - \bar{x_2}}{\sqrt{S_p^2(\frac{1}{n_1} + \frac{1}{n_2})}} = \frac{\bar{x_1} - \bar{x_2}}{S_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

```
# calculate the test statistic
t_stat <- (mu1-mu2)/sqrt((pooled_var*(1/n1 + 1/n2)))
t_stat
```

```
## [1] 0.6573726
```

$$= \frac{36.4 - 30.75}{\sqrt{18378.93(\frac{1}{10} + \frac{1}{12})}} = 0.6573$$

- Test statistic: $t = 0.6573$

```
# calculate the critical value
p = 0.05
df = n1+n2-2
t_critical <- qt(p/2, df)
t_critical
```

```
## [1] -2.085963
```

- Critical value: $t_{\alpha=0.05,df=20} = 2.086$

Conclusion: Since the test statistic is less than the critical value, we fail to reject the null hypothesis. Therefore, the data did not provide enough evidence to prove that the average Price/Earning ratios for Japanese and US companies are not equal. There is no statistically significant difference between the average ratios, hence we can assume that the two means are equal.

The pooled variance is also used to calculate the confidence interval, which takes the following form;

$$(\bar{x_1} - \bar{x_2}) \pm t_{\frac{\alpha}{2}} \sqrt{s_p^2(\frac{1}{n_1} + \frac{1}{n_2})}$$

The 94% confidence interval is then calculated as;

```
# 95% confidence interval for mu1-m2
lower_limit <- (mu1-mu2) - abs(t_critical) * sqrt(pooled_var*(1/n1 + 1/n2))
upper_limit <- (mu1-mu2) + abs(t_critical) * sqrt(pooled_var*(1/n1 + 1/n2))
lower_limit
```

```
## [1] -12.27848
```

```
upper_limit
```

```
## [1] 23.57848
```

$$(96.4 - 30.75) \pm 2.086\sqrt{402.93(\frac{1}{10} + \frac{1}{12})} = (-12.28, 23.57)$$

The confidence interval for $\mu_1 - \mu_2$ is $[-12.28, 23.57]$. This means that we can be 95% confident that the average differences of price and earning ratio between Japanese and the US companies lies between -12.28 and 23.57.

Part(c): Bayesian Inference with JAGS, Model (2).

The description of Bayesian model (2) are as follows:

$$-Likelihoods$$

$$Y_{1i} \sim N(\mu_1, \tau), i = 1, 2, ...n_1 = 10$$

$$Y_{2i} \sim N(\mu_2, \tau), i = 1, 2, ...n_2 = 10$$

$$-Priors$$

$$\mu_1 \sim N(0, 0.0001)$$

$$\mu_2 \sim N(0, 0.0001)$$

$$\tau \sim Gamma(0.001, 0.001)$$

$$\sigma^2 \sim \frac{1}{\tau}$$

Using *R2jags* package in r, JAGS takes these descriptions plus the data and in return it generates MCMC samples from posterior distribution. JAGS follows 5 basic steps:

1. Load the required data to be used. in this case the data is loaded as *Japan* and *US*.
2. Define the model using the model's descriptions(likelihood and priors).
3. Use JAGS function to compile the model.
4. Simulate samples from posterior distribution.
5. Summarizes the simulated samples.

```r
#library(R2jags)

# Data
y1 = Japan
y2 = US

# Model (2)

model1 <- "model{
# samples
for(i in 1:n1){
y1[i] ~ dnorm(mu1, tau)
}

for(j in 1:n2){
y2[j] ~ dnorm(mu2, tau)
}

# priors
mu1 ~ dnorm(0, 0.0001)
mu2 ~ dnorm(0,0.0001)
sigma2 <- 1/tau
mu1_mu2 <- mu1 - mu2
tau ~ dgamma(0.001, 0.001)
}"

params1 <- c("mu1", "mu2", "sigma2", "mu1_mu2")

# bundle the data
data1 <- list("y1","y2","n1","n2")

inits <- function (){
  list("mu1" = 0, "mu2" = 0, "tau" = 1)
}

# Write
writeLines(model1, "model1.jags")

set.seed(123)

# posterior
```

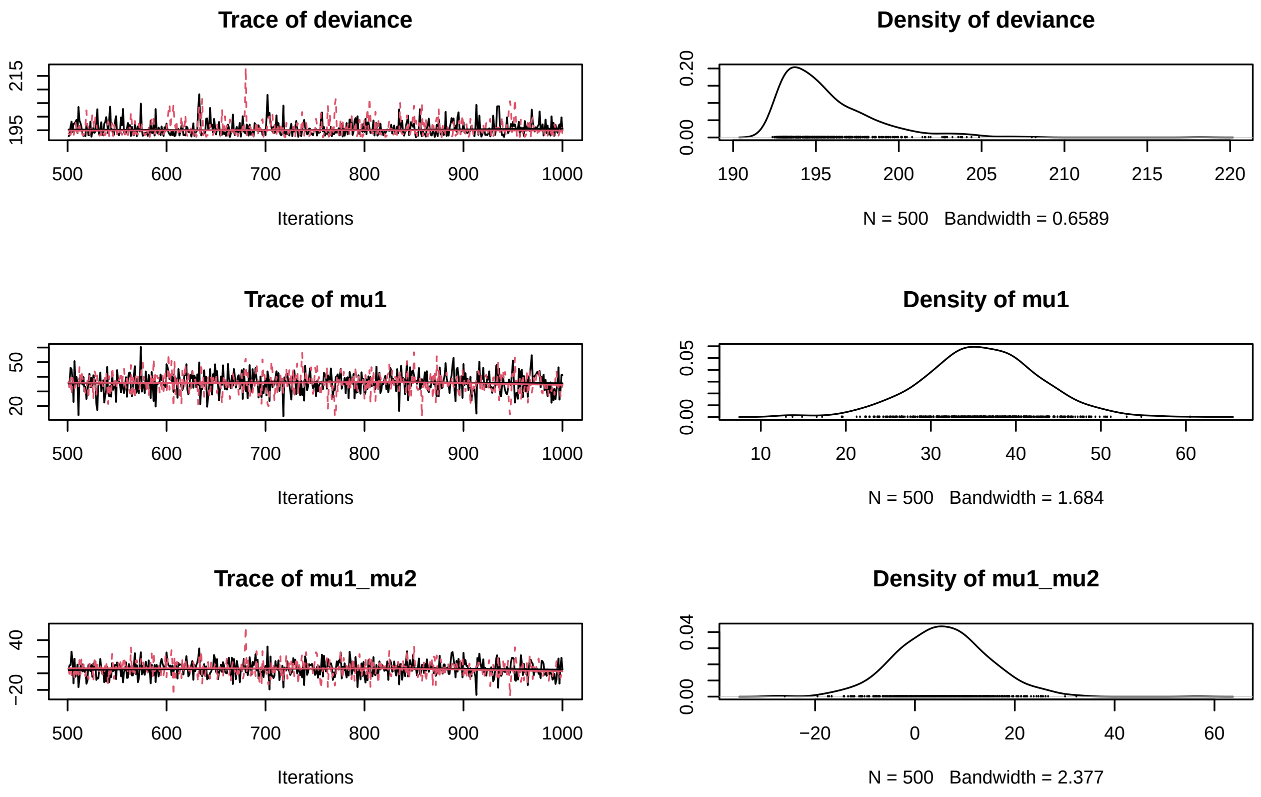
```r
model1.jags <- jags(data = data1,
                    inits = inits,
                    parameters.to.save = params1,
                    model.file = "model1.jags",
                    n.iter = 1000,
                    n.chains = 2)
```

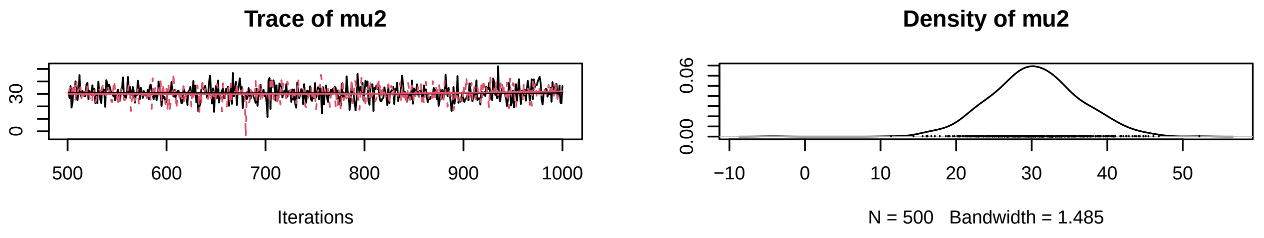
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 22
##    Unobserved stochastic nodes: 3
##    Total graph size: 33
##
## Initializing model
```

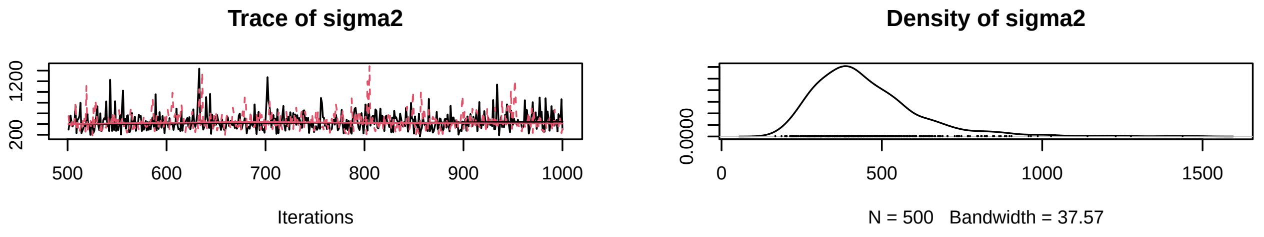
Part(d): Model (2) Posterior distributions.

In this section, *coda* package has been used to generate posterior distribution outputs by leveraging MCMC algorithm where we convert the JAGS to a language that can be understood by *coda* using *as.mcmc*() function. A *plot*() is then used to visualize the samples graphically.

```r
# Posterior
model1_mcmc <- as.mcmc(model1.jags)
plot(model1_mcmc)
```







Based on the posterior distribution plots above, it can be observed that μ_1 and μ_2 are highly likely to be greater than 20. Also, μ_1 is highly likely to be less than 50 while μ_2 is more likely to less than 40. This suggests that, on average, P/E ratio in Japan is more likely to be greater than the P/E ration of the US companies. Additionally, there is a greater likelihood for $\mu_1 - \mu_2$ to fall between -20 and 20.

The plots also clear shows that the μ_j of posterior distributions is normally distribution, compared to the density plots from Part(a) which were skewed.

The plot below displays the posterior distribution of $\mu_1 - \mu_2$ and the 95% credible interval.

```r
# 95% credible interval for mu1-mu2
plot_post(model1_mcmc[, "mu1_mu2",], main = "95% mu1-mu2 credible interval",
          xlab = expression(mu1-mu2), cenTend = "mean", ROPE = c(0.45, 0.55),
          compVal = 0.5, credMass = 0.95, quietly = TRUE)
```

The 95% credible interval, also called Bayesian confidence interval, for $\mu_1 - \mu_2$ is $[-10.9, 25.7]$. Comparing this interval to the the one in part (b), $[-12.28, 23.57]$, we can arrive at a reasonable conclusion that the true parameter $\mu_1 - \mu_2$ lies within the 95% credible interval with 95% probability. Noteworthy, the upper and lower limits for the 95% credible interval is slightly higher than the confidence interval. This is because the sample mean from posterior distribution is a weighted multiplicative combinations of the sample and prior means. This explains why the sample mean(μ) tend to be more weighted than the the prior means, not only because of the effect of number of observations, but also the sample means is considerably affected by both sample and prior variances.

The posterior median of σ^2 is 447 with a 95% credible interval $[185, 775]$.

```r
CRI_sigma <- plot_post(model1_mcmc[, "sigma2"], centre = c("median"),
                main = "sigma^2", xlab = expression(sigma^2), showCurve = TRUE,
                cenTend = "median")
```

```r
CRI_sigma$posterior
```

```
##           ESS     mean   median     mode
## var1 837.2686 450.8414 416.8979 387.3379
```

Part(e): Bayesian Statistics, Model (3).

$$Y_{1i} \sim N(\mu_1, \tau_1), i = 1, 2, ...n_1 = 10$$

$$Y_{2i} \sim N(\mu_2, \tau_2), i = 1, 2, ...n_2 = 10$$

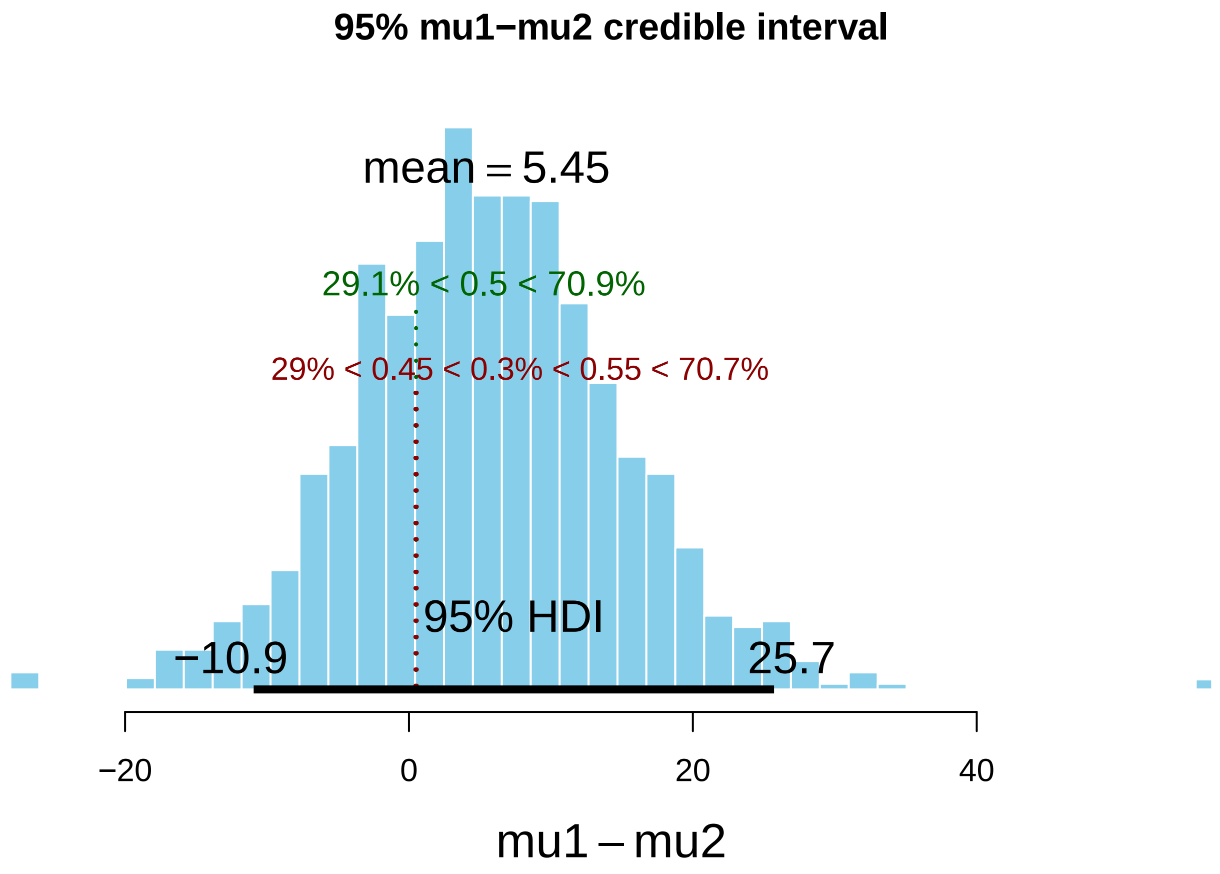


Figure 8: Posterior distribution of mu1-mu2 with 95% credible interval

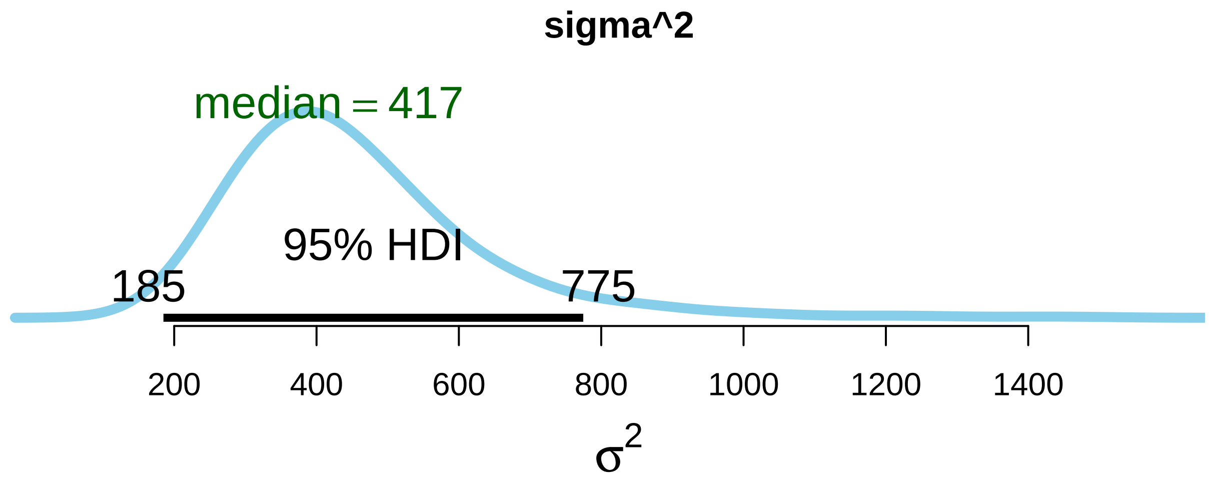


Figure 9: Posterior distribution of sigmaˆ2.

$$-Priors$$

$$\mu_1 \sim N(0, 0.0001)$$

$$\mu_2 \sim N(0, 0.0001)$$

$$\tau_1 \sim Gamma(0.001, 0.001)$$

$$\tau_2 \sim Gamma(0.001, 0.001)$$

$$\sigma_1^2 \sim \frac{1}{\tau_1}$$

$$\sigma_2^2 \sim \frac{1}{\tau_2}$$

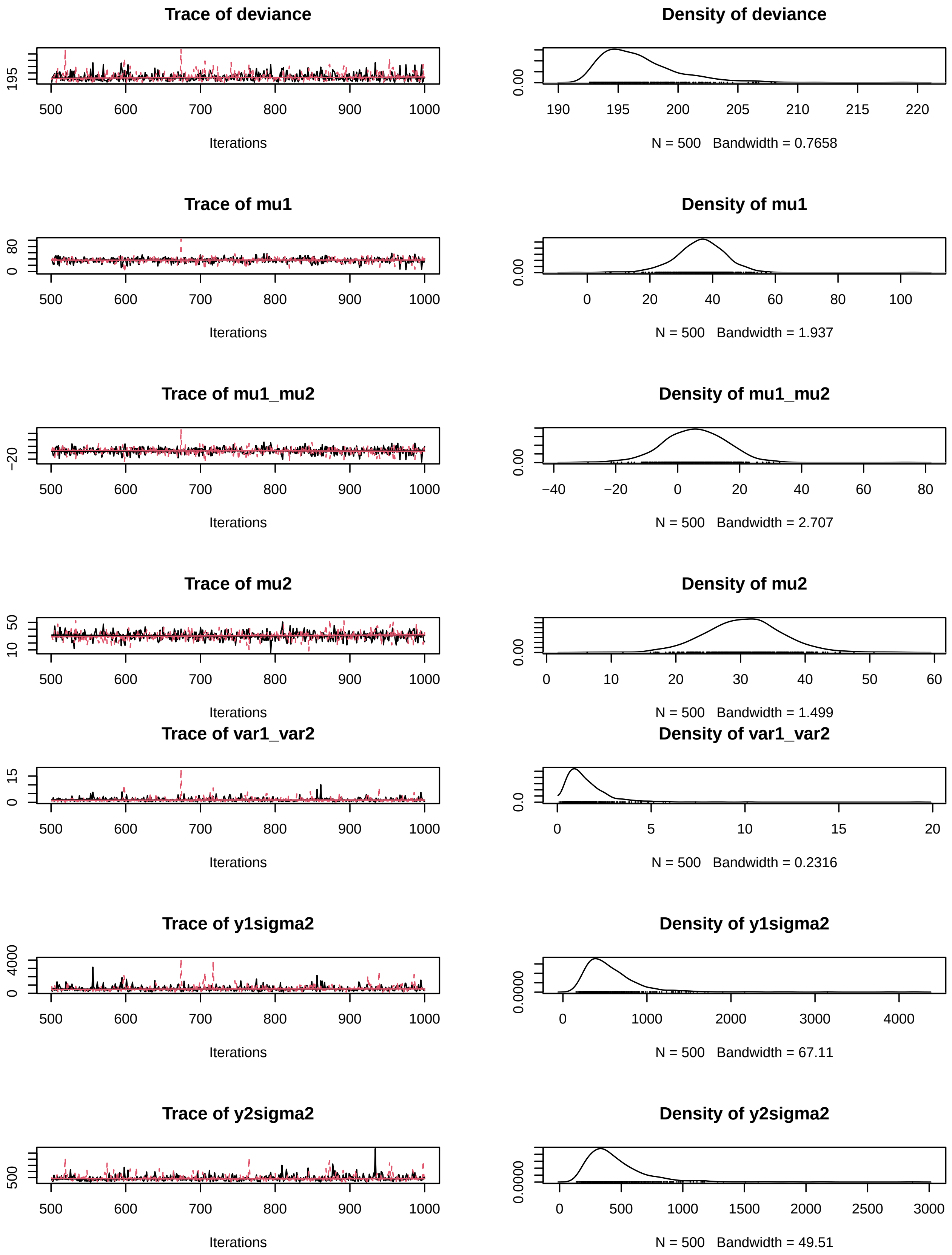
```
# Model 3: The economist's proposed model
model2 <- "model{
# samples for model (3)
for(i in 1:n1){
y1[i] ~ dnorm(mu1, tau1)
}
for(j in 1:n2){
y2[j] ~ dnorm(mu2, tau2)
}

# priors
mu1 ~ dnorm(0, 0.0001)
mu2 ~ dnorm(0,0.0001)
y1sigma2 <- 1/tau1
y2sigma2 <- 1/tau2
mu1_mu2 <- mu1 - mu2
var1_var2 <- y1sigma2 / y2sigma2
tau1 ~ dgamma(0.001, 0.001)
tau2 ~ dgamma(0.001, 0.001)
}"

params2 <- c("mu1", "mu2", "mu1_mu2", "y1sigma2", "y2sigma2", "var1_var2")

# bundle the data
data2 <- list("y1","y2","n1","n2")

inits2 <- function (){
  list("mu1" = mu1, "mu2" = mu2, "tau1" = 1, "tau2" = 1)
}

# Write
writeLines(model2, "model2.jags")
```

```r
set.seed(123)
# posterior
model2.jags <- jags(data = data2,
                   inits = inits2,
                   parameters.to.save = params2,
                   model.file = "model2.jags",
                   n.iter = 1000,
                   n.chains = 2)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 22
##    Unobserved stochastic nodes: 4
##    Total graph size: 36
##
## Initializing model
```

Part(f):

```r
# Posterior distributions model 2
model2_mcmc <- as.mcmc(model2.jags)
plot(model2_mcmc)
```



From posterior distributions plots above, the posterior parameters μ_1, μ_2, σ_1^2, σ_2^2, $\mu_1 - \mu_2$ are no longer normally distributed as they were in model (2). The μ are right-skewed while variances σ^2 are skewed left. This implies that the more parameters added added more weight to the distribution, which tends to stretch these parameters.

```r
# 95% credible interval for mu1-mu2
plot_post(model2_mcmc[, "mu1_mu2", ], main = "95% CRI mu1-mu2", xlab = expression(mu - mu),
         cenTend = "mean", ROPE = c(0.45, 0.55), compVal = 0.5,
         credMass = 0.95, quietly = TRUE)
```

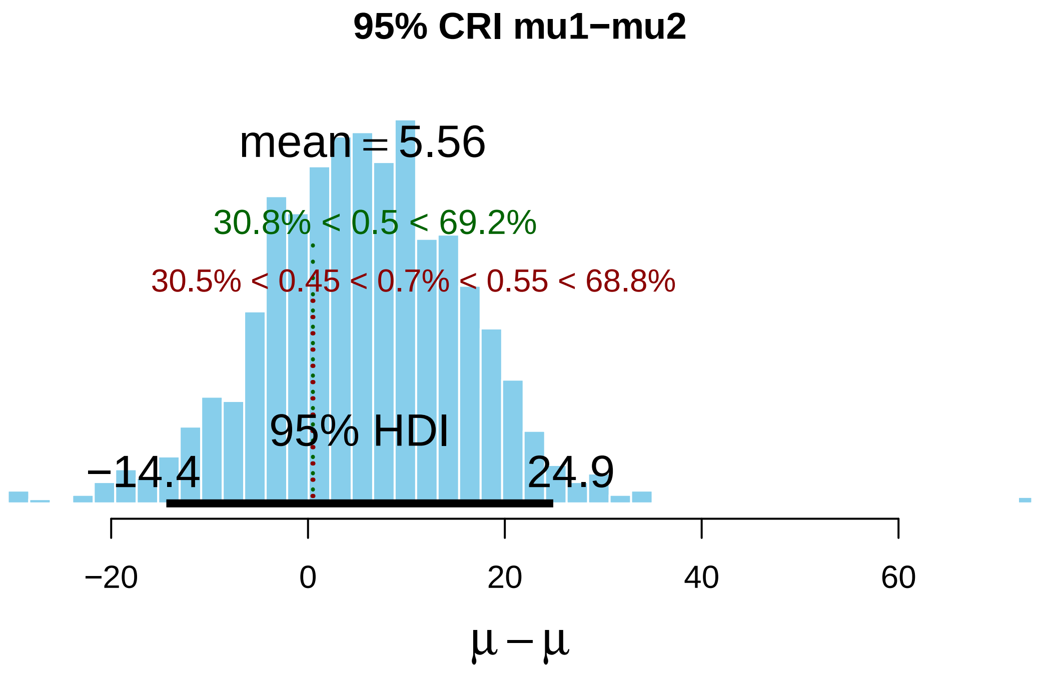


Figure 10: Model (3) 95% credible interval for mu1-mu2.

The credible interval for $\mu_1 - \mu_2$ is (-14,24.9), which is wider when compared to (-10.9, 25.7) from model(2). This shows that model (2) has a higher degree of precision than model (3).

Homogeneity of variances.

```r
Var_Ratio <- plot_post(model2_mcmc[, "var1_var2"], centre = c("median"),
                main = "Variance Ratio", xlab = expression(sigma^2/sigma^2), showCurve = TRUE,
                cenTend = "median")
```

```r
Var_Ratio$posterior
```

```
##            ESS     mean   median      mode
## var1 759.7399 1.575913 1.243781 0.8780958
```

The 95% credible interval of the ratio σ_1^2/σ_2^2 is $[0.179, 3.85]$. Since we are working with a confidence interval of two independent samples with, we assume the variance of the population from which the samples come from are equal. By the rule of thumb, if the ratio of σ_1^2 to σ_2^2 is less than 4, then we say that the two variances are approximately equal. Since most of the ratios of variances from posterior distribution are more likely to lie between 0.18 to 3.89, then the assumption of homogeneity of variance holds.

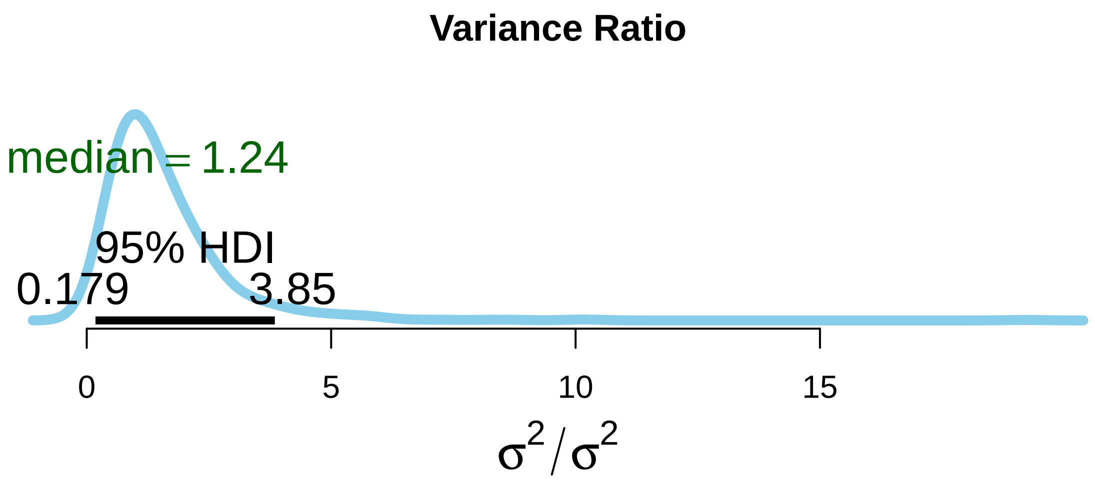


Figure 11: Posterior distribution of sigmaˆ2.

Part (g): comparing credible intervals

A 95% confidence interval obtained using *var.test* function, $0.339, 4.76]$, though slightly higher also implies that variances in the prior distribution are equal. However, the $p - value = 0.7465$ suggests that these results are not statistically significant.

```r
# 95% CI for the ratio of variances
var.test(y1, y2, conf.level=0.95)
```

```
##
##  F test to compare two variances
##
## data:  y1 and y2
## F = 1.2168, num df = 9, denom df = 11, p-value = 0.7465
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.3391455 4.7602893
## sample estimates:
## ratio of variances
##            1.21682
```

Bayesian statistics Part (h).

Model (2): From the analysis above, a hypothesis suggested that there is no statistically significant difference between the average P/E ratio between Japan and US. Given that the priors likelihoods were described as normally distributed, then adding more priors led to high leverage values which made the posterior distribution to be skewed. The distribution of σ_1^2/σ_2^2 ratio implied that the variance of the price and earning ratio are equal between the two groups.

Therefore, I would prefer model (2) to (3) because it satisfies the assumption of homogeneity of variance and the posterior distributions are normal. Having a narrower confidence interval hence providing a higher degree of accuracy.