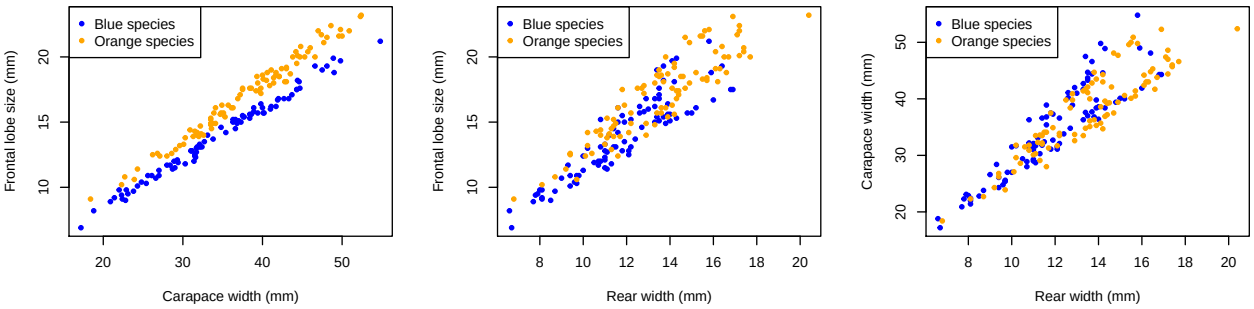
MATH501 Coursework Machine Learning and Bayesian

10630583, 10502673, 10603288, 10473010

1 Machine Learning Task

Machine Learning Part (a)

```
def.col <- rep('blue', 180)
def.col[spec == 'O'] <- 'orange'
par(mfrow=c(1,3))
#
## Carapace width vs. frontal love size
#
plot(c.width,f.size,col=def.col,xlab='Carapace width (mm)',ylab='Frontal lobe size (mm)',pch=20)
legend(x='topleft',legend = c('Blue species','Orange species'),col = c('blue','orange'),pch=20)
#
## Rear width vs. frontal lobe size
#
plot(r.width,f.size,col=def.col,xlab='Rear width (mm)',ylab='Frontal lobe size (mm)',pch=20)
legend(x='topleft',legend = c('Blue species','Orange species'),col = c('blue','orange'),pch=20)
#
## Rear width vs. carapace width
#
plot(r.width,c.width,col=def.col,xlab='Rear width (mm)',ylab='Carapace width (mm)',pch=20)
legend(x='topleft',legend=c('Blue species','Orange species'),col = c('blue', 'orange'),pch=20)
```



From the graph paired with the Carapace width (mm) and Frontal lobe size (mm) variables, it can be concluded that the orange shellfish have longest frontal lobe size than the blue shellfish within the same carapace width. Futhermore, all these three plots show a high correlation between variables, however there is a more clear classification between species when using carapace width and frontal lobe size as determinants. Therefore, the carapace width and frontal lobe size variables would be sufficient when determining the difference between the two species.

Good comments here.

Machine Learning Part (b)

```
## Call:
## lda(spec ~ f.size + c.width, data = shellfish)
##
```

```
## Prior probabilities of groups:
##         B         O
## 0.4888889 0.5111111
##
## Group means:
##     f.size c.width
## B 13.93295  34.425
## O 16.97065  37.825
##
## Coefficients of linear discriminants:
##                LD1
## f.size   2.2523093
## c.width -0.8929664
```

Thus the linear discriminant function is:

$$spec = -0.8929664 * c.width + 2.2523093 * f.size$$

From the results of the linear discriminant analysis function it can be concluded that the orange species are longer than the blue species in both carapace width and frontal lobe size. The results being an average of 38 mm of carapace width for the orange specie and 34 mm for the blue specie. Additionally, the results for the frontal lobe size show that on average the orange species is 17 mm and the blue species is 14 mm. Moreover, the LDA function also shows that for this sample there are slightly more orange species than blue, having 49% of blue species and 51% orange species.

LDA - Calculate Training Error

```
# Confusion Matrix
lda.pred <- predict(shellfish.lda)  # predictions for the training data
lda.class <- lda.pred$class     # class predictions for training data
lda.tab <- table(lda.class, spec)   # this function is called a "confusion matrix"
lda.tab
```

```
##          spec
## lda.class  B  O
##         B 88  2
##         O  0 90
```

```
(lda.tab[1,2] + lda.tab[2,1]) /
  sum(lda.tab) # overall fraction of incorrectly classified data
```

```
## [1] 0.01111111
```

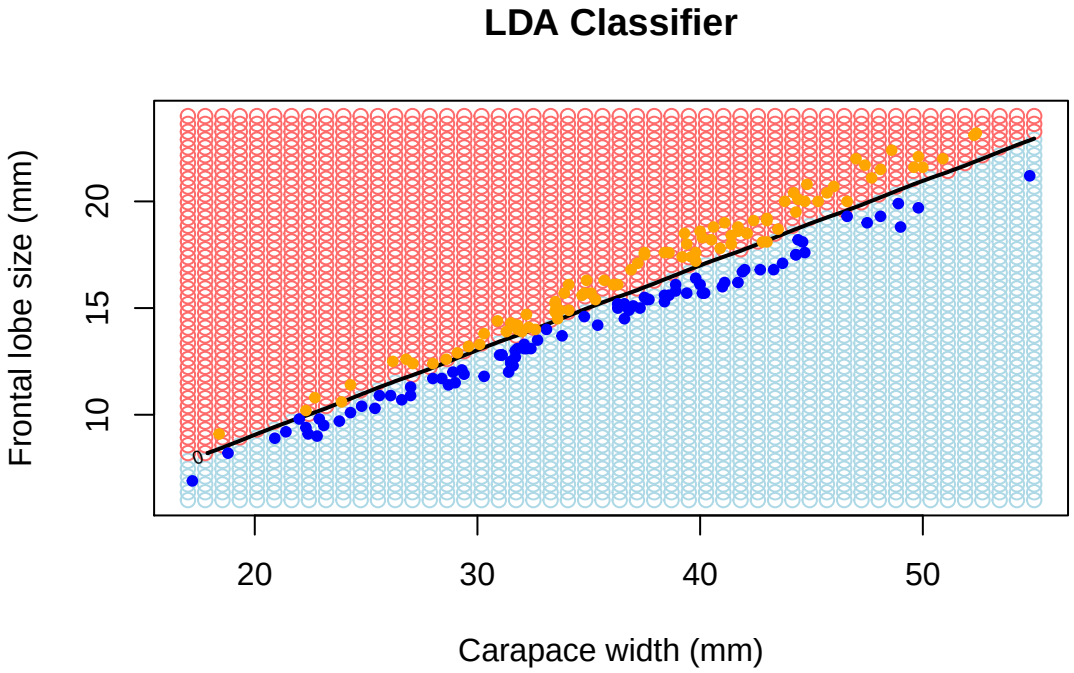
Correct

Therefore, approximately only 1.1% data is incorrectly classified by LDA classifier.

Scatter Plot of LDA

```
len <- 50
# points covering the range of carapace width
xp <- seq(17, 55, length = len)
# points covering the range of frontal lobe size
yp <- seq(6, 24, length = len)
xygrid <- expand.grid(c.width = xp, f.size = yp)
grid.lda <- predict(shellfish.lda, xygrid)
col_lda <- rep('lightblue', len*len)
for (i in 1:(len*len)) if (grid.lda$class[i]  == 'O') col_lda[i] <- 'indianred1'
zp <- grid.lda$posterior[ , 1] - grid.lda$posterior[ ,2]
plot(xygrid, col = col_lda, main = 'LDA Classifier', xlab = 'Carapace width (mm)',
```

```
                                                    ylab = 'Frontal lobe size (mm)')
contour(xp, yp, matrix(zp, len), level = 0, add = TRUE, lwd = 2)
points(c.width, f.size, pch = 20, col = def.col)
```



Good plot! Some comments would be helpful.

Machine Learning Part (c)

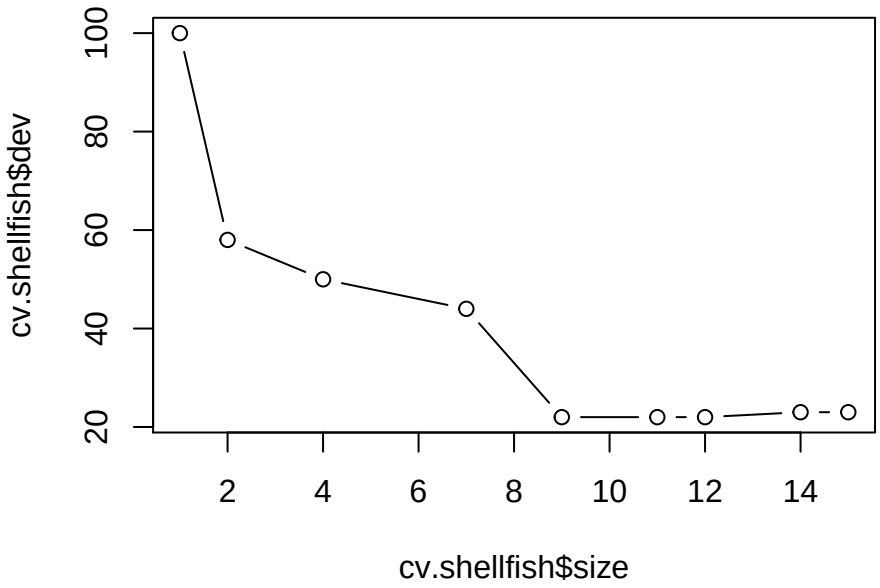
```
##
## Classification tree:
## tree(formula = spec ~ f.size + c.width, data = shellfish)
## Number of terminal nodes:  15
## Residual mean deviance:  0.2257 = 37.24 / 165
## Misclassification error rate: 0.04444 = 8 / 180
```

A small deviance indicates a tree that provides a good fit to the data. The residual mean deviance reported is simply the deviance divided by n minus the number of leaves, which in this case 180 - 15 = 165. Based on the above calculation the training error is 4.4% (Misclassification error rate).

Prune Tree

Using the **cv.tree()** function to find the optimal number of nodes.

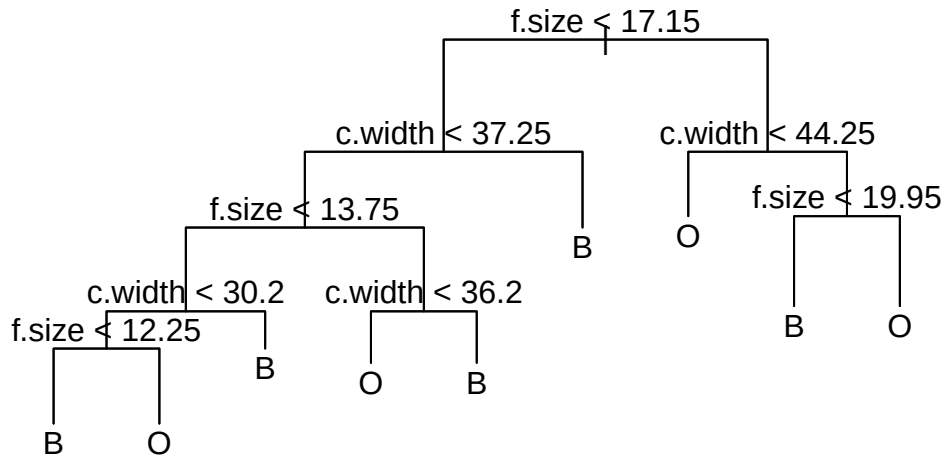
```
set.seed(3)
cv.shellfish <- cv.tree(shellfish.tree, FUN = prune.misclass)
plot(cv.shellfish$size, cv.shellfish$dev, type = 'b')
```



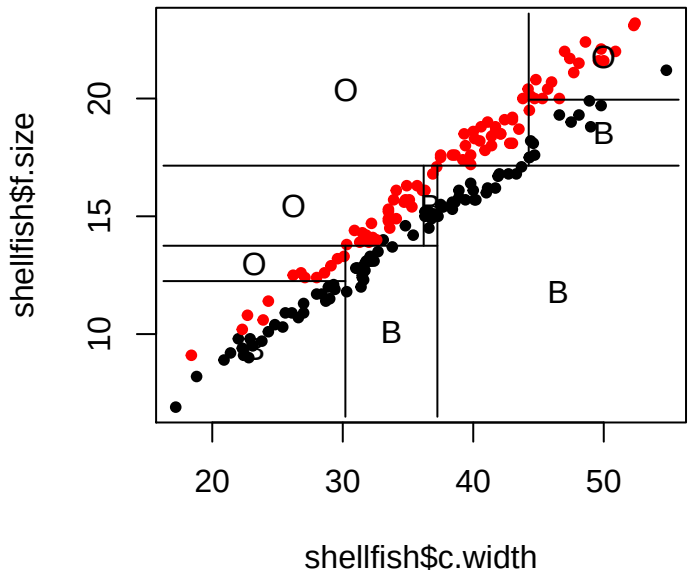
Based on the above figure, the lowest cross-validation error rate happens when the node is equal to 9. Therefore, 9 would be the optimal tree size.

Now the **prune.misclass()** function is apllied in order to obtain the tree with the optimal size of 9.

```
prune.shellfish <- prune.misclass(shellfish.tree, best = 9)
plot(prune.shellfish)
text(prune.shellfish, pretty = 0)
```




```
plot(shellfish$c.width, shellfish$f.size, pch = 20, col = shellfish$spec)
partition.tree(prune.shellfish, ordvars=c("c.width","f.size"), add=TRUE)
```



The plot is correct but it would be good if you give better axes labels

Machine Learning Part (d)

To apply principal components analysis, **princomp()function** will do the job which is based on given data numeric and returns the result as an object of class.

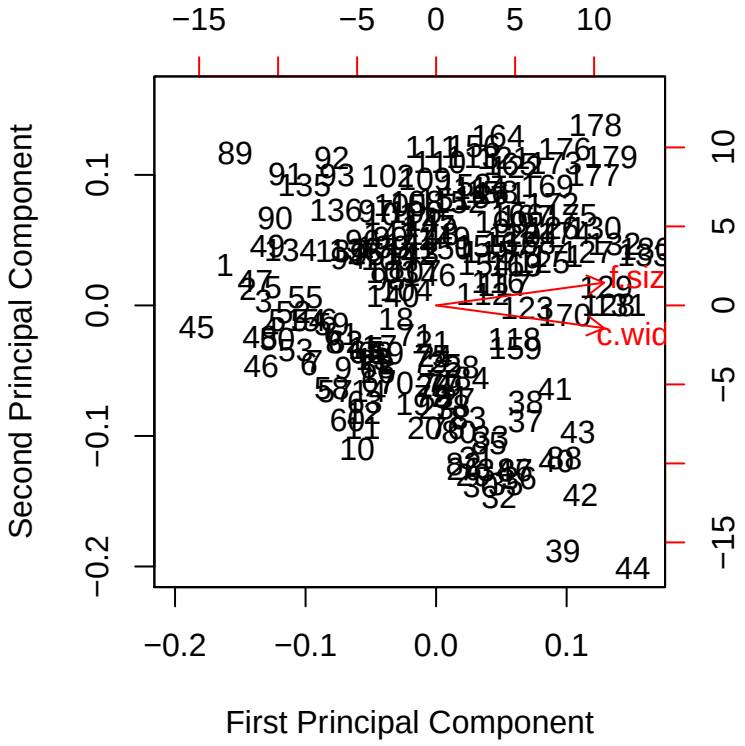
```
shellfish_pca <- princomp(shellfish[, -c(1,4)], cor = TRUE)
summary(shellfish_pca)
```

```
## Importance of components:
##                           Comp.1     Comp.2
## Standard deviation     1.401450 0.18957296
## Proportion of Variance 0.982031 0.01796895
## Cumulative Proportion  0.982031 1.00000000
```

The first principle component explains 98.2% of the variability of the data. On the other hand, component 2 explains 1.80% of the data variablity.

Plot interpretaion PCA

```
biplot(shellfish_pca,
       xlab = "First Principal Component",
       ylab = "Second Principal Component")
```



Based on the above plot, both c.width and f.size variables have positive contribution to the first principal component. In contrast, for the second principal component, there is a negative contribution from c.width variable and a positive contribution from the f.size variable.

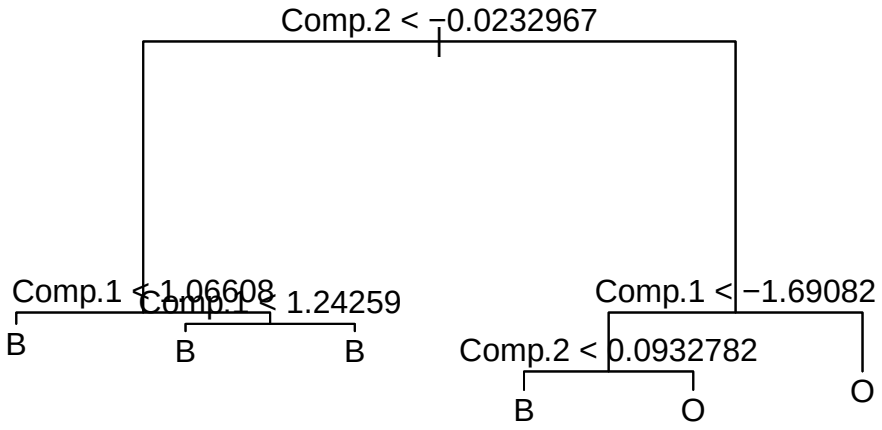
Correct but a bit more detail would be good here. For example, you could say that both variables make positive AND ROUGHLY EQUAL contributions into the first principal component.

Machine Learning Part (e)

The aim for this section is to produce Decision Tree based on the two principal component obtained in PCA section.

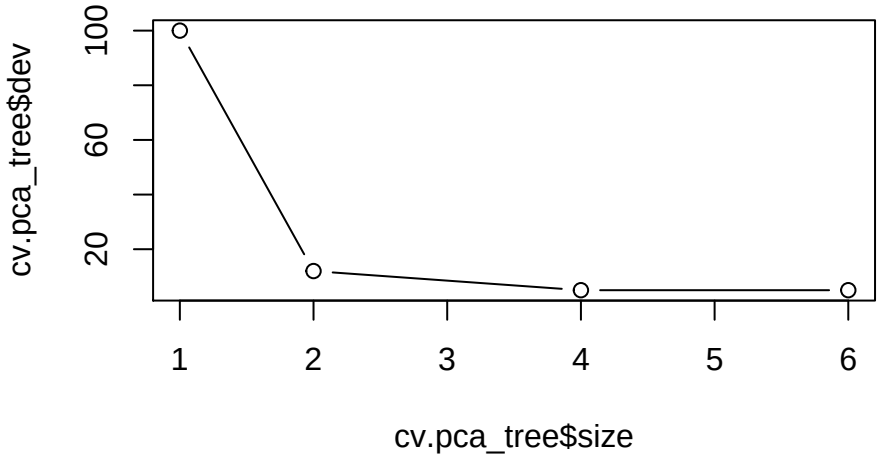
```
# create a data frame that including pca scores
new_variables_df <- data.frame(shellfish_pca$scores)
# combine spec and scores
new_variables_df$spec <- shellfish$spec
pca_tree <- tree(spec ~ . , data = new_variables_df)
summary(pca_tree)
```

```
##
## Classification tree:
## tree(formula = spec ~ ., data = new_variables_df)
## Number of terminal nodes:  6
## Residual mean deviance:  0.06975 = 12.14 / 174
## Misclassification error rate: 0.01667 = 3 / 180
```



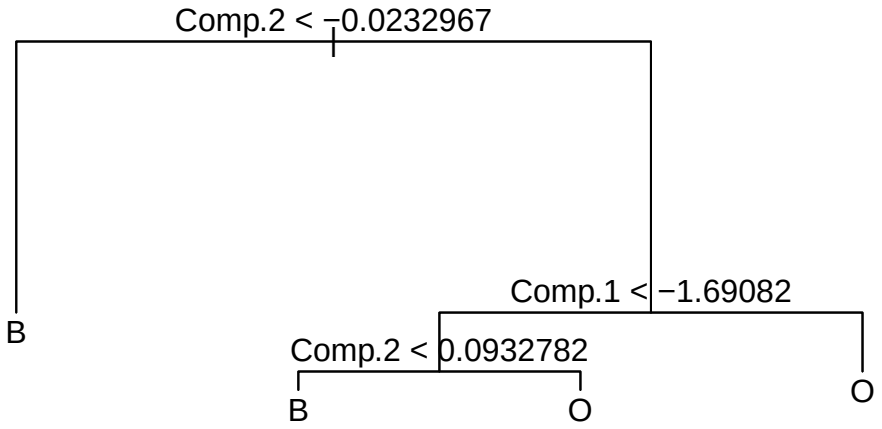
Using **cv.tree()** function to determine the optimal size of the tree.

```
set.seed(3)
cv.pca_tree <- cv.tree(pca_tree, FUN = prune.misclass)
plot(cv.pca_tree$size, cv.pca_tree$dev, type = 'b')
```

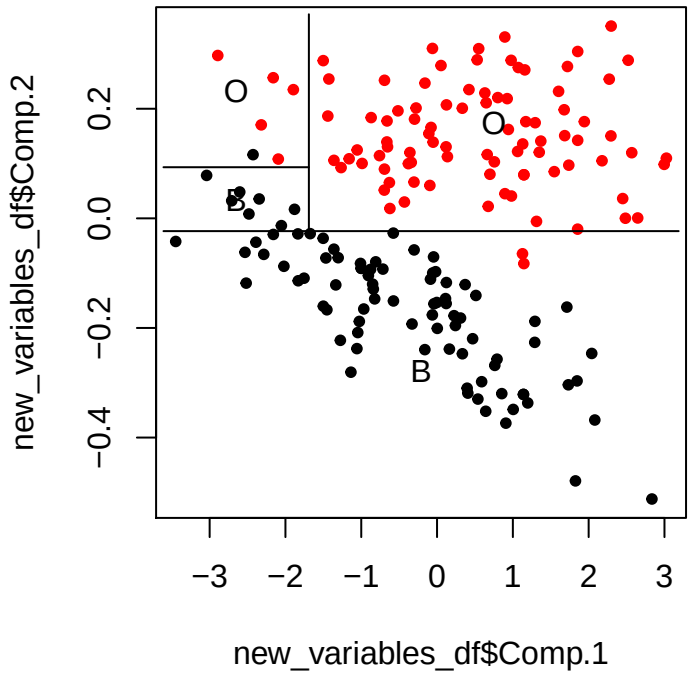


The graph shows that 4 is the optimal number of tree nodes.The next step is adding the number founded on the graph to produce a new tree for the pruning.

```
prune.pca_tree<- prune.misclass(pca_tree, best = 4)
plot(prune.pca_tree)
text(prune.pca_tree, pretty = 0)
```



```
plot(new_variables_df$Comp.1, new_variables_df$Comp.2, pch = 20, col = new_variables_df$spec)
partition.tree(prune.pca_tree, ordvars=c("Comp.1","Comp.2"), add=TRUE)
```



Based on the summary of the obtained tree, the misclassification error rate is 1.6%. Therefore, compare to the original tree (4.4%) PCA tree has less error which is showing a better performance. The PCA changes the decision tree by using the principle components instead of the variables, it rotates and scales the data which improves the accuracy of the result. Principal Component Analysis (PCA) is one of the most common ways of extracting features and reducing the dimensionality of your data set. Using transformed predictors can lead to an improved decision tree due to the fact that the noisy data and irrelevant features were eliminated from the data during the process. Moreover, by the reduction of the irrelevant data, the accuracy of the decision tree is improved (Nasution, 2018).

Machine Learning Part (f)

Leave-one-out cross-validation for part (b)

```
n <- nrow(shellfish)
cv.predictions <- rep('O', n)
for(i in 1:n){
  shellfish.lda <- lda(spec ~ f.size + c.width, data=shellfish[-i,])
  cv.predictions[i] <- predict(shellfish.lda, newdata = shellfish[i, ])$class
}
tab <- table(cv.predictions, spec)
tab
```

```
##                spec
## cv.predictions  B  O
##              1 88  4
##              2  0 88
```

```
cv.error = (tab[1,2] + tab[2,1]) / sum(tab)
cv.error
```

```
## [1] 0.02222222
```

Leave-one-out cross-validation for part (c)

```
n <- nrow(shellfish)
cv.predictions <- rep('O', n)
for(i in 1:n) {
  tree.fit <- tree(spec ~ f.size + c.width, data = shellfish[-i, ])
  cv.predictions[i] <- predict(tree.fit, newdata = shellfish[i,], type = "class")
}
tab_DT <- table(cv.predictions, spec)
tab_DT
```

```
##                spec
## cv.predictions  B  O
##              1 78 12
##              2 10 80
```

```
cv.error = (tab_DT[1,2] + tab_DT[2,1]) / sum(tab_DT)
cv.error
```

```
## [1] 0.1222222
```

Leave-one-out cross-validation for part (e)

```
new_variables_df <- data.frame(shellfish_pca$scores)
new_variables_df$spec <- shellfish$spec
n <- nrow(new_variables_df)
cv.predictions <- rep('O', n)
for(i in 1:n) {
  pca_tree <- tree(spec ~ . , data = new_variables_df[-i,])
  cv.predictions[i] <- predict(pca_tree, newdata = new_variables_df[i,], type = "class")
}
tab_PCA <- table(cv.predictions, spec)
tab_PCA
```

```
##                spec
## cv.predictions  B  O
##              1 85  3
##              2  3 89
```

```
cv.error = (tab_PCA[1,2] + tab_PCA[2,1]) / sum(tab_PCA)
cv.error
```

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

The above result is showing that the rule in *part b* has a less classification error compare to rules in *part e* and *part c*. There is a lower classification error for LDA than the original decision tree because from the part (a) we observed that there is a linear boundary and LDA is a better method for linear classification while the decision tree is a non-parametric method.

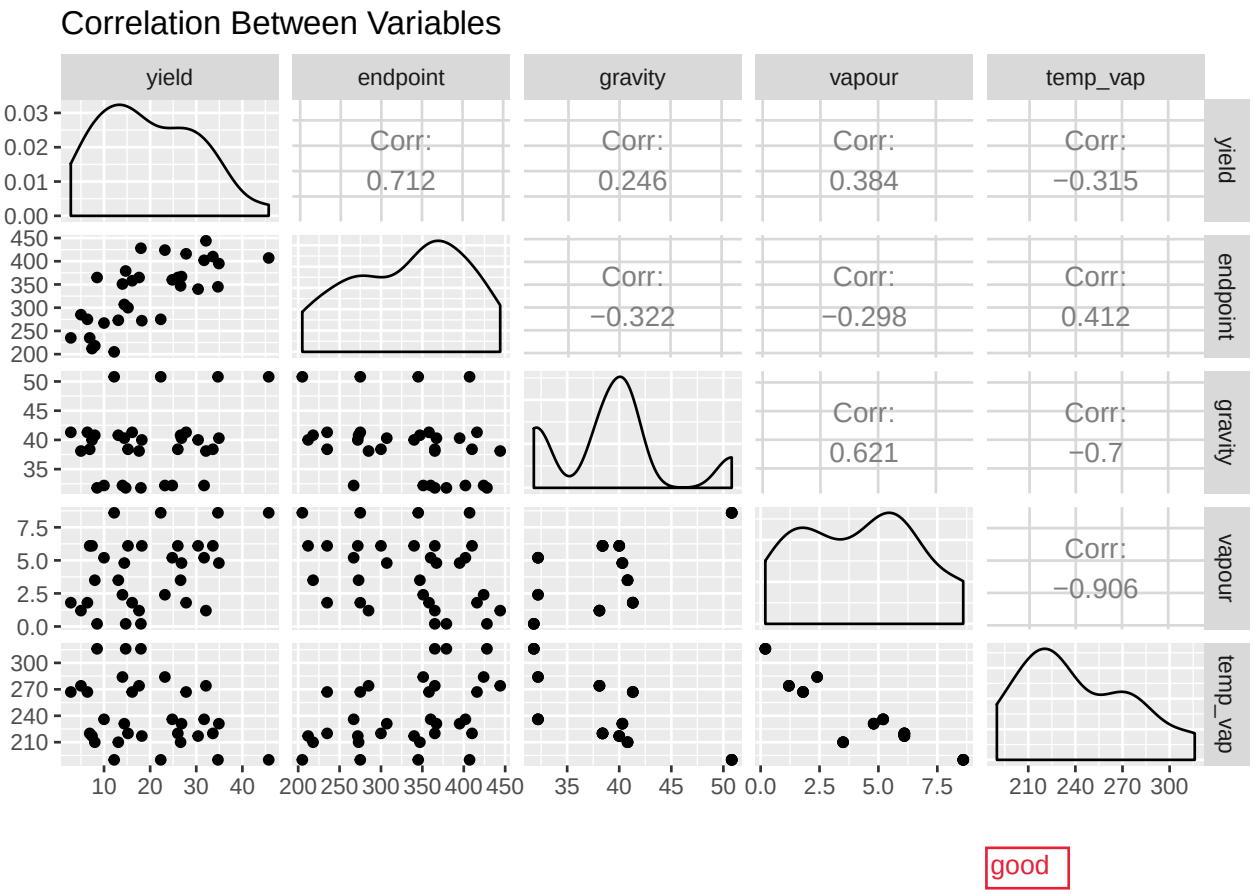
2 Bayesian Statistics Task

First Sub-Task

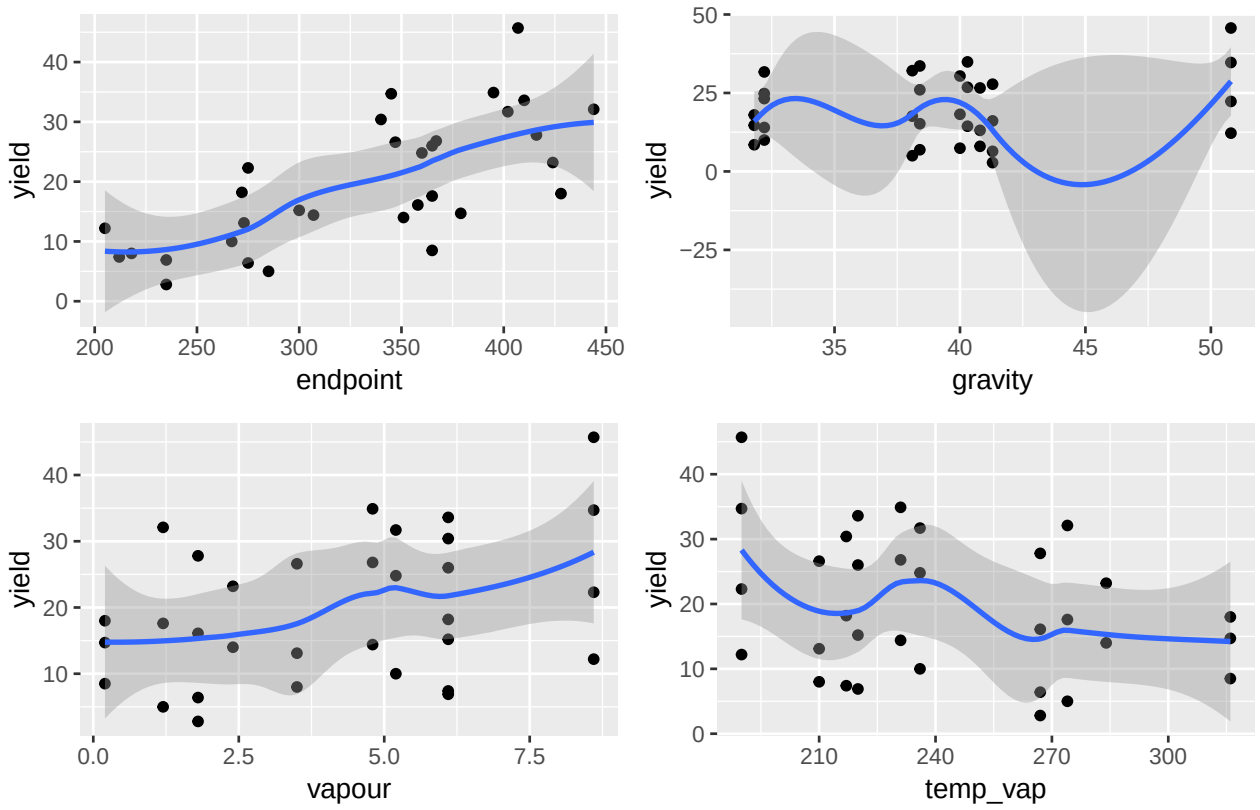
Bayesian Statistics Part (a)

Create a scatter plot to display the petrol data.

```
ggpairs(petrol, title = "Correlation Between Variables")
```



good



From the above graphs, we observed that the endpoint has the highest correlation (0.7) with yield while the gravity has the lowest correlation (0.2). Moreover, temp_vap shows a negative correlation of -0.3 with the petrol yield.

Useful comments on the correlation coefficients. However, what about the plots?

Bayesian Statistics Part (b)

The statistical model for the data is:

$$y_i = \beta_0 + \beta_1 endpoint_i + \beta_2 gravity_i + \beta_3 vapour_i + \beta_4 temp\_vap_i + \epsilon_i \quad (1)$$

yield

typo

you only need to consider beta1, not beta0 + beta1

β_1 represents the expected change in y when *endpoint* increases one unit and *gravity*, *vapour* and $temp_vap$ remain fixed. More technically, β_1 represents the expected chainge in yield of crude oil $(\beta_0 + \beta_1)$ if the endpoint variable increases one degree Fahrenhiet and the variables gravity, vapour, and temp_vap remain fixed.

Bayesian Statistics Part (c)

Fit the above model in the frequentist framework and report $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$.

```
m <- lm(yield ~ ., data = petrol)
# intepret betas
kable(summary(m)$coef, digits = 3)
```

Very nice table! However, you should have pointed out what are, from the table, the values of the beta hats.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-6.821	10.123	-0.674	0.506
endpoint	0.155	0.006	23.992	0.000
gravity	0.227	0.100	2.274	0.031
vapour	0.554	0.370	1.498	0.146
temp_vap	-0.150	0.029	-5.116	0.000

The model with the estimated values of parameters is:

$$y_i = -6.82 + 0.16 endpoint_i + 0.23 gravity_i + 0.55 vapour_i - 0.15 temp\_vap_i \quad (2)$$

Why mu_v? Here the test is based on beta3!

What are the "underlying mean scores? You are getting confused with the ANOVA model

The *null hypotheses* is $H_0 : \mu_v = 0$, in which μ_v is the underlying mean scores from vapour pressure. The *alternative* hypothesis is $H_1 : \mu_v \neq 0$.

The p-value for vapour is 0.15 which is greater than 0.05, and t value is 1.50 which is less than 2 so we fail to reject the null hypothesis.

```
confint(m)
```

```
##                    2.5 %      97.5 %
## (Intercept) -27.59176589 13.95021775
## endpoint      0.14142431  0.16787587
## gravity       0.02219291  0.43229900
## vapour       -0.20494210  1.31239452
## temp_vap     -0.20950898 -0.08956226
```

The 95% confidence interval for vapour is [-0.2, 1.3] includes zero which makes it not statistically significant. Therefore, we can conclude that vapour does not have an effect on the petrol yield.

Bayesian Statistics Part (d)

Use jags/BUGS code to perform inference about the following related statistical model in the Bayesian framework.

```
bayesian_regression_model <- function(){
  for(i in 1:n){
    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] + beta_3 * x3[i] + beta_4 * x4[i]
    }
  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
  beta_3 ~ dnorm(0.0, 1.0E-4) # Prior on beta_3 is normal with low precision
  beta_4 ~ dnorm(0.0, 1.0E-4) # Prior on beta_4 is normal with low precision
  tau ~ dgamma(1.0E-3, 1.0E-3) # Prior on tau is gamma with small shape and rate parameters
```

```
  sigma <- 1.0 / sqrt(tau)
}
# prepare data for jags
x1 <- endpoint
x2 <- gravity
x3 <- vapour
x4 <- temp_vap
y <- yield
n <- length(y)
data_regression <- list('x1', 'x2', 'x3', 'x4', 'y', 'n')
# perform bayesian inference
bayesian_regression <- jags(data = data_regression,
                            parameters.to.save = c("beta_0","beta_1",
                                                   "beta_2","beta_3",
                                                   "beta_4"),
                            n.chains = 3,
                            n.iter = 100000,
                            model.file = bayesian_regression_model)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 32
##    Unobserved stochastic nodes: 6
##    Total graph size: 261
##
## Initializing model
```

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

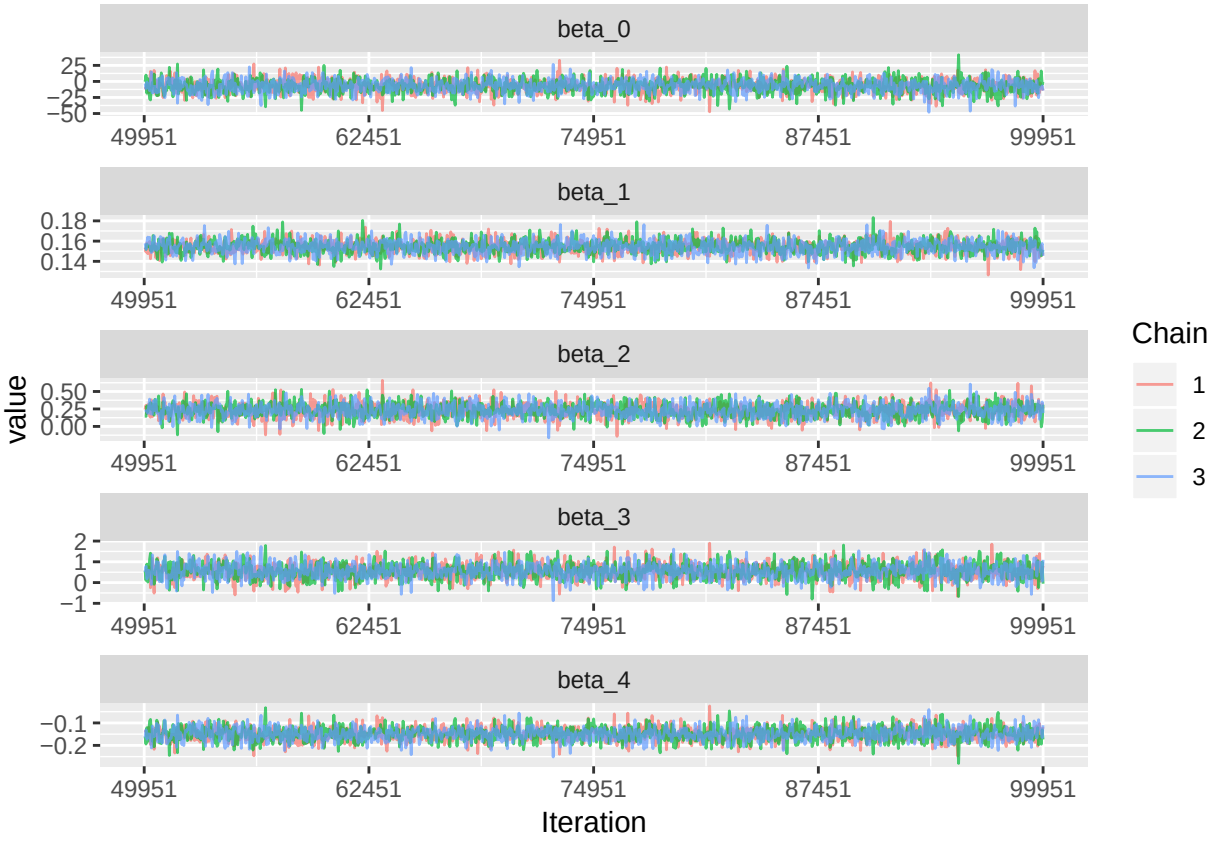
```
## Inference for Bugs model at "/var/folders/yj/r_thtqks72vb9fglwpzyn8qc0000gn/T//RtmpPJDQGw/model17c5a
##  3 chains, each with 1e+05 iterations (first 50000 discarded), n.thin = 50
##  n.sims = 3000 iterations saved
##          mu.vect sd.vect    2.5%     50%   97.5%  Rhat n.eff
## beta_0    -6.851  10.586 -27.898  -6.569  13.382 1.001  3000
## beta_1     0.155   0.007   0.141   0.155   0.168 1.001  2700
## beta_2     0.227   0.104   0.028   0.225   0.436 1.001  3000
## beta_3     0.553   0.382  -0.215   0.550   1.326 1.001  3000
## beta_4    -0.149   0.031  -0.208  -0.150  -0.087 1.001  3000
## deviance 143.432   3.718 138.286 142.771 152.076 1.001  3000
##
## 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 = 6.9 and DIC = 150.3
## DIC is an estimate of expected predictive error (lower deviance is better).
```

good

Bayesian Statistics Part (e)

Graphical presentations of the posterior distributions of $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$.

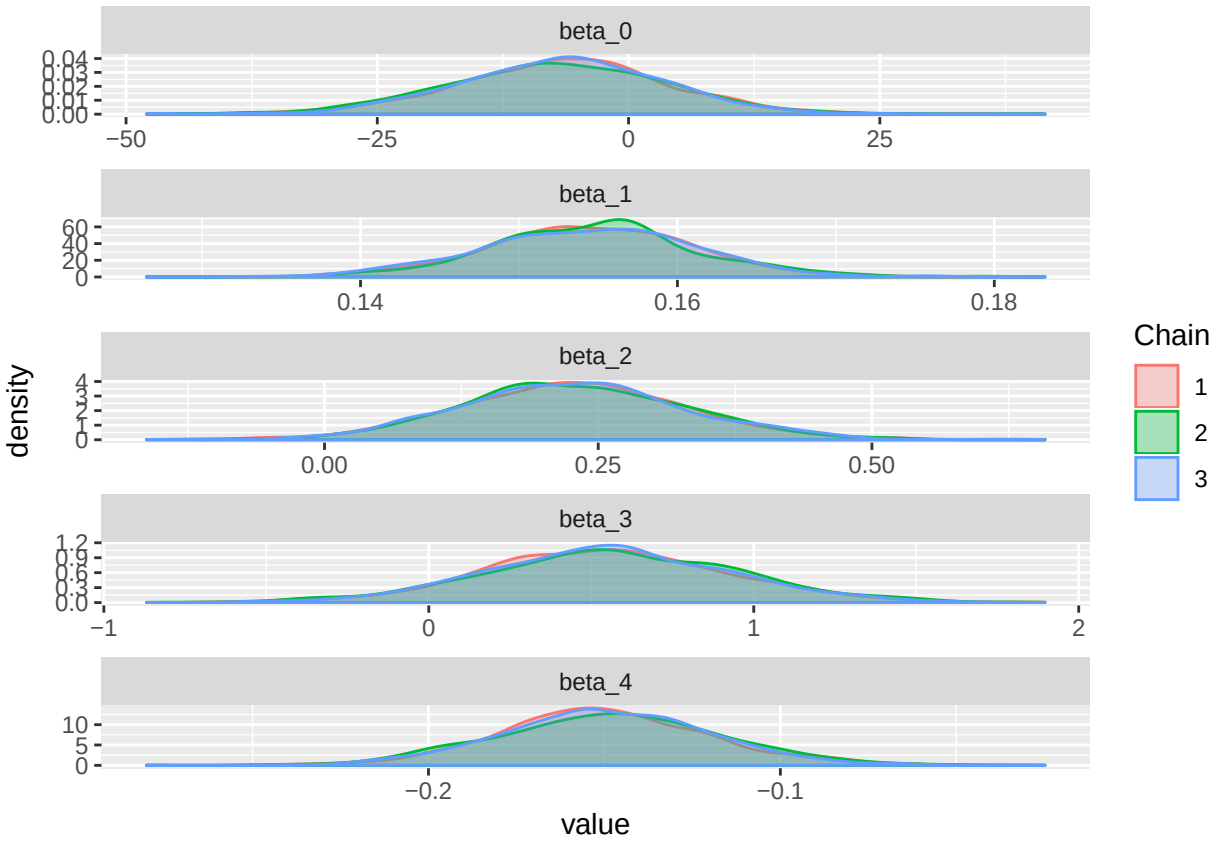
```r
bayesian_regression.mcmc <- as.mcmc(bayesian_regression)
# Create a ggs object
bayesian_regression.ggs <- ggs(bayesian_regression.mcmc)
# Display the traceplot
ggs_traceplot(bayesian_regression.ggs, family = "^beta")
```



The variability between the chains is low which is good because it shows that the sampled values have settled.

```r
# Display posterior probability density functions
ggs_density(bayesian_regression.ggs, family = "^beta")
```

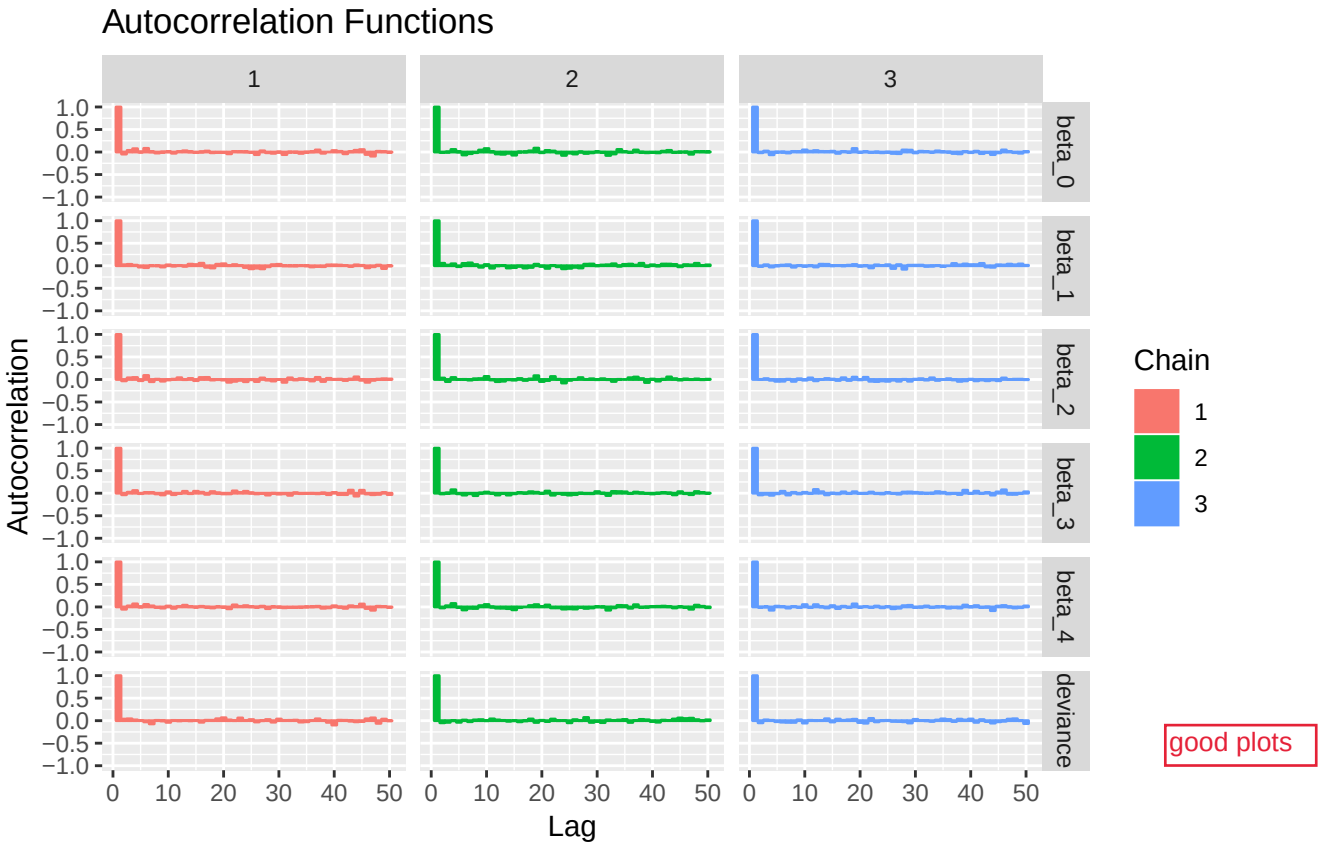
very good



The Gibbs Sampler algorithm was runned using n.chains = 3 (three MCMC chains) from different starting points. The results from the algorithm display similarity between them, which shows that the results do not depend on the starting point of the algorithm.

good

```
# Display the autocorrelation functions
ggs_autocorrelation(bayesian_regression.ggs) +
  ggtitle("Autocorrelation Functions")
```



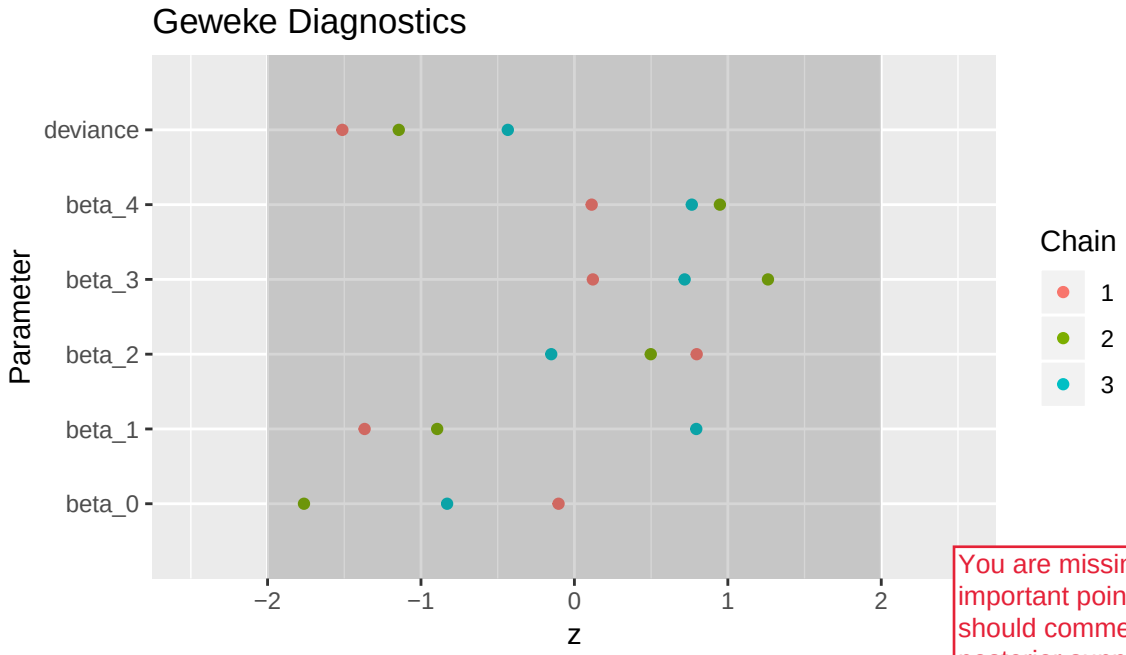
Because the plots fall into zero straightaway after the first spike (the value one), it can be concluded that the samples from the posteriors are independent.

```
geweke.diag(bayesian_regression.mcmc)
```

```
## [[1]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##    beta_0    beta_1    beta_2    beta_3    beta_4  deviance
##  0.005161 -1.468314  0.700268  0.284085  0.018944 -1.310055
##
##
## [[2]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##   beta_0   beta_1   beta_2   beta_3   beta_4 deviance
##  -1.8332  -1.2449   1.0312   0.9429   1.0301  -1.6325
##
##
## [[3]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
```

```
##    beta_0    beta_1    beta_2    beta_3    beta_4 deviance
##   -0.9297    1.1372   -0.1226    0.8507    0.7393   -0.9069
```

```
ggs_geweke(bayesian_regression.ggs)
```



You have included a wide range of useful plots of your results

Our result is desirable since at least 95% of the values should be between -2 and 2.

You are missing an important point: you should comment on the posterior support to zero of each parameter.

Bayesian Statistics Part (f)

The 95% credible interval for β_3 is:

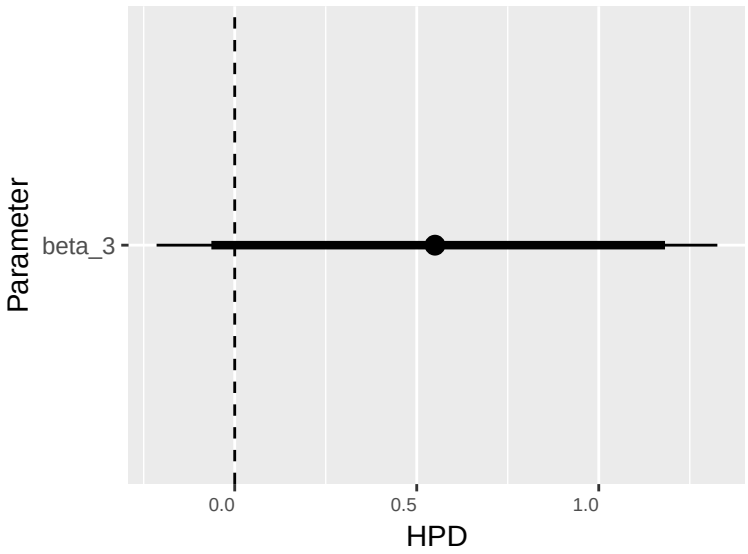
```
bayesian_regression$BUGSoutput$summary["beta_3", c("2.5%", "97.5%")]
```

```
##      2.5%     97.5%
## -0.214594  1.325783
```

A graphical presentation and the numerical values of a 95% credible interval for β_3.

```
ggs_caterpillar(bayesian_regression.ggs, family = "^beta_3") + geom_vline(xintercept=0, lty=2)
```

Nice idea to include a vertical line at zero.



For the results from part (b), the 95% confidence interval of β_3 is [-0.20, 1.31] and the 95% credible interval for β_3 has similarity to the 95% confidence internval of β_3. They both include zero in the interval which means β_3 is not significant.

Are the intervals similar?

The concept of "significance" is used in the frequentist framework. In the Bayesian framework you should talk about posterior support to zero.

Bayesian Statistics Part (g)

Perform the reduced Bayesian model.

```
bayesian_regression_model <- function(){
  for(i in 1:n){
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- beta_0 + beta_1 * x1[i] + beta_4 * x4[i]
  }
  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_4 ~ dnorm(0.0, 1.0E-4) # Prior on beta_4 is normal with low precision
  tau ~ dgamma(1.0E-3, 1.0E-3)
  sigma <- 1.0 / sqrt(tau)
}
library(R2jags)
x1 <- endpoint
x4 <- temp_vap
y <- yield
n <- length(x1)
data_regression <- list('x1', 'x4', 'y', 'n')
# perform bayesian inference
bayesian_regression_reduced <- jags(data = data_regression,
                            parameters.to.save = c("beta_0","beta_1","beta_4"),
                            n.chains = 3,
                            n.iter = 100000,
                            model.file = bayesian_regression_model)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
```

```
##    Observed stochastic nodes: 32
##    Unobserved stochastic nodes: 4
##    Total graph size: 177
##
## Initializing model
```

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

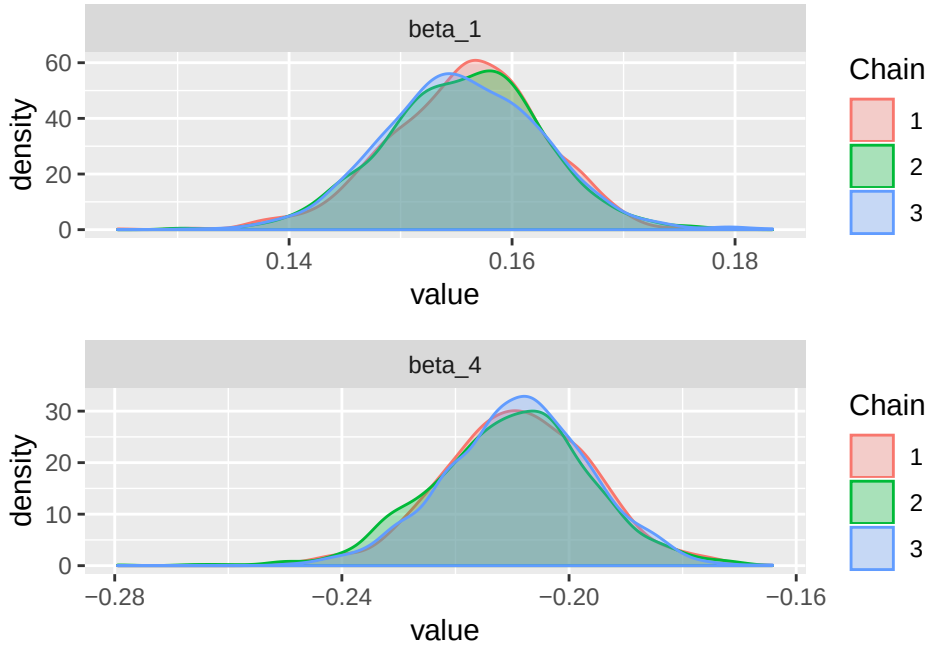
```
## Inference for Bugs model at "/var/folders/yj/r_thtqks72vb9fglwpzyn8qc0000gn/T//RtmpPJDQGw/model17c5a
##  3 chains, each with 1e+05 iterations (first 50000 discarded), n.thin = 50
##  n.sims = 3000 iterations saved
##          mu.vect sd.vect    2.5%     50%   97.5%  Rhat n.eff
## beta_0    18.516   3.122  12.429  18.483  24.767 1.004   660
## beta_1     0.156   0.007   0.142   0.156   0.169 1.001  3000
## beta_4    -0.209   0.013  -0.236  -0.209  -0.184 1.004   820
## deviance 148.634   3.060 144.880 147.925 156.537 1.001  3000
##
## 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 = 4.7 and DIC = 153.3
## DIC is an estimate of expected predictive error (lower deviance is better).
```

good

Bayesian Statistics Part (h)

The graphical presentations of 95% credible intervals for β_1 and β_4 are:

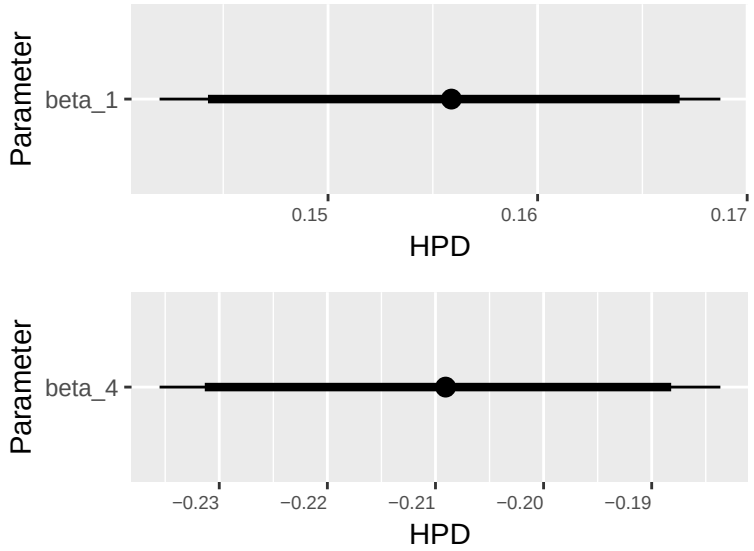
```
bayesian_regression_reduced.mcmc <- as.mcmc(bayesian_regression_reduced)
bayesian_regression_reduced.ggs <- ggs(bayesian_regression_reduced.mcmc)
grid.arrange(ggs_density(bayesian_regression_reduced.ggs, family = "beta_1"),
ggs_density(bayesian_regression_reduced.ggs, family = "beta_4"))
```



The Gibbs Sampler algorithm was runned using n.chains = 3 (three chains) from different starting points.

The results from the algorithm display similarity between them, which shows that the results do not depend on the starting point of the algorithm. Moreover, for β_1, most of the points are concentrated between 0.14 and 0.17. On the other hand, for β_4 most of the points are concentrated within -0.24 an -0.18.

```
caterpillar_beta1 <- ggs_caterpillar(bayesian_regression_reduced.ggs, family = "^beta_1")
caterpillar_beta4 <- ggs_caterpillar(bayesian_regression_reduced.ggs, family = "^beta_4")
grid.arrange(caterpillar_beta1, caterpillar_beta4, nrow = 2)
```



The numerical values of 95% credible intervals for β_1 is:

```
bayesian_regression_reduced$BUGSoutput$summary["beta_1", c("2.5%", "97.5%")]
```

```
##      2.5%     97.5%
## 0.1419627 0.1687203
```

The numerical values of 95% credible intervals for β_4 is:

```
bayesian_regression_reduced$BUGSoutput$summary["beta_4", c("2.5%", "97.5%")]
```

```
##       2.5%      97.5%
## -0.2355204 -0.1836496
```

see my previous comment

Both 95% credible intervals do not include zero which means endpoint and temp_vap are significant. In addition, the interval for β_1 is positive while the interval for β_4 is negative. Thus, endpoint has a positive effect on the petrol yield and temp_vap has a negative impact.

Bayesian Statistics Part (i)

By following this rule **petrol yield value when endpoint is 460 and temp_vap is 180**. We produce BUGS/jags model as follow:

```
bayesian_regression_model_predict <- function(){
  for(i in 1:n){
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- beta_0 + beta_1 * x1[i] + beta_4 * x4[i]
  }
  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_4 ~ dnorm(0.0, 1.0E-4) # Prior on beta_4 is normal with low precision
```

```
  tau ~ dgamma(1.0E-3, 1.0E-3)
  sigma <- 1.0 / sqrt(tau)
  mu_new <- beta_0 + beta_1 * x1_new + beta_4 * x4_new
  y_new ~ dnorm(mu_new, tau)
}
x1_new <- 460
x4_new <- 180
data_regression_predict <- list("x1", "x4", "y", "n", "x1_new", "x4_new")
bayesian_regression_predict <- jags(data = data_regression_predict,
                                    parameters.to.save = c("beta_0", "beta_1", "beta_4","mu_new", "y_new
                                    n.iter = 100000,
                                    n.chains = 3,
                                    model.file = bayesian_regression_model_predict)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 32
##    Unobserved stochastic nodes: 5
##    Total graph size: 183
##
## Initializing model
```

You could have included a graphical representation of these results

```
bayesian_regression_predict
```

```
## Inference for Bugs model at "/var/folders/yj/r_thtqks72vb9fglwpzyn8qc0000gn/T//RtmpPJDQGw/model17c5a
##  3 chains, each with 1e+05 iterations (first 50000 discarded), n.thin = 50
##  n.sims = 3000 iterations saved
##          mu.vect sd.vect    2.5%     25%     50%     75%   97.5%  Rhat
## beta_0    18.415   3.037  12.414  16.486  18.388  20.411  24.449 1.002
## beta_1     0.156   0.007   0.142   0.151   0.156   0.160   0.169 1.001
## beta_4    -0.209   0.013  -0.235  -0.218  -0.209  -0.200  -0.182 1.001
## mu_new    52.419   1.536  49.349  51.419  52.422  53.411  55.561 1.001
## y_new     52.422   2.861  46.720  50.604  52.416  54.280  58.037 1.001
## deviance 148.584   3.049 144.859 146.361 147.863 150.000 156.309 1.002
##          n.eff
## beta_0    2600
## beta_1    3000
## beta_4    3000
## mu_new    3000
## y_new     3000
## deviance  1900
##
## 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 = 4.6 and DIC = 153.2
## DIC is an estimate of expected predictive error (lower deviance is better).
```

At the temperature of vapourization of 460 degrees Fahrenheit and at the crude oil 10% point of 180 degrees Fahrenheit, the percentage of the petrol yield is expected to be 52.4. In other words, if the temperature for the both varaibles is as stated above it can be estimated that around 52% of the crude oil will be converted to petrol after fractional distillation. The posterior mean is 52.5 and the 95% credible interval is [49.5, 55.5]

very good interpretation

Is this an example of interpolation or extrapolation? Your comments could be expanded.

and the prediction interval for the petrol yield is [46.4, 58.3], which estimates that an observation with a similar temperature values for endpoint and temp_vap yield will fall within this range. We also obseved in part (h) that the temperature of vaporization has a positive effect on yield and the crude oil 10% point has a negative effect. Thus, a higher endpoint and a lower temp_vap will generate higher petrol yield.

Bayesian Statistics Part (j)

To decide which model is better, we need to compare the Deviance Information Criterion.

The DIC for full model is:

```
bayesian_regression$BUGSoutput$DIC
```

```
## [1] 150.3465
```

The DIC for the reduced model is:

```
bayesian_regression_predict$BUGSoutput$DIC
```

```
## [1] 153.231
```

One way of choosing a model is thourgh its DIC value. The lower the DIC value the better due to the fact that DIC represent how bad is the model fit. Therefore, the model from part d (the full model) is preferred. Its DIC value is 151 and the reduced model from part g DIC value is 153.

good

Second Sub-Task

Bayesian Statistics Part (k)

```
bayesian_binary_logistic_model <- function(){
  for(i in 1:n){
    y[i] ~ dbin(p[i], 1)
    # logit(p) in BUGS give log(p / (1 - p))
    # Linear predictor
    logit(p[i]) <- eta[i]
    eta[i] <- beta_0 + beta_1 * x[i]
  }
  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
}
```

```
n <- nrow(test)
x <- wais
y <- is_senility
data_binary_logistic <- list("n","y","x")
bayesian_binary_logistic <- jags(data = data_binary_logistic,
                                 parameters.to.save = c("beta_0", "beta_1"),
                                 n.iter = 10000,
                                 n.chains = 3,
                                 model.file = bayesian_binary_logistic_model)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
```

```
##    Observed stochastic nodes: 54
##    Unobserved stochastic nodes: 2
##    Total graph size: 165
##
## Initializing model
```

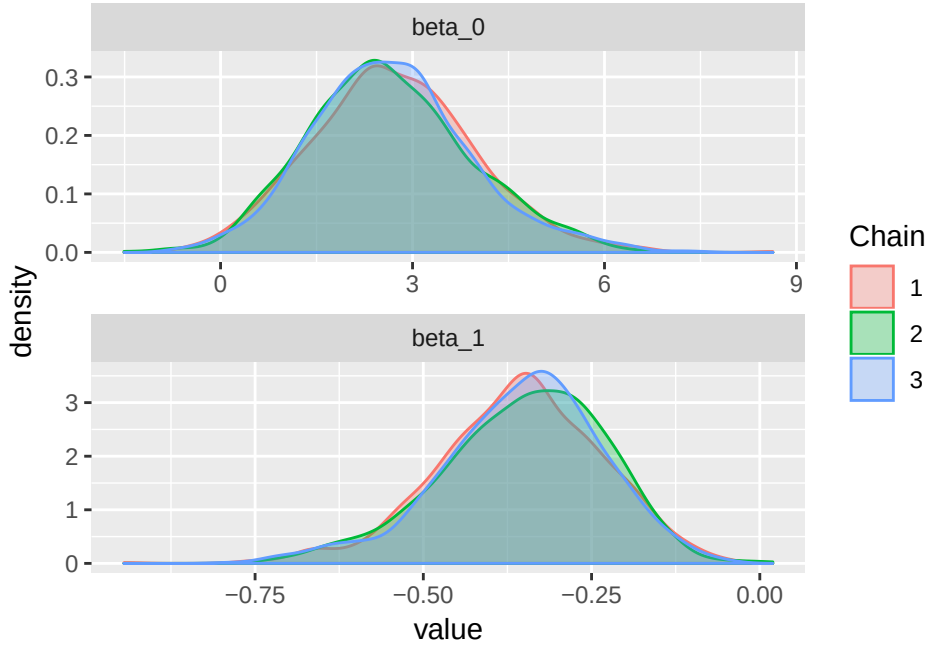
```
print(bayesian_binary_logistic , intervals = c(0.025, 0.5, 0.975))
```

```
## Inference for Bugs model at "/var/folders/yj/r_thtqks72vb9fglwpzyn8qc0000gn/T//RtmpPJDQGw/model17c5a
##  3 chains, each with 10000 iterations (first 5000 discarded), n.thin = 5
##  n.sims = 3000 iterations saved
##          mu.vect sd.vect   2.5%    50%  97.5%  Rhat n.eff
## beta_0     2.669   1.262  0.363  2.585  5.400 1.001  3000
## beta_1    -0.353   0.121 -0.626 -0.345 -0.140 1.001  3000
## deviance  53.074   2.037 51.067 52.425 58.702 1.001  3000
##
## 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 = 2.1 and DIC = 55.1
## DIC is an estimate of expected predictive error (lower deviance is better).
```

typo

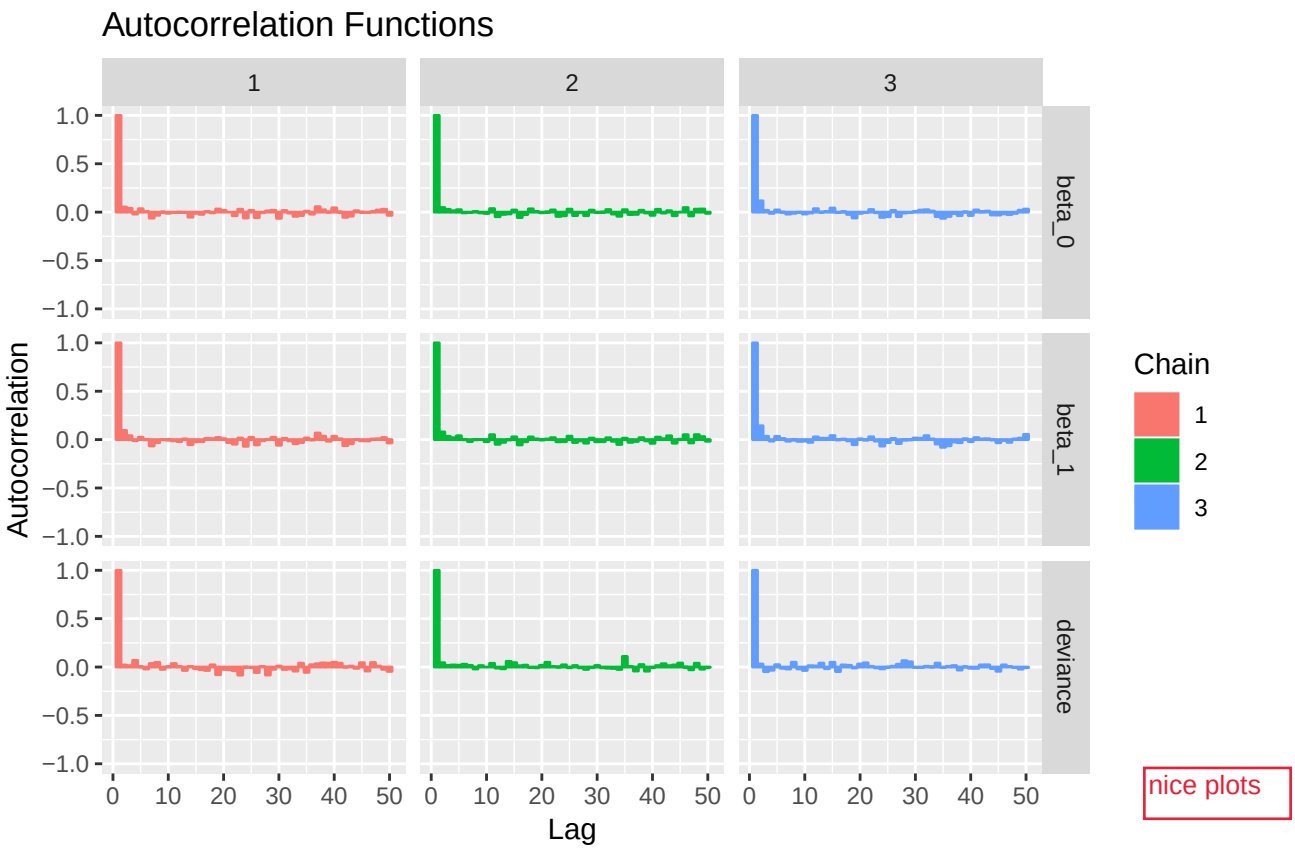
The 95% credible interval for β_0 is [0.5, 5.3] and the 95% credible interval for β_1 is [-0.6, -0.1]. The 95% credible internals for both parameters are not including zero. In addition, the 95% credible internals for β_1 contains negative values which means the WAIS score would impact the senility negatively.

```
bayesian_regression.mcmc <- as.mcmc(bayesian_binary_logistic)
bayesian_regression.ggs <- ggs(bayesian_regression.mcmc)
ggs_density(bayesian_regression.ggs, family = "^beta")
```



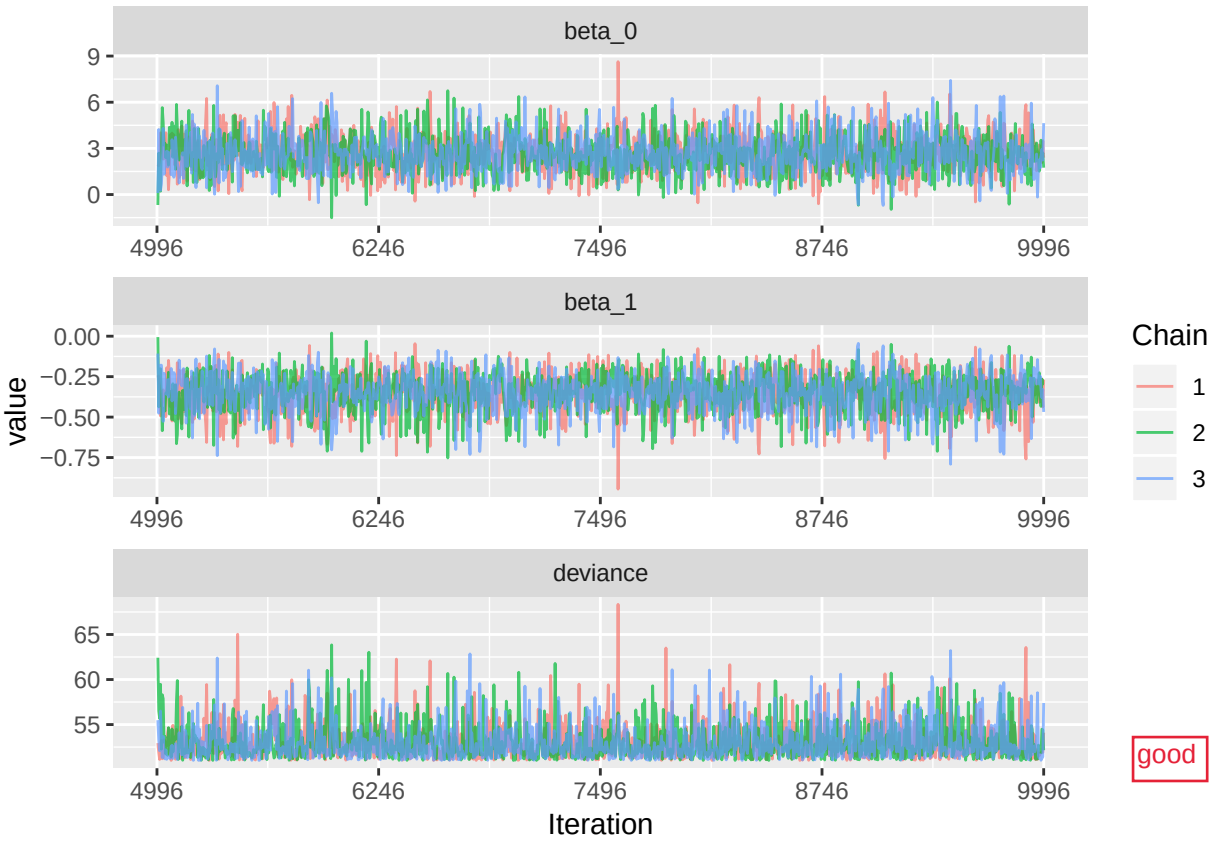
The results from the algorithm display similarity between chains, which shows that the results do no depend on the starting point of the algorithm. Moreover, for β_0, most of the points are concentrated between 0.5 and 5. On the other hand, for β_1 most of the points are concentrated between -0.6 and -0.1.

```
ggs_autocorrelation(bayesian_regression.ggs) +
  ggtitle("Autocorrelation Functions")
```



The above plots show that because the plots fall into zero directly after the first spike (the value one), therefore, the samples from the posteriors are independent. In addition, when the autocorrelation hit zero immediately, the chains are said to explore the posterior distribution well.

```
ggs_traceplot(bayesian_regression.ggs)
```



The trace plot shows that the value of parameter took during the runtime of the chain. In addition, the variability between the chains is low which is good because it shows that the sampled values have settled.

Bayesian Statistics Part (l)

```
bayesian_binary_logistic_model <- function(){
  for(i in 1:n){
    y[i] ~ dbin(p[i], 1)
    logit(p[i]) <- eta[i]
    eta[i] <- beta_0 + beta_1 * x[i]
  }
  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
  eta_new <- beta_0 + beta_1 * x_new
  p_new <- exp(eta_new) / (1 + exp(eta_new))
}
```

```
x_new <- 0
data_binary_logistic_predict <- list("n", "y", "x", "x_new")
#
bayesian_binary_logistic <- jags(data = data_binary_logistic_predict,
                                 parameters.to.save = c("beta_0",
                                                        "beta_1",
                                                        "p_new"),
                                 n.iter = 100000,
                                 n.chains = 3,
```

```
                                  model.file = bayesian_binary_logistic_model)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 54
##    Unobserved stochastic nodes: 2
##    Total graph size: 171
##
## Initializing model
```

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

```
## Inference for Bugs model at "/var/folders/yj/r_thtqks72vb9fglwpzyn8qc0000gn/T//RtmpPJDQGw/model17c5a8
##  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     2.702   1.289  0.437  2.622  5.441 1.001  3000
## beta_1    -0.356   0.124 -0.621 -0.345 -0.143 1.001  3000
## p_new      0.896   0.108  0.607  0.932  0.996 1.002  3000
## deviance  53.135   2.128 51.057 52.496 59.032 1.001  3000
##
## 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 = 2.3 and DIC = 55.4
## DIC is an estimate of expected predictive error (lower deviance is better).
```

good

```
x_new <- 20
data_binary_logistic_predict <- list("n", "y", "x", "x_new")
#
bayesian_binary_logistic <- jags(data = data_binary_logistic_predict,
                                 parameters.to.save = c("beta_0",
                                                        "beta_1",
                                                        "p_new"),
                                 n.iter = 100000,
                                 n.chains = 3,
                                 model.file = bayesian_binary_logistic_model)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 54
##    Unobserved stochastic nodes: 2
##    Total graph size: 169
##
## Initializing model
```

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

```
## Inference for Bugs model at "/var/folders/yj/r_thtqks72vb9fglwpzyn8qc0000gn/T//RtmpPJDQGw/model17c5a
##  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      2.609   1.251  0.224  2.561  5.140 1.001  3000
## beta_1     -0.349   0.120 -0.594 -0.341 -0.131 1.001  3000
## p_new       0.023   0.029  0.001  0.014  0.103 1.001  3000
## deviance   53.083   2.009 51.078 52.460 58.452 1.001  3000
##
## 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 = 2.0 and DIC = 55.1
## DIC is an estimate of expected predictive error (lower deviance is better).
```

What can you say about these results?

Bayesian Statistics Part (m)

There is another reason, which is related to the sign of beta1.

What about the data collection method?

Based on the resulting calculation, the people with lower WAIS score are most likely to show senility symptoms because the mean probability at x = 0 is close to one (0.9) while people with high WAIS score are most likely not showing senility symptoms because the mean probability at x = 20 is close to zero. The limitation is that because the sample size is too small. To improve this we would recommend to increase the sample size.

Bayesian Statistics Part (n)

With $p = 0.5$, we could transform our logistic regression model into:

$$0 = \quad beta_0 + beta_1 WAIS_i \tag{3}$$

```
bayesian_binary_logistic_model <- function(){
  for(i in 1:n){
    y[i] ~ dbin(p[i], 1)
    logit(p[i]) <- eta[i]
    eta[i] <- beta_0 + beta_1 * x[i]
  }
  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
  x_new <- - beta_0 / beta_1
}
n <- nrow(test)
x <- wais
y <- is_senility
data_binary_logistic_new <- list("n", "y", "x")
init_values_Regression <- list(
  list(beta_0 = 0.3, beta_1 = -0.6),
  list(beta_0 = 0.3, beta_1 = -0.3),
  list(beta_0 = 3, beta_1 = -0.6),
  list(beta_0 = 3, beta_1 = -0.3))
#
bayesian_regression_2 <- jags(data = data_binary_logistic_new,
                              inits = init_values_Regression,
                              parameters.to.save = c("beta_0",
                                                     "beta_1",
                                                     "x_new"),
```

```
                              n.iter = 100000,
                              n.chains = 4,
                              model.file = bayesian_binary_logistic_model)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 54
##    Unobserved stochastic nodes: 2
##    Total graph size: 167
##
## Initializing model
```

Using the Gelman-Rubin diagnostic for checking convergence in MCMC:

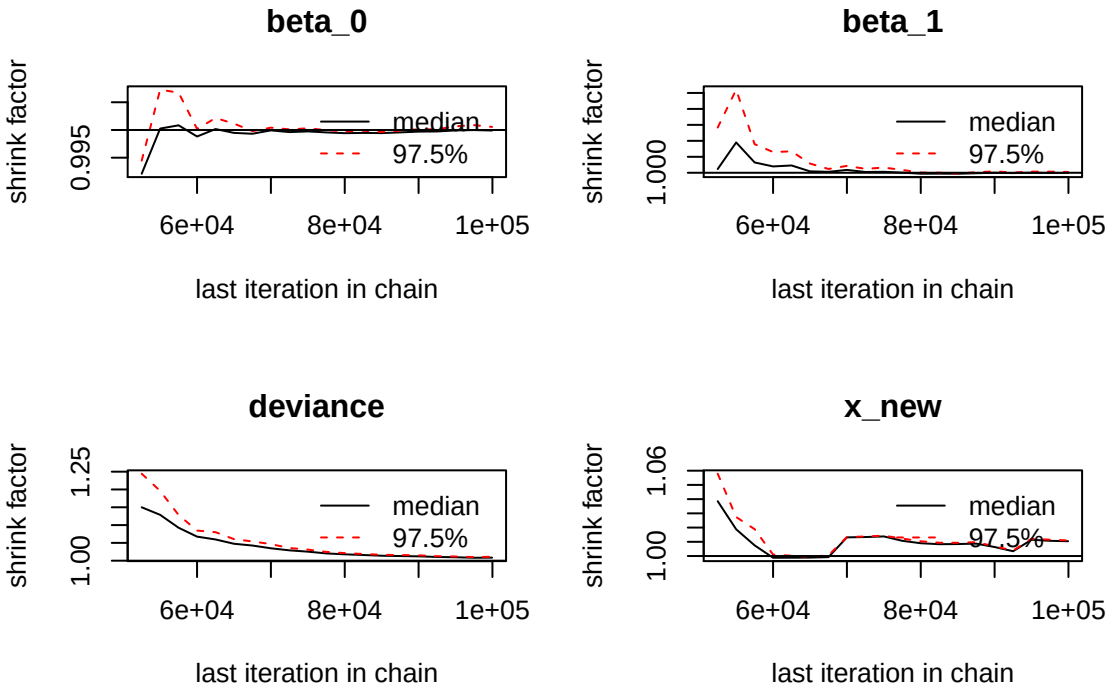
```
bayesian_regression_2.mcmc <- as.mcmc(bayesian_regression_2)
gelman.diag(bayesian_regression_2.mcmc)
```

```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## beta_0         1.00       1.00
## beta_1         1.00       1.00
## deviance       1.01       1.01
## x_new          1.01       1.01
##
## Multivariate psrf
##
## 1
```

It is very good to include the Gelman-Rubin diagnostic with plots. However, why showing it only for this analysis and not for the others?

The results are desirable. The Gelman-Rubin diagnostics show that the values are close to 1, which indicate that the chains have forgotten the initial values and have settled down.

```
gelman.plot(bayesian_regression_2.mcmc)
```



The above graphs also show that values are close to 1. In addition, values get smaller by running chains longer.

```
print(bayesian_regression_2 , intervals = c(0.025, 0.5, 0.975))
```

```
## Inference for Bugs model at "/var/folders/yj/r_thtqks72vb9fglwpzyn8qc0000gn/T//RtmpPJDQGw/model17c5a
##  4 chains, each with 1e+05 iterations (first 50000 discarded), n.thin = 50
##  n.sims = 4000 iterations saved
##          mu.vect sd.vect   2.5%    50%  97.5%  Rhat n.eff
## beta_0     2.653   1.267  0.329  2.592  5.259 1.001  4000
## beta_1    -0.353   0.123 -0.612 -0.344 -0.132 1.001  4000
## x_new      7.092   2.519  2.346  7.523  9.494 1.010  4000
## deviance  53.146   2.255 51.066 52.475 58.871 1.005  2400
##
## 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 = 2.5 and DIC = 55.7
## DIC is an estimate of expected predictive error (lower deviance is better).
```

The values reported are not exactly equal to those printed in the output. I suspect that you copied and pasted the values from a previous simulations. However, you can include R results in the text in R Markdown. Please, have a look at the MATH500 material.

The posterior median of the WAIS score is 7.1, and the 95% credible interval is [2.6, 9.5].

What can you say about these results?

3 Reference

References

Nasution, M., Sitompul, O. and Ramli, M. (2018). PCA based feature reduction to improve the accuracy of decision tree c4.5 classification. *Journal of Physics: Conference Series*, 978, p.012-058.