University of Edinburgh

School of Mathematics

Bayesian Data Analysis, 2024/2025, Semester 2

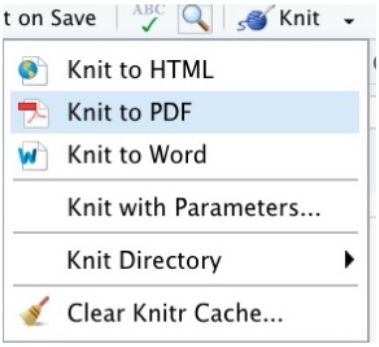
Assignment 1

IMPORTANT INFORMATION ABOUT THE ASSIGNMENT

In this paragraph, we summarize the essential information about this assignment. The format and rules for this assignment are different from your other courses, so please pay attention.

1) Deadline: The deadline for submitting your solutions to this assignment is 24 February 12:00 noon Edinburgh time.

2) Format: You will need to submit your work as 2 components: a PDF report, and your R Markdown (.Rmd) notebook (this can be in a zip file if you include additional images). There will be two separate submission systems on Learn: Gradescope for the report in PDF format, and a Learn assignment for the code in Rmd format. You need to write your solutions into this R Markdown notebook (code in R chunks and explanations in Markdown chunks), and then select Knit/Knit to PDF in RStudio to create a PDF report.



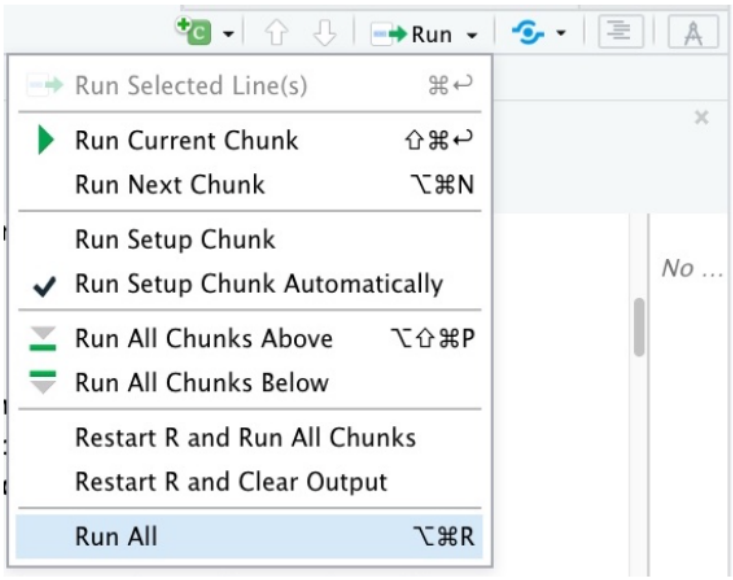
The compiled PDF needs to contain everything in this notebook, with your code sections clearly visible (not hidden), and the output of your code included. Reports without the code displayed in the PDF, or without the output of your code included in the PDF will be marked as 0, with the only feedback "Report did not meet submission requirements".

You need to upload this PDF in Gradescope submission system, and your Rmd file in the Learn assignment submission system. You will be required to tag every sub question on Gradescope.

Some key points that are different from other courses:

a) Your report needs to contain written explanation for each question that you solve, and some numbers or plots showing your results. Solutions without written explanation that clearly demonstrates that you understand what you are doing will be marked as 0 irrespective of whether the numerics are correct or not.

b) Your code has to be possible to run for all questions by the Run All in RStudio, and reproduce all of the numerics and plots in your report (up to some small randomness due to stochasticity of Monte Carlo simulations). The parts of the report that contain material that is not reproduced by the code will not be marked (i.e. the score will be 0), and the only feedback in this case will be that the results are not reproducible from the code.



c) Multiple Submissions are allowed BEFORE THE DEADLINE are allowed for both the report, and the code.
However, multiple submissions are NOT ALLOWED AFTER THE DEADLINE.
YOU WILL NOT BE ABLE TO MAKE ANY CHANGES TO YOUR SUBMISSION AFTER THE DEADLINE.
Nevertheless, if you did not submit anything before the deadline, then you can still submit your work after the deadline, but late penalties will apply. The timing of the late penalties will be determined by the time you have submitted BOTH the report, and the code (i.e. whichever was submitted later counts).

We illustrate these rules by some examples:

Alice has spent a lot of time and effort on her assignment for BDA. Unfortunately she has accidentally introduced a typo in her code in the first question, and it did not run using Run All in RStudio. - Alice will get 0 for the part of the assignments that do not run, with the only feedback "Results are not reproducible from the code".

Bob has spent a lot of time and effort on his assignment for BDA. Unfortunately he forgot to submit his code. He will get one reminder to submit his code. If he does not do it, Bob will get 0 for the whole assignment, with the only feedback "Results are not reproducible from the code, as the code was not submitted."

Charles has spent a lot of time and effort on his assignment for BDA. He has submitted both his code and report in the correct formats. However, he did not include any explanations in the report. Charles will get 0 for the whole assignment, with the only feedback "Explanation is missing."

3) Group work: This is an INDIVIDUAL ASSIGNMENT. You can talk to your classmates to clarify questions, but you have to do your work individually and cannot copy parts from other students. Students who submit work that has not been done individually will be reported for Academic Misconduct, which can lead to severe consequences. Each question will be marked by a single instructor, and submissions will be compared by advanced software tools, so we will be able to spot students who copy.

4) Piazza: During the assignments, the instructor will change Piazza to allow messaging the instructors only, i.e. students will not see each others messages and replies.

Only questions regarding clarification of the statement of the problems will be answered by the instructors. The instructors will not give you any information related to the solution of

the problems, such questions will be simply answered as "This is not about the statement of the problem so we cannot answer your question."

THE INSTRUCTORS ARE NOT GOING TO DEBUG YOUR CODE, AND YOU ARE ASSESSED ON YOUR ABILITY TO RESOLVE ANY CODING OR TECHNICAL DIFFICULTIES THAT YOU ENCOUNTER ON YOUR OWN.

5) Office hours: There will be one office hour per week during the 2 weeks for this assignment. This is in JCMB 5608. I will be happy to discuss the course/workshop materials. However, I will only answer questions about the assignment that require clarifying the statement of the problems, and will not give you any information about the solutions.

6) Late submissions and extensions: UP TO A MAXIMUM OF 3 CALENDAR DAYS EXTENSION IS ALLOWED FOR THIS ASSIGNMENT IN THE ESC SYSTEM. You need to apply before the deadline.

If you submit your solutions on Learn before the deadline, the system will not allow you to update it even if you have received an extension. There is only 1 submission allowed after the deadline.

Students who have existing Learning Adjustments in Euclid will be allowed to have the same adjustments applied to this course as well, but they need to apply for this BEFORE THE DEADLINE on the website.

https://www.ed.ac.uk/student-administration/extensions-special-circumstances

by clicking on "Access your learning adjustment". This will be approved automatically.

Students who submit their work late will have late submission penalties applied by the ESC team automatically (this means that even if you are 1 second late because of your internet connection was slow, the penalties will still apply). The penalties are 5% of the total mark deduced for every day of delay started (i.e. one minute of delay counts for 1 day). The course instructors do not have any role in setting these penalties, we will not be able to change them.

```
rm(list = ls(all = TRUE))
#Do not delete this!
#It clears all variables to ensure reproducibility
```

Problem 1) Auld Reekie rain

Figure 1: Our dataset consists of monthly mean weather variables sampled in Edinburgh.

This question looks at modelling rainfall in Edinburgh. You can load the dataset using the below code:

```
# Load data
Edi.rain <- read.csv("Edinburgh_rainfall.csv")
```

The dataset consists of 12*83=996 monthly observations of different environmental variables sampled in Edinburgh. These variables includes:

- daily rainfall total (mm)
- average windspeed (m/s)
- average temperature (K)
- year: (this is stored as a real number between 1940 and 2023, i.e. 1950.0 and 1950.5 correspond to January and July for 1950.)

All environmental variables are provided as monthly averages (so rainfall is recorded as the monthly average of the daily total rainfall). Note that the last 12 observations of rainfall, corresponding to the year 2022, are missing (set to NA).

Q1.1)[12 marks]

Fit a Bayesian Linear regression model in INLA (with Gaussian likelihood) using rainfall as the response. Use all covariates, including the year, in a linear model. Scale all of the non-categorical covariates that you use in your model. You should do this before fitting the INLA model.

For the regression coefficients and intercept, use zero mean Normal priors with standard deviation 100. For the Gaussian variance, use an inverse Gamma prior with parameters 1 and 0.01.

Print out the model summary and interpret the posterior means of the regression coefficients. Plot the marginal posterior densities for the regression coefficients.

```r
require(INLA)
```

```
## Loading required package: INLA
```

```
## Warning: package 'INLA' was built under R version 4.4.2
```

```
## Loading required package: Matrix
```

```
## This is INLA_24.12.11 built 2024-12-11 20:07:43 UTC.
##  - See www.r-inla.org/contact-us for how to get help.
##  - List available models/likelihoods/etc with inla.list.models()
##  - Use inla.doc(<NAME>) to access documentation
```

```r
# scale covariates data, keep response variable unchanged.
scaled_data <- scale(Edi.rain)
scaled_df <- data.frame(rainfall = Edi.rain$rainfall, windspeed = scaled_data[,"windspeed"],
                        temperature = scaled_data[,"temperature"], year = scaled_data[,"year"])
# lm1 <- lm(rainfall ~ windspeed + temperature + year, data = scaled_df)
# summary(lm1)

# Fit Bayesian linear reg with gaussian likelihood in INLA
sd <- 100
prior.beta <- list(mean.intercept = 0, prec.intercept = 1/(sd^2),
                   mean = 0, prec = 1/(sd)^2)
# prior.beta <- list(mean.intercept = 0, prec.intercept = 1/(sd^2),
#                    mean = list(windspeed = 0, temperature = 0, year = 0),
#                    prec = list(windspeed = 1/(sd^2), temperature = 1/(sd^2), year = 1/(sd^2)))

prec.prior <- list(prec = list(prior = "loggamma", param = c(1, 0.01)))

rainfall.reg <- inla(rainfall ~ windspeed + temperature + year, data = scaled_df,
                     family = "gaussian", control.family = list(hyper = prec.prior),
                     control.fixed = prior.beta,
                     control.compute = list(config = TRUE, waic = TRUE, cpo = TRUE),
                control.predictor = list(compute = TRUE))
summary(rainfall.reg) # WAIC = 2744.95
```

```
## Time used:
##     Pre = 0.829, Running = 0.291, Post = 0.0241, Total = 1.14
## Fixed effects:
##              mean    sd 0.025quant 0.5quant 0.975quant  mode kld
## (Intercept) 2.240 0.031      2.179    2.240      2.301 2.240   0
## windspeed   0.081 0.032      0.019    0.081      0.143 0.081   0
## temperature 0.027 0.032     -0.035    0.027      0.089 0.027   0
## year        0.156 0.032      0.094    0.156      0.218 0.156   0
##
## Model hyperparameters:
##                                          mean    sd 0.025quant 0.5quant
## Precision for the Gaussian observations 1.06 0.048      0.967     1.06
##                                         0.975quant mode
## Precision for the Gaussian observations       1.15 1.06
##
```

```
## Watanabe-Akaike information criterion (WAIC) ...: 2744.95
## Effective number of parameters .................: 5.27
##
## Marginal log-Likelihood:  -1406.34
## CPO, PIT is computed
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

```
# waic
rainfall.reg$waic$waic #2744.951
```

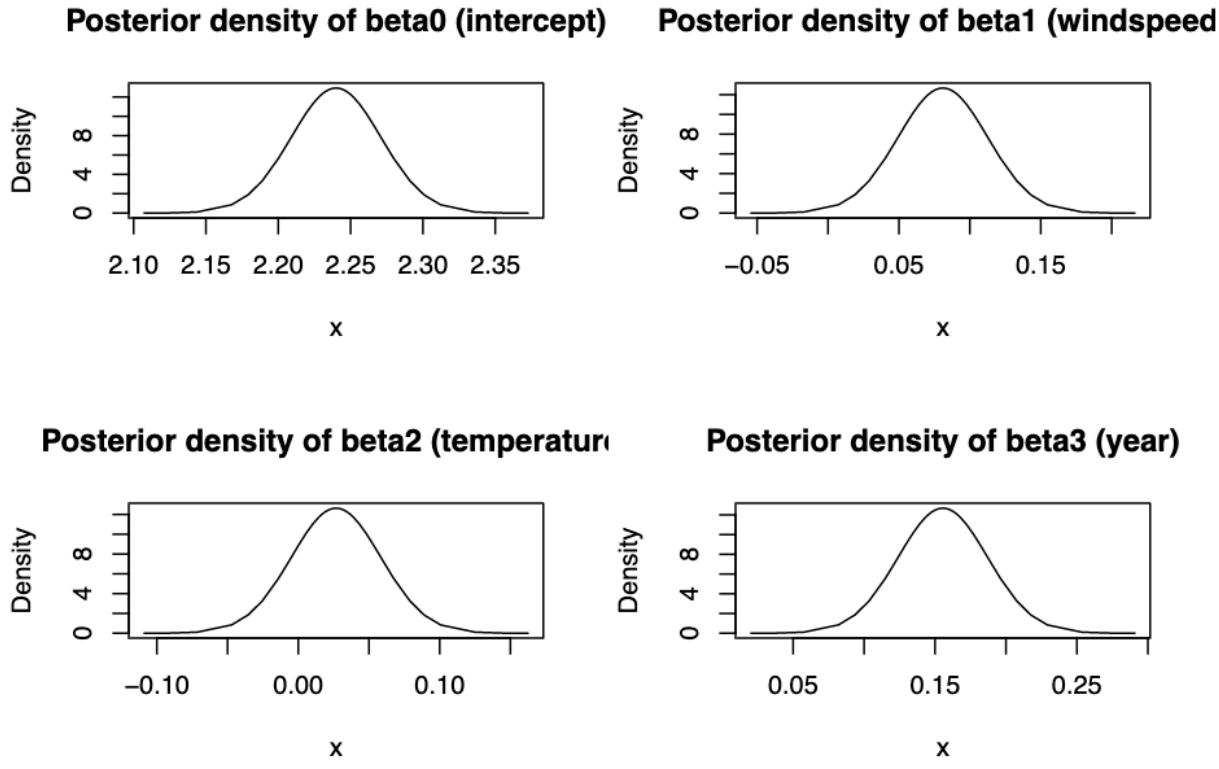
```
## [1] 2744.951
```

```
# nslcpo
nslcpo_norm <- -sum(log(rainfall.reg$cpo$cpo)[!is.na(rainfall.reg$cpo$cpo)])
nslcpo_norm #1372.475
```

```
## [1] 1372.475
```

Plotting marginal posterior densities

```
par(mfrow=c(2,2))
plot(rainfall.reg$marginals.fixed$`(Intercept)`, type = 'l',xlab="x",ylab="Density",main="Posterior den
plot(rainfall.reg$marginals.fixed$'windspeed',type='l',xlab="x",ylab="Density",main="Posterior density
plot(rainfall.reg$marginals.fixed$`temperature`, type = 'l',xlab="x",ylab="Density",main="Posterior den
plot(rainfall.reg$marginals.fixed$`year`, type = 'l',xlab="x",ylab="Density",main="Posterior density of
```



Explanation: (write your explanation here)

Interpretation of each Coefficients: - Intercept (mean = 2.240): If all of the predictors are fixed at their mean values, the expected rainfall is 2.240.

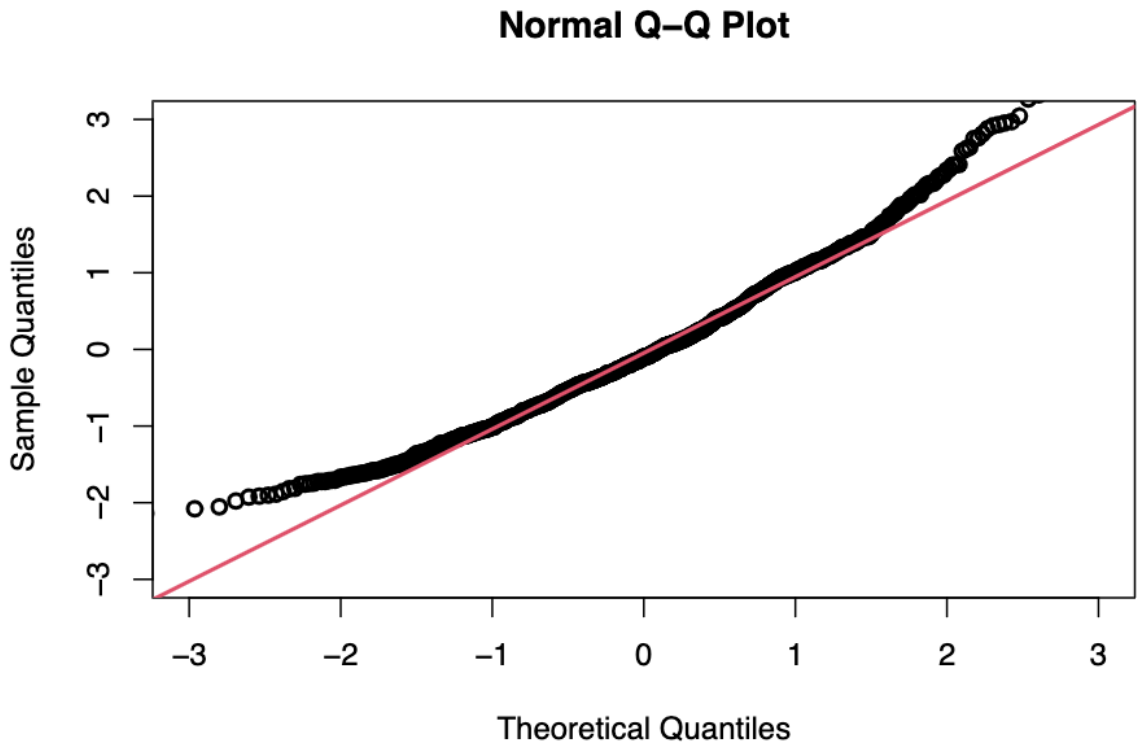
- windspeed (mean = 0.081): An increase in 1 standard deviation in windspeed means an increase chance in average monthly rainfall by 0.081mm.
- temperature (mean = 0.027): An increase in 1 standard deviation of temperature suggest a higher chance of average monthly rainfall by 0.027mm but the value here is close to 0 suggesting it may not have the same power as windspeed or year.
- year (mean = 0.156): This means that rainfall rate has increased over time.

Q1.2)[8 marks]

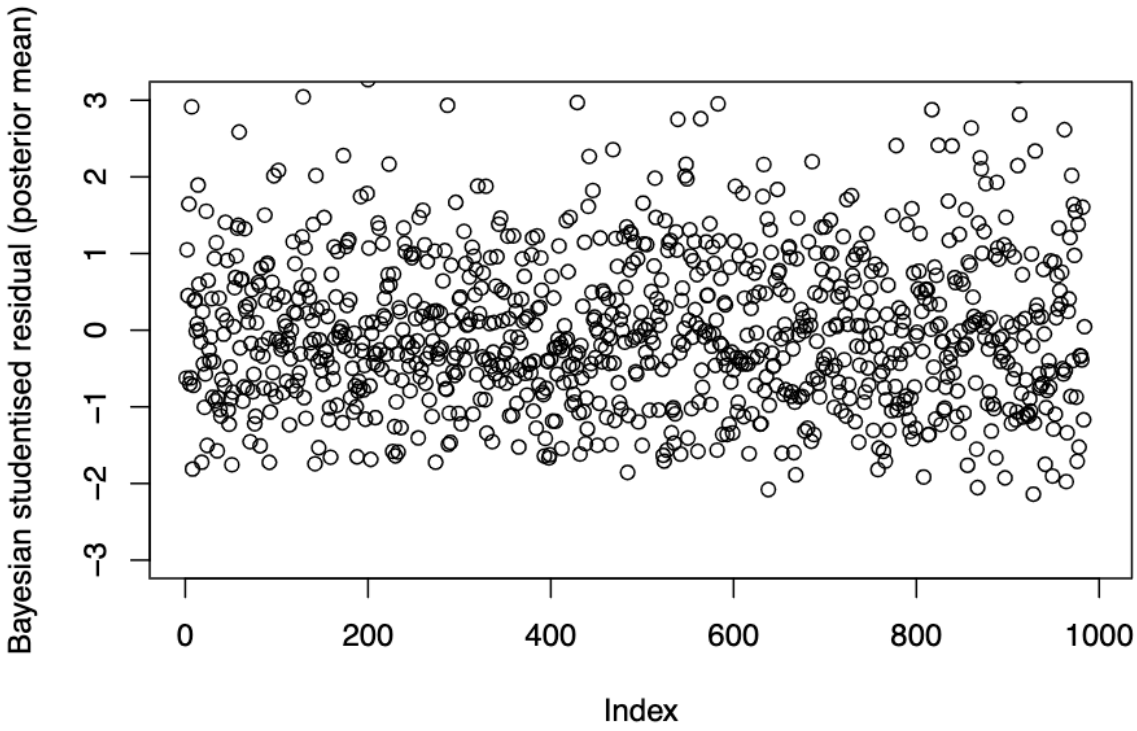
Perform the necessary model checks for the linear regression model in Q1.1 using the Studentized residuals. Interpret your results.

```
sample <- 10000
rain.samples <- inla.posterior.sample(n = sample, result = rainfall.reg)
#We extract the samples for the 4 regression coefficients
beta0=inla.posterior.sample.eval(function(...) {(Intercept)},
  rain.samples)
beta1=inla.posterior.sample.eval(function(...) {windspeed},
  rain.samples)
beta2=inla.posterior.sample.eval(function(...) {temperature},
  rain.samples)
beta3=inla.posterior.sample.eval(function(...) {year},
  rain.samples)

#The precision hyperparameter is called theta, so we can get sigma2=1/theta
sigma2=1/(inla.posterior.sample.eval(function(...) {theta},
  rain.samples))

n <- nrow(Edi.rain)
X <- cbind(rep(1,n), scaled_df$windspeed, scaled_df$temperature, scaled_df$year)
H = X%*%solve((t(X)%*%X))%*%t(X)
y <- scaled_df$rainfall

fitted_val_rain <- inla.posterior.sample.eval(function(...) {Predictor},
                                              rain.samples)

studentisedred=matrix(0,nrow=n,ncol=sample)
for(l in 1:sample){
  for(i in 1:n){
  studentisedred[i,l]=(y[i]-fitted_val_rain[i,l])/(sqrt(sigma2[l]*(1-diag(H)[i])))
  }
}

#posterior mean of studentised residuals
studentisedredm=numeric(n)
for(i in 1:n){
studentisedredm[i]=mean(studentisedred[i,])
}
```

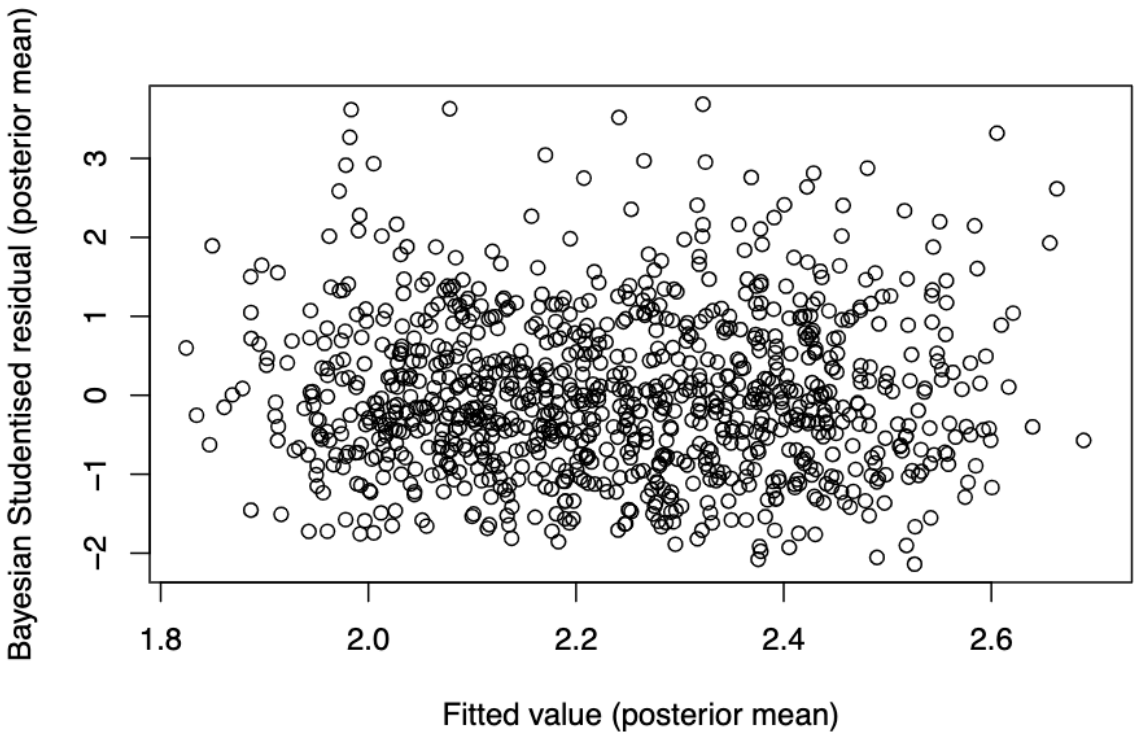
```r
#QQ-plot
qqnorm(studentisedredm,xlim=c(-3,3),ylim=c(-3,3),lwd=2)
qqline(studentisedredm,col=2,lwd=2)
```




```r
#checking independence of error terms
#seq_along(studentisedredm) creates a vector with the indices of studentisedredm, i.e. 1,2,...,32
#we could also write 1:n instead of seq_along(studentisedredm)
plot(seq_along(studentisedredm),studentisedredm,xlab="Index",
     ylab="Bayesian studentised residual (posterior mean)",ylim=c(-3,3))
```




```
#posterior mean fitted values
fittedvaluesm=rainfall.reg$summary.fitted.values$mean

plot(fittedvaluesm,studentisedredm,xlab="Fitted value (posterior mean)",
     ylab="Bayesian Studentised residual (posterior mean)")
```



Explanation: (write your explanation here)

1. Checking Normality assumption:

Using the qqplot, we can assess the normality assumption of the residuals. From the plot, we see a heavy tail in the lower and upper tail, suggesting that there is a few extreme data influencing the distribution, hence this means the distribution may have a heavier tails than a normal distribution.

2. Checking independence of the error term: Bayesian Studentized residuals against the index:

From the residuals plot, it seems the data looks randomly scattered with no clear pattern. However, one thing to point out is that some residuals reaches 3 while the lower bound is around -2, suggesting a potential large residuals that might be an outlier. But overall, the plot still shows an independence in the residuals.

3. To check the assumptions of linearity and constant variance, we can plot the bayesian studentized residuals against the posterior mean of the fitted values:

The plot looks randomly scattered with no obvious patterns with most residuals being in range of -2 to 2, with some particular large residuals but still no clear pattern. Suggesting that the linearity and constant variance assumption is may be met.

Q1.3)[20 marks]

Robust regression:

Using the same linear predictor specification as in Q1.1), find a robust regression model that provides an improved fit to the data. You can do this by changing the INLA family for the

error distribution. For the family, choose from the (non-Gaussian) two parameter models considered by the authors in this paper https://doi.org/10.1002/2016WR019276; note that a copy of the PDF for this paper is also in the assignment .zip. Use only models that are implemented in INLA.

Choose and specify appropriate priors for your chosen family. For the fixed effect coefficients, you may use the same priors as in Q1.1).

You should quantify the goodness-of-fits using the WAIC and NSLCPO.

Write down the full formulation of the Bayesian hierarchical model that you have chosen to use, including your choice of prior distributions.

```
sd_gam <- 100
prior.beta_gam <- list(mean.intercept = 0, prec.intercept = 1/(sd_gam^2),
                       mean = 0, prec = 1/(sd_gam)^2)
prior.prec_gam <- list(prec = list(prior = "loggamma", param = c(1,0.01)))

rainfall.reg.gamma <- inla(rainfall ~ windspeed + temperature + year, data = scaled_df,
                      family = "gamma", control.family = list(hyper = prior.prec_gam),
                      control.fixed = prior.beta_gam,
                      control.compute = list(config = TRUE, waic = TRUE, cpo = TRUE),
                control.predictor = list(compute = TRUE, link = 1))

summary(rainfall.reg.gamma)
```

```
## Time used:
##     Pre = 0.629, Running = 0.283, Post = 0.0161, Total = 0.928
## Fixed effects:
##              mean    sd 0.025quant 0.5quant 0.975quant  mode kld
## (Intercept) 0.804 0.015      0.775    0.804      0.832 0.804   0
## windspeed   0.033 0.015      0.004    0.033      0.062 0.033   0
## temperature 0.013 0.015     -0.016    0.013      0.043 0.013   0
## year        0.069 0.015      0.040    0.069      0.098 0.069   0
##
## Model hyperparameters:
##                                                  mean   sd 0.025quant 0.5quant
## Precision-parameter for the Gamma observations 4.81 0.21       4.41     4.80
##                                                  0.975quant mode
## Precision-parameter for the Gamma observations       5.23 4.80
##
## Watanabe-Akaike information criterion (WAIC) ...: 2689.75
## Effective number of parameters .................: 4.85
##
## Marginal log-Likelihood:  -1380.54
## CPO, PIT is computed
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

```
# waic
rainfall.reg.gamma$waic$waic #2689.746
```

```
## [1] 2689.746
```

```r
# nslcpo
nslcpo_rainfall_gamma <- -sum(log(rainfall.reg.gamma$cpo$cpo[!is.na(rainfall.reg.gamma$cpo$cpo)]))
nslcpo_rainfall_gamma #1344.873
```

```
## [1] 1344.873
```

Explanation: (write your explanation here)

Based on the paper above, we can use Gamma likelihood as a two parameter models to be a robust regression model for our case.

$$y_i \sim Gamma(shape = \alpha = \frac{\mu_i}{Var(y_i)}, rate = \frac{\alpha}{\mu_i}))$$

where we use parameterization of the distribution by its rate $\frac{\alpha}{\mu_i}$ and shape parameter $\alpha = \frac{\mu_i}{Var(y_i)}$.

For Gamma regression, we use a log link to connect the random component and the linear predictor:

$$log(\mu_i) = \beta_0 + \beta_1 windspeed_i + \beta_2 temperature_i + \beta_3 year_i$$

with the expected rainfall to be:

$$\mu_i = exp(\beta_0 + \beta_1 windspeed_i + \beta_2 temperature_i + \beta_3 year_i)$$

The hierarchical model can be represented as:

$$y_i \sim Gamma(shape = \alpha = \frac{\mu_i}{Var(y_i)}, rate = \frac{\alpha}{\mu_i}), \quad i = 1, \ldots, n$$

$$\frac{1}{Var(y_i)} \sim Gamma(1, 0.01) \quad \beta_j \sim N(\mu = 0, \tau = 1/100^2), \quad i = 1, \ldots, n, \quad j = 0, 1, 2, 3$$

$$\mu_i = exp(\beta_0 + \beta_1 windspeed_i + \beta_2 temperature_i + \beta_3 year_i).$$

We choose gamma prior for α due to its support/range of positive continous parameters, gamma ensures that $\alpha > 0$. While Normal prior are chosen for β coefficients as instructed in the question.

We have fitted the Bayesian Linear Regression using Gamma likelihood, hierarchical model representation are given above. Based on the model goodness fit, we see that the gamma likelihood model has lower WAIC and NSLCPO compared to the normal likelihood, with WAIC of Gamma likelihood being 2689.746 compared to 2744.951 for normal and NSLCPO for gamma being 1344.873 compared to 1372.475.This indicates a better fit for the gamma likelihood compared to the normal, suggesting the data might have heavier tails than what normal distribution can handle.

Q1.4)[15 marks]

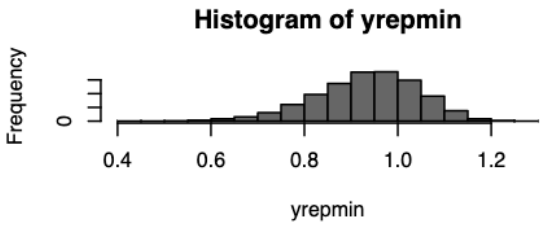
With the best-fitting model from Q1.3), perform posterior predictive checks. Interpret the output of your checks. [Hint: when generating from the posterior predictive distribution, be careful of your choice of family]

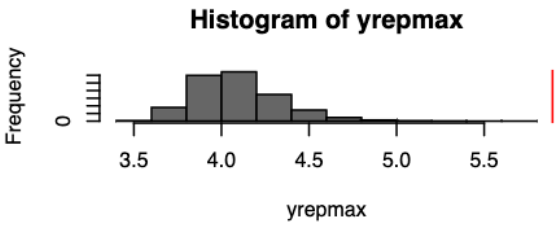
```r
# samples
nsamp <- 10000
# get samples, select a 27th obs
rainfall_gamma.samples <- inla.posterior.sample(n = nsamp, result = rainfall.reg.gamma)
# rainfall_gamma.samples <- inla.posterior.sample(n = nsamp, result = rainfall.reg.gamma, selection = l

predictor.samples_gamma =inla.posterior.sample.eval(function(...) {Predictor},
   rainfall_gamma.samples)
```

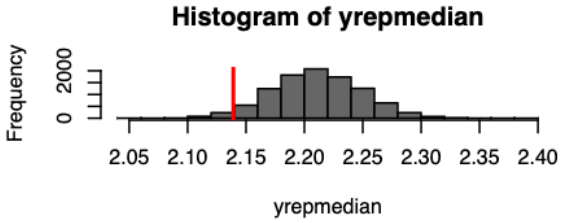
```
precision.samples <- inla.posterior.sample.eval(function(...){theta},
                                                rainfall_gamma.samples)
n <- 984
yrep=matrix(0,nrow=n,ncol=nsamp)
y=na.omit(scaled_df$rainfall)
for(row.num in 1:n){
  mu <- exp(predictor.samples_gamma[row.num, ])
  rate.samples <- mu * precision.samples
  yrep[row.num, ]<- rgamma(nsamp, shape = mu*rate.samples, rate = rate.samples)
}

#statistics of interest in this case (and different from the ones included in the model)
yrepmin=apply(yrep,2,min)
yrepmax=apply(yrep,2,max)
yrepmedian=apply(yrep,2,median)

require(fBasics)
```

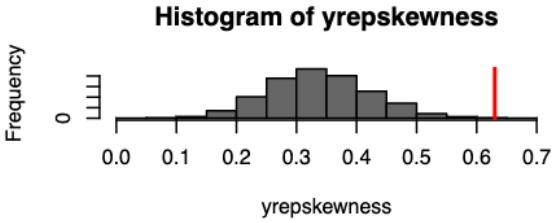
```
## Loading required package: fBasics
```

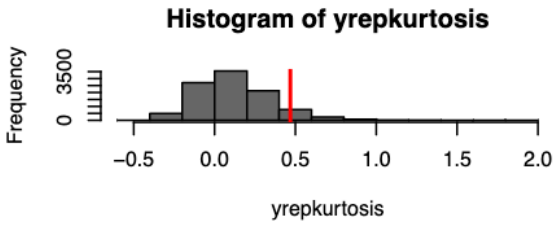
```
yrepskewness=apply(yrep,2,skewness)
yrepkurtosis=apply(yrep,2,kurtosis)

par(mfrow=c(3,2))
#Predictive checks using replicated data - minimum
hist(yrepmin,col="gray40")
abline(v=min(y),col="red",lwd=2)

#Predictive checks using replicated data - maximum
hist(yrepmax,col="gray40")
abline(v=max(y),col="red",lwd=2)

#Predictive checks using replicated data - median
hist(yrepmedian,col="gray40")
abline(v=median(y),col="red",lwd=2)

#Predictive checks using replicated data - skewness
hist(yrepskewness,col="gray40")
abline(v=skewness(y),col="red",lwd=2)

#Predictive checks using replicated data - kurtosis
hist(yrepkurtosis,col="gray40")
abline(v=kurtosis(y),col="red",lwd=2)
```











Explanation: (write your explanation here)

From the posterior predictive check, we plot the histogram of the distribution of the statistic (min, max, median, skewness, and kurtois) against the true data.

- For the histogram of yrepmin (the distribution of the lowest value of the replicated data), we see that the red line is not on the scale that the model have generated, indicating that the model did not model the lower tail of the replicated data well.
- For the histogram of yrepmax, the observed maximum is in the right tail of the replicated maximum distribution of the model, suggesting that the model often predicts lower maximum compared to what was observed. This mean that the model has a lower upper tail than the replicated data.
- For the histogram of yrepmedian, the redline for the observed median line lies around the left bulk of the replicated data, meaning that the model gives out result that have a higher median to the replicated data.
- For the histogram of yrepskewness, the redline lies around the right side of the replicated data, meaning that the model gives out result that have a lower skewness to the replicated data.
- For the histogram of yrepkurtosis, the redline is around the right side of the bulk of kurtosis distribution of the model, suggesting that the model replicate data that have a smaller peak compared to the replicated data.

Using the same model, calculate the posterior predictive probability that the total rainfall in 2022 will exceed the previously recorded maximum of the annual total rainfall throughout the observation period. Ignore leap years in your calculations. [hint: the recorded values are monthly averages of the daily rainfall totals.]

```r
#rainfall.reg.gamma
nbsamp <- 10000
#get missing index from data
missing_index <- which(is.na(scaled_df$rainfall))

rainfall.sample.2022 <- inla.posterior.sample(nbsamp, rainfall.reg.gamma,
                                              selection = list(Predictor = missing_index))

predictor.samples_gamma.2022 <- matrix(unlist(lapply(rainfall.sample.2022, function(x) x$latent[1:lengt
                                    nrow = length(missing_index),
                                    ncol = nbsamp)

precision.samples.2022 <- sapply(rainfall.sample.2022, function(x) x$hyperpar[1])

n.miss <- length(missing_index)

month <- c(31,28,31,30,31,30,31,31,30,31,30,31)

post.pred.samples=matrix(0,nrow=n.miss,ncol=nbsamp)
for(i in 1:n.miss){
  mu <- exp(predictor.samples_gamma.2022[i,])
  rate.samples.2022 <- mu * precision.samples.2022
  post.pred.samples[i,] = rgamma(nbsamp, shape = rate.samples.2022*mu, rate = rate.samples.2022) * montl
}

simulated_total_rainfall.2022 <- colSums(post.pred.samples)

total_year <- length(which(!is.na(Edi.rain$rainfall)))/12

#get the total rainfall each year from Edi.rain data
yearly_sum <- numeric(total_year)

for(i in 1:total_year){
  pointer.start <- 12*(i-1) + 1
  pointer.end <- 12*i

  monthly_avg <- Edi.rain$rainfall[pointer.start:pointer.end]

  sum_year_i <-0
  for(m in 1:12){
    sum_year_i <- sum_year_i + monthly_avg[m]*month[m]
  }

  yearly_sum[i] <- sum_year_i
}


#posterior prob
prob_post.2022 <- mean(max(yearly_sum) < simulated_total_rainfall.2022)

cat("Posterior Predictive Probabilty of Year 2022 Exceeding the maximum :", prob_post.2022, "\n")
```

```
## Posterior Predictive Probabilty of Year 2022 Exceeding the maximum : 0.0089
```

Hence, the posterior predictive probability that the total rainfall in 2022 will exceed the previously recorded maximum of the annual total rainfall throughout the observation period are given as above!

Problem 2) South American frogs

Figure 2: Our dataset consists of call recordings for South American frog species.

We will be using a subset of data from the Anuran Calls repository. The data consists of 6882 individual syllables found in the recordings of frog calls; there are 55 unique frogs in this dataset, and their ID is provided as `recordID`. For each syllable, the audio signal was decomposed into 22 Mel-frequency cepstrum coefficients (MFCCs) (we have removed the first coefficient). Each coefficient describes the spectral decomposition of the audio signal, i.e., it's shape. The dataset also includes the frogs' family, genus, and species.

```
#Load in the data
frogs<-read.csv("Frogs.csv")
```

We are going to perform a Bayesian clustering of the MFCC values in order to identify different frogs.

Q2.1) [10 marks] **Write a JAGS model to fit a bivariate normal distribution, with the following prior specification**: let $\mathbf{Y}_i = (Y_{1i}, Y_{2i}) \sim \text{BVN}(\boldsymbol{\mu}, \Sigma)$, where

$$\begin{aligned} \boldsymbol{\mu} &\sim \text{BVN}(\mathbf{0}, 100^2 * \text{I}_2), \\ \Sigma_{11} = \sigma_1^2, \quad \Sigma_{22} = \sigma_2^2, \quad \Sigma_{12} &= \Sigma_{21} = \sigma_1 \sigma_2 \rho, \\ \rho \sim \text{Unif}(-1, 1), \quad \sigma_1^2, \sigma_2^2 &\sim \text{Inverse-Gamma}(1, 0.1). \end{aligned}$$

Retrieve the `MFCCs_.2` and `MFCCs_11` coefficients for frogs 1 and 46 (note that frog 46 is a Leptodactylus Fuscus, which is pictured above), and collect these in a new dataset. Fit your bivariate normal model to these data. Use 4000 burnin, generate 10000 samples, and keep only every fifth sample. Plot the posterior densities and trace plots for μ_1, μ_2, σ_1^2, σ_2^2, and ρ, and print the model summary. Interpret these outputs.

```
require(rjags)
```

```
## Loading required package: rjags
```

```
## Loading required package: coda
```

```
## Linked to JAGS 4.3.2
```

```
## Loaded modules: basemod,bugs
```

```
require(dplyr)
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

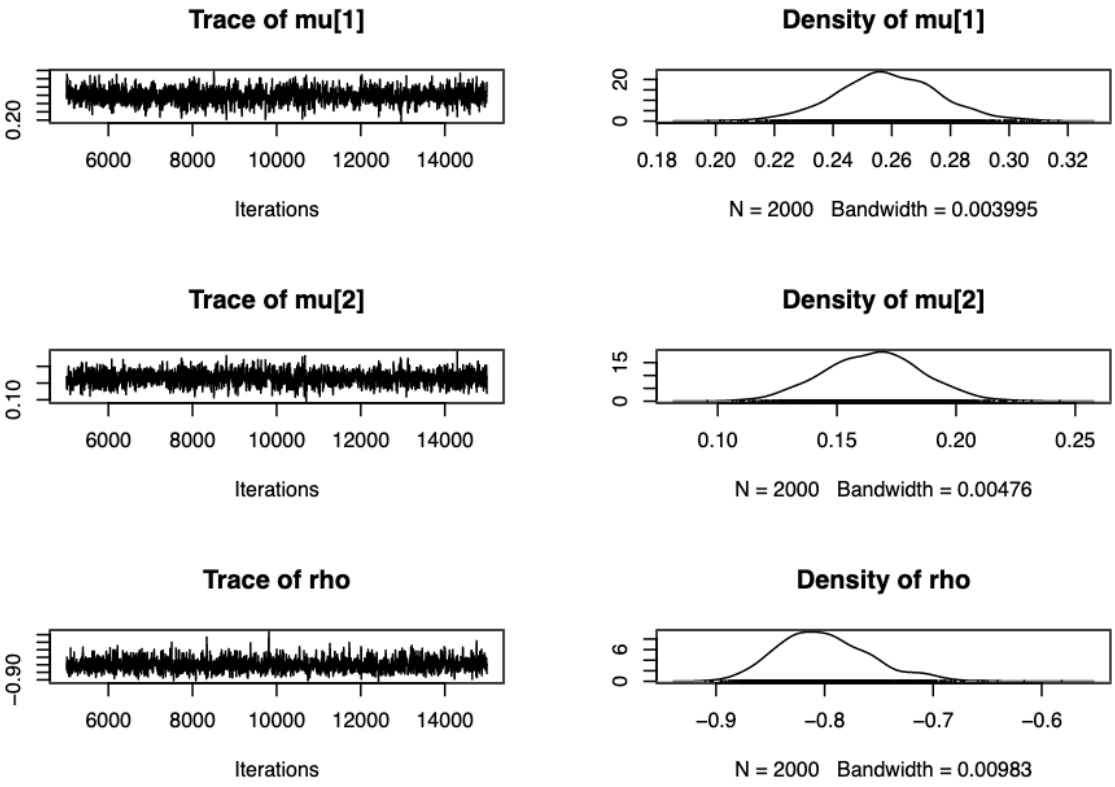
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

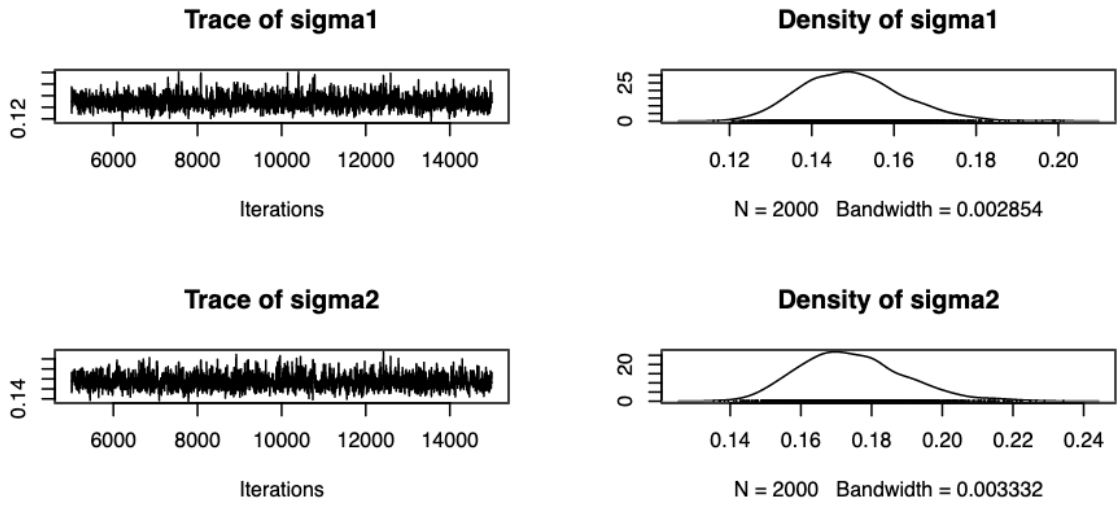
```
# retrieve MFCCs 2 and MFCCs 11 & frog 1 and 46
subset_data <- frogs %>% filter(RecordID %in% c(1,46)) %>%
  select(RecordID, MFCCs_.2, MFCCs_11)


bvn_model <- "model{
  for(i in 1:n){
    Y[i, ] ~ dmnorm(mu[], tau[, ])
  }

  mu[1:2] ~ dmnorm(amu[], tau_mu[,])

  rho ~ dunif(-1, 1)

  tau1 ~ dgamma(1, 0.1)
  tau2 ~ dgamma(1, 0.1)

  sigma1 <- 1 / sqrt(tau1)
  sigma2 <- 1 / sqrt(tau2)

  Sigma[1,1] <- 1 / tau1
  Sigma[2,2] <- 1 / tau2
  Sigma[1,2] <- rho * sigma1 * sigma2
  Sigma[2,1] <- Sigma[1,2]
```

```
  tau[1:2, 1:2] <- inverse(Sigma[,])

  # hyperprior of mean of mu
  amu[1] <- 0
  amu[2] <- 0

  # hyperprior of tau of mu
  tau_mu[1,1] <- 1 / (100^2)
  tau_mu[2,2] <- 1 / (100^2)
  tau_mu[1,2] <- 0
  tau_mu[2,1] <- 0
}"

data_jags <- list(n = nrow(subset_data),
                  Y = cbind(subset_data$MFCCs_.2, subset_data$MFCCs_11))

jags_model <- jags.model(textConnection(bvn_model), data = data_jags)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 73
##    Unobserved stochastic nodes: 4
##    Total graph size: 97
##
## Initializing model
```

```
#burn in
update(jags_model, 4000)

# generate 10000 samples and keep every fifth sample
mcmc_samples <- coda.samples(jags_model, variable.names = c("mu", "sigma1", "sigma2", "rho"),
                             n.iter = 10000, thin = 5)
# plot traceplot + densities
plot(mcmc_samples)
```


Trace of mu[1]
0.20
6000 8000 10000 12000 14000
Iterations
Density of mu[1]
0 20
0.18 0.20 0.22 0.24 0.26 0.28 0.30 0.32
N = 2000 Bandwidth = 0.003995
Trace of mu[2]
0.10
6000 8000 10000 12000 14000
Iterations
Density of mu[2]
0 15
0.10 0.15 0.20 0.25
N = 2000 Bandwidth = 0.00476
Trace of rho
−0.90
6000 8000 10000 12000 14000
Iterations
Density of rho
0 6
−0.9 −0.8 −0.7 −0.6
N = 2000 Bandwidth = 0.00983




```
summary(mcmc_samples)
```

```
##
## Iterations = 5005:15000
## Thinning interval = 5
## Number of chains = 1
## Sample size per chain = 2000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean      SD  Naive SE Time-series SE
## mu[1]   0.2586 0.01750 0.0003913      0.0003913
## mu[2]   0.1652 0.02053 0.0004591      0.0004591
## rho    -0.8001 0.04289 0.0009590      0.0010512
## sigma1  0.1498 0.01239 0.0002771      0.0003054
## sigma2  0.1745 0.01485 0.0003320      0.0003603
##
## 2. Quantiles for each variable:
##
##           2.5%     25%     50%     75%   97.5%
## mu[1]   0.2234  0.2472  0.2579  0.2703  0.2936
## mu[2]   0.1259  0.1514  0.1659  0.1789  0.2047
## rho    -0.8736 -0.8301 -0.8041 -0.7733 -0.7044
## sigma1  0.1282  0.1410  0.1491  0.1575  0.1762
## sigma2  0.1496  0.1641  0.1732  0.1833  0.2072
```

```
# effectiveSize(mcmc_samples)
```

Explanation: (write your explanation here)

The traceplot looks scattered randomly, showing no sign of pattern, this suggest a good convergence in the MCMC and no significant autocorrelation is observed. The density plot shows the uncertainty in the estimate from its spread, with most of the parameters being spread not too wide or narrow. Furthermore, the density plot peak shows the most probable posterior parameter value, with the peak coincides with the mean value seen in summary(mcmc_samples)

The summary function shows the estimated mean, sd as well as the quantiles for each variables.

- mu[1] (mean: 0.2589, sd: 0.0177): represents the expected value (mean) of MFCC_2 across the two frog species, the low SD here indicates low uncertainty in the estimates.
- mu[2] (mean: 0.1654, sd: 0.02056): represents the expected value (mean) of MFCC_11 across the two frog species, the low SD here indicates low uncertainty in the estimates.
- rho (mean: -0.8000, sd: 0.04430): this value suggests a strong negative correlation between MFCC_2 and MFCC_11. This means, the two variables move in opposite directions.
- sigma1 (mean: 0.1497, sd: 0.01242): this represent the standard deviation of MFCC_2, showing a moderate variability in the data.
- sigma2 (mean: 0.1746, sd: 0.01457): this represent the standard deviation of MFCC_11, showing a moderate variability in the data.

The Quantiles in summary allow us to see the credible intervals of our estimates by looking can the difference between the upper and lower quantiles we are interested in:

- the credible intervals shows not too much of uncertainty based on how narrow the intervals are. Furthermore, we can infer from the density plot that most of the estimates density are symmetric, suggesting not too much skewness in the distribution. Lastly, None of the intervals includes 0, suggesting all the effects are significant!

Rerun the model with 3 chains and no initial starting values. Choose the number of steps in the burn-in period and the number of MCMC iterations in a way that you ensure that the effective sample size for ρ is above 2500. Once this is ensured, evaluate, for ρ, μ_1, and μ_2: i) their autocorrelation function estimates, ii) the Gelman-Rubin statistic estimates, and iii) the Gelman-Rubin diagnostic plots. Interpret these plots and values.

```
# using 3 chains
jags_model_chain3 <- jags.model(textConnection(bvn_model), data = data_jags, n.chains = 3)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 73
##    Unobserved stochastic nodes: 4
##    Total graph size: 97
##
## Initializing model
```

```r
#burn in
update(jags_model_chain3, 4000)

# generate 10000 samples and keep every fifth sample
mcmc_samples_chain3 <- coda.samples(jags_model_chain3, variable.names = c("mu", "sigma1", "sigma2", "rho
                                    n.iter = 10000, thin = 5)

print(effectiveSize(mcmc_samples_chain3))
```

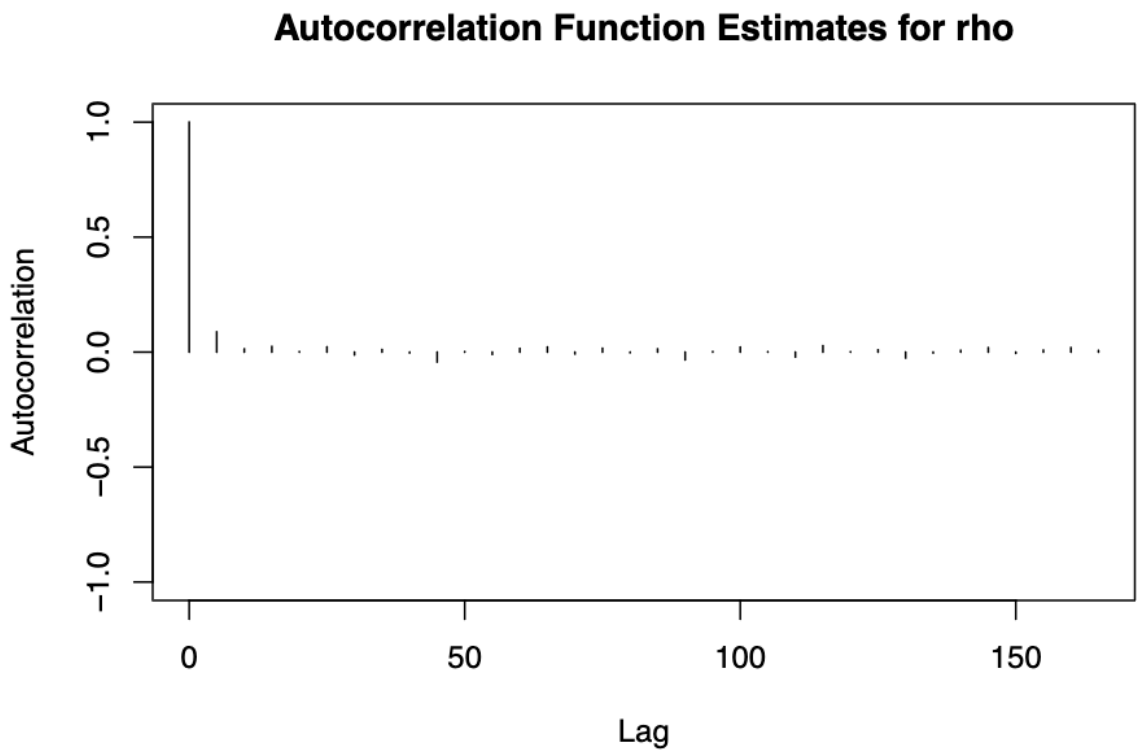
```
##    mu[1]    mu[2]      rho   sigma1   sigma2
## 6000.000 5363.102 4579.151 5232.181 4317.030
```

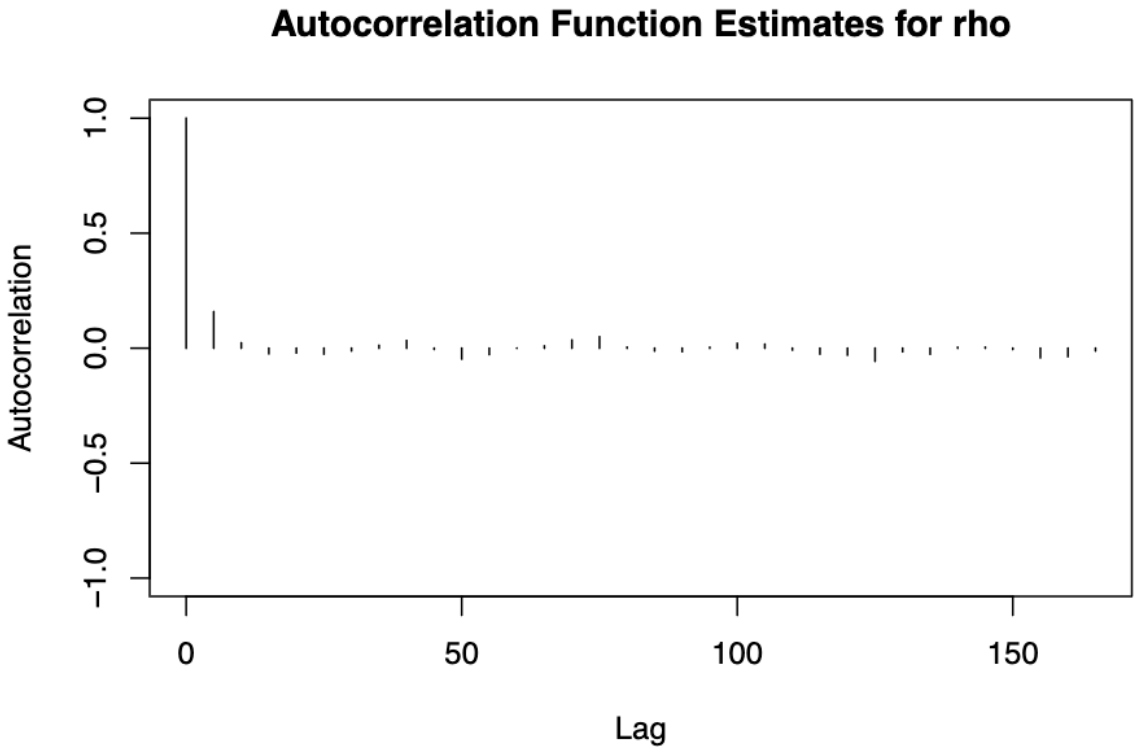
Explanation: (write your explanation here)

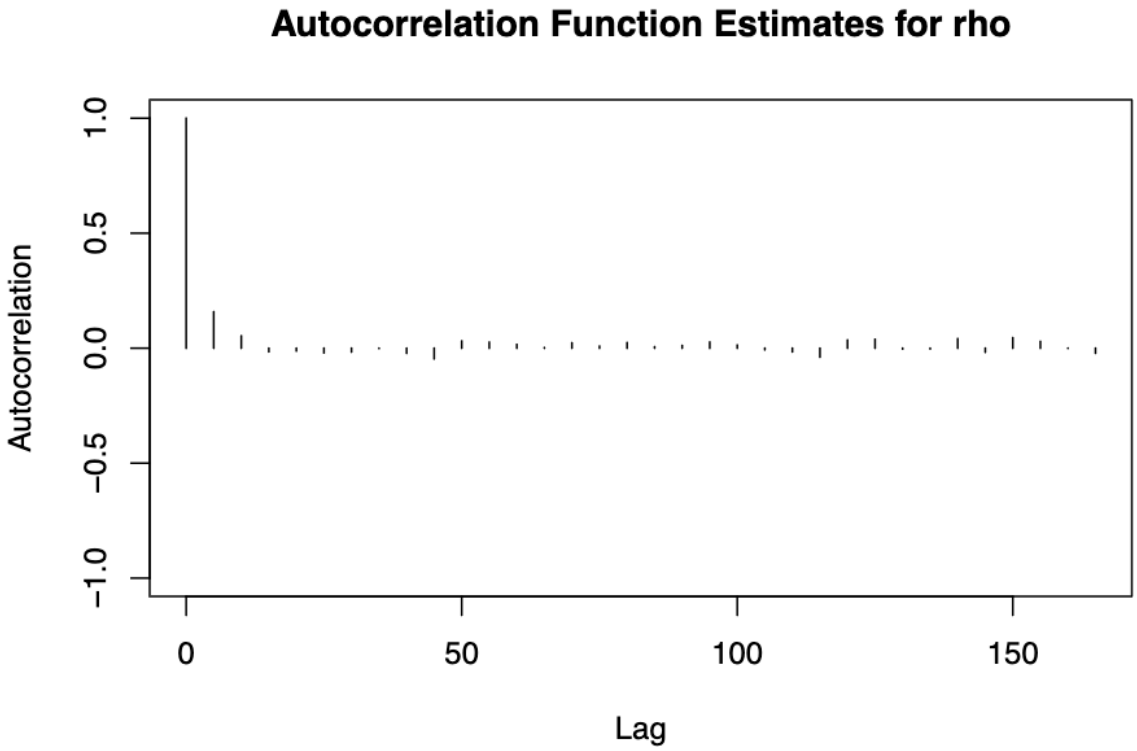
ESS for $\rho \approx 4700$ Iteration number chosen: 10000 Burn in: 4000 Chain: 3

Auto Correlation Function Plot:

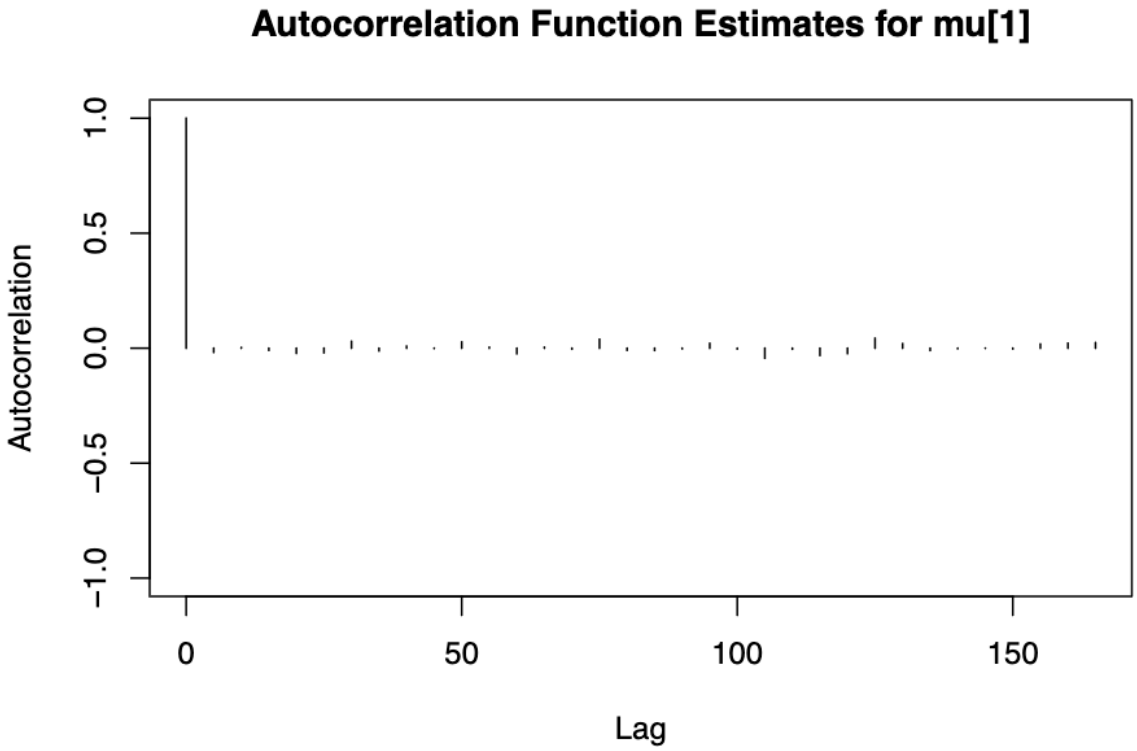
```r
autocorr.plot(mcmc_samples_chain3[, "rho"], main = "Autocorrelation Function Estimates for rho")
```

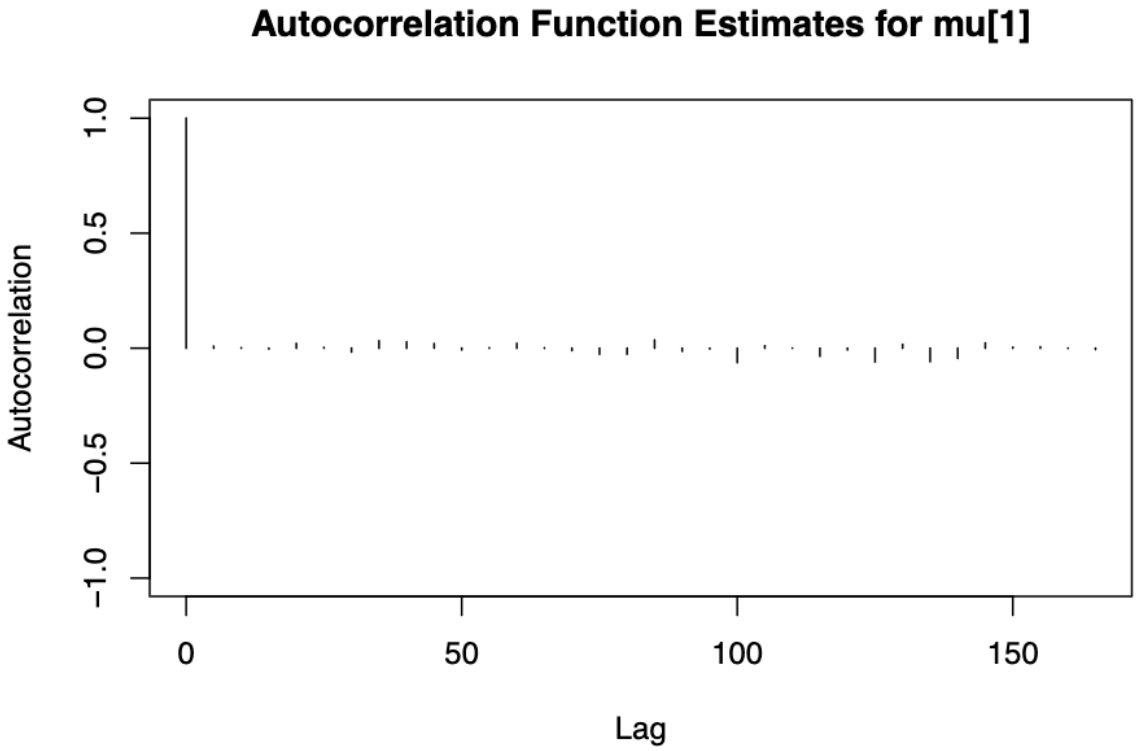


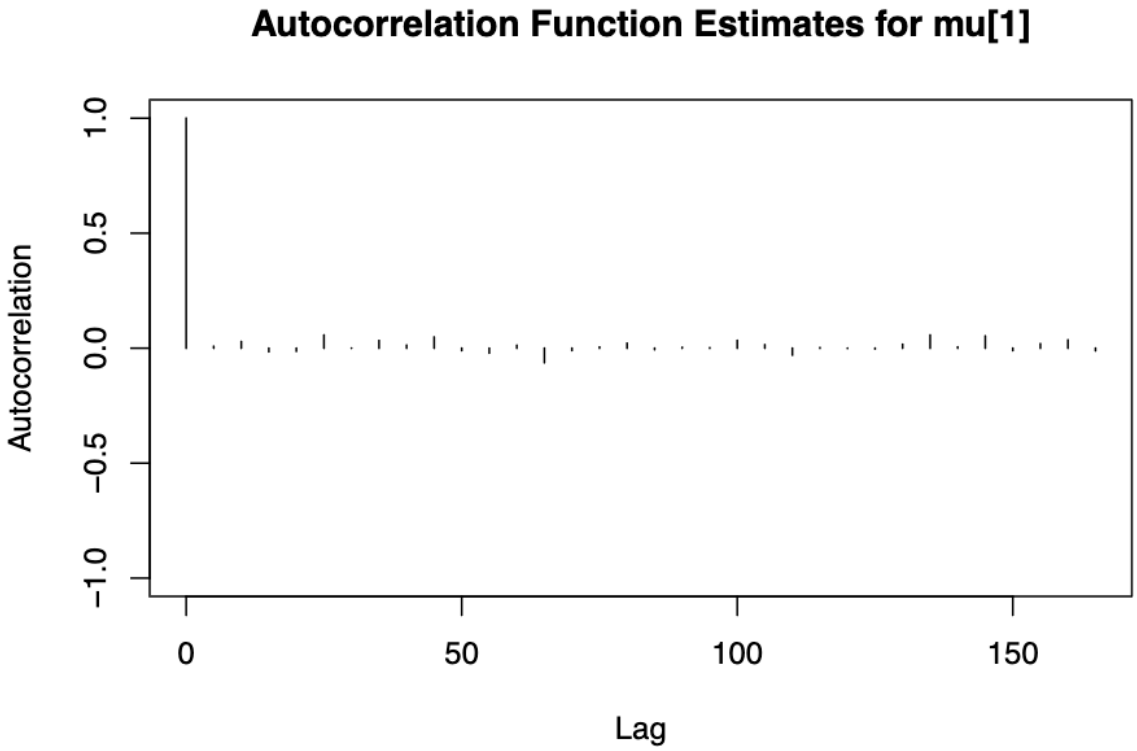

Autocorrelation Function Estimates for rho
Autocorrelation
1.0
0.5
0.0
−0.5
−1.0
0
50
100
150
Lag



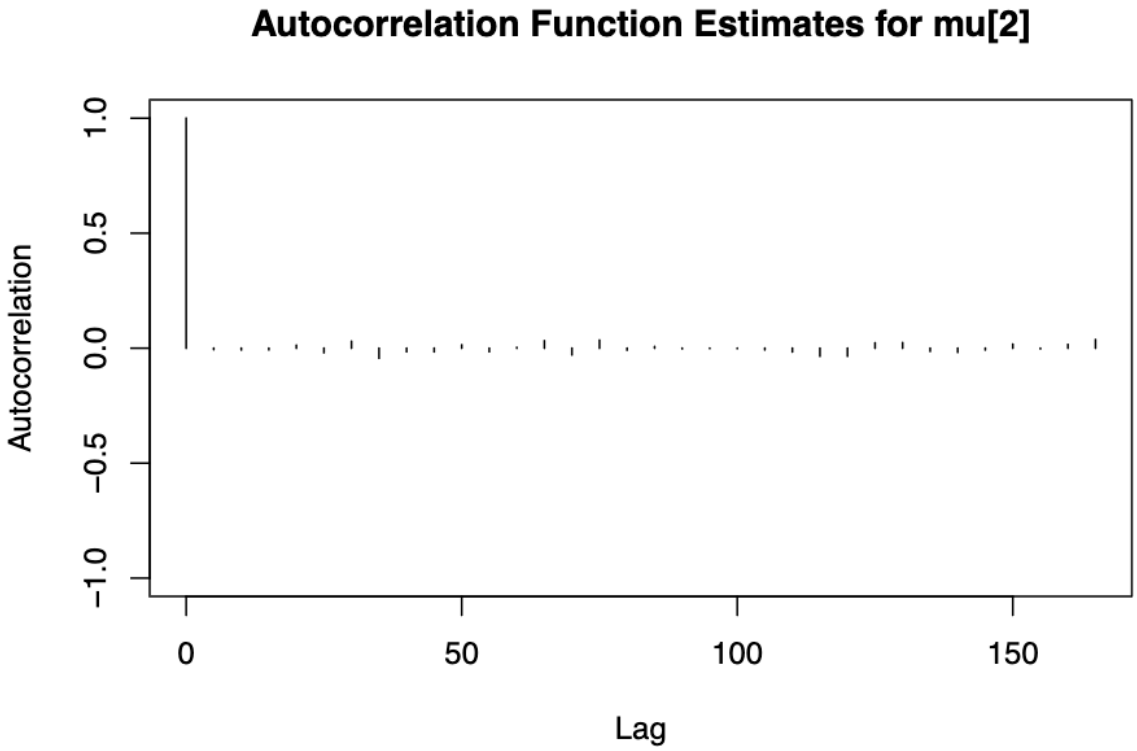

```
autocorr.plot(mcmc_samples_chain3[, "mu[1]"], main = "Autocorrelation Function Estimates for mu[1]")
```

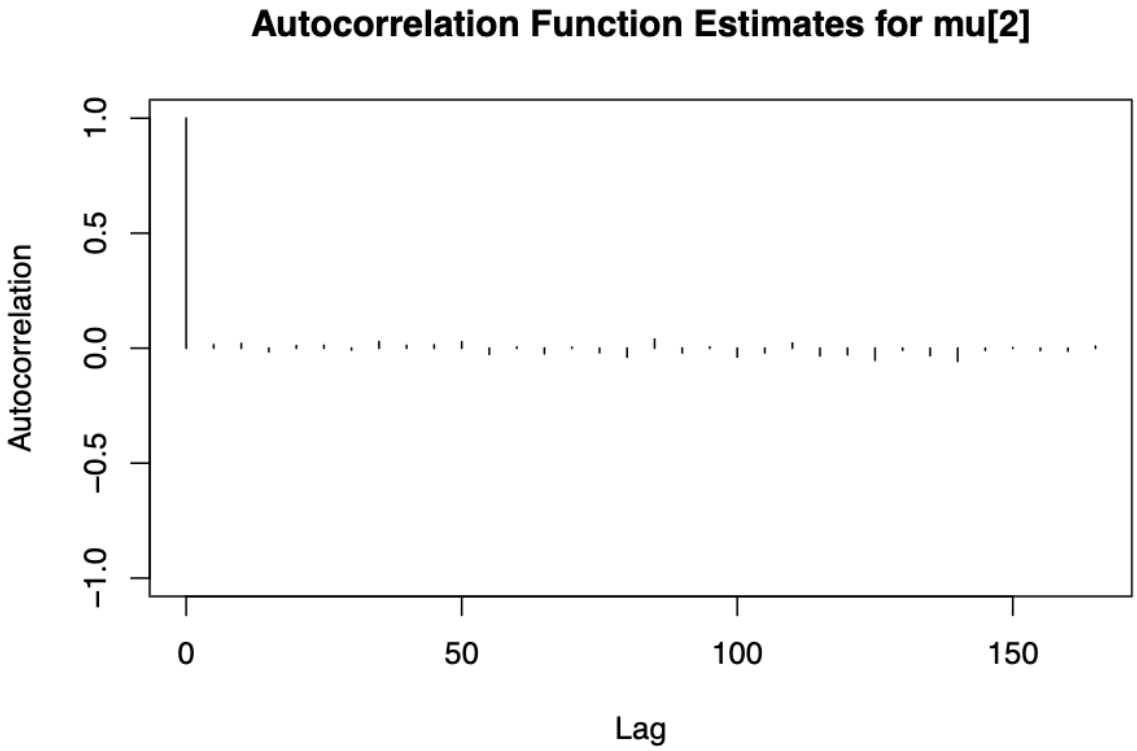

Autocorrelation Function Estimates for mu[1]
Autocorrelation
1.0
0.5
0.0
−0.5
−1.0
0
50
100
150
Lag

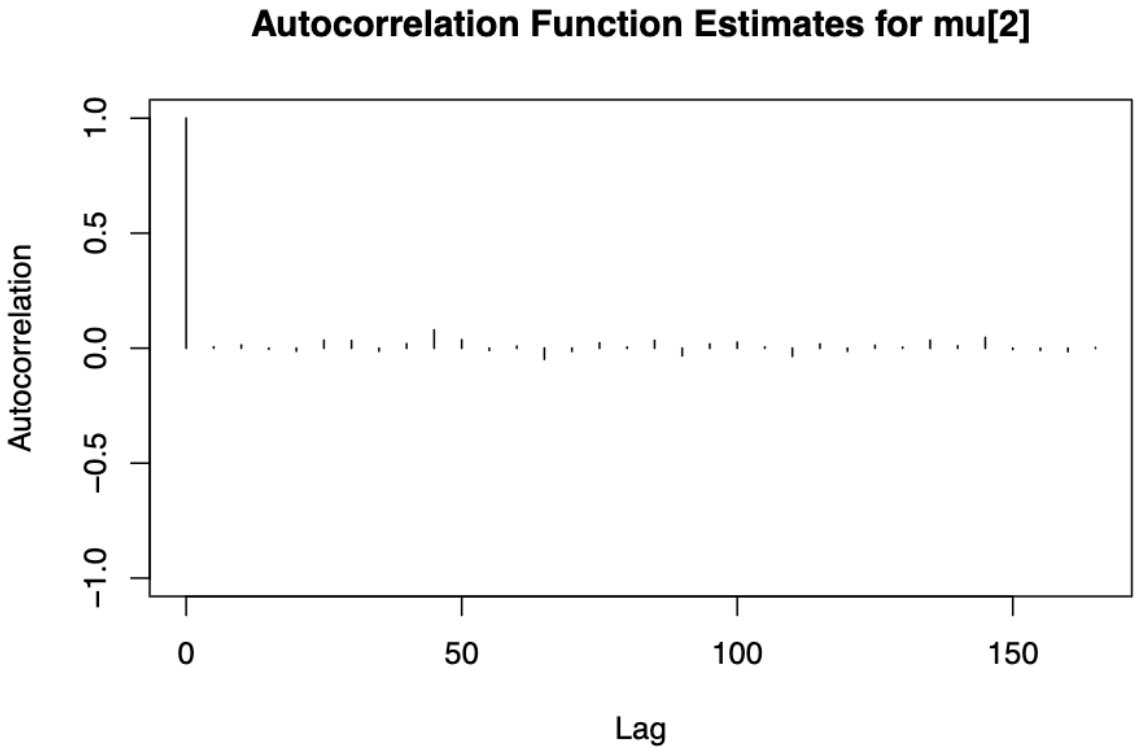

Autocorrelation Function Estimates for mu[1]
Autocorrelation
1.0
0.5
0.0
−0.5
−1.0
0
50
100
150
Lag




```
autocorr.plot(mcmc_samples_chain3[, "mu[2]"], main = "Autocorrelation Function Estimates for mu[2]")
```


Autocorrelation Function Estimates for mu[2]
Autocorrelation
1.0
0.5
0.0
−0.5
−1.0
0
50
100
150
Lag


Autocorrelation Function Estimates for mu[2]
Autocorrelation
1.0
0.5
0.0
−0.5
−1.0
0
50
100
150
Lag



From the autocorrelation function plot of ρ, μ_1, μ_2, the autocorrelation drops quickly to near zero after within a few lags, suggesting that the chain is mixing well and that the MCMC samples are relatively independent.

Gelman-Rubin statitic estimates:

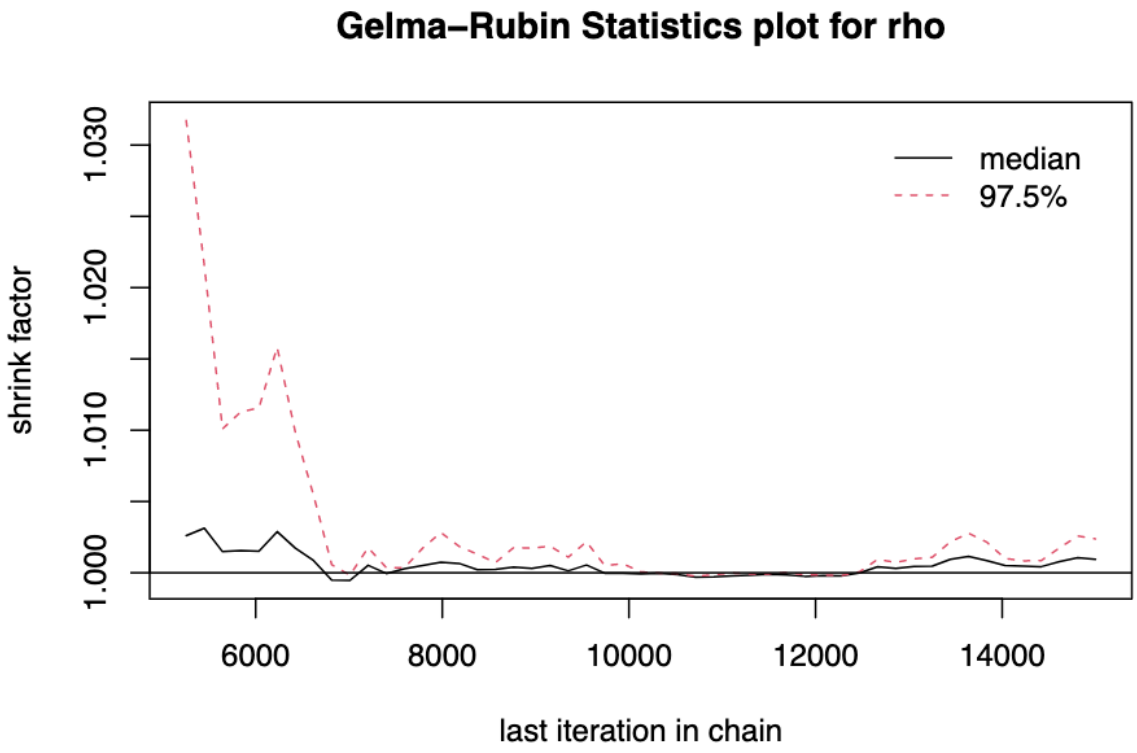
```
gelman.diag(mcmc_samples_chain3)
```

```
## Potential scale reduction factors:
##
##        Point est. Upper C.I.
## mu[1]           1       1.00
## mu[2]           1       1.00
## rho             1       1.00
## sigma1          1       1.01
## sigma2          1       1.00
##
## Multivariate psrf
##
## 1
```

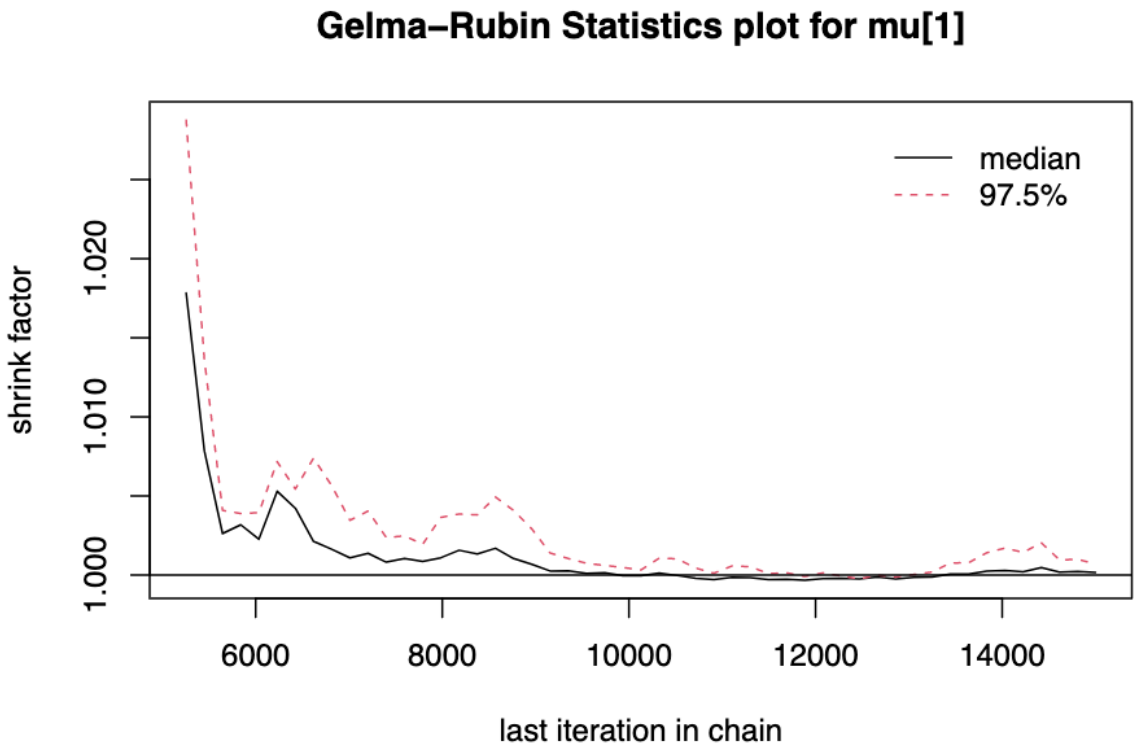
From the Gelman-Rubin statistics, we can see that the point estimate of ρ, μ_1, μ_2 are all 1 with the Upper C.I. being all 1 too, this indicates a good convergence in our chain and that our iteration + burn in number is sufficient.

Gelman-Rubin diagnostic plot:

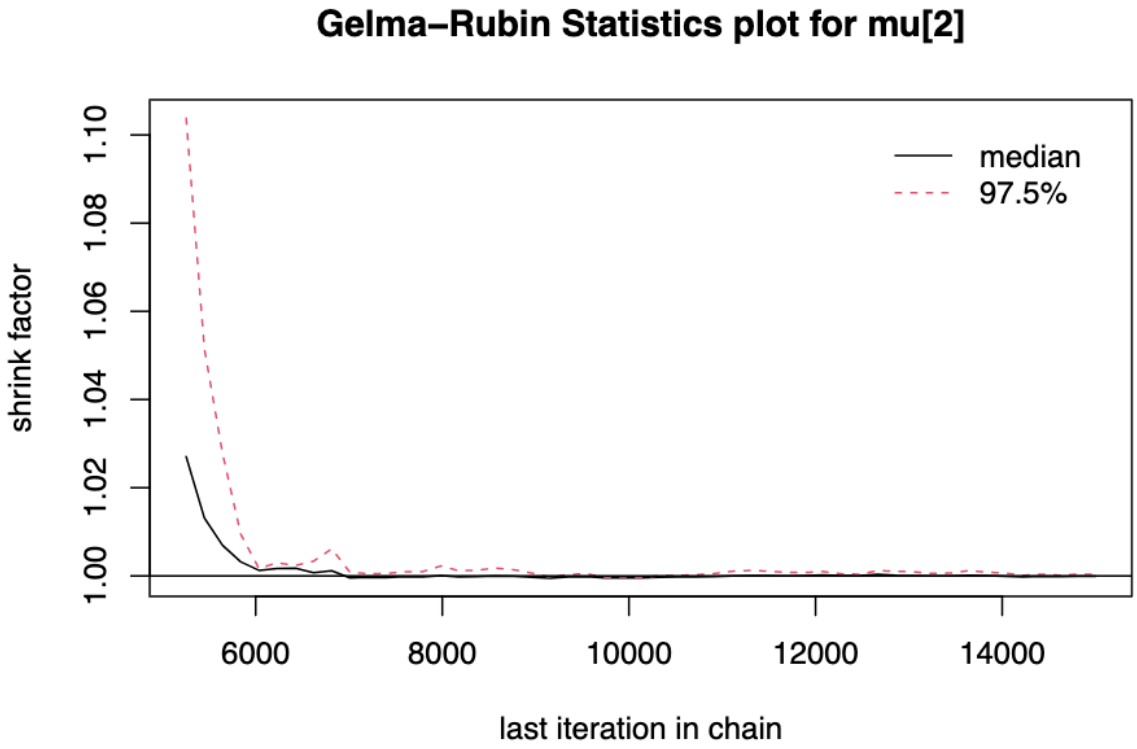
```
gelman.plot(mcmc_samples_chain3[, "rho"], main = "Gelma-Rubin Statistics plot for rho")
```




```
gelman.plot(mcmc_samples_chain3[, "mu[1]"], main = "Gelma-Rubin Statistics plot for mu[1]")
```




```
gelman.plot(mcmc_samples_chain3[, "mu[2]"], main = "Gelma-Rubin Statistics plot for mu[2]")
```



All 3 Gelman-Rubin diagonsitc plots shows that the shrink factor / PSFR drops to below 1.01 after a certain iterations, this indicate that there is no evidence that all 3 parameters have not converge.

Q2.2) [15 marks] We are now going to fit a bivariate normal mixture model. For K mixture components, our model will have the form:

$$\begin{aligned}
&\mathbf{Y}_i|(Z_i = k) \sim BVN(\boldsymbol{\mu}^{(k)}, \Sigma) \\
&\Pr\{Z_i = k\} = p_{i,k}, \quad \mu_1^{(k)} \sim N(0, 100^2), \quad \mu_2^{(k)} \sim N(0, 100^2), \quad k = 1, \ldots, K, \\
&\sum_{k=1}^{K} p_{i,k} = 1, \quad \text{for all } i = 1, \ldots, N, \\
&\Sigma_{11} = \sigma_1^2, \quad \Sigma_{22} = \sigma_2^2, \quad \Sigma_{12} = \Sigma_{21} = \sigma_1 \sigma_2 \rho, \\
&\rho \sim \text{Unif}(-1, 1), \quad \sigma_1^2, \sigma_2^2 \sim \text{Inverse-Gamma}(1, 0.1),
\end{aligned}$$

where all $\mu_k^{(k)}, j = 1, 2, k = 1, \ldots, K$ are independent a priori.

The latent cluster label Z_i tells us which cluster observation Y_i belongs to. Note that the mean vector of the bivariate normal distribution changes with the cluster, but the covariance matrix is the same for all clusters.

Write a JAGS model for a bivariate normal mixture with $K = 2$ components. Note that your model will suffer from label switching if you do not impose any constraints on the parameters see this article. This can easily be circumvented by ordering the first component of the mean vector, i.e., by forcing $\mu_1^{(1)} \leq \mu_1^{(2)} \leq \cdots \leq \mu_1^{(k)}$. [hint 1: generate $\mu_1^{(1)}, \ldots, \mu_1^{(k)}$ from their priors, then just `sort` the values.][hint 2: you can generate the cluster labels Z_i using `dcat`]

Fit this model to the same data as in Q2.1). Scale the data before fitting the model. Use a Beta(0.5,0.5) prior on $p_{1,i}, i = 1, \ldots, N$; note that $p_{i,2} = 1 - p_{i,1}$. Place the same priors on σ_1^2, σ_2^2,

and ρ as in Q2.1). Use 5000 burn-in samples to update the model, with one chain.

```
bmm_model <- "model{
  for(i in 1:n){
    z[i] ~ dcat(p[])

    Y[i, 1:2] ~ dmnorm(mu[z[i], ], tau[,])
  }

  for(j in 1:2){
  mu_r[j, 1] ~ dnorm(0, 1/100^2)
  mu_r[j, 2] ~ dnorm(0, 1/100^2)

  }
  #sort mu_1
  mu_sort[1:2, 1] <- sort(mu_r[1:2, 1])

  for(k in 1:2){
  mu[k,1] <- mu_sort[k, 1]
  mu[k,2] <- mu_r[k,2]
  }

  #construct covariance matrix
  rho ~ dunif(-1, 1)
  tau1 ~ dgamma(1, 0.1)
  tau2 ~ dgamma(1, 0.1)

  sigma1 <- 1 / sqrt(tau1)
  sigma2 <- 1 / sqrt(tau2)

  Sigma[1,1] <- 1 / tau1
  Sigma[2,2] <- 1 / tau2
  Sigma[1,2] <- rho * sigma1 * sigma2
  Sigma[2,1] <- Sigma[1,2]

  tau[1:2, 1:2] <- inverse(Sigma[,])

  p[1] ~ dbeta(0.5,0.5)
  p[2] <- 1-p[1]
}"

scaled_subset_data <- scale(subset_data)

data_mixture_K2 <- list(n = nrow(scaled_subset_data),
                  Y = cbind(scaled_subset_data[,"MFCCs_.2"], scaled_subset_data[,"MFCCs_11"]))

jags_model_mixture_2 <- jags.model(textConnection(bmm_model), data = data_mixture_K2)
```

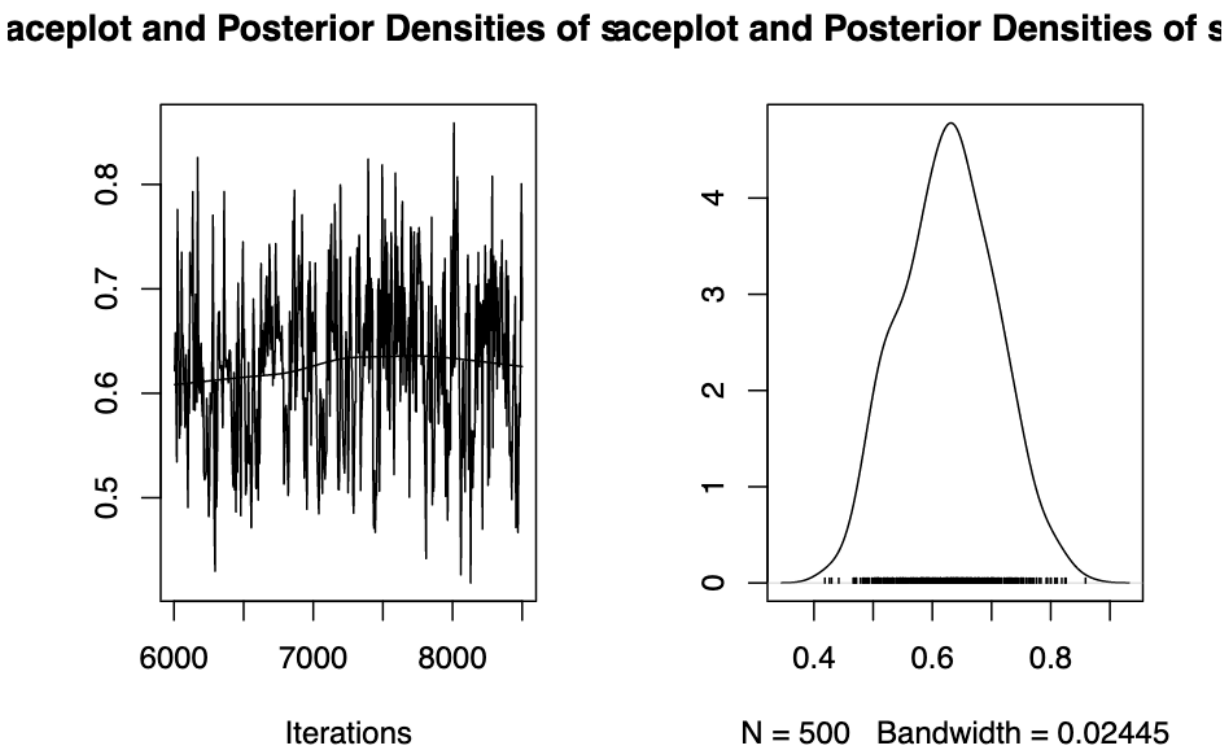
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 73
##    Unobserved stochastic nodes: 81
```

```
##    Total graph size: 254
##
## Initializing model
```

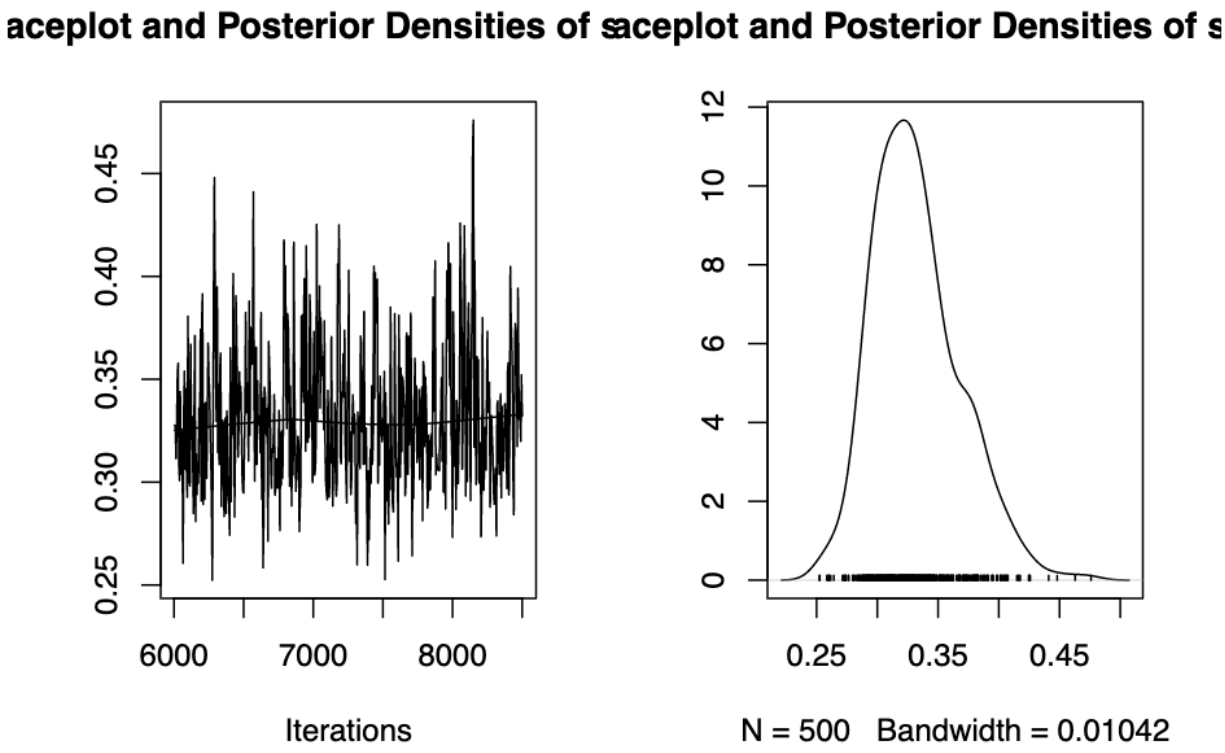
```
#burn in
update(jags_model_mixture_2, 5000)
```

Generate 2500 samples with thin=5. Plot the traceplots and posterior densities for the standard deviations and correlation parameter of the bivariate normal mixture components.

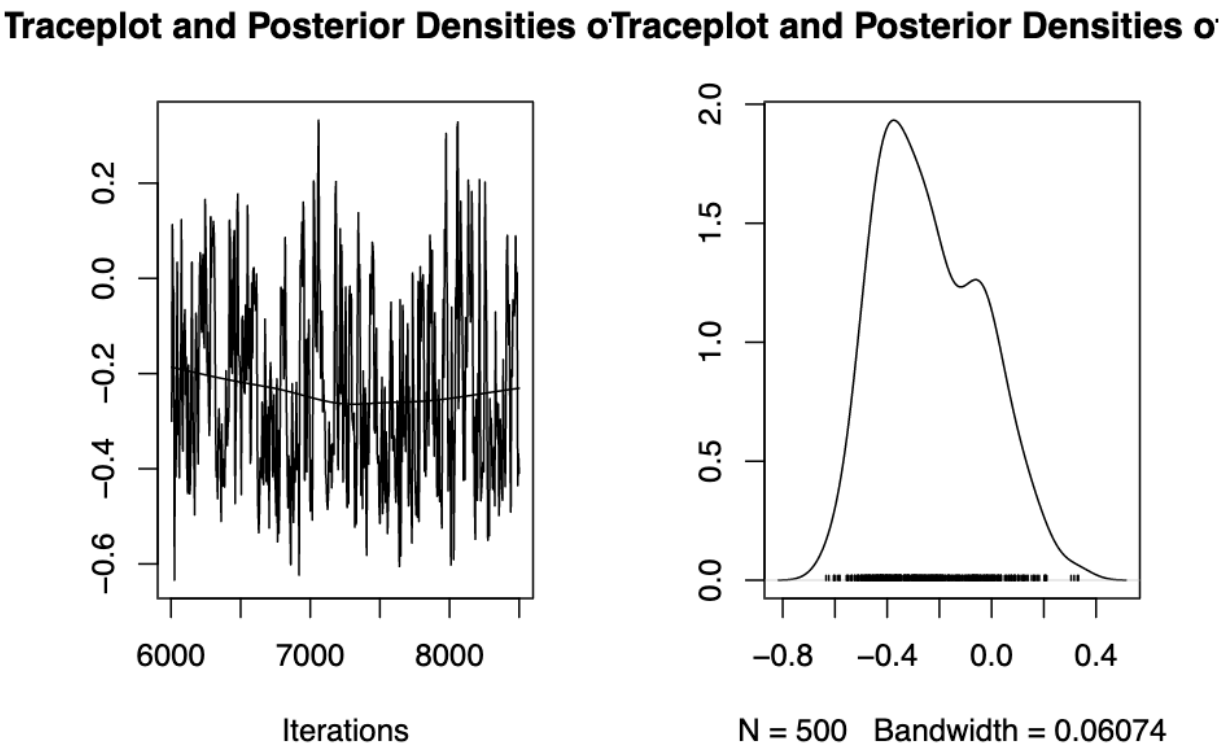
```
# generate 2500 samples and keep every fifth sample
mcmc_samples_mixture_K2 <- coda.samples(jags_model_mixture_2,
                                        variable.names = c("mu", "sigma1", "sigma2", "rho", "p", "z"),
                                        n.iter = 2500, thin = 5)

plot(mcmc_samples_mixture_K2[, 'sigma1'], main = "Traceplot and Posterior Densities of sigma1")
```




```
plot(mcmc_samples_mixture_K2[, 'sigma2'], main = "Traceplot and Posterior Densities of sigma2")
```

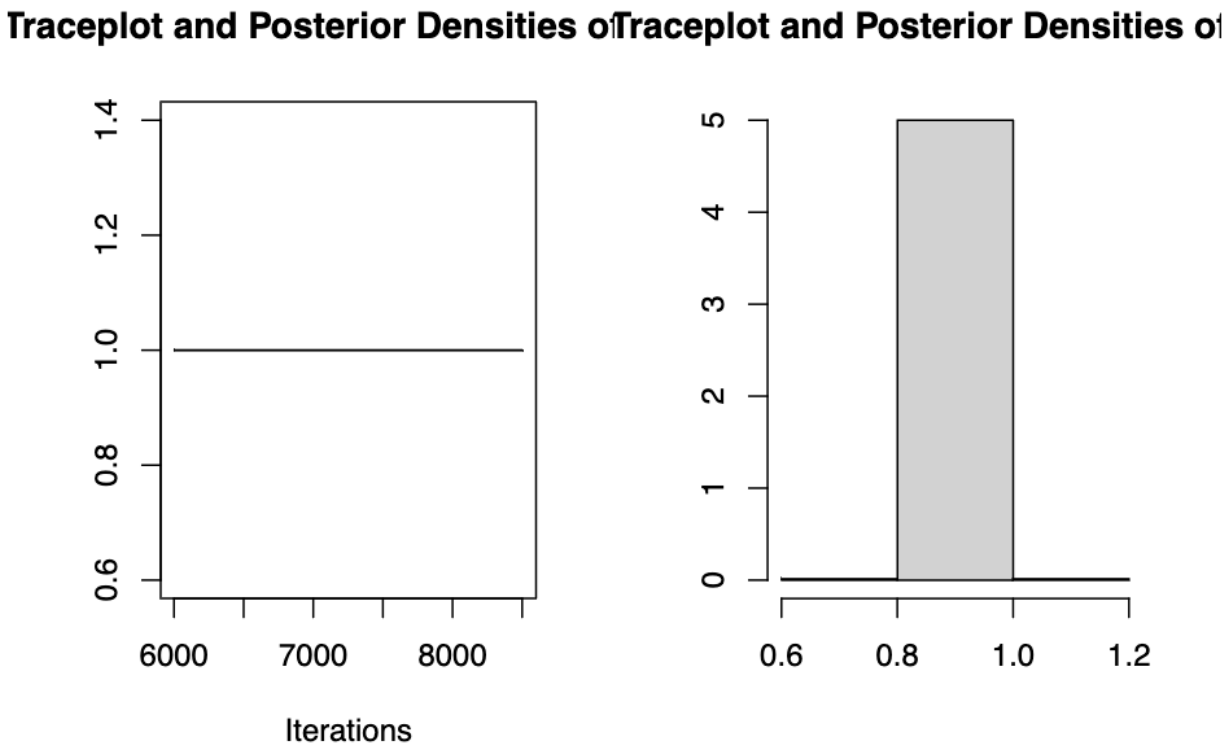



```
plot(mcmc_samples_mixture_K2[, 'rho'], main = "Traceplot and Posterior Densities of rho")
```

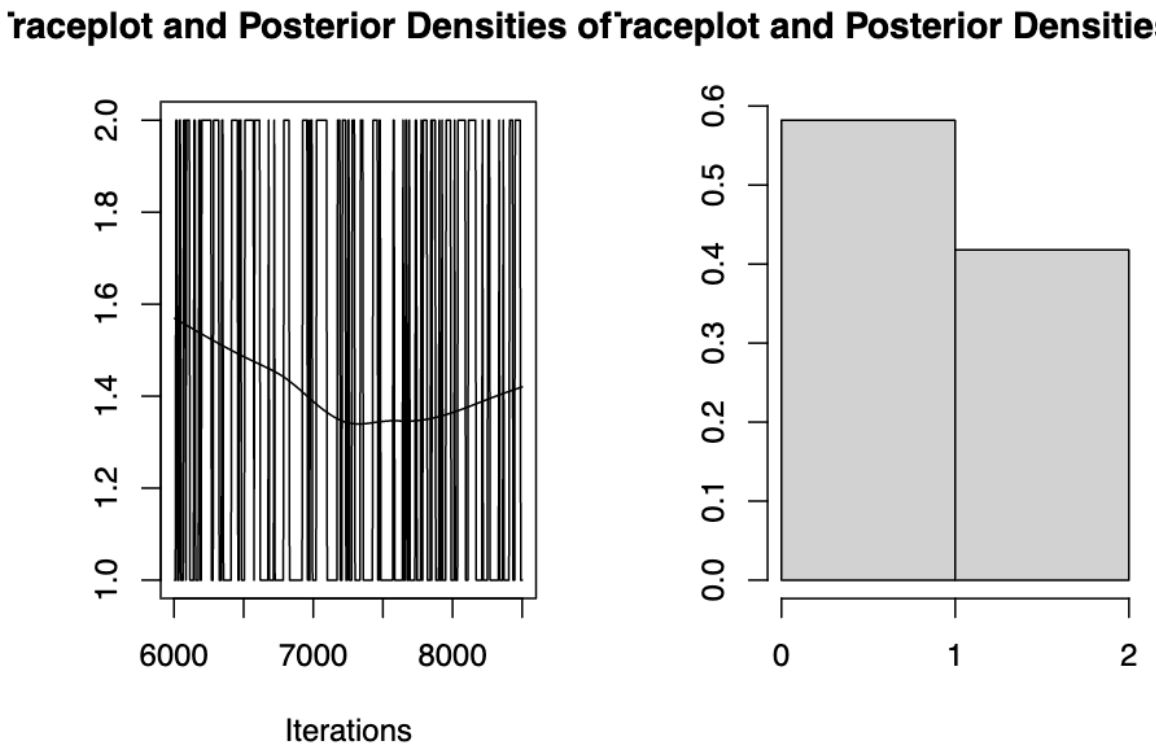


Plot the traceplots and posterior densities for latent label components Z_1 and Z_{48}, and the corresponding cluster probabilities, $p_{1,1}$ and $p_{48,2}$. Interpret you posterior density estimates.

```
plot(mcmc_samples_mixture_K2[, 'z[1]'], main = "Traceplot and Posterior Densities of Z_1")
```




```
plot(mcmc_samples_mixture_K2[, 'z[48]'], main = "Traceplot and Posterior Densities of Z_48")
```




```
# cluster probability of data 1 for cluster 1
p1_1 <- mean(as.matrix(mcmc_samples_mixture_K2[,'z[1]']) == 1)

# cluster probability of data 48 for cluster 2
p48_2 <- mean(as.matrix(mcmc_samples_mixture_K2[,'z[48]']) == 2)

cat(sprintf("Cluster probability of data point 1 in cluster 1: %.4f\n", p1_1))
```

```
## Cluster probability of data point 1 in cluster 1: 1.0000
```

```
cat(sprintf("Cluster probability of data point 48 in cluster 2: %.4f\n", p48_2))
```

```
## Cluster probability of data point 48 in cluster 2: 0.4180
```

Explanation: (write your explanation here)

The posterior densities of Z_1 and Z_48 shows the probability that each observation are belonging to each cluster. For both Z_1 and Z_48, we see that the probability are much higher for cluster 2, indicating that observation 1 and 48 belongs to cluster 2.

Furthermore, the cluster probability for $p_{1,1}$ and $p_{48,2}$ are 0.0080 and 1.000, suggesting that both observation 1 and 48 are both more suited to cluster 2 (or that the model thinks cluster 2 suits them more).

The density Q2.3) [20 marks] **Adapt your JAGS code for a general $K = K$ mixture model. You should write your code so that K can be supplied as an argument to data. Use an appropriate**

prior on $(p_{i,1}, p_{i,2}, \ldots, p_{i,k})$ to ensure they sum to one. You may choose your own priors for the other model parameters.

Include (in your dataset from Q2.1 and Q2.2) the `MFCCs_.2` and `MFCCs_11` coefficients for frog 56. Fit three mixture models to these data, with number of mixture components $K = 2$, $K = 3$, and $K = 4$. Use 10000 burn-in iterations and two chains.

```
#general K model
bmm_model_general <- "model{
  for(i in 1:n){
    z[i] ~ dcat(p[])
    Y[i, 1:2] ~ dmnorm(mu[z[i], ], tau[,])
  }

  for(j in 1:K){
  mu_r[j, 1] ~ dnorm(0, 1/100^2)
  mu_r[j, 2] ~ dnorm(0, 1/100^2)
  }
  #sort mu_1
  mu_sort[1:K, 1] <- sort(mu_r[1:K, 1])

  for(k in 1:K){
  mu[k,1] <- mu_sort[k, 1]
  mu[k,2] <- mu_r[k,2]
  }

  #construct covariance matrix
  rho ~ dunif(-1, 1)
  tau1 ~ dgamma(1, 0.1)
  tau2 ~ dgamma(1, 0.1)

  sigma1 <- 1 / sqrt(tau1)
  sigma2 <- 1 / sqrt(tau2)

  Sigma[1,1] <- 1 / tau1
  Sigma[2,2] <- 1 / tau2
  Sigma[1,2] <- rho * sigma1 * sigma2
  Sigma[2,1] <- Sigma[1,2]

  tau[1:2, 1:2] <- inverse(Sigma[,])

  #dirichlet dist with alpha 1
  p[1:K] ~ ddirich(rep(1,K))
}"


subset_data2 <- frogs %>% filter(RecordID %in% c(1,46, 56)) %>%
  select(RecordID, MFCCs_.2, MFCCs_11)
scaled_subset_data2 <- scale(subset_data2)

# fit K = 2,3,4
data_mixture_K2 <- list(n = nrow(scaled_subset_data2),
                  Y = cbind(scaled_subset_data2[,"MFCCs_.2"], scaled_subset_data2[,"MFCCs_11"]),
                  K = 2)
data_mixture_K3 <- list(n = nrow(scaled_subset_data2),
```

```
                  Y = cbind(scaled_subset_data2[,"MFCCs_.2"], scaled_subset_data2[,"MFCCs_11"]),
                  K = 3)

data_mixture_K4 <- list(n = nrow(scaled_subset_data2),
                  Y = cbind(scaled_subset_data2[,"MFCCs_.2"], scaled_subset_data2[,"MFCCs_11"]),
                  K = 4)

#set.seed jags
inits_seed = list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 123)

jags_model_mixture_K2 <- jags.model(textConnection(bmm_model_general),
                                    data = data_mixture_K2, n.chains = 2,
                                    inits = inits_seed)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 119
##    Unobserved stochastic nodes: 127
##    Total graph size: 391
##
## Initializing model
```

```
jags_model_mixture_K3 <- jags.model(textConnection(bmm_model_general),
                                    data = data_mixture_K3, n.chains = 2,
                                    inits = inits_seed)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 119
##    Unobserved stochastic nodes: 129
##    Total graph size: 395
##
## Initializing model
```

```
jags_model_mixture_K4 <- jags.model(textConnection(bmm_model_general),
                                    data = data_mixture_K4, n.chains = 2,
                                    inits = inits_seed)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 119
##    Unobserved stochastic nodes: 131
##    Total graph size: 399
##
## Initializing model
```

```
#burn in
update(jags_model_mixture_K2, 10000)
update(jags_model_mixture_K3, 10000)
update(jags_model_mixture_K4, 10000)
```

Evaluate the DIC for all three model fits, using 5000 samples. Which model fits the data better?

```
set.seed(123)
#k = 2 dic
dic_bmm_K2 <- dic.samples(jags_model_mixture_K2,n.iter = 5000)
dic_bmm_K2 #368.4
```

```
## Mean deviance:  368.4
## penalty 0
## Penalized deviance: 368.4
```

```
#k=3 dic
dic_bmm_K3 <- dic.samples(jags_model_mixture_K3,n.iter = 5000)
dic_bmm_K3 #262.6
```

```
## Mean deviance:  262.6
## penalty 0
## Penalized deviance: 262.6
```

```
#k=4 dic
dic_bmm_K4 <- dic.samples(jags_model_mixture_K4,n.iter = 5000)
dic_bmm_K4 #137.9
```

```
## Mean deviance:  137.9
## penalty 0
## Penalized deviance: 137.9
```

Explanation: (write your explanation here)

The DIC is a model selection criterion which indicate how good our model is fitting the data with an additional penalized term due to the complexity of the parameter. A lower DIC value indicate a "better" overall fit in terms of model complexity and general fit.

In our case:

The DIC for K = 2 is 368.4, for K = 3 is 262.6 , and for K = 4 is 137.9. This suggests that K = 4 is our preferred model since it has the lwoest DIC of all model we compared.

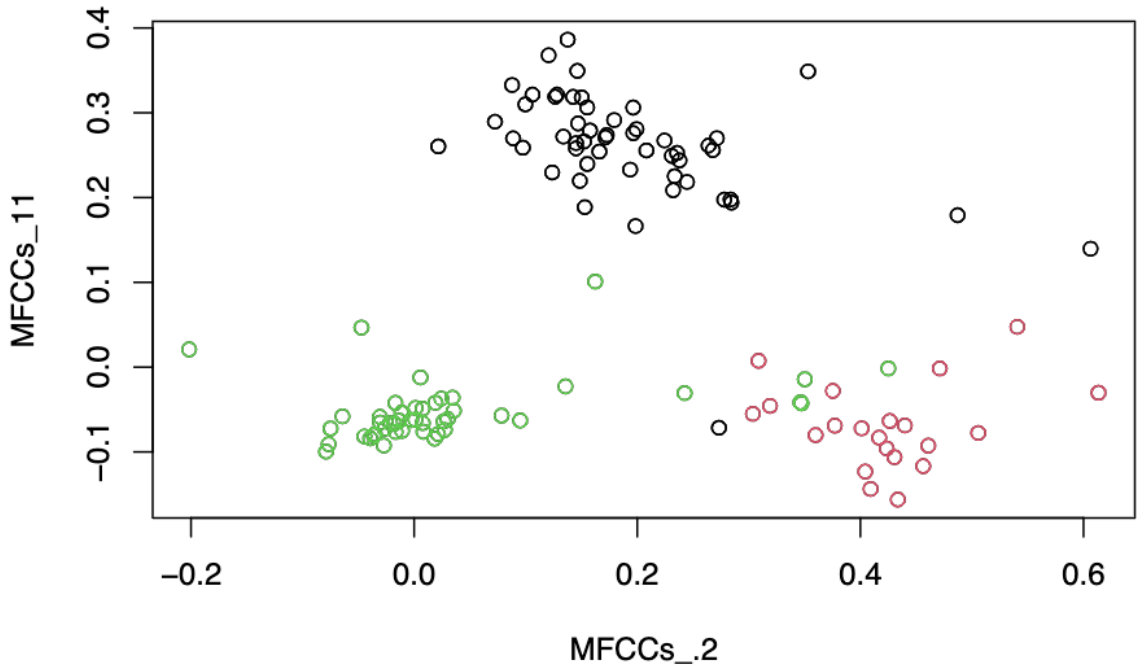
Generate 2500 samples of the latent cluster labels from your preferred model. Evaluate the posterior mode of the latent cluster variable for each syllable.

```
z_samples_mixture_K4 <- coda.samples(jags_model_mixture_K4, variable.names = c("z"),n.iter = 2500)

z_samples_mat <- as.matrix(z_samples_mixture_K4)
#mode function
Mode <- function(x) {
  ux <- unique(x)
```

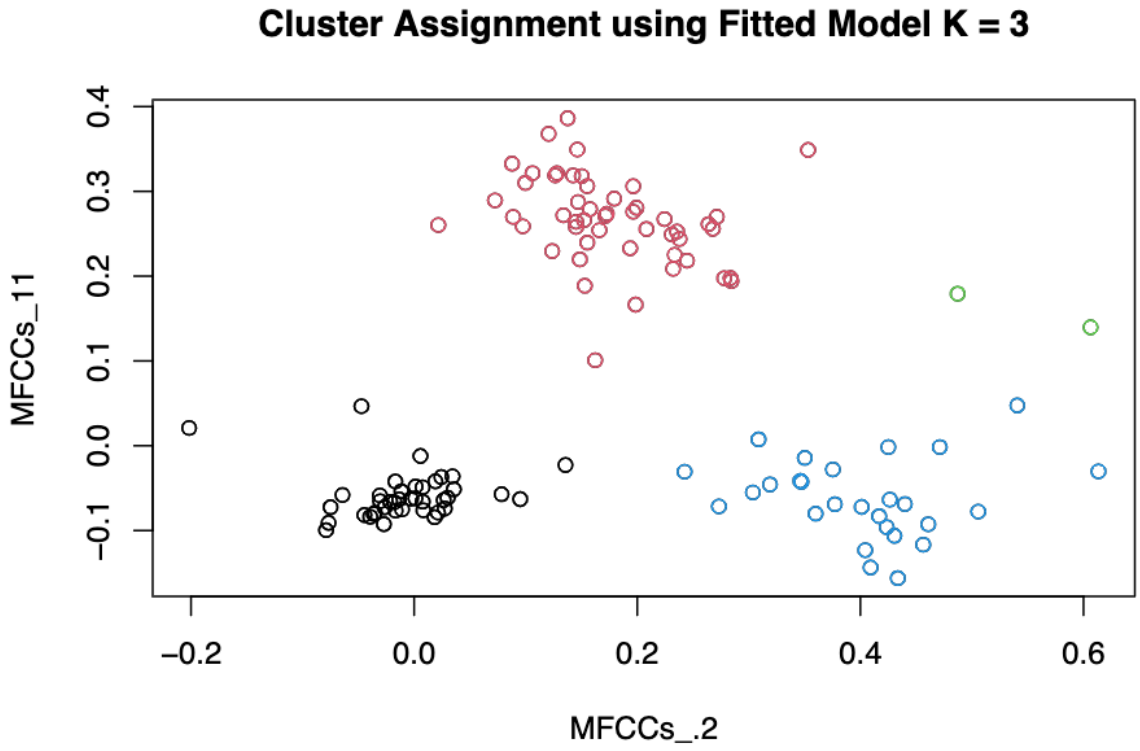
```
  ux[which.max(tabulate(match(x, ux)))]
}

mode_obs <- numeric(ncol(z_samples_mat))
for(i in 1:ncol(z_samples_mat)){
  mode_obs[[i]] <- Mode(z_samples_mat[,i])
}
mode_obs
```

```
##   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [38] 2 2 2 2 2 2 2 2 2 3 3 2 2 2 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2
##  [75] 1 1 1 1 4 1 1 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 4 1 1 1 1 1 4 1 1
## [112] 1 1 1 1 1 1 1 4
```

Below is a plot of your data, with each point coloured according to the true frog label (black = 1, red = 46, green = 56). Produce a similar plot, but with the points instead coloured by the posterior mode of the cluster label from your fitted model; use `col=1` up until 'col=K', where K is the optimal number of clusters. Interpret the estimates on these plots, and comment on the efficacy of your clustering model (relative to the true clusters).




```
plot(subset.data, main = "Cluster Assignment using Fitted Model K = 3")
points(subset.data, col=mode_obs)
```



Explanation: (write your explanation here)

From the cluster assignment plot, we notice that there are 4 cluster assignments instead of 3 frog type in our data set, however the 4th cluster seems to only consist of a small amount data. Overall, the mixture model were able to capture the 3 frog type with the black dot representing frog 56, red dot representing frog 1, while the blue dot represent frog 46. Furthermore, we notice that in the true cluster assignment, there are some of frog 1 and frog 56 that are in the frog 46 cluster and in our model it is classified as the frog 46 (blue dot), this might be due to the fact that these frogs have similar characteristic to frog 46 as seen from their data point location in the graph, hence our model classify it as frog 46. Overall, the mixture model were able to capture the underlying data type with a relative minor error, this might be due to the frog inherent characteristic.