**University of Edinburgh**

**School of Mathematics**

**Bayesian Data Analysis, 2024/2025, Semester 2**

**Assignment 1**

```
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:

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

```
##   rainfall windspeed temperature     year
## 1 1.239984  1.313964    272.0122 1940.000
## 2 2.902047  1.651040    275.0046 1940.083
## 3 2.412593  2.863953    277.3972 1940.167
## 4 3.493554  1.250086    280.3240 1940.250
## 5 1.249396  1.378372    284.0099 1940.333
## 6 1.388603  2.024482    287.8907 1940.417
```

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.**

```
library(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
```

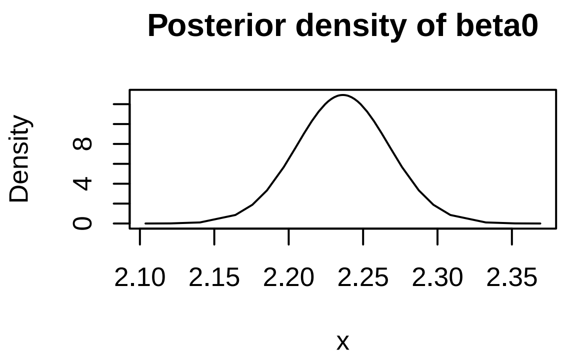
```
library(dplyr)
```

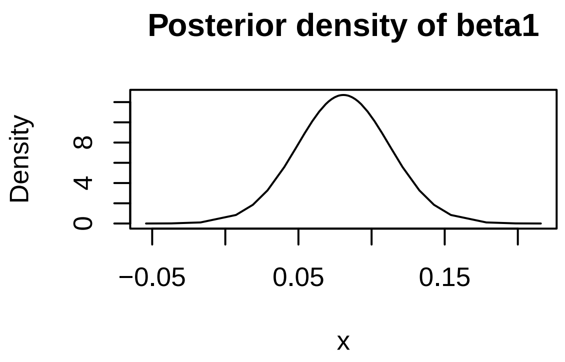
```
##
## Attaching package: 'dplyr'

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

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

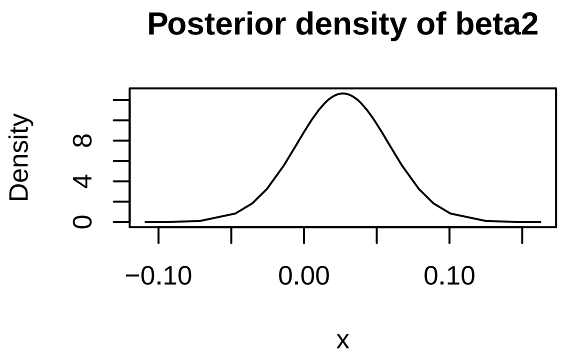
```
# Data preprocessing
Edi.rain_new <- na.omit(Edi.rain)
Edi.rain_new <- Edi.rain_new %>%
  mutate(
    windspeed_scale = scale(windspeed),
    temperature_scale = scale(temperature),
    year_scale = scale(year)
  )
rainfall <- Edi.rain_new$rainfall
windspeed_scale <- Edi.rain_new$windspeed_scale
temperature_scale <- Edi.rain_new$temperature_scale
year_scale <- Edi.rain_new$year_scale
# Fit model
prec.prior <- list(prec=list(prior = "loggamma", param = c(1, 0.01)))
prior.beta <- list(
  mean.intercept = 0, prec.intercept = 0.0001,
  mean = 0, prec =  0.0001
)
data=data.frame(rainfall,windspeed_scale,temperature_scale,year_scale)
fomular <- rainfall ~ windspeed_scale + temperature_scale + year_scale
m1I <- inla(fomular, data = data,
            control.family = list(hyper = prec.prior),
            control.fixed = prior.beta,
            control.compute = list(config = TRUE))
summary(m1I)
```

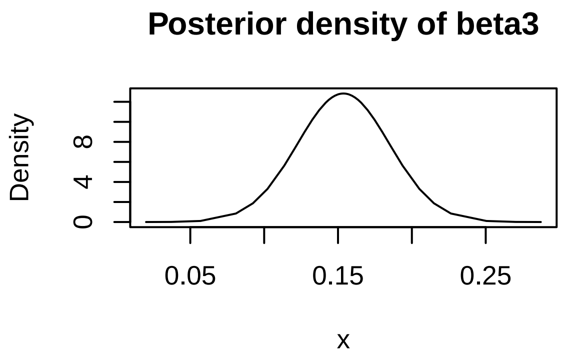
```
## Time used:
##     Pre = 0.645, Running = 1.26, Post = 0.0216, Total = 1.93
## Fixed effects:
##                    mean    sd 0.025quant 0.5quant 0.975quant  mode kld
## (Intercept)       2.236 0.031      2.176    2.236      2.297 2.236   0
## windspeed_scale   0.081 0.032      0.019    0.081      0.143 0.081   0
## temperature_scale 0.027 0.032     -0.035    0.027      0.089 0.027   0
## year_scale        0.154 0.031      0.092    0.154      0.215 0.154   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
##
## Marginal log-Likelihood:  -1406.36
##  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)')
```

```
#Plot
par(mfrow = c(2,2))
plot(m1I$marginals.fixed$`(Intercept)`, type ="l",xlab="x",ylab="Density",
main='Posterior density of beta0')
plot(m1I$marginals.fixed$windspeed_scale, type ="l",xlab="x",ylab="Density",
main='Posterior density of beta1')
plot(m1I$marginals.fixed$temperature_scale, type ="l",xlab="x",ylab="Density",
main='Posterior density of beta2')
plot(m1I$marginals.fixed$year_scale, type ="l",xlab="x",ylab="Density",
main='Posterior density of beta3')
```









**Explanation**: In this question, we fit a Bayesian linear regression model to the Edinburgh rainfall dataset using INLA, with daily rainfall total as the response and average windspeed, average temperature, and year as covariates. We remove any rows containing missing values and scale all continuous covariates, storing the cleaned and scaled data in the new data frame *Edi.rain_new*.

For the Gaussian error variance $\sigma^2$, we assume an InvGamma$(\alpha, \beta)$ prior. Defining $\tau = 1/\sigma^2$ implies $\sigma^2 = 1/\tau$, and hence $\tau \sim \text{Gamma}(\alpha, \beta)$. We then specify:

$$\beta_j \sim N(0,\ 100^2) \quad \text{and} \quad \tau \sim \text{Gamma}(1,\ 0.01).$$

The Bayesian linear regression model fitted via INLA is:

$$\text{rainfall} \ = \ \beta_0 \ + \ \beta_1\,\text{windspeed} \ + \ \beta_2\,\text{temperature} \ + \ \beta_3\,\text{year} \ + \ \epsilon.$$

We then print out the model summaries, including the posterior means of the fixed effects. From the results, the posterior means (with standard deviations in parentheses) are approximately:

$$\beta_0 \approx 2.236\,(0.031), \quad \beta_1 \approx 0.081\,(0.032), \quad \beta_2 \approx 0.027\,(0.032), \quad \beta_3 \approx 0.154\,(0.031).$$

Except for *temperature* variable, the posterior standard deviations are relatively small compared to their means, and zero is excluded from all $\mu \pm \sigma$ intervals. This indicates that most of the coefficient estimates are reasonably precise and differ from zero in a consistent direction.

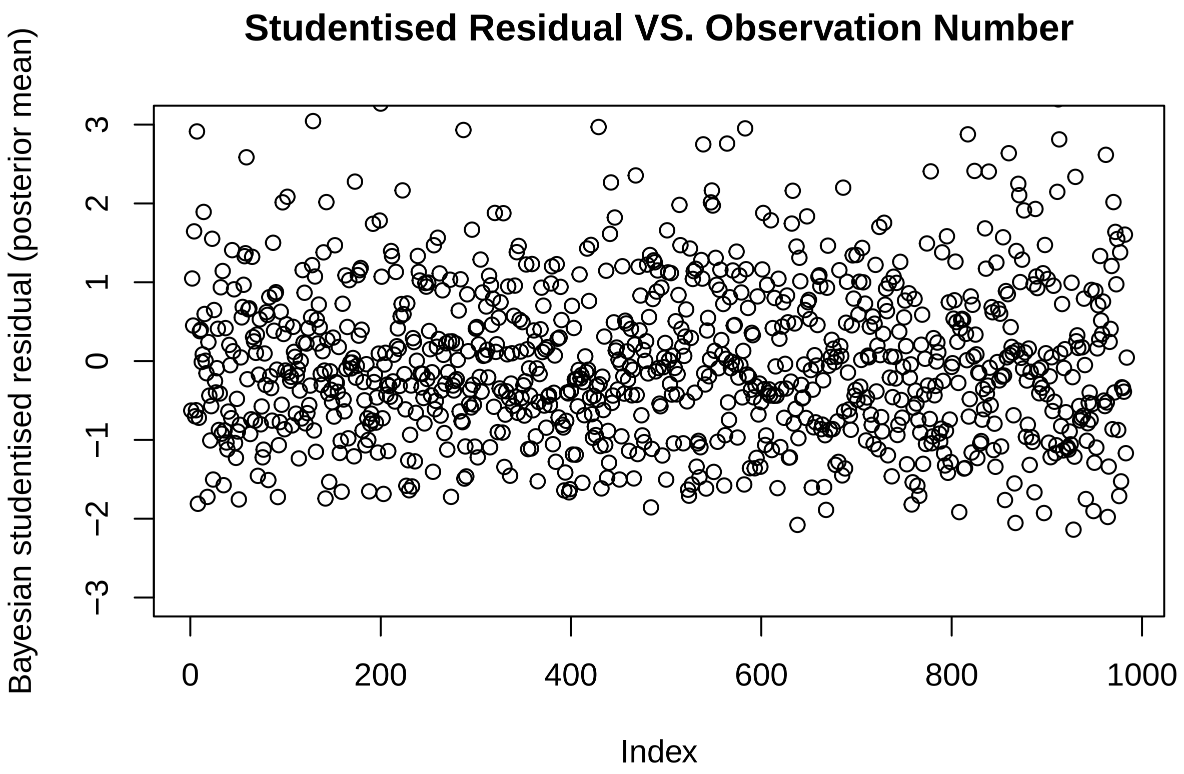
From these findings, we can draw some initial insights:

- Higher windspeed, higher temperature, and later years are each associated with higher rainfall.
- The *year* variable has a clear main effect on rainfall, suggesting a possible temporal trend.

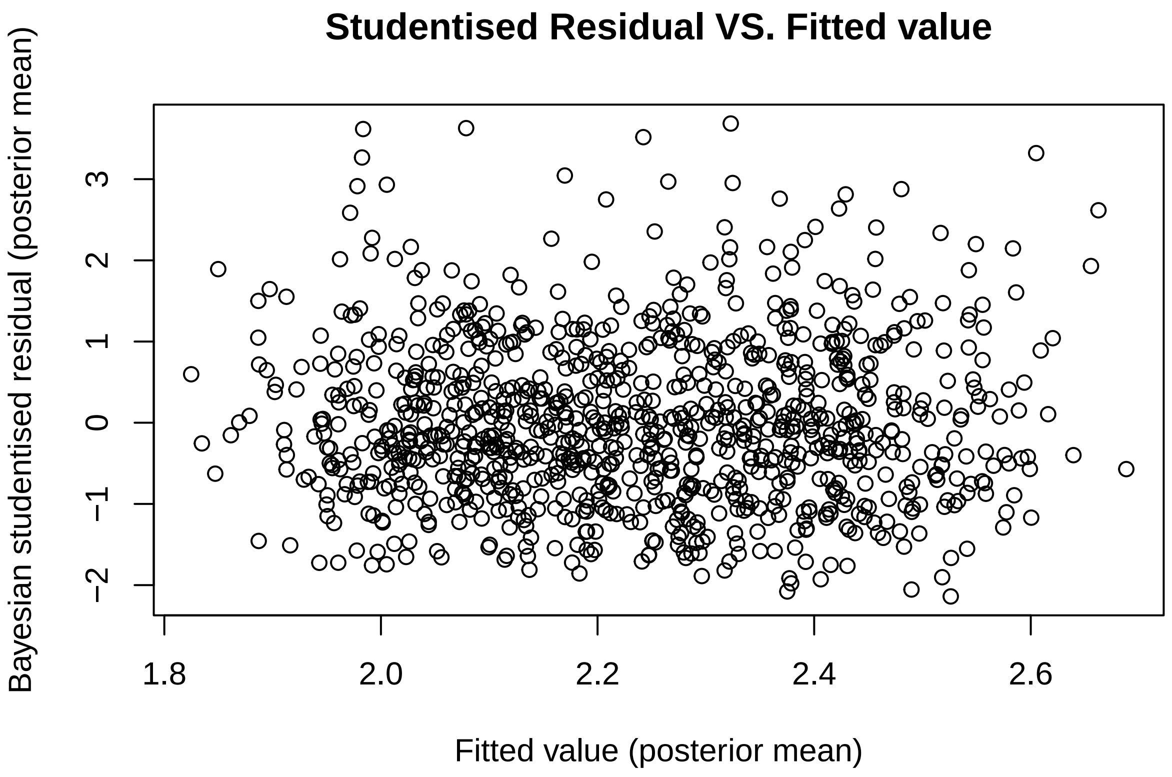
Q1.2)[8 marks]

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

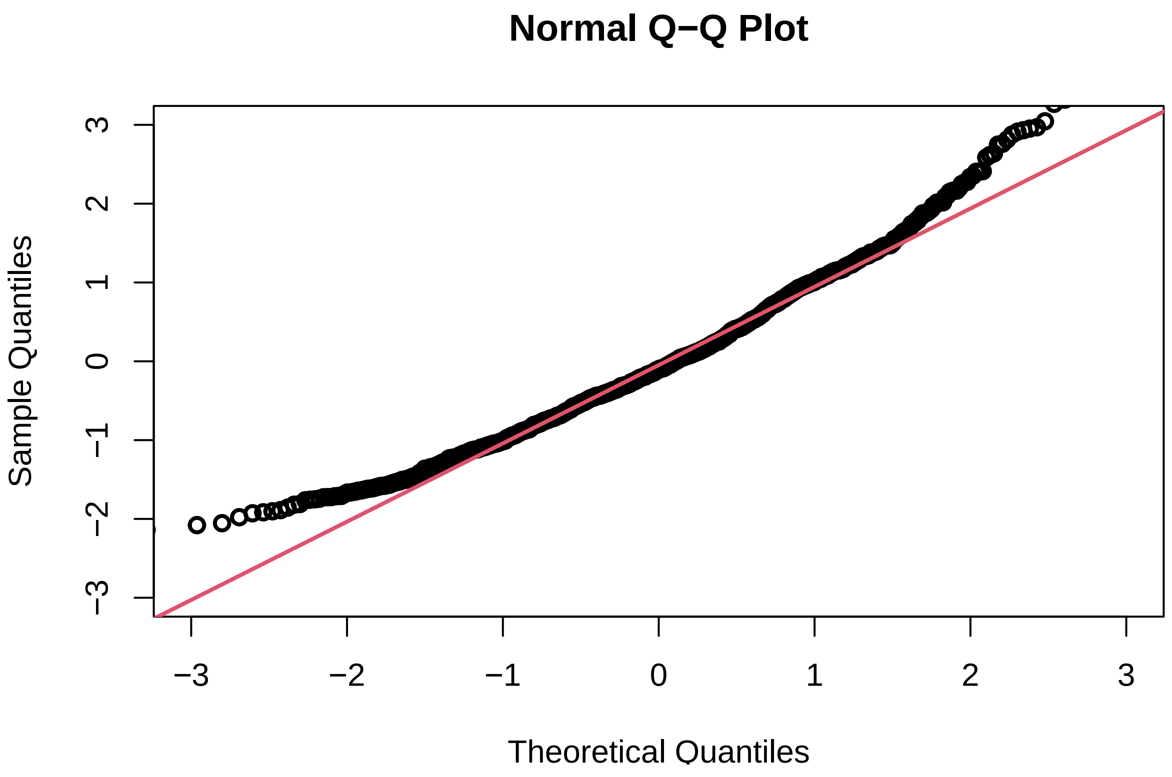
```r
#Resample and caculate
nbsamp=10000
samp <- inla.posterior.sample(nbsamp, m1I)
sigma=1/sqrt(inla.posterior.sample.eval(function(...) {theta},samp))
fittedvalues=inla.posterior.sample.eval(function(...) {Predictor},
samp)
n=nrow(Edi.rain_new)
x=cbind(rep(1,n),windspeed_scale, temperature_scale, year_scale)
H=x%*%solve((t(x)%*%x))%*%t(x)
studentisedred=matrix(0,nrow=n,ncol=nbsamp)
y=rainfall
ymx=as.matrix(y)%*%matrix(1,nrow=1,ncol=nbsamp);
resid=ymx-fittedvalues;
for(l in 1:nbsamp){
  studentisedred[,l]=resid[,l]/sigma[l];
}
for(i in 1:n){
  studentisedred[i,]=studentisedred[i,]/sqrt(1-H[i,i]);
}
studentisedredm=numeric(n)
for(i in 1:n){
  studentisedredm[i]=mean(studentisedred[i,])
}

#Plot
par(mfrow=c(1,1))
plot(seq_along(studentisedredm),studentisedredm,xlab="Index",ylab="Bayesian studentised residual (poste
```



```
fittedvaluesm=numeric(n)
for(i in 1:n){
fittedvaluesm[i]=mean(fittedvalues[i,])
}
plot(fittedvaluesm,studentisedredm,xlab="Fitted value (posterior mean)",ylab="Bayesian studentised resi
```



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



**Explanation**:

In our multiple linear regression model, we assume:normality, independence, constant variability and linearity. To check these assumptions, we examine the Studentized residuals, which are often used in frequentist analyses and are defined by

$$\hat{\varepsilon}_i = \frac{y_i - \hat{y}_i}{\hat{\sigma}\sqrt{1 - h_{ii}}}, \quad i = 1, \ldots, n$$

where $\hat{y}_i = \sum_{j=0}^{p} \hat{\beta}_j x_{ij}$ and $h_{ii}$ is the $i$-th diagonal entry of the 'hat' matrix.

We check for independence by plotting the Studentized residuals against the index. There is no noticeable pattern or trend, this generally supports the assumption of independence. We next plot the Studentized residuals against the fitted values. The residuals do not exhibit any clear structure (such as curvature or funnel-shaped spread), then linearity and homoscedasticity assumptions are reasonably satisfied. We construct a Q-Q plot of the Studentized residuals to assess whether they align with a normal distribution. In our analysis, the points lie fairly close to the diagonal for lower quantiles, but deviate in the tails, suggesting heavier tails than a normal distribution.

Overall, the standard diagnostic plots for the Studentized residuals suggest that the linearity, homoscedasticity, and independence assumptions are reasonably met, while the normality assumption shows mild violations in the tails.

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.**

```r
prior.beta<-list(mean.intercept=0,prec.intercept=1/100^2,mean=0,prec=1/100^2)
prec.prior<-list(prior="loggamma", param=c(1, 0.01))
fit_model<-function(family){
  model<-inla(rainfall ~ windspeed_scale + temperature_scale + year_scale,
              data = Edi.rain_new,
              family=family,
              control.family=list(hyper=list(theta=prec.prior)) ,
             control.fixed= prior.beta,
             control.compute=list(waic=TRUE,cpo=TRUE) )
   WAIC <- model$waic$waic
  NSLCPO <- -sum(log(model$cpo$cpo), na.rm = TRUE)

    return(c(WAIC = WAIC, NSLCPO = NSLCPO))

}
  gamma_results <- fit_model("gamma")
  lognormal_results <- fit_model("lognormal")
  weibull_results <- fit_model("weibull")

results_df <- data.frame(
  Model = c("Gamma", "Lognormal", "Weibull"),
  WAIC = c(gamma_results["WAIC"], lognormal_results["WAIC"], weibull_results["WAIC"]),
  NSLCPO = c(gamma_results["NSLCPO"], lognormal_results["NSLCPO"], weibull_results["NSLCPO"])
)
print(results_df)
```

```
##       Model     WAIC   NSLCPO
## 1     Gamma 2689.746 1344.873
## 2 Lognormal 2778.006 1389.003
## 3   Weibull 2707.396 1353.700
```

**Explanation**:

In this task, we seek a robust regression model by replacing the standard Gaussian error family with one of the two-parameter distributions (Gamma, Lognormal, or Weibull) available in INLA. These distributions can handle the strictly positive nature of daily rainfall better than the unconstrained Gaussian or Gumbel distributions.

Firstly, keep the same linear predictor specification as in Q1.1:

$$\mu/\lambda = \beta_0 + \beta_1 \cdot windspeed + \beta_2 \cdot temperature + \beta_3 \cdot year \quad \text{and} \quad \beta_j \sim N(0,\ 100^2)$$

***gamma_model***

$$rainfall \sim \text{Gamma}(\alpha, \lambda) \quad \text{and} \quad \theta \sim \text{Gamma}(1, 0.01)$$

***lognormal_model***

$$\log(rainfall) \sim N(\mu, \sigma^2) \quad \text{and} \quad \tau = \frac{1}{\sigma^2} \sim \text{Gamma}(1, 0.01)$$

***weibull_model***

$$rainfall \sim \text{Weibull}(\lambda, k) \quad \text{and} \quad \text{Gamma}(1, 0.01)$$

We use the WAIC and NSLCPO to quantify the goodness-of-fits, their mathematical formulas are as follows.

$$\text{WAIC}(y, \Theta) = -2\left(\text{lppd} - \sum_i \text{Var}_\theta \log p(y_i \mid \theta)\right),$$

where $\text{lppd}(y, \Theta) = \sum_i \log \frac{1}{S} \sum_s p(y_i \mid \Theta_s)$.

WAIC balances fit and complexity, aiming to measure out-of-sample predictive accuracy.

$$\text{NSLCPO} = -\sum_{i=1}^{n} \log(\text{CPO}_i),$$

where $\text{CPO}_i = p(y_i|y_{(-i)})$.

$\text{CPO}_i$ quantifies how likely is the observation i given the rest of the observations given the model

; NSLCPO sums it over all data points. Smaller NSLCPO indicates better predictive performance.

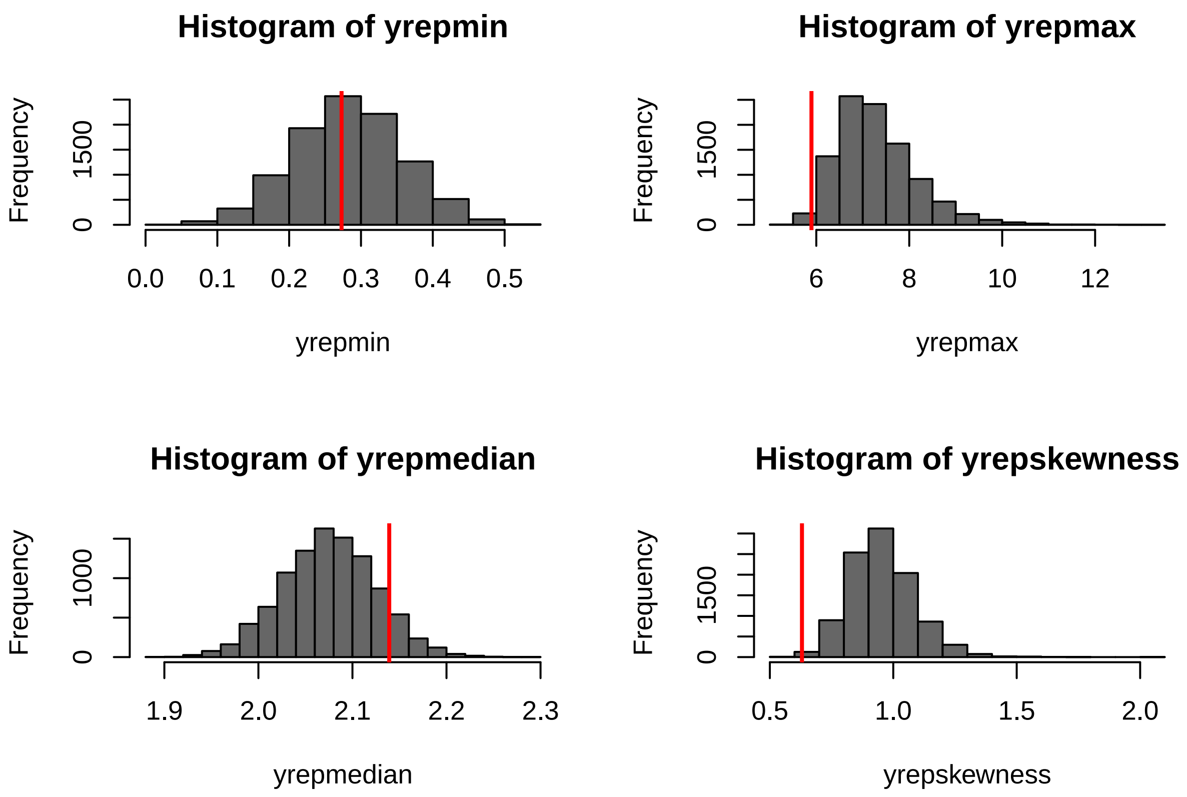
After fitting the Gamma, Lognormal, and Weibull models, we compile their respective WAIC (Gamma: 2689, Lognormal: 2778, Weibull: 2707) and NSLCPO (Gamma: 1344, Lognormal: 1389, Weibull: 1353) values. From these summary statistics, the Gamma model appears to be the best-fitting among the three candidates.

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
library(fBasics)
require(fBasics)

# Define prior distributions
prior.beta<-list(mean.intercept=0, prec.intercept=1/100^2,mean=0, prec=1/100^2)
prec.prior<-list(prior="loggamma", param=c(1,0.01))
best_model<-inla(rainfall~windspeed_scale+temperature_scale+year_scale,
                   data=Edi.rain_new,
                   family="gamma",
                  control.family = list(link="log",hyper = list(theta = prec.prior)),
                   control.fixed=prior.beta,
                   control.compute=list(config=TRUE),
                   control.predictor=list(compute=TRUE))


# Sample from the posterior distribution
nsamp<-10000  # Number of posterior samples
n<-nrow(Edi.rain_new)  # Number of observations
samples<-inla.posterior.sample(nsamp, result=best_model)
```

```r
# Extract linear predictor samples ( _i)
predictor.samples<-inla.posterior.sample.eval(function(...){Predictor}, samples)
# Extract precision parameter samples (theta)
theta.samples<-inla.posterior.sample.eval(function(...){theta},samples)
# Generate posterior predictive replicates
yrep<-matrix(0, nrow=n, ncol=nsamp)
y=Edi.rain_new$rainfall
for(i in 1:n){
     yrep[i, ]<-rgamma(n=nsamp, shape=theta.samples, rate=theta.samples/exp(predictor.samples[i,]))
}


# Compute summary statistics for posterior predictive samples
yrepmin<-apply(yrep, 2, min)
yrepmax<-apply(yrep, 2, max)
yrepmedian<-apply(yrep, 2, median)
yrepskewness<-apply(yrep, 2, skewness)
 par(mfrow=c(2,2))
  hist(yrepmin, col = "gray40", main = "Histogram of yrepmin")
  abline(v = min(y), col = "red", lwd = 2)
  hist(yrepmax, col = "gray40", main = "Histogram of yrepmax")
  abline(v = max(y), col = "red", lwd = 2)
  hist(yrepmedian, col = "gray40", main = "Histogram of yrepmedian")
  abline(v = median(y), col = "red", lwd = 2)
  hist(yrepskewness, col = "gray40", main = "Histogram of yrepskewness")
  abline(v = skewness(y), col = "red", lwd = 2)
```



**Explanation**:

We fit a Gamma model for the response variable rainfall using the INLA framework. The likelihood can be written as:

$$Y_i \sim \text{Gamma}(\alpha, \lambda)$$

where $\alpha$ is the shape parameter, and $\lambda$ is the rate parameter. In our INLA setup, we define predictor.samples as $\eta$, and define theta.samples as $\theta$.

$$\eta = log(\mu) \quad \mu = exp(\eta)$$

$$\text{mean} : \frac{\alpha}{\lambda} = exp(\eta) \quad \text{variance} : \frac{\alpha}{\lambda^2} = \frac{1}{\theta}$$

$$\text{shape} : \alpha = \theta \quad \text{rate} : \lambda = \frac{\theta}{exp(\lambda)}$$

So we use scale and shape in *rgamma* function to gain 10000 samples. To evaluate how well the model captures the data's distributional characteristics, we compute several summary statistics from each set of posterior predictive samples (e.g., min, max, median, skewness, etc.). From the histograms, the observed statistics (red lines) lie within high-density regions of the posterior predictive distributions. This indicates that the fitted Gamma model (with log link) can replicate important distributional features of the observed rainfall data, including extremes (min, max) and distribution shape (median, skewness).

**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.**]

```
Edi.rain <- Edi.rain %>%
  mutate(
    windspeed_scale = scale(windspeed),
    temperature_scale = scale(temperature),
    year_scale = scale(year)
  )

mI<-inla(rainfall~windspeed_scale+temperature_scale+year_scale,
                   data=Edi.rain,
                   family="gamma",
                  control.family = list(link="log",hyper = list(theta = prec.prior)),
                   control.fixed=prior.beta,
                   control.compute=list(config=TRUE),
                   control.predictor=list(compute=TRUE))

nbsamp=10000
mI.samp = inla.posterior.sample(nbsamp, mI,selection = list(Predictor=985:996))
predictor.samples=matrix(unlist(lapply(mI.samp, function(x)(x$latent[1:12]))),nrow=12,ncol=nbsamp)
theta.samples=unlist(lapply(mI.samp, function(x)(x$hyperpar[1])))
post.pred.samples=matrix(0,nrow=12,ncol=nbsamp)
for (i in 1:12){
    post.pred.samples[i,]=rgamma(n=nsamp, shape=theta.samples, rate=theta.samples/exp(predictor.samples
  }

month_days <- c(31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)
```

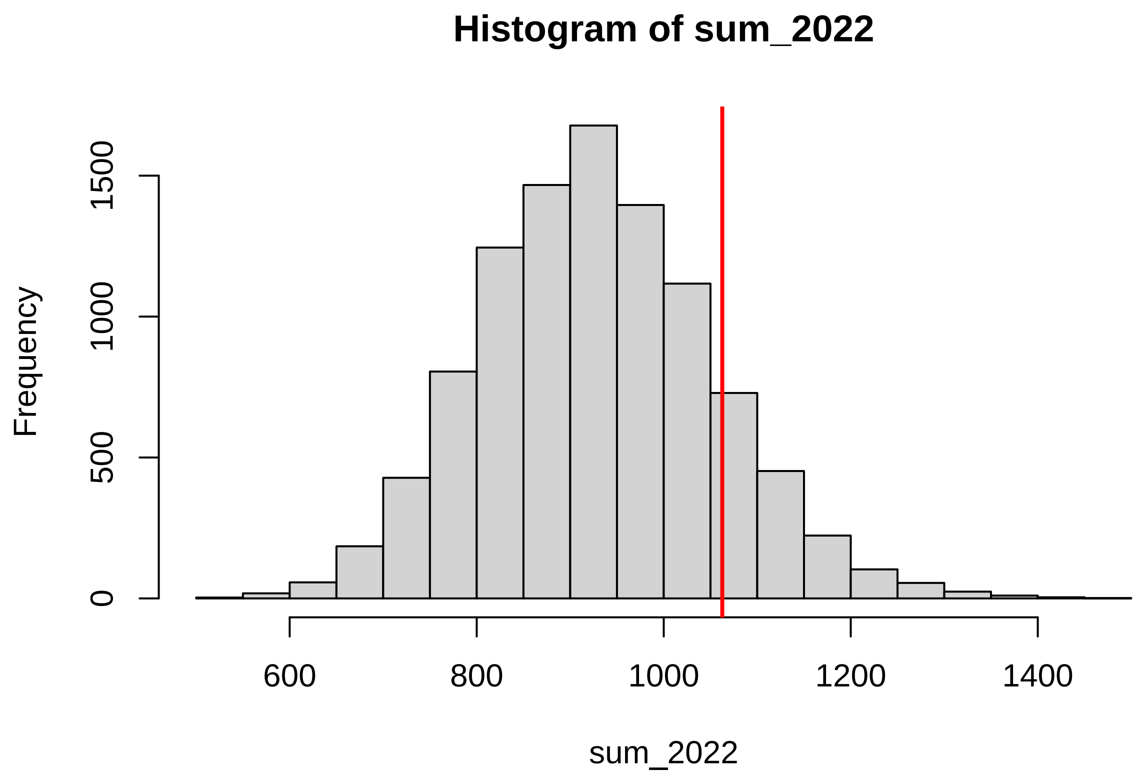
```r
sum_2022 <- colSums(post.pred.samples * month_days)


Edi.rain$group <- ifelse((1:nrow(Edi.rain)) %% 12 == 0, 12, (1:nrow(Edi.rain)) %% 12)
Edi.rain$annual <- ((1:nrow(Edi.rain))-1) %/% 12 +1940
annual_rainfall <- Edi.rain %>%
  mutate(days_in_month = month_days[group]) %>%
  group_by(annual) %>%
  summarise(
    total_rainfall = sum(rainfall * days_in_month)
  )
max_annual_rainfall <- max(annual_rainfall$total_rainfall, na.rm = TRUE)
cat("Maximum recorded annual rainfall:", max_annual_rainfall, "\n")
```

```
## Maximum recorded annual rainfall: 1062.594
```

```r
p.2022.exceeded=mean(sum_2022>max_annual_rainfall)
cat("Posterior probability of exceeding the record of 2022:",p.2022.exceeded)
```

```
## Posterior probability of exceeding the record of 2022: 0.1391
```

```r
hist(sum_2022)
abline(v = max_annual_rainfall , col = "red", lwd = 2)
```



**Explanation**:

We now use the original Edi.rain dataset (including missing data for 2022) and apply the mI model to generate monthly rainfall estimates for January through December 2022 (corresponding to indices 985 to 996 in the dataset). Because these estimates are monthly predictions, each one is multiplied by the number of days in the respective month (treating February as 28 days and ignoring leap years) and then summed to obtain the total annual rainfall.

From the resulting plot, most simulated annual totals cluster between 900 and 1100 mm, although some exceed the historical record (1065 mm), indicating approximately a 0.15 probability that 2022 will surpass the previous maximum.

**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")
head(frogs)
```

```
##     MFCCs_.2    MFCCs_.3  MFCCs_.4  MFCCs_.5  MFCCs_.6    MFCCs_.7   MFCCs_.8
## 1 0.15293630 -0.10558590 0.2007219 0.3172011 0.2607639 0.100944641 -0.1500626
## 2 0.17153426 -0.09897474 0.2684252 0.3386719 0.2683531 0.060835087 -0.2224746
## 3 0.15231709 -0.08297267 0.2871280 0.2760141 0.1898668 0.008713957 -0.2422342
## 4 0.22439245  0.11898466 0.3294317 0.3720880 0.3610046 0.015501040 -0.1943475
## 5 0.08781691 -0.06834489 0.3069667 0.3309229 0.2491439 0.006883713 -0.2654234
## 6 0.09970374 -0.03340782 0.3498951 0.3445353 0.2475688 0.022406957 -0.2137672
##      MFCCs_.9  MFCCs_10  MFCCs_11    MFCCs_12   MFCCs_13    MFCCs_14  MFCCs_15
## 1 -0.17112763 0.1246764 0.1886541 -0.07562172 -0.1564359  0.08224512 0.1357520
## 2 -0.20769267 0.1708829 0.2709583 -0.09500394 -0.2543415  0.02278623 0.1633201
## 3 -0.21915332 0.2325383 0.2660645 -0.07282719 -0.2373836  0.05079074 0.2073384
## 4 -0.09818067 0.2703754 0.2672789 -0.16225825 -0.3170842 -0.01156743 0.1004128
## 5 -0.17269981 0.2664343 0.3326951 -0.10074854 -0.2985239  0.03743889 0.2191528
## 6 -0.12791598 0.2773526 0.3098613 -0.13452792 -0.2951227  0.01248602 0.1806410
##      MFCCs_16    MFCCs_17    MFCCs_18     MFCCs_19    MFCCs_20    MFCCs_21
## 1 -0.02401665 -0.10835111 -0.07762252 -0.009567802  0.05768398  0.11868014
## 2  0.01202228 -0.09097401 -0.05650952 -0.035303357  0.02013996  0.08226299
## 3  0.08353570 -0.05069143 -0.02359023 -0.066721549 -0.02508323  0.09910840
## 4 -0.05022373 -0.13600940 -0.17703701 -0.130498133 -0.05476640 -0.01869145
## 5  0.06283723 -0.04888462 -0.05307351 -0.088550403 -0.03134557  0.10860983
## 6  0.05524178 -0.08048746 -0.13008922 -0.171477611 -0.07156940  0.07764295
##     MFCCs_22          Family     Genus        Species RecordID
## 1 0.01403845 Leptodactylidae Adenomera AdenomeraAndre        1
## 2 0.02905574 Leptodactylidae Adenomera AdenomeraAndre        1
## 3 0.07716238 Leptodactylidae Adenomera AdenomeraAndre        1
## 4 0.02395431 Leptodactylidae Adenomera AdenomeraAndre        1
## 5 0.07924433 Leptodactylidae Adenomera AdenomeraAndre        1
## 6 0.06490259 Leptodactylidae Adenomera AdenomeraAndre        1
```

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 $\mathbf{Y}_i = (Y_{1i}, Y_{2i}) \sim \text{BVN}(\mu, \Sigma)$, where

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

**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 $\mu_1$, $\mu_2$, $\sigma_1^2$, $\sigma_2^2$, and $\rho$,** and print the model summary. Interpret these outputs.

```r
library(rjags)
```

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

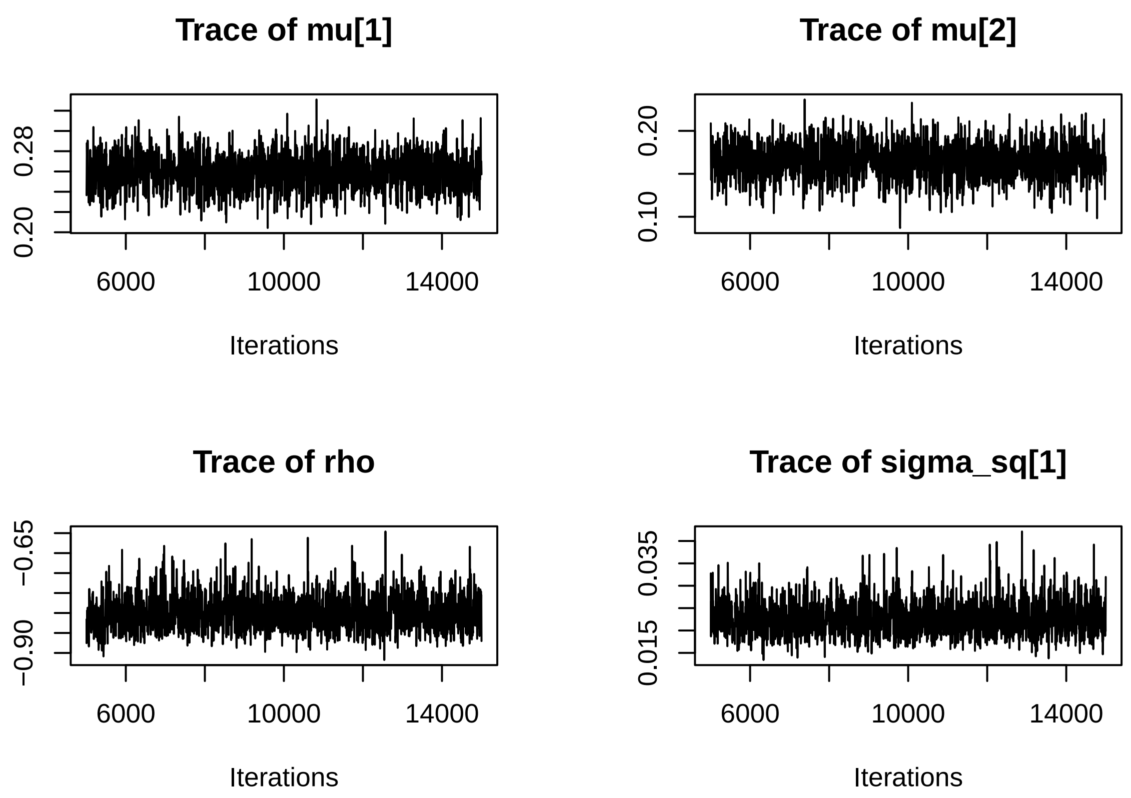
```
## Linked to JAGS 4.3.2
```

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

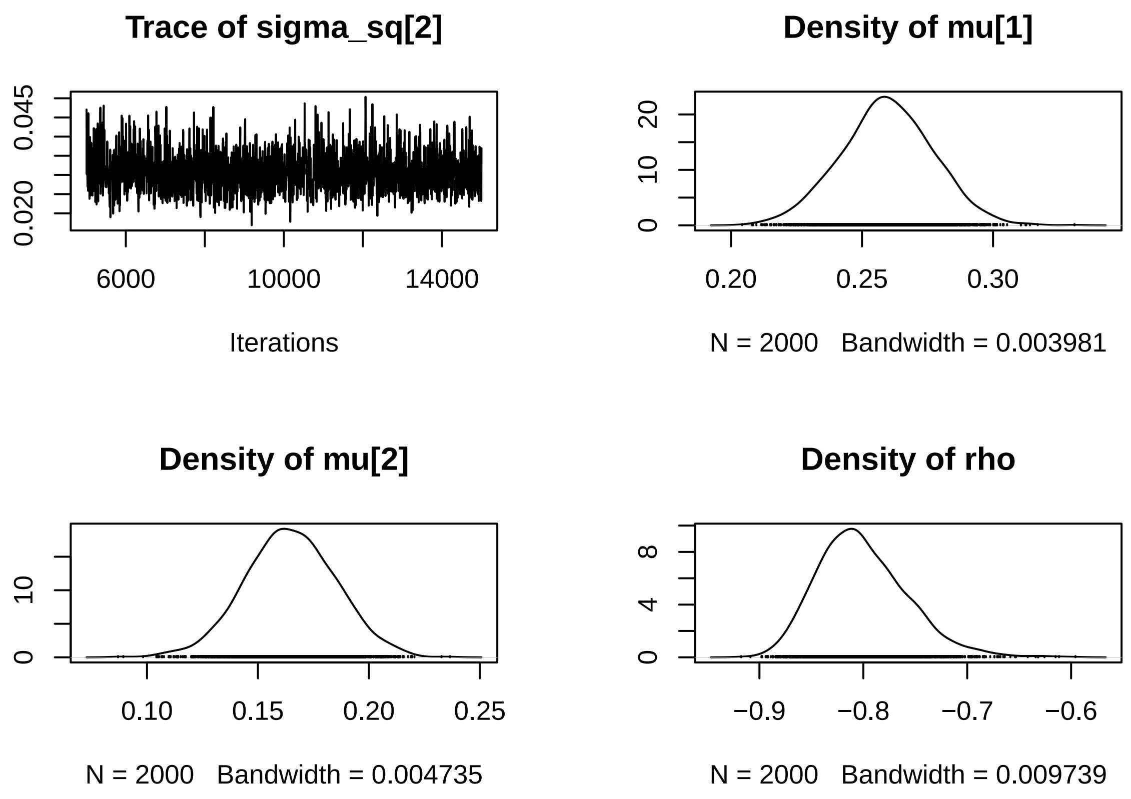
```r
# 1 Create model block for JAGS
BivariateNormal.model <- "
model {
  # Priors
  mu[1] ~ dnorm(0, 0.0001)
  mu[2] ~ dnorm(0, 0.0001)
  rho   ~ dunif(-1, 1)
  tau1  ~ dgamma(1, 0.1)
  tau2  ~ dgamma(1, 0.1)

  #  Likelihood
  for(i in 1:n) {
    Y[i, 1:2] ~ dmnorm(mu[], Omega)
  }

  sigma_sq[1] <- 1 / tau1
  sigma_sq[2] <- 1 / tau2
  Omega[1,1] <- tau1 / (1 - rho^2)
  Omega[2,2] <- tau2 / (1 - rho^2)
  Omega[1,2] <- - rho * sqrt(tau1 * tau2) / (1 - rho^2)
  Omega[2,1] <- Omega[1,2]
}
"
# 2  Create data input and initial values for JAGS
selected<-frogs$RecordID %in% c(1, 46)
M2  <- frogs$MFCCs_.2[selected]
M11 <- frogs$MFCCs_11[selected]
n <- length(M2)
Y <- cbind(M2, M11)
Frog.data <- list(n = n, Y = Y)
Frog.inits <- list(
  list(
    mu   = c(0, 0),
    rho  = 0,
    tau1 = 0.05,
    tau2 = 0.05
  )
)
#  3 Execute JAGS model
results.A <- jags.model(
  file = textConnection(BivariateNormal.model),
  data = Frog.data,
  inits = Frog.inits,
  n.chains = 1
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 73
##    Unobserved stochastic nodes: 5
##    Total graph size: 98
##
## Initializing model
```

```r
# 4 Burn-in
update(results.A, n.iter = 4000)
# 5 MCMC Sampling
results.B <- coda.samples(
  results.A,
  variable.names = c("mu", "rho", "sigma_sq"),
  n.iter = 10000,
  thin = 5
)
#  trace plot, density plot
par(mfrow=c(2,2))
traceplot(results.B)
```



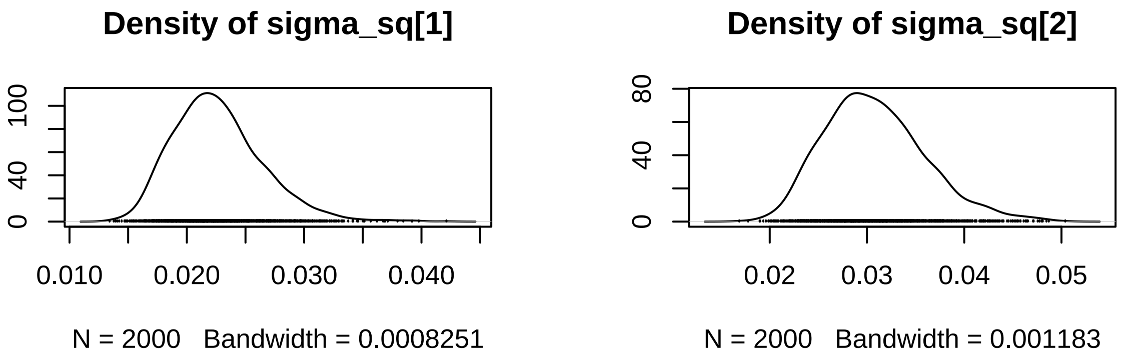
```r
densplot(results.B)
```



```r
summary(results.B)
```

```
##
```

```
## 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]         0.25955 0.017578 3.930e-04      0.0004423
## mu[2]         0.16459 0.020429 4.568e-04      0.0004979
## rho          -0.80073 0.043234 9.667e-04      0.0011057
## sigma_sq[1]   0.02263 0.003823 8.549e-05      0.0000839
## sigma_sq[2]   0.03081 0.005159 1.154e-04      0.0001346
##
## 2. Quantiles for each variable:
##
##                  2.5%      25%      50%      75%    97.5%
## mu[1]         0.22446  0.24827  0.25951  0.27128  0.29378
## mu[2]         0.12457  0.15087  0.16423  0.17828  0.20531
## rho          -0.87098 -0.83091 -0.80532 -0.77461 -0.70484
## sigma_sq[1]   0.01653  0.01999  0.02227  0.02476  0.03132
## sigma_sq[2]   0.02233  0.02717  0.03033  0.03401  0.04242
```



**Explanation**:

We fit a bivariate normal model to the MFCCs_.2 and MFCCs_.11 coefficients for frogs 1 and 46. From the summary, the variable $\mu_1$, $\mu_2$, $\sigma_1^2$, $\sigma_2^2$, and $\rho$ (with standard deviations in parentheses) are approximately:

$$\mu_1 \approx 0.258(0.018), \quad \mu_2 \approx 0.165(0.021)$$
$$\sigma_1^2 \approx 0.022(0.003), \quad \sigma_2^2 \approx 0.031(0.005), \quad \rho \approx -0.799(0.044)$$

the variables' standard deviations are relatively small compared to their means, and zero is excluded from all $\mu \pm \sigma$ intervals. This indicates that most of the variables estimates are reasonably precise and differ from zero in a consistent direction.

From the trace plot of each variable, all values are within a zone without strong periodicity and (especially) a trend, and it is consistent with convergence to a stationary posterior distribution. Then from the density plots of each variable, the shapes are unimodal and fairly *tight* around the means, reinforcing that the parameters are not jumping between multiple modes and estimated with decent precision.

**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** $\rho$ is above 2500. Once this is ensured, evaluate, for $\rho$, $\mu_1$, and $\mu_2$: i) their autocorrelation function estimates, ii) the Gelman-Rubin statistic estimates, and iii) the Gelman-Rubin diagnostic plots. Interpret these plots and values.

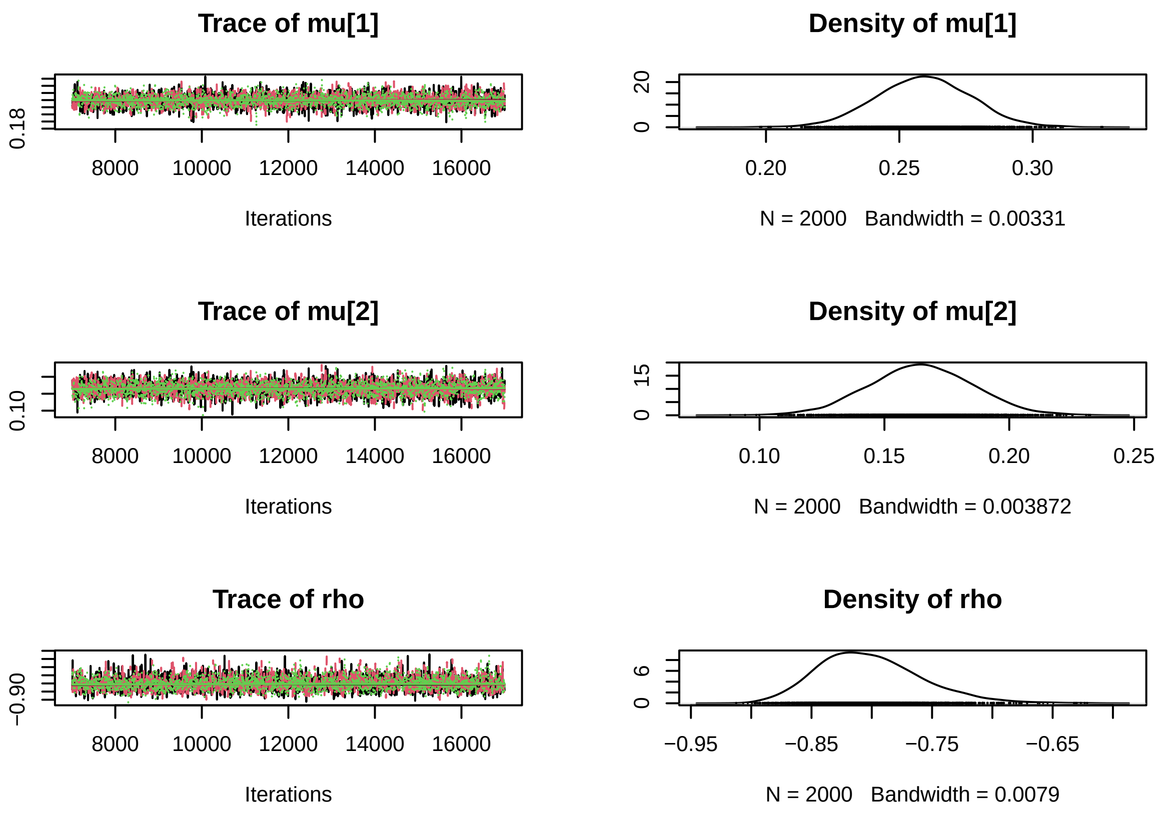
```
model.jags <- jags.model(file = textConnection(BivariateNormal.model),
                         data = Frog.data,
                         n.chains = 3)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 73
##    Unobserved stochastic nodes: 5
##    Total graph size: 98
##
## Initializing model
```

```
update(model.jags, n.iter = 6000)
results.C <- coda.samples(model.jags,
                        variable.names = c("mu", "rho"),
                        n.iter = 10000,
                        thin = 5)
ess <- effectiveSize(results.C)
cat('The effective sample size for rho is', ess[3], '\n')
```

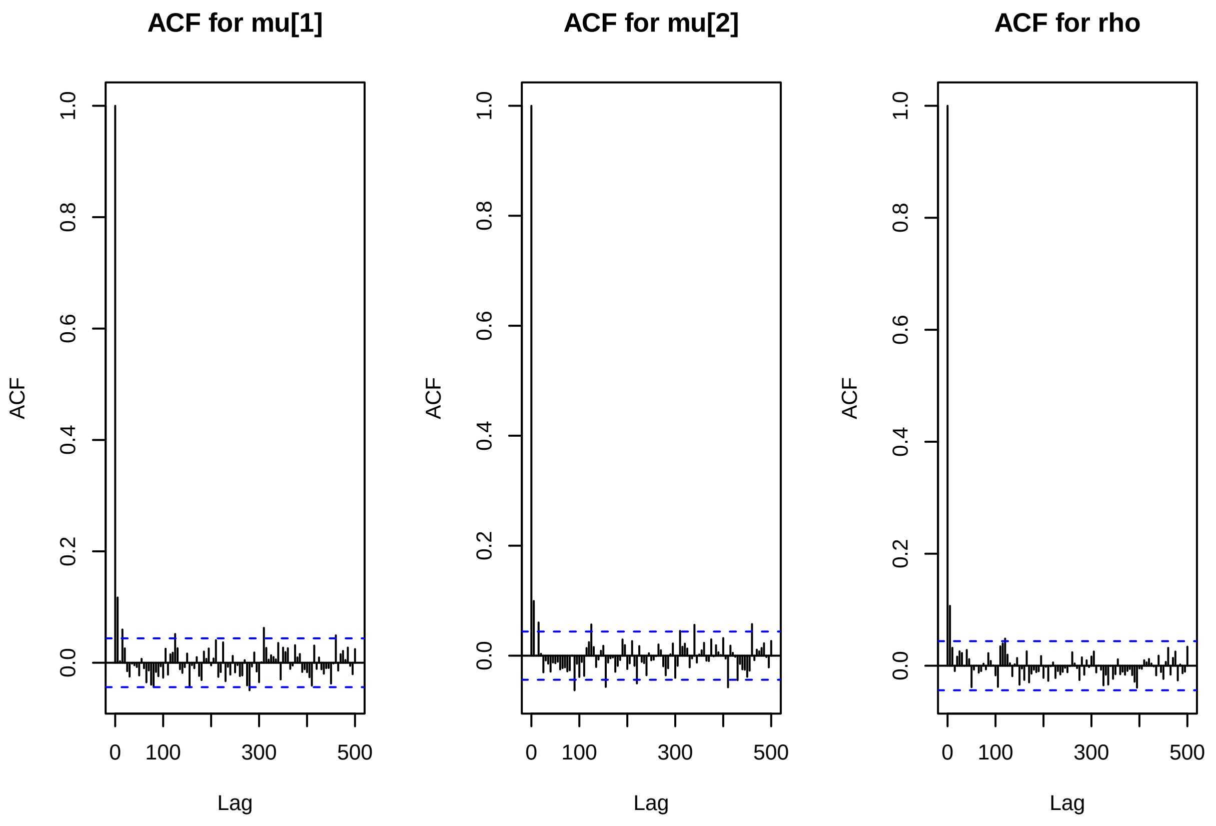
```
## The effective sample size for rho is 4728.61
```

```
plot(results.C)
```

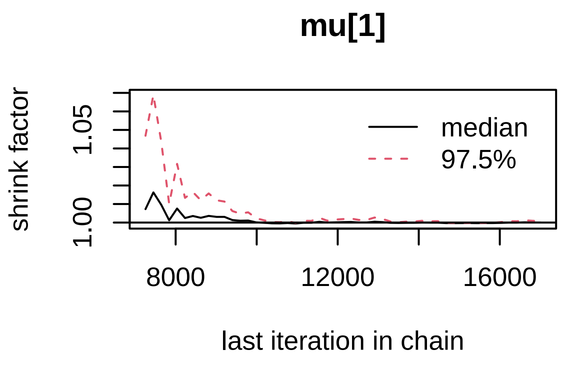


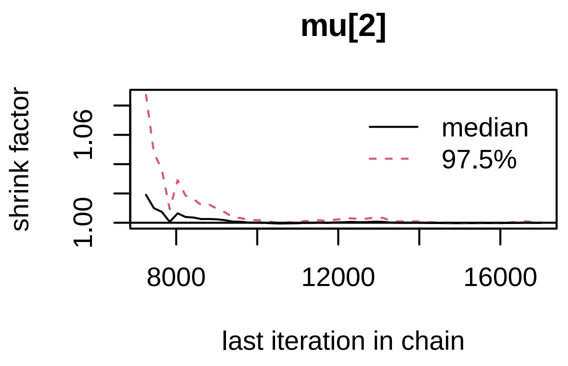
```
# Evaluate autocorrelation functions
par(mfrow=c(1,3))
```

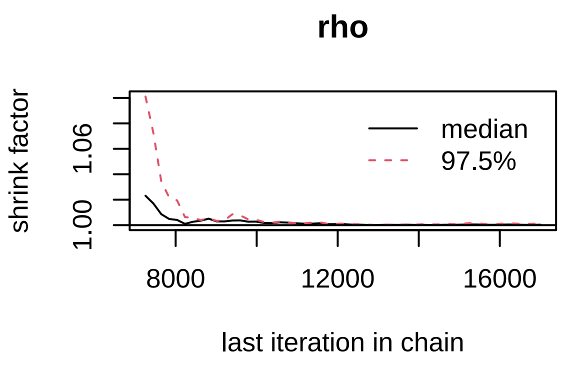
```r
acf(results.C[[1]][, "mu[1]"], lag.max = 100, main = "ACF for mu[1]")
acf(results.C[[1]][, "mu[2]"], lag.max = 100, main = "ACF for mu[2]")
acf(results.C[[1]][, "rho"],   lag.max = 100, main = "ACF for rho")
```



```r
gelman.plot(results.C)
```







```r
gelman.diag(results.C)
```

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

**Explanation**:

We reran the model using three independent chains with no specified initial values. And we set the burn-in period and the total number of MCMC iterations as 6000 and 10000 so that the effective sample size (ESS) for $\rho$ exceeded 2500, which is 4591.

Examining the autocorrelation function (ACF) estimates for $\mu_1$, $\mu_2$ and $\rho$, we observe that the correlations at various lags remain near zero and largely fall within the 95% confidence bounds. This indicates that successive samples in the Markov chain are only weakly dependent on each other, reflecting *good mixing* of the sampler.

The potential scale reduction factors (Gelman-Rubin statistic) for $\mu_1$, $\mu_2$ and $\rho$ are all reported as 1.0 (with upper confidence limits at or near 1.0). A PSRF close to 1.0 suggests that the three chains have converged to the same target distribution, with minimal between-chain variance relative to within-chain variance. The Gelman-Rubin diagnostic plots reinforce this conclusion. In these plots, the "shrink factor" curves for all parameters drop below 1.05 relatively early in the sampling process. This pattern indicates that the three chains have become well mixed and are converging to a common posterior distribution.

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:

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

where all $\mu_k^{(k)}, j = 1, 2, k = 1, \dots, K$ 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 $\mu_1^{(1)} \leq \mu_1^{(2)} \leq \dots \leq \mu_1^{(k)}$. [hint 1: generate $\mu_1^{(1)}, \dots, \mu_1^{(k)}$ 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** $p_{1,i}, i = 1, \dots, N$; note that $p_{i,2} = 1 - p_{i,1}$. Place the same priors on $\sigma_1^2$, $\sigma_2^2$, and $\rho$ as in Q2.1). Use 5000 burn-in samples to update the model, with one chain.

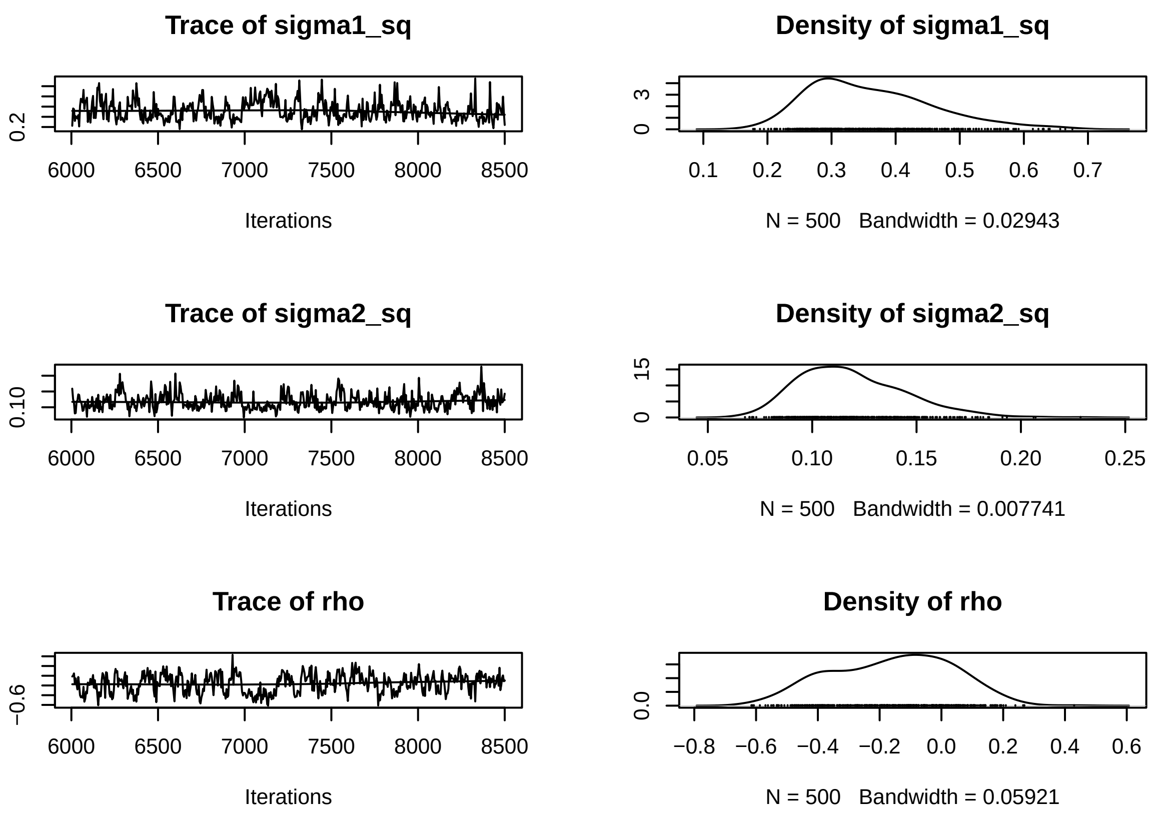
```
BivariateMixture.model <- "
model {
  # Priors
  for(k in 1:2){
  mu_unsorted[k,1]~ dnorm(0, 1.0/10000)
  }

  mu[1,1] <- min(mu_unsorted[,1])
  mu[2,1] <- max(mu_unsorted[,1])
  mu[1,2] ~ dnorm(0, 1/10000)
  mu[2,2] ~ dnorm(0, 1/10000)
  rho ~ dunif(-1, 1)
  tau1 ~ dgamma(1, 0.1)
  tau2 ~ dgamma(1, 0.1)

  # Likelihood
  for(i in 1:n) {
    p[i,1] ~  dbeta(0.5, 0.5)
    p[i,2]=1-p[i,1]
    Z[i] ~ dcat(p[i,1:2])
    Y[i, ] ~ dmnorm(mu[Z[i],], Omega.inv)
  }


  Omega.inv[1:2,1:2] <- inverse(Omega)
  sigma1_sq <- 1/tau1
  sigma2_sq <- 1/tau2
  Omega[1,1] <- sigma1_sq
  Omega[2,2] <- sigma2_sq
  Omega[1,2] <- rho * sqrt(sigma1_sq *sigma2_sq)
  Omega[2,1] <- Omega[1,2]
}
"

# 2  Create data input and initial values for JAGS
n <- length(M2)
Y <- cbind(scale(M2), scale(M11))
Frognew.data <- list(n = n, Y = Y)

#  3 Execute JAGS model
results.D <- jags.model(
  file = textConnection(BivariateMixture.model),
  data = Frognew.data,
  n.chains = 1
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 73
##    Unobserved stochastic nodes: 153
##    Total graph size: 465
##
## Initializing model
```

```
# 4 Burn-in
update(results.D, n.iter = 5000)
```

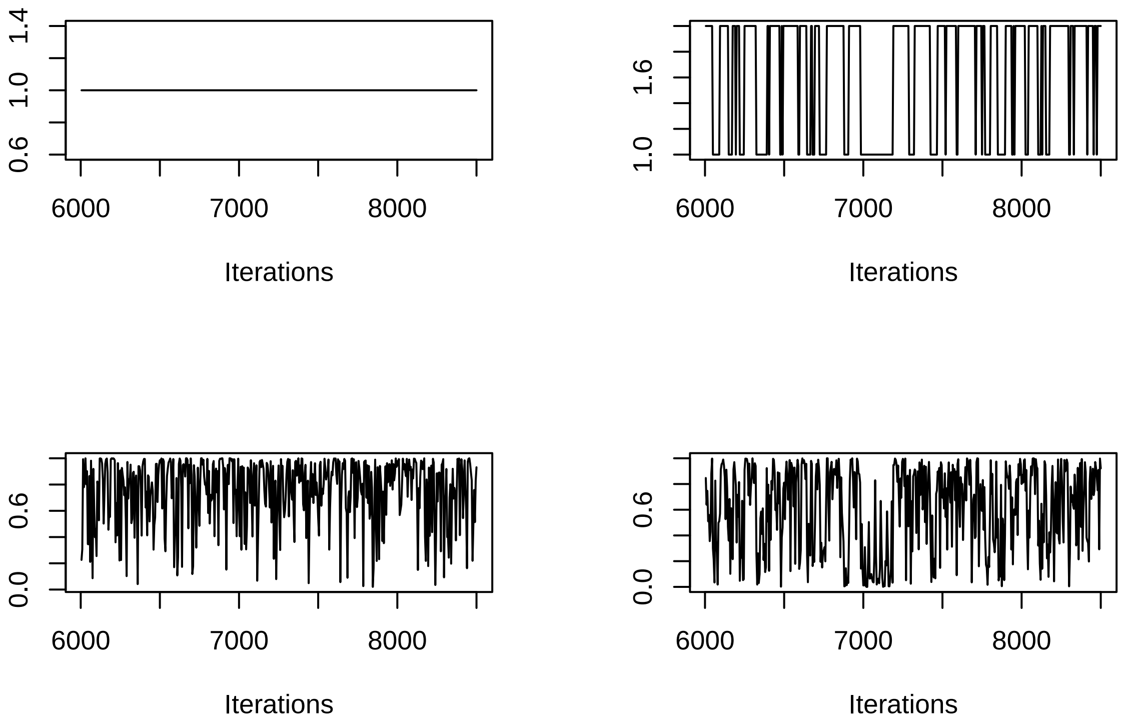
**Generate 2500 samples with thin=5. Plot the traceplots and posterior densities for the standard deviations and correlation parameter of the bivariate normal mixture components.**

```
results.E <- coda.samples(
  results.D,
  variable.names = c( "rho", "sigma1_sq","sigma2_sq","p","Z"),
  n.iter = 2500,
  thin = 5
)
plot(results.E[, c("sigma1_sq","sigma2_sq","rho")])
```

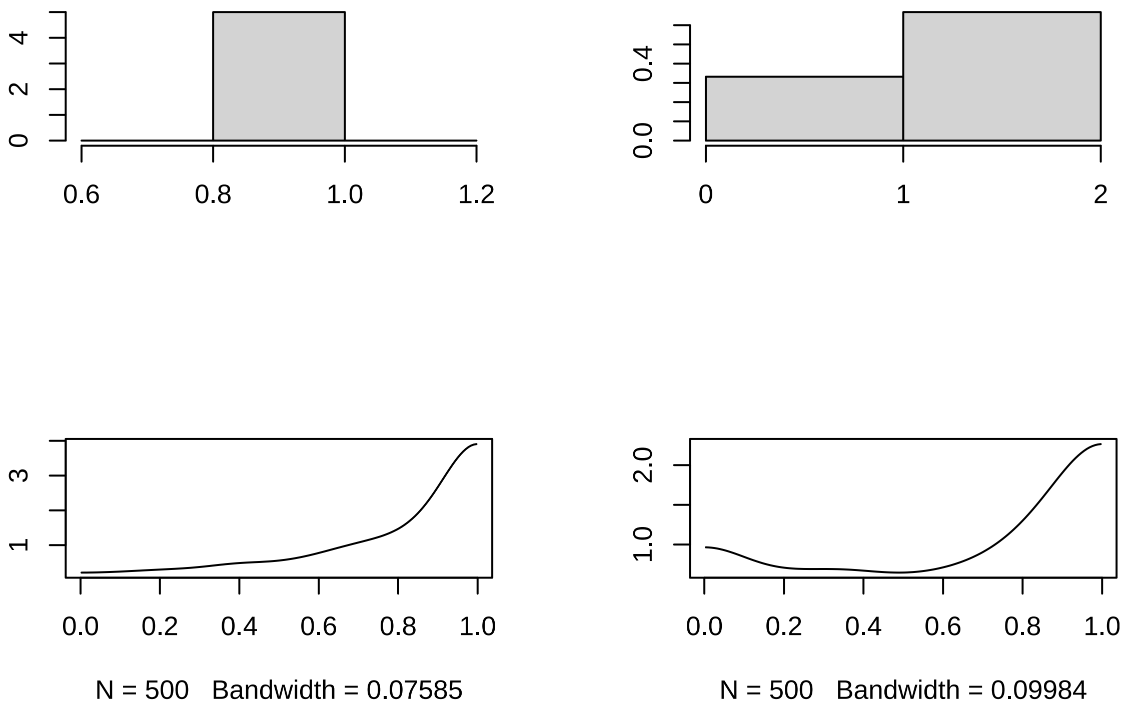


**Plot the traceplots and posterior densities for latent label components** $Z_1$ and $Z_{48}$, and the corresponding cluster probabilities, $p_{1,1}$ and $p_{48,2}$. Interpret you posterior density estimates.

```
par(mfrow=c(2,2))
traceplot(results.E[,"Z[1]"])
traceplot(results.E[,c("Z[48]")])
traceplot(results.E[,"p[1,1]"])
traceplot(results.E[,"p[48,2]"])
```



```
densplot(results.E[,c("Z[1]")])
densplot(results.E[,c("Z[48]")])
densplot(results.E[,"p[1,1]"])
densplot(results.E[,"p[48,2]"])
```



**Explanation**:

We extracted two features (MFCCs_.2 and MFCCs_.11) from two frogs (ID = 1 and ID = 46) to form our dataset. These features were used to train a Markov Chain Monte Carlo (MCMC) model, which produced 73 clustering outcomes (52 data points from frog ID = 1 and 21 from frog ID = 46). The model then determined whether each data point belonged to frog ID = 1 or frog ID = 46.

After fitting the model, the trace plots for $\sigma_1^2$, $\sigma_2^2$, and $\rho$ appear stable with no discernible trends or large fluctuations, suggesting that the Markov chains have converged. Each parameter's posterior density is unimodal, indicating the sampler is not jumping between multiple modes. For $\sigma_1^2$, $\sigma_2^2$, and $\rho$ , the mode is approximately 0.3, 0.1, and 0, respectively. Additionally, $\rho$ exhibits a slightly heavier tail on the negative side.

Regarding Z[1] and Z[48], we found that their most frequent values (modes) were 1 and 2, respectively. However, both actually belong to frog #1. This indicates that while the model mostly consistently assigns data points from frog #1 to the first cluster, some points are occasionally misclassified. As the MCMC process converges, the probability of Z[1] being assigned to cluster 1 and Z[48] to cluster 2 follows a clear, monotonic trend, reflecting the model's increasing confidence in these cluster assignments.

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 $(p_{i,1}, p_{i,2}, \dots, p_{i,k})$ 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.

```
MixtureK.model <- "
model {
  for (k in 1:K) {
    mu_raw[k,1] ~ dnorm(0, 1.0/10000)
    mu[k,2] ~ dnorm(0, 1.0/10000)
  }
  mu_order[1:K] <- rank(mu_raw[1:K,1])
  for (k in 1:K) {
    mu[k,1] <- mu_raw[mu_order[k],1]
  }

  rho  ~ dunif(-1, 1)
  tau1 ~ dgamma(1, 0.1)
  tau2 ~ dgamma(1, 0.1)
  alpha=rep(1,K)


  for(i in 1:n) {
    p[1:K,i] ~ ddirch(alpha)   #  (Dirichlet)
    Z[i] ~ dcat( p[1:K,i] )
    Y[i,] ~ dmnorm(mu[Z[i], ], Omega.inv)
  }

  Omega.inv[1:2,1:2] <- inverse(Omega)
  sigma1_sq <- 1/tau1
  sigma2_sq <- 1/tau2
  Omega[1,1] <- sigma1_sq
  Omega[2,2] <- sigma2_sq
  Omega[1,2] <- rho * sqrt(sigma1_sq * sigma2_sq)
  Omega[2,1] <- Omega[1,2]
}
"

selected<-frogs$RecordID %in% c(1,46,56)
```

```r
M2  <- frogs$MFCCs_.2[selected]
M11 <- frogs$MFCCs_11[selected]
n <- length(M2)
Y <- cbind(scale(M2), scale(M11))

Frogmix2.data <- list(n = n, Y = Y, K=2)
Frogmix3.data <- list(n = n, Y = Y, K=3)
Frogmix4.data <- list(n = n, Y = Y, K=4)

results.F2 <- jags.model(
  file = textConnection(MixtureK.model),
  data = Frogmix2.data,
  n.chains = 2
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 119
##    Unobserved stochastic nodes: 245
##    Total graph size: 507
##
## Initializing model
```

```r
results.F3 <- jags.model(
  file = textConnection(MixtureK.model),
  data = Frogmix3.data,
  n.chains = 2
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 119
##    Unobserved stochastic nodes: 247
##    Total graph size: 512
##
## Initializing model
```

```r
results.F4 <- jags.model(
  file = textConnection(MixtureK.model),
  data = Frogmix4.data,
  n.chains = 2
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 119
##    Unobserved stochastic nodes: 249
##    Total graph size: 517
##
```

```
## Initializing model
```

```
update(results.F2, n.iter = 10000)
update(results.F3, n.iter = 10000)
update(results.F4, n.iter = 10000)
```

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

```
dic.F2 <- dic.samples(model = results.F2, n.iter = 5000, type = "pD")
dic.F3 <- dic.samples(model = results.F3, n.iter = 5000, type = "pD")
dic.F4 <- dic.samples(model = results.F4, n.iter = 5000, type = "pD")
dic.F2
```

```
## Mean deviance:  411
## penalty 902.7
## Penalized deviance: 1314
```

```
dic.F3
```

```
## Mean deviance:  223.7
## penalty 173.7
## Penalized deviance: 397.4
```

```
dic.F4
```

```
## Mean deviance:  115.6
## penalty 29.11
## Penalized deviance: 144.8
```

**Explanation**:

We use the Dirichlet function to make sure the sum of $p$ is 1 and use the rank function to ensure that we order the first component of the mean vector, $\mu_1^{(1)} \leq \mu_1^{(2)} \leq \cdots \leq \mu_1^{(k)}$. The DIC function provides three indicators:

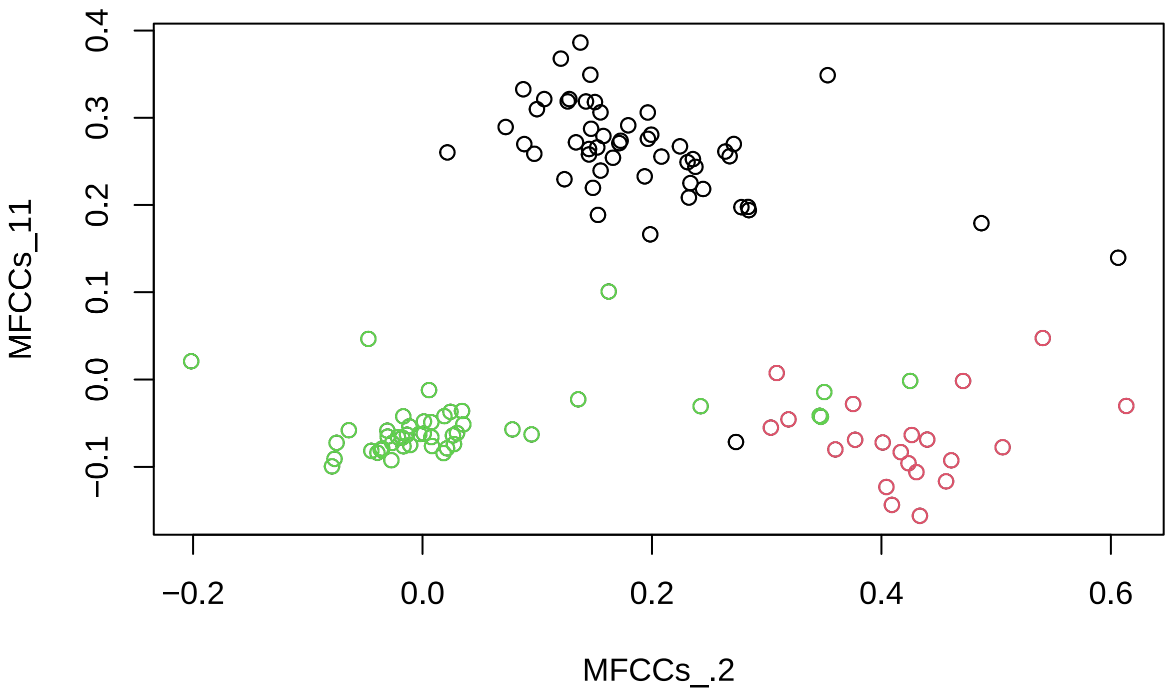
- The Mean deviance represents the average discrepancy between the observed data and the model's predictions, serving as an overall measure of model fit.
- The penalty, on the other hand, quantifies the effective complexity of the model—essentially penalizing models with more parameters or higher flexibility.
- The Penalized deviance is the sum of the mean deviance and the penalty; it is used to balance fit and complexity when comparing models.

In the results, comparing the penalized deviance, the model with K=3 yields the lowest Penalized deviance (-303.2) compared to K=2 (-160.2) and K=4 (-291.2), making it the preferred model based on this criterion.

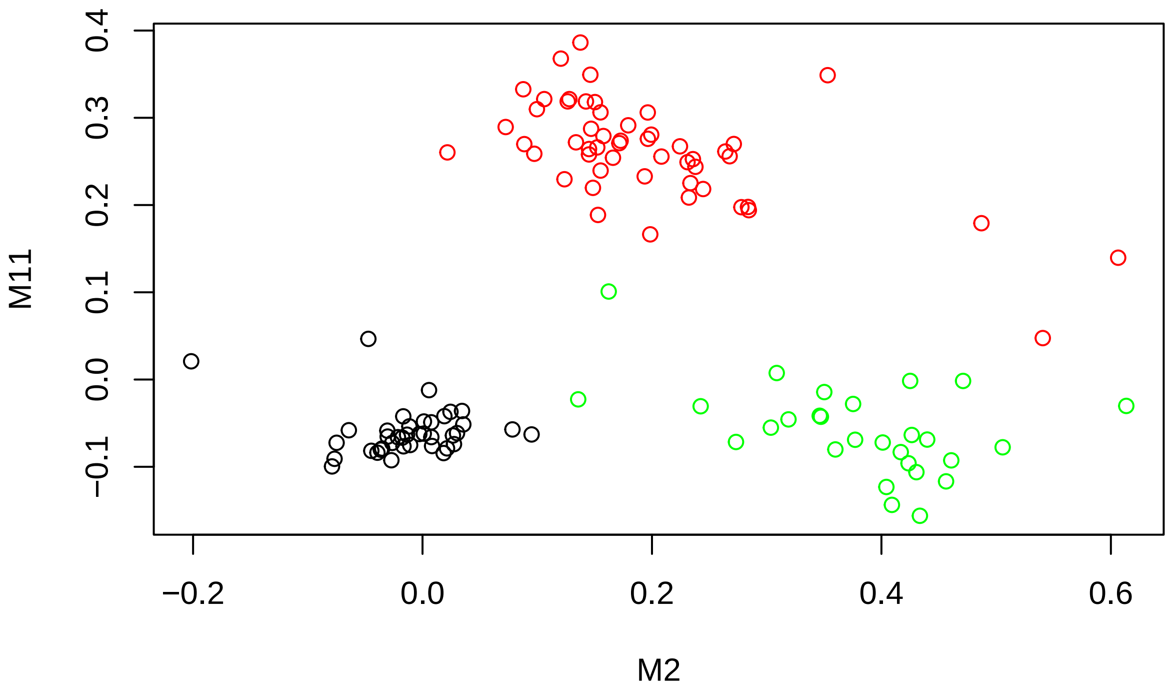
From a theoretical perspective, our training data consist of the MFCCs_.2 and MFCCs_11 coefficients for frogs with IDs 1, 46, and 56. This means that the underlying structure of the data naturally corresponds to three distinct frog identities. Therefore, when the model with K=3 clusters is chosen based on the Penalized deviance criterion, it aligns with the real-world grouping of the frogs.

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

**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).



```
samples<-coda.samples(results.F3 ,variable.names = c('Z','p'),n.iter = 2500)
samples_matrix <- as.matrix(samples[[1]])
z_cols <- grep("^Z\\[", colnames(samples_matrix), value = TRUE)
Z_est <- apply(samples_matrix[, z_cols], 2, function(x) {
  uniq <- unique(x)
  freq <- tabulate(match(x,uniq))
  mf<-max(freq)
  return(uniq[freq==mf])
 })

colors <- c("black","green" ,"red")

plot(M2, M11, col = colors[Z_est], xlab = "M2", ylab = "M11")
```



**Explanation**:

In the provided code, we first extract the MCMC samples and store them in samples_matrix. Then identify the columns corresponding to each Z[i]. For each column (i.e., each data point), we picks the label with the highest frequency count—this becomes the posterior mode of that data point's cluster label.

Next, we plot the original data (the MFCCs_.2 and MFCCs_.11 coefficients) with points colored by the "true" frog ID (black for frog 1, red for frog 46, green for frog 56). Then we also produce a similar scatter plot, but this time color each point by the posterior mode of its cluster assignment.

By comparing the two plots, we found the colored regions from the posterior mode plot largely match the frog IDs in the "true" label plot, so the model is doing a good job distinguishing between the three frogs based on the MFCC features.