Solutions to BDA Assignment 2, 2020/2021 Semester 2

Chen Tao, s2022911

IMPORTANT: EVERY ANSWER NEEDS TO HAVE AT LEAST ONE SENTENCE OF EXPLANATION CLEARLY DEMONSTRATING THAT YOU UNDERSTAND WHAT YOU ARE DOING. SOLUTIONS WITHOUT EXPLANATION WILL BE MARKED AS 0, IRRESPECTIVELY WHETHER THE COMPUTATIONS ARE CORRECT OR NOT.

FOR THE RESULTS, YOU CAN COPY PASTE THE PLOTS AND NUMBERS FROM KAGGLE. THE SOLUTION HAS TO BE CLEARLY EXPLAINED AND READABLE WITHOUT LOOKING AT YOUR CODE.

BESIDES SUBMITTING THIS REPORT, YOU ARE ALSO REQUIRED TO SUBMIT YOUR IPYNB NOTEBOOK FROM KAGGLE. YOUR CODE HAS TO BE ABLE TO RUN THROUGH ALL QUESTIONS AND REPRODUCE EVERY RESULT IN THIS REPORT.

Please note that the results of the R code for the questions of this assignment may have some (slight) variability over time due to randomness.

1 Problem 1 - Earthquakes in Scotland

Question (a)

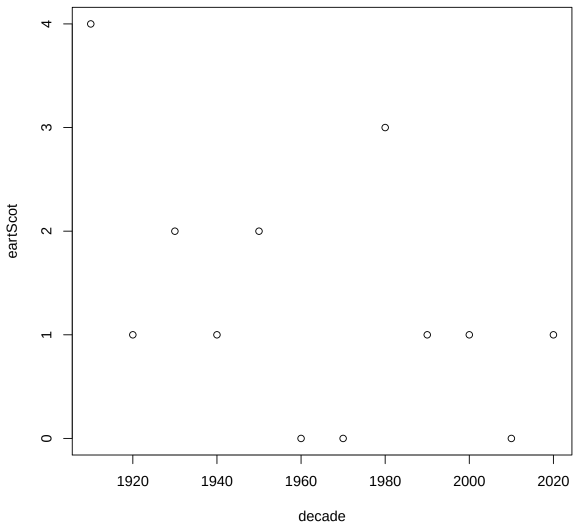
The summary statistics of the data set `earthQ` are shown below.

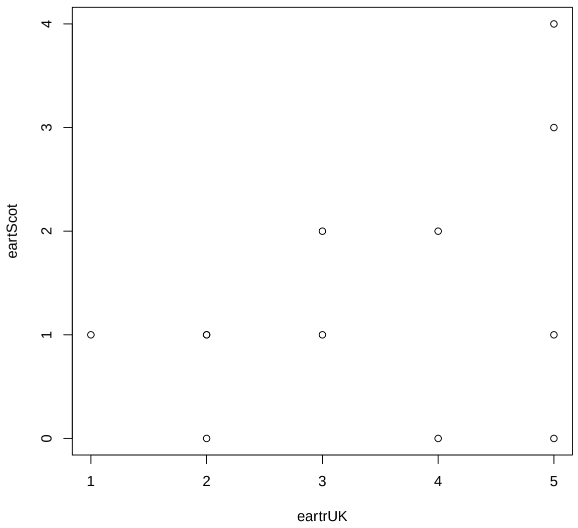
```
summary(earthQ)
```

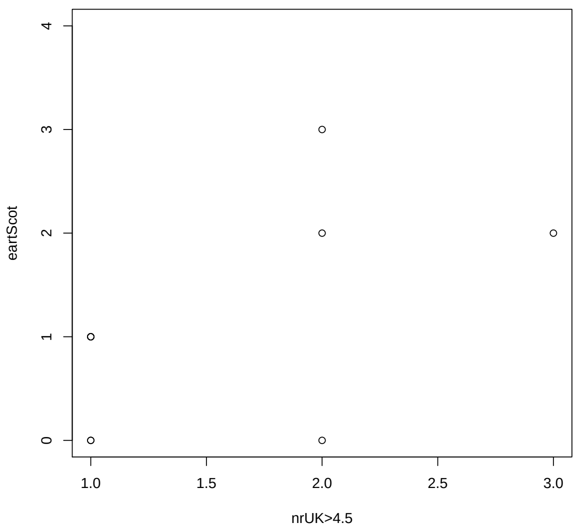
```
     decade         eartScot         eartrUK          MLrUK             dist      
 Min.   :1910   Min.   :0.000   Min.   :1.000   Min.   :4.200   Min.   :  3.0  
 1st Qu.:1938   1st Qu.:0.750   1st Qu.:2.000   1st Qu.:4.355   1st Qu.:120.0  
 Median :1965   Median :1.000   Median :3.500   Median :4.515   Median :165.0  
 Mean   :1965   Mean   :1.333   Mean   :3.417   Mean   :4.510   Mean   :163.8  
 3rd Qu.:1992   3rd Qu.:2.000   3rd Qu.:5.000   3rd Qu.:4.577   3rd Qu.:202.5  
 Max.   :2020   Max.   :4.000   Max.   :5.000   Max.   :5.230   Max.   :380.0  

   nrUK>4.5  
 Min.   :1.0  
 1st Qu.:1.0  
 Median :1.0  
 Mean   :1.5  
 3rd Qu.:2.0  
 Max.   :3.0  
 NA's   :2    
```

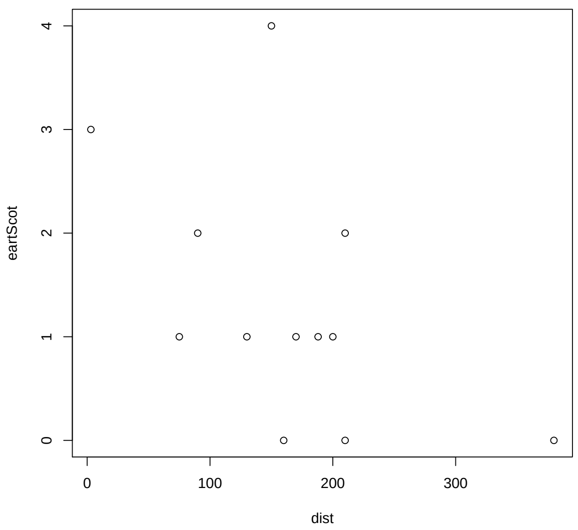
Then, I have created the scatter plot between the response variable `eartScot` and each individual variable in the data set `earthQ` respectively. Furthermore, I also calculate the (Pearson's) correlation coefficient between the response `eartScot` and each covariate in `earthQ`. The R code and the results are shown as follows.

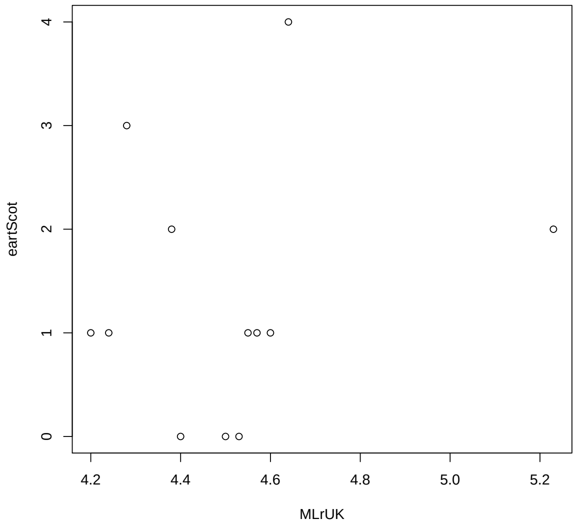
```
attach(earthQ)
plot(decade, eartScot)
plot(eartrUK, eartScot)
plot(`nrUK>4.5`, eartScot)
plot(dist, eartScot)
plot(MLrUK, eartScot)
detach(earthQ)
```











From these scatter plots, we can see that the response `eartScot` has an obvious overall trend of decrease with the increase of the variable `decade`, which means as time pass by, the earthquakes in Scotland are less likely to happen. Then, `eartScot` has an apparent trend of increase with the rising of the variable `earhrUK`, which means when there are more earthquakes in the rest of the UK, there are likely more earthquakes in Scotland. Meanwhile, it seems that the response `eartScot` also has a upward tendency with the increase of the variable `nrUK>4.5`. Moreover, the `eartScot` has an obvious downward trend with the increase of the covariate `dist`, meaning that the farther the distance of the nearest non-Scotish earthquake to the Scottish border, the lower the frequency of the Scotish earthquakes will be. On the other hand, it seems that `eartScot` has no evident trend as the increase of the variable `MLrUK`, which means there is no distinct relationship between these two variables.

```
attach(earthQ)
cor(decade, eartScot)
-0.471124231979254
cor(eartrUK, eartScot)
0.375233260785874
cor(`nrUK>4.5`[is.na(`nrUK>4.5`) == FALSE],
    eartScot[is.na(`nrUK>4.5`) == FALSE])
0.553053040329973
cor(dist, eartScot)
-0.520211680864338
cor(MLrUK, eartScot)
0.169793992826601
detach(earthQ)
```

The results of the calculation of the correlation coefficients are also consistent with the above analysis on scatter plots. We can see that the correlations between `eartScot` and `decade` and between `eartScot` and `dist` are negative, and the correlations between `eartScot` and the other three covariates are all positive. Moreover, the correlation between `eartScot` and `MLrUK` is the lowest and much lower than others. These results indicate that there are negative relationships between `eartScot` and `decade` and between `eartScot` and `dist`, and there are positive relationships between eartScot and eartrUK and between eartScot and nrUK>4.5. Also, there is no evident relationship between eartScot and MLrUK.

Hence, all in all, the covariates decade, eartrUK, nrUK>4.5 and dist might be useful to explain the number of earthquakes in Scotland between 1900 and 2020. Also, it seems that the response eartScot is positively correlated with the covariates eartrUK and nrUK>4.5, and eartScot is negatively correlated with the covariates decade and dist.

Question (ii a)

Under the scenario of generalized linear model of Poisson distribution with log link function, the relationship between the mean μ_i of the response variable and the covariate x_i is

$$\mu_i = \exp(\beta_0 + \beta_1 x_i) = \exp(\beta_0) * \exp(\beta_1 x_i).$$

Therefore, in this situation, if the mean parameter μ_i of the number of earthquakes in Scotland is proportional to the number of nonScot earthquakes $eartrUK_i$, we can obtain the relations that $\lambda = \exp(\beta_0)$, $\beta_1 = 1$ and the covariate $x_i = \log(eartrUK_i)$. The R code for carrying out the frequentist analysis using the glm function, the Poisson family and the default log link function is shown below.

```
glm.model1 = glm(eartScot ~ 1 + offset(log(eartrUK)),
                 family = poisson(link = "log"), data = earthQ)
```

To fix the regression coefficient β_1 to be 1, we can use the offset function in the formula when specifying the glm model. To specify the Poisson family with the default log link function, we need to set the argument family of glm function as poisson(link = "log"). The maximum likelihood estimate (MLE) of the mean parameter μ_i for each decade in the original non-transformed scale can be expressed as

$$\widehat{\mu_i} = \widehat{\lambda} * eartrUK_i = \exp(\widehat{\beta_0}) * eartrUK_i, \quad i = 1, 2, \ldots, 12,$$

where $\widehat{\beta_0}$ is the MLE of the intercept β_0. The MLE of β_0 can be accessed by the command model.glm1$coefficients. Thus, the MLE of the mean parameter μ_i for each decade ($i = 1, 2, \ldots, 12$) in the original non-transformed scale can be calculated by the code below, and the results are also shown in the following.

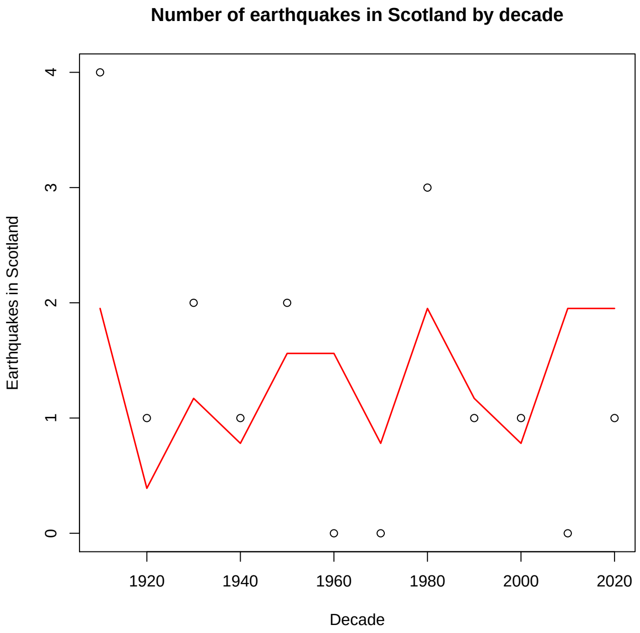
```
MLE = exp(glm.model1$coefficients)*(earthQ$eartrUK)
round(MLE, 5)
1.95122 0.39024 1.17073 0.78049 1.56098 1.56098
0.78049 1.95122 1.17073 0.78049 1.95122 1.95122
```

The expected number of earthquakes in Scotland between 1970 and 1980 can be accessed by the code glm.model1$fitted.values[8] and it is 1.95122.

```
round(glm.model1$fitted.values[8], 5)
1.95122
```

The observed data by decade with the fitted line added in red can be plotted by the R code below, and the figure is shown as follows.

```r
plot(earthQ$decade, earthQ$eartScot, xlab = "Decade",
     ylab = "Earthquakes in Scotland",
     main = "Number of earthquakes in Scotland by decade")
lines(earthQ$decade, glm.model1$fitted.values, col = "red", lwd = 1.5)
```



From the figure, we can see that the fitted line overall depicts the trend of the number of earthquakes in Scotland by decades, although there are some outliers in the figure not accounted by the model. Hence, model (a1) fits the data relatively fine.

Question (b)

By the requirements on the values of μ_a and μ_b of this question, we have

$$\begin{cases} E(\lambda) = \mu_a/\mu_b = 0.5, \\ CV(\lambda) = \sqrt{Var(\lambda)}/E(\lambda) = 1/\sqrt{\mu_a} = 0.5. \end{cases}$$

After solving the equation above, we can obtain $\mu_a = 4$ and $\mu_b = 8$. For the selection of values for τ_a and τ_b, we have

$$\begin{cases} \tau_a = 1/\sigma_a^2 = 1/\log(CV(a)^2 + 1) = 1/\log(1.64), \\ \tau_b = 1/\sigma_b^2 = 1/\log(CV(b)^2 + 1) = 1/\log(1.64). \end{cases}$$

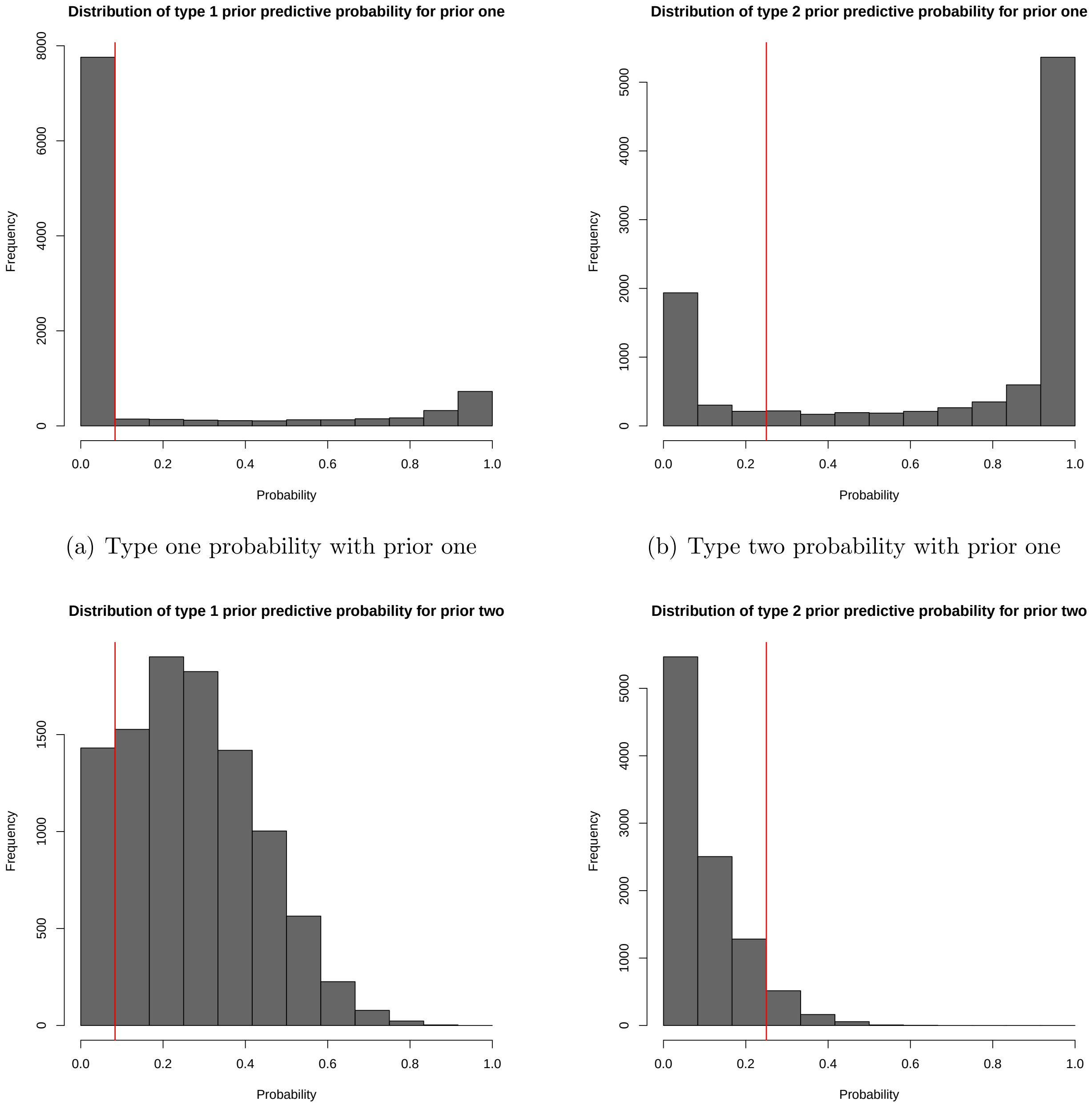
To conduct prior predictive checks, we need to generate replicated data sets of the same size as the original data set from the prior predictive distribution of the response `eartScot`, and then compute the two kinds of probabilities required in this question for each replicated data set and compare with the results derived from the observed data. This process is completely similar with posterior predictive checks, except that we first simulate data from the prior distribution of λ instead of the posterior distribution. To achieve this process, we first take N samples from the three different priors of λ separately. Then, for each prior sample of λ, we take a sample from the distribution of the response eartScot_i, the Poisson distribution with mean parameter $\mu_i = \lambda * \text{eartrUK}_i$, for every decade (i.e. for $i = 1, 2, \ldots, 12$). Now, the samples that we obtain are N samples of the replicated data set of the response eartScot from the prior predictive distribution. The implementation of this process in R is shown below.

```
# Generate replicated data sets for the response `eartScot`
mu.a = 4; mu.b = 8;
sigma.a = sigma.b = sqrt(log(1.64))
tau.a = tau.b = 1/log(1.64)
N = 10000
lambda1 = rgamma(N, 0.1, 0.1)
lambda2 = rgamma(N, 16, 20)
a = rlnorm(N, meanlog = log(mu.a), sdlog = sigma.a)
b = rlnorm(N, meanlog = log(mu.b), sdlog = sigma.b)
lambda3 = rgamma(N, shape = a, rate = b)
n = dim(earthQ)[1]
prior.pred1 = sapply(1:N, function(k){rpois(n, lambda = lambda1[k]*(earthQ$eartrUK))})
prior.pred2 = sapply(1:N, function(k){rpois(n, lambda = lambda2[k]*(earthQ$eartrUK))})
prior.pred3 = sapply(1:N, function(k){rpois(n, lambda = lambda3[k]*(earthQ$eartrUK))})
```

Then, for each replicated data set of eartScot, I calculate the two kinds of probabilities required by this question, the probability for the number of Scottish earthquakes to be greater than 3 and the probability of observing no earthquake during a decade. Then, for each kind of prior of λ and each kind of required probability, I create a histogram for the probabilities obtained from all the replicated data sets, with the red vertical line denoting the result derived from the observed data set. The R code and the figures are shown as follows.

```
# The histograms that show the distribution of the prior
# predictive probability for the number of Scottish earthquakes
# to be greater than 3 for the three priors respectively
# The observed probability for the number of Scottish earthquakes
# to be greater than 3 is shown as the red line
prior.pred1.prob1 = apply(prior.pred1, MARGIN = 2, FUN = function(x){mean(x > 3)})
prior.pred2.prob1 = apply(prior.pred2, MARGIN = 2, FUN = function(x){mean(x > 3)})
prior.pred3.prob1 = apply(prior.pred3, MARGIN = 2, FUN = function(x){mean(x > 3)})
hist(prior.pred1.prob1, breaks = seq(0, 1, 1/12), col = "gray40",
     xlab = "Probability", ylab = "Frequency",
     main = "Distribution of type 1 prior predictive probability for prior one")
abline(v = mean(earthQ$eartScot > 3), col = "red", lwd = 1.5)
hist(prior.pred2.prob1, breaks = seq(0, 1, 1/12), col = "gray40",
     xlab = "Probability", ylab = "Frequency",
     main = "Distribution of type 1 prior predictive probability for prior two")
abline(v = mean(earthQ$eartScot > 3), col = "red", lwd = 1.5)
hist(prior.pred3.prob1, breaks = seq(0, 1, 1/12), col = "gray40",
     xlab = "Probability", ylab = "Frequency",
     main = "Distribution of type 1 prior predictive probability for prior three")
abline(v = mean(earthQ$eartScot > 3), col = "red", lwd = 1.5)

# The histograms that show the distribution of the prior predictive probability
# of observing no earthquake during a decade for the three priors respectively
# The observed probability of observing no earthquake during a decade
# is shown as the red line
```

```
prior.pred1.prob2 = apply(prior.pred1, MARGIN = 2, FUN = function(x){mean(x == 0)})
prior.pred2.prob2 = apply(prior.pred2, MARGIN = 2, FUN = function(x){mean(x == 0)})
prior.pred3.prob2 = apply(prior.pred3, MARGIN = 2, FUN = function(x){mean(x == 0)})
hist(prior.pred1.prob2, breaks = seq(0, 1, 1/12), col = "gray40",
     xlab = "Probability", ylab = "Frequency",
     main = "Distribution of type 2 prior predictive probability for prior one")
abline(v = mean(earthQ$eartScot == 0), col = "red", lwd = 1.5)
hist(prior.pred2.prob2, breaks = seq(0, 1, 1/12), col = "gray40",
     xlab = "Probability", ylab = "Frequency",
     main = "Distribution of type 2 prior predictive probability for prior two")
abline(v = mean(earthQ$eartScot == 0), col = "red", lwd = 1.5)
hist(prior.pred3.prob2, breaks = seq(0, 1, 1/12), col = "gray40",
     xlab = "Probability", ylab = "Frequency",
     main = "Distribution of type 2 prior predictive probability for prior three")
abline(v = mean(earthQ$eartScot == 0), col = "red", lwd = 1.5)
```

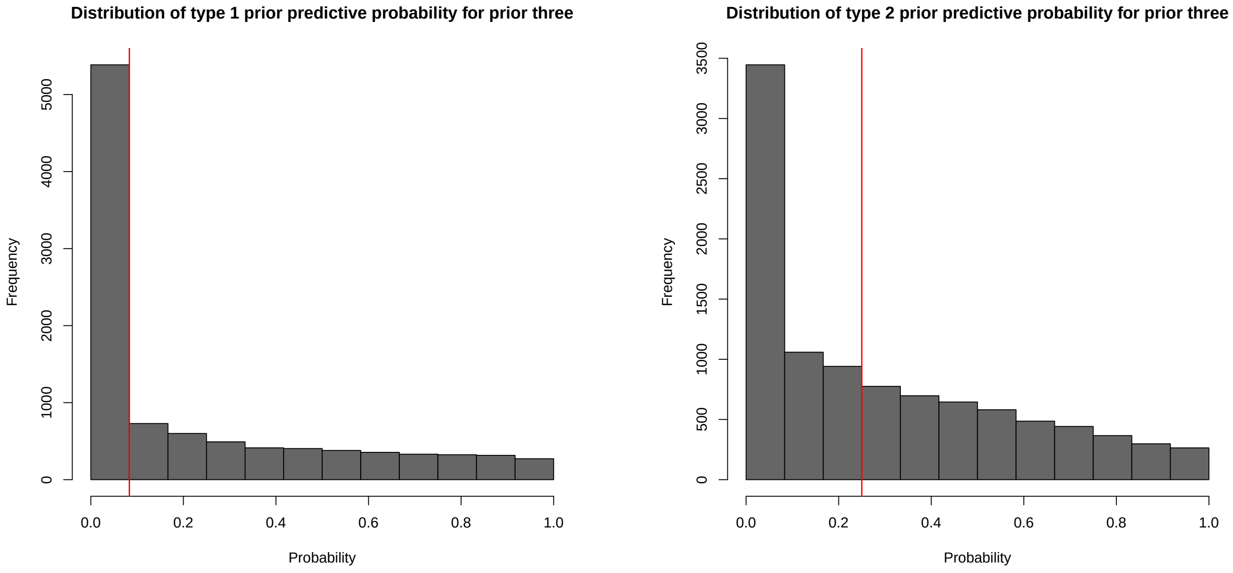


(a) Type one probability with prior one

(b) Type two probability with prior one

(c) Type one probability with prior two

(d) Type two probability with prior two



(e) Type one probability with prior three

(f) Type two probability with prior three

There are six histograms in total. The two figures in each row show the results for each kind of prior of λ, and the three figures in each column display the results of each kind of probability that we are required to calculate for the three different kinds of prior of λ. Because there are only 12 observations in the original data set `earthQ`, all the probabilities calculated here will be a multiple of $1/12$, so I divide the interval $[0, 1]$ into 12 parts equally and each part is the position of a bin of the histogram. The probabilities that fall into the same bin of the histogram are all the same (a multiple of $1/12$), and the value corresponding to the red line is coincide with the value of the probabilities falling in the bin to the right of the red line. From these figures, we can see that under the scenarios of the first and third prior for λ, the distribution of the prior predictive probability for the number of Scottish earthquakes to be greater than 3 is consistent with the observed value, which is concentrated near 0. However, under the scenario of the second prior, the distribution of the prior predictive probability is obviously deviates from the observed value. The mean of it is 0.3056 which is much larger than the observed value 0.0833. For the distribution of the prior predictive probability of observing no earthquake during a decade, the results for the first and second prior are all significantly deviates from the observed value, where the result for the first prior is too large which are concentrated near 1 and the result for the second prior is too small which are concentrated near 0. The result for the third prior is obviously better than the previous two priors, though it is also a little concentrated near 0. Also, under the third prior, the mean of the distribution of the prior predictive probability is 0.3197 which is very near to the observed value 0.25. Hence, all in all, according to the results of the prior predictive checks, the third kind of prior, which is

$$\lambda \sim \text{Gamma}(a, b); \quad a \sim \text{Lognormal}(ln(\mu_a), \tau_a),\ b \sim \text{Lognormal}(ln(\mu_b), \tau_b),$$

is the best prior that presents the prior information acquired from the observed data set, and so I will choose this prior. Now, let me run the model with the third kind of prior for λ using JAGS. The BUGS code for specifying the model and the R code to run JAGS for this model are shown in the following.

```
P1.jags.model.b = "model{

# The likelihood
for (i in 1:n){
  eartScot[i] ~ dpois(mu[i])
  mu[i] <- lambda*eartrUK[i]
}

# The priors for parameters
lambda ~ dgamma(a, b)

# The priors for hyper-parameters
a ~ dlnorm(log(mu.a), tau.a)
b ~ dlnorm(log(mu.b), tau.b)

# The values of hyper-hyperparameters
mu.a <- 4; mu.b <- 8
tau.a <- 1/log(1.64)
tau.b <- 1/log(1.64)
}
"

model.b.data = list(n = dim(earthQ)[1], eartScot = earthQ$eartScot,
                    eartrUK = earthQ$eartrUK)
model.b = jags.model(file = textConnection(P1.jags.model.b),
                     data = model.b.data, n.chain = 2)
update(model.b, n.iter = 5000)
sample.b = coda.samples(model.b, variable.names = c("lambda", "a", "b"),
                        n.iter = 60000, thin = 3)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 12
   Unobserved stochastic nodes: 3
   Total graph size: 41

Initializing model
```

Now, let me perform the model diagnostics to check the convergence and mixing of the chains. At first, I run the model for 20000 iterations with no thinning, but I find that the autocorrelation functions for the model parameters `a` and `b` decrease relatively slow and the effective sample sizes of these two parameters are only 3088.314 and 2947.657 respectively. Therefore, the mixing of the chains for these two parameters are poor. Thus, I increase the number of iterations to 60000 with `thin=3` so that the number of the posterior samples is also 20000 for each parameter. After this change, the results have significantly improved. The autocorrelation functions for these two parameters decrease much faster and the effective sample sizes of them are 8149.093 and 8377.870 respectively. Thereby, the mixing of the chains are good after the adjustments. The following two figures show the changes of the autocorrelation functions for `a` and `b` before and after increasing the number of iterations and the value of the argument `thin`.

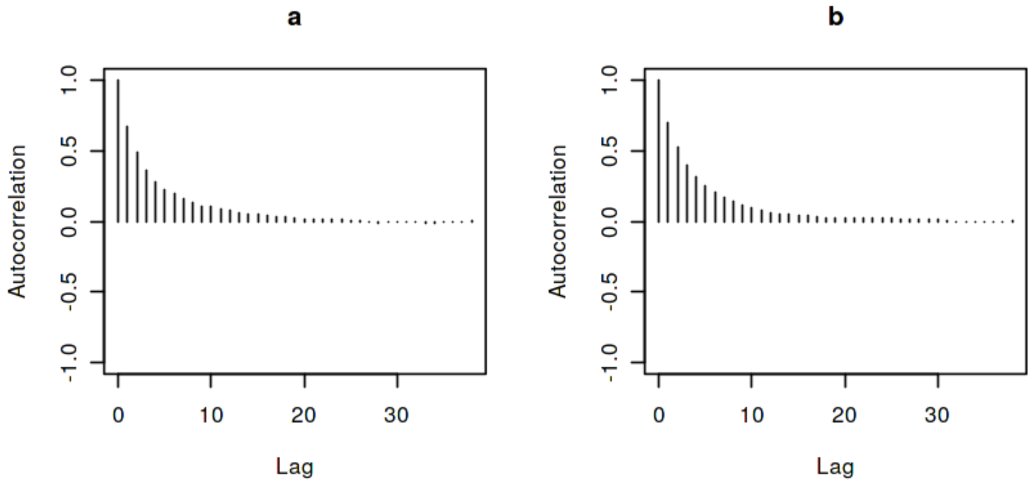


Figure 1: Autocorrelation functions for parameters a and b with 20000 iterations and no thinning

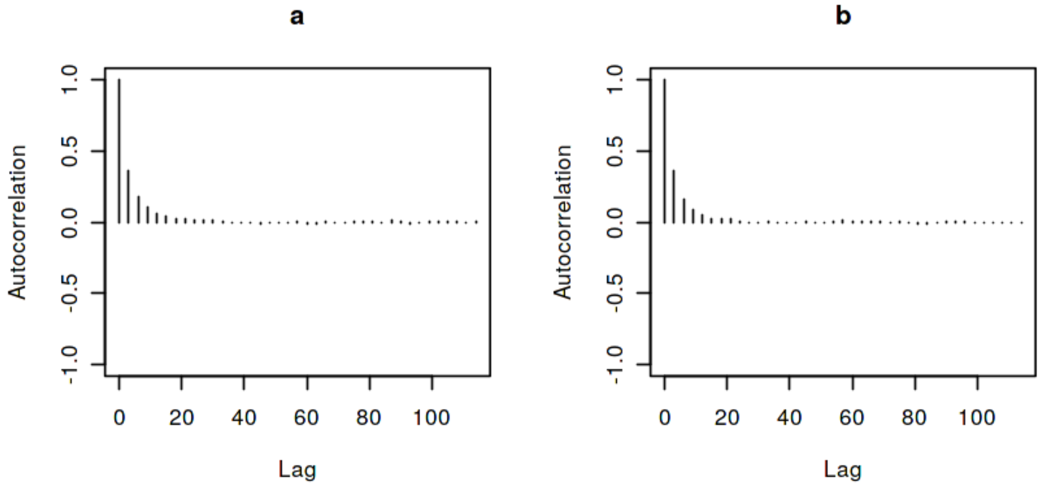


Figure 2: Autocorrelation functions for parameters a and b with 60000 iterations and thin = 3

Furthermore, when setting the number of iterations to be 60000 and the argument `thin` to be 3, I also find that the trace plots for all the parameters vibrate in a certain range with no obvious outliers and the plots of the Gelman-Rubin-Brooks statistics for all the parameters are also extremely close to 1 (much lower than 1.1). These phenomena indicate good convergence of the chains.

Finally, the summary statistics of the posterior samples of the model parameters are shown below. We can see that the ratio between the standard deviation and the MCMC error (Time-series SE) for every parameter of this model is all greater than 20.

```
summary(sample.b)
```

```
Iterations = 6003:66000
Thinning interval = 3
Number of chains = 2
Sample size per chain = 20000
```

```
1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

          Mean      SD  Naive SE Time-series SE
a       4.3396 2.41130 0.0120565      0.0185368
b      10.5719 6.26671 0.0313336      0.0516939
lambda  0.3966 0.09619 0.0004809      0.0004896

2. Quantiles for each variable:

         2.5%    25%    50%     75%   97.5%
a      1.2752 2.6594 3.8173  5.4277 10.4111
b      2.8774 6.2244 9.1506 13.3699 26.3928
lambda 0.2307 0.3286 0.3886  0.4564  0.6063
```

Question (c)

I will implement the model (c1) in JAGS for this question. The model statement in BUGS code is shown as follows. As required by this question, the rate parameter λ is modelled as an exponentiated linear function of the (centered) covariate `dist` using the command `lambda[i] <- exp(beta0 + beta1*(dist[i] - mean(dist[])))`, and I set Gaussian priors with zero mean and variance 5 for the parameters β_0 and β_1.

```
P1.jags.model.c1 = "model{

# The likelihood
for (i in 1:n){
  eartScot[i] ~ dpois(mu[i])
  mu[i] <- lambda[i]*eartrUK[i]
  lambda[i] <- exp(beta0 + beta1*(dist[i] - mean(dist[])))
}

# The priors for parameters
beta0 ~ dnorm(0, tau)
beta1 ~ dnorm(0, tau)

# The hyper-parameters
tau <- 1/5
}
"
```

Then, let me prepare the data for this model, and I will take standard normal random variables as initial values for the parameters β_0 and β_1. Afterwards, I call from R to JAGS using `jags.model` function. After a burn-in period of 5000 iterations, I run two chains for the parameters β_0 and β_1 of this model respectively, where each chain has 20000 iterations. The R code is shown below.

```
model.c1.data = list(n = dim(earthQ)[1], eartScot = earthQ$eartScot,
                     eartrUK = earthQ$eartrUK, dist = earthQ$dist)
model.c1.inits = function(){list(beta0 = rnorm(1, 0, 1), beta1 = rnorm(1, 0, 1))}
model.c1 = jags.model(file = textConnection(P1.jags.model.c1),
                      data = model.c1.data, n.chain = 2)
update(model.c1, n.iter = 5000)
sample.c1 = coda.samples(model.c1, variable.names = c("beta0", "beta1"), n.iter = 20000)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 12
   Unobserved stochastic nodes: 2
   Total graph size: 101

Initializing model
```

Now, let me perform the model diagnostics to check the convergence and mixing of the chains before looking at the summary statistics. I have created the trace plots and density plots, the plots of Gelman-Rubin statistics, the plots of the autocorrelation functions and calculated the effective sample sizes for the parameters β_0 and β_1. I find that the trace plots for all the parameters vibrate in a certain range with no obvious outliers and the plots of the Gelman-Rubin-Brooks statistics for all the parameters are also extremely close to 1 (much lower than 1.1), which all indicate good convergence of the chains. Moreover, the autocorrelation functions for all the parameters decrease quickly, and the effective sample sizes of model parameters β_0 and β_1 in one chain are 7800.335 and 9082.511 respectively, which indicate good mixing of the chains.

```
plot(sample.c1)
gelman.plot(sample.c1)
autocorr.plot(sample.c1[[1]])
round(effectiveSize(sample.c1[[1]]), 5)
```

The summary statistics for the posterior samples of β_0 and β_1 of this model are shown below. We can see that the ratio between the standard deviation and the MCMC error for every parameter of this model is all greater than 20.

```
summary(sample.c1)
```

```
Iterations = 6001:26000
Thinning interval = 1
Number of chains = 2
Sample size per chain = 20000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

           Mean       SD  Naive SE Time-series SE
beta0 -1.103377 0.279739 1.399e-03      2.222e-03
beta1 -0.004357 0.003044 1.522e-05      2.314e-05

2. Quantiles for each variable:
```

```
             2.5%        25%        50%       75%      97.5%
beta0 -1.68451 -1.283369 -1.089428 -0.90855 -0.591243
beta1 -0.01033 -0.006412 -0.004343 -0.00229  0.001575
```

Finally, let me interpret the results using appropriate transformations for the intercept and the regressor. From the calculation by the R code below, we can obtain that the expectation of $\exp \beta_0$ is 0.34469, and the expectation of $\exp \beta_1$ is 0.99565. It means that 0.34469 is the posterior expected ratio between the number of Scottish earthquakes with a Local Magnitude scale ≥ 4 and the number of earthquakes in the rest of the UK, under the condition that the distance of the nearest nonScot earthquake to the Scottish border holds average. Also, for a unit increase in the distance (in miles) of the nearest nonScot earthquake to the Scottish border, the expected number of Scottish earthquakes with a Local Magnitude scale ≥ 4 decreases of 0.99565 times, i.e. a 0.435% decrease, while the variable `eartrUK`, the number of earthquakes in the rest of the UK, holds constant.

```
sample.c1.df = do.call(rbind.data.frame, sample.c1)
cat("E[exp(beta0)] is", round(mean(exp(sample.c1.df$beta0)), 5), "\n")
E[exp(beta0)] is 0.34469
cat("E[exp(beta1)] is", round(mean(exp(sample.c1.df$beta1)), 5))
E[exp(beta1)] is 0.99565
```

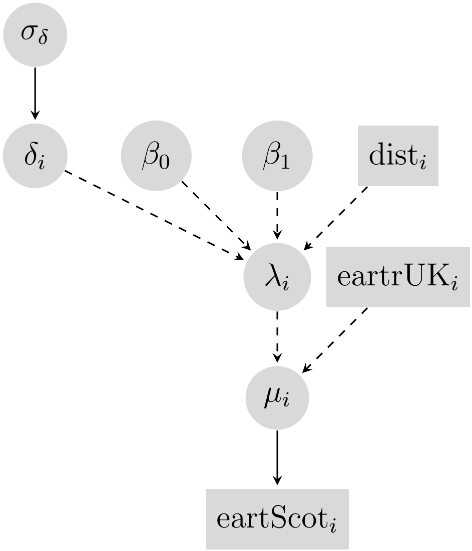
Question (d)

The formula of this new model can be written as follows.

$$
\begin{gathered}
\text{eartScot}_i \mid \mu_i, \mathbf{x}_i \sim \text{Possion}(\mu_i), \quad i = 1, 2 \ldots, n, \\
g(\mu_i) = \lambda_i * \text{eartrUK}_i, \quad i = 1, 2 \ldots, n, \\
\lambda_i = \exp(\beta_0 + \beta_1 * (\text{dist}_i - mean(\text{dist})) + \exp(\delta_i), \quad i = 1, 2 \ldots, n, \\
\beta_j \sim N(0, 5), \quad j = 0, 1, \\
\delta_i \sim N(0, \sigma_\delta^2), \quad i = 1, 2 \ldots, n, \\
\sigma_\delta \sim \text{Uniform}(0, 5),
\end{gathered}
$$

where $n = 12$ is the number of observations in the data set `earthQ`. Note that in the formulas of Bayesian models, the deterministic relations are represented by the equal sign "=", and the stochastic relations are represented by the symbol "$\sim$".

The Direct Acyclic Graph (DAG) representation of this new model is shown as follows (in the next page). Note that in DAG representation of Bayesian models, nodes and edges are of different types. On one hand, stochastic nodes are variables with a distribution that depends stochastically on other nodes. They may be observed (correspond to data), or unobserved (parameters or random effects). On the other hand, deterministic nodes are nodes that are deterministic functions of other nodes. They cannot be observed. For the arrows (edges), there are two types of them. stochastic relationships between nodes are shown as solid arrows, and deterministic relationships between nodes are shown as dashed arrows. Moreover, in DAG representation, the nodes for observed data or constants are represented by squares, while the nodes for parameters and random effects of the model are represented by circles.



Question (e)

According to the illustration of this question, the model that we need to consider in this question is that the number of earthquakes in Scotland comes from a Poisson distribution with unknown mean parameter μ_i, and

$$\mu_i = \lambda * (\text{`}nrUK > 4.5\text{'}_i),$$
$$\lambda = \exp(\beta_0 + \beta_1 * (dist_i - mean(dist)).$$

(1) Model (e1)

Firstly, let me use the conditional mean imputation to impute the missing values in the variable `nrUK>4.5`, which means I will fit a Poisson regression model using the `glm` function and take `nrUK>4.5` as the response variable and take the variable `MLrUK` as the explanatory variable. Then, I use the function `predict` to access the fitted values of this model. Note that I add the argument `type = "response"` to the function `predict`, so that the fitted values of the mean parameter $\mu_i = \exp(\widehat{\beta_0} + \widehat{\beta_1}\text{MLrUK}_i)$ of Poisson regression can be accessed, otherwise the linear predictor $\eta_i = \widehat{\beta_0} + \widehat{\beta_1}\text{MLrUK}_i$ will be returned. Next, I will impute the missing values in `nrUK>4.5` using the fitted values of the mean parameter μ_i, and create a new data set named `earthQ.e1` after imputation. The R code for this imputation process is shown below.

```
# The imputation part for model (e1)
impu.e1 = glm(`nrUK>4.5` ~ MLrUK, data = earthQ, family = poisson)
prediction = predict(impu.e1, newdata = earthQ, type = "response")
`nrUK>4.5.e1` = ifelse(is.na(earthQ$`nrUK>4.5`), prediction, earthQ$`nrUK>4.5`)
earthQ.e1 = earthQ
earthQ.e1$`nrUK>4.5` = `nrUK>4.5.e1`
```

Then, the model statement in JAGS of model (e1) is shown as follows. Please note that because there cannot be greater-than symbol > in variable names in BUGS code, I adjust the name of the variable `nrUK>4.5` to `nrUK_g4.5`.

```
# The model (e1) in JAGS
P1.jags.model.e1 = "model{
```

```
# The likelihood
for (i in 1:n){
  eartScot[i] ~ dpois(mu[i])
  mu[i] <- lambda[i]*nrUK_g4.5[i]
  lambda[i] <- exp(beta0 + beta1*(dist[i] - mean(dist[])))
}

# The priors for parameters
beta0 ~ dnorm(0, tau)
beta1 ~ dnorm(0, tau)

# The hyper-parameters
tau <- 1/5
}
"
```

Now, I will run model (e1) with JAGS and sample from the posterior distributions of β_0 and β_1.

```
# Run model (e1) with JAGS
model.e1.data = list(n = dim(earthQ.e1)[1], eartScot = earthQ.e1$eartScot,
                     nrUK_g4.5 = earthQ.e1$`nrUK>4.5`, dist = earthQ.e1$dist)
model.e1 = jags.model(file = textConnection(P1.jags.model.e1),
                      data = model.e1.data, n.chain = 2)
update(model.e1, n.iter = 5000)
sample.e1 = coda.samples(model.e1, variable.names = c("beta0", "beta1"),
                         n.iter = 20000)
sample.e1.df = do.call(rbind.data.frame, sample.e1)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 12
   Unobserved stochastic nodes: 2
   Total graph size: 101

Initializing model
```

Afterwards, let me perform the model diagnostics to check the convergence and mixing of the chains. I find that the trace plots for all the parameters of the model vibrate in a certain range with no obvious outliers and the plots of the Gelman-Rubin-Brooks statistics for all the parameters are also extremely close to 1 (much lower than 1.1), which all indicate good convergence of the chains. Also, the autocorrelation functions for all the parameters decrease quickly, and the effective sample sizes of model parameters β_0 and β_1 in one chain are 8005.541 and 8069.642 respectively, which indicate good mixing of the chains. Moreover, the summary statistics for this model are shown as follows. We can see that the ratio between the standard deviation and the MCMC error (Time-series SE) for every parameter of this model is all greater than 20.

```
# Model diagnostics and summary
plot(sample.e1)
gelman.plot(sample.e1)
autocorr.plot(sample.e1[[1]])
round(effectiveSize(sample.e1[[1]]), 5)
summary(sample.e1)

Iterations = 6001:26000
Thinning interval = 1
Number of chains = 2
Sample size per chain = 20000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

          Mean       SD  Naive SE Time-series SE
beta0 -0.274309 0.281380 1.407e-03      2.194e-03
beta1 -0.004846 0.002995 1.497e-05      2.331e-05

2. Quantiles for each variable:

         2.5%       25%       50%       75%     97.5%
beta0 -0.86283 -0.455590 -0.259804 -0.078136 0.2407762
beta1 -0.01073 -0.006885 -0.004822 -0.002812 0.0009827
```

(2) Model (e2)

The Bayesian solution that I propose for this question is that I will adopt the model with the same structure of the main model in this question for the number of earthquakes in Scotland, `eartScot`, but I will take the two missing observations of the variable `nrUK>4.5` as unobserved stochastic nodes, just like the model parameters β_0 and β_1, and specify priors on these two stochastic nodes. Then, the posterior samples of these two missing values can be obtained using JAGS. Hence, the imputation process will be automatically included when running the model with JAGS, because we can impute the missing values using their posterior distributions which can quantify the uncertainty about these missing values, or impute them using the means of their posterior distributions respectively. Because in the original data set, the range of `nrUK>4.5` is $\{1, 2, 3\}$, I specify a non-informative prior, the uniform distribution Uniform(0,4), on these two missing values. The model statement in JAGS of model (e2) is shown in the following.

```
# The model (e2) in JAGS
P1.jags.model.e2 = "model{

# The likelihood
for (i in 1:n){
  eartScot[i] ~ dpois(mu[i])
  mu[i] <- lambda[i]*nrUK_g4.5[i]
  lambda[i] <- exp(beta0 + beta1*(dist[i] - mean(dist[])))
}

# The priors for parameters
beta0 ~ dnorm(0, tau)
beta1 ~ dnorm(0, tau)

# The hyper-parameters
tau <- 1/5

# The priors for the missing values of the explanatory variable `nrUK>4.5`
nrUK_g4.5[1] ~ dunif(0, 4)
nrUK_g4.5[4] ~ dunif(0, 4)
}
"
```

Next, let me run model (e1) with JAGS. Because the two missing values in `nrUK>4.5` are stochastic parameters now, I will take samples from the posterior distributions of them, together with the posterior samples of β_0 and β_1.

```
# Run model (e2) with JAGS
model.e2.data = list(n = dim(earthQ)[1], eartScot = earthQ$eartScot,
                     nrUK_g4.5 = earthQ$`nrUK>4.5`, dist = earthQ$dist)
model.e2 = jags.model(file = textConnection(P1.jags.model.e2),
                      data = model.e2.data, n.chain = 2)
update(model.e2, n.iter = 5000)
sample.e2 = coda.samples(model.e2, variable.names = c("beta0", "beta1",
                         "nrUK_g4.5[1]", "nrUK_g4.5[4]"), n.iter = 20000)
sample.e2.df = do.call(rbind.data.frame, sample.e2)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 12
   Unobserved stochastic nodes: 4
   Total graph size: 102

Initializing model
```

Now, let me perform the model diagnostics to check the convergence and mixing of the chains. For the model (e2), I find that the trace plots for all the model parameters vibrate in a certain range with no obvious outliers and the plots of the Gelman-Rubin-Brooks statistics for all the parameters are also extremely close to 1 (much lower than 1.1), which all indicate good convergence of the chains. Also, the autocorrelation functions for all the parameters decrease quickly, and the effective sample sizes for parameters β_0, β_1, `nrUK>4.5[1]` and `nrUK>4.5[4]` in one chain are 7535.551, 8366.515, 7516.810 and 8786.118 respectively which are all above 7000. These phenomena indicate good mixing of the chains. Moreover, the summary statistics for this model are shown as follows. We can see that the ratio between the standard deviation and the MCMC error (Time-series SE) for every parameter of this model is all greater than 20.

```
# Model diagnostics and summary
plot(sample.e2)
```

```
gelman.plot(sample.e2)
autocorr.plot(sample.e2[[1]])
round(effectiveSize(sample.e2[[1]]), 5)
summary(sample.e2)
```

```
Iterations = 6001:26000
Thinning interval = 1
Number of chains = 2
Sample size per chain = 20000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

                   Mean       SD  Naive SE Time-series SE
beta0         -0.379766 0.289776 1.449e-03      0.0023391
beta1         -0.004847 0.003155 1.578e-05      0.0000244
nrUK_g4.5[1]   3.022656 0.719737 3.599e-03      0.0059386
nrUK_g4.5[4]   1.607634 0.969604 4.848e-03      0.0072387

2. Quantiles for each variable:

                  2.5%       25%       50%       75%    97.5%
beta0         -0.98598 -0.568120 -0.367435 -0.176936 0.151041
beta1         -0.01102 -0.006959 -0.004874 -0.002719 0.001359
nrUK_g4.5[1]   1.37098  2.559732  3.165116  3.615538 3.960582
nrUK_g4.5[4]   0.21091  0.822730  1.429086  2.272378 3.723837
```

The posterior means of the two missing values in `nrUK>4.5`, `nrUK>4.5[1]` and `nrUK>4.5[4]`, are 3.023 and 1.608 respectively, which can be used as imputations of them.

```
round(mean(sample.e2.df$`nrUK_g4.5[1]`), 5)
3.02266
round(mean(sample.e2.df$`nrUK_g4.5[4]`), 5)
1.60763
```

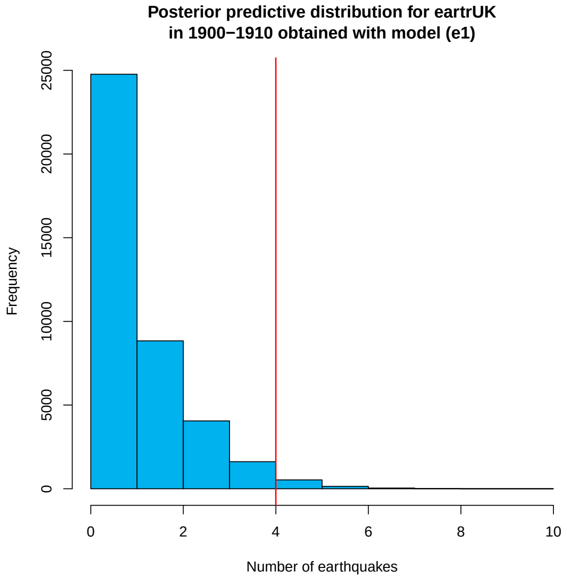
(3) Comparison of posterior predictive distributions obtained with model (e1) and (e2)

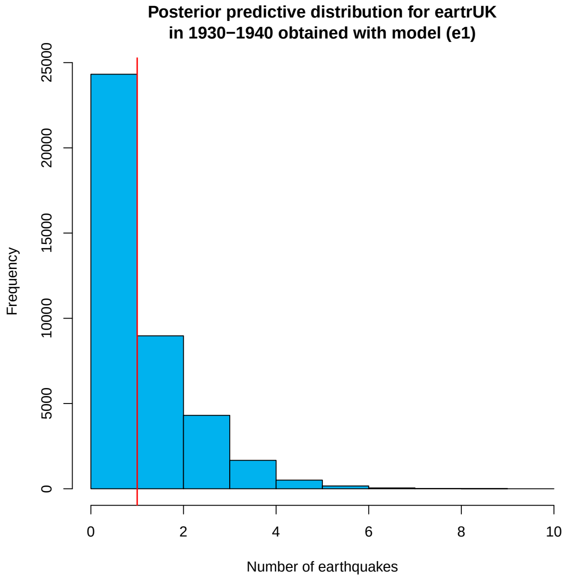
Finally, let me compare how well the posterior predictive distributions for the number of earthquakes in Scotland in the decade 1900-1910 and 1930-1940 obtained with model (e1) and (e2) respectively fit the original data. For the variable `dist`, I use its values in the original data set `earthQ`. However, the treatments on the values of the variable `nrUK>4.5` are different for the models (e1) and (e2). For the model (e1), the imputed values for the missing values of `nrUK>4.5` are used. After taking samples of size N from the posterior distributions of β_0 and β_1, we can obtain the posterior samples of the mean parameter μ_i for the decade 1900-1910 and 1930-1940 respectively by the formula:

$$\mu_i = \exp(\beta_0 + \beta_1 * (dist_i - mean(dist)) * `nrUK > 4.5'_i, \quad i = 1, 4.$$

For each posterior sample of the mean parameter μ_i, I take a sample from the Poisson distribution with mean μ_i. Then, we can obtain N samples in total, and these are the samples from the posterior predictive distribution of the number of earthquakes in Scotland obtained with model (e1). After running the R code below, we can obtain that the means of the posterior predictive distributions of the response eartScot obtained with model (e1) in the decade 1900-1910 and 1930-1940 are 1.35715 and 1.36325 respectively. Also, the two histograms in the following (in the next page) show the posterior predictive distributions of the number of earthquakes in Scotland (eartScot) obtained with model (e1) in 1900-1910 and in 1930-1940 respectively, with the red line indicating the observed number of earthquakes in Scotland. The observed value of eartScot in 1900-1910 is 4 and in 1930-1940 is 1. We can see that the results for the decade 1900-1910 obtained by model (e1) significantly deviate from the observed value of eartScot. However, the results for the decade 1930-1940 obtained by model (e1) are close to the observed value of eartScot. The reason why the results for the decade 1900-1910 have so much deviation is that the imputed value for nrUK>4.5 in this decade is only 1.601, which is relatively small in the data set. For this decade, there is a big gap in the relative size of the response eartScot and the covariate nrUK>4.5.

```
# The posterior predictive distributions for the number of earthquakes in
# Scotland in the decade 1900-1910 and 1930-1940 obtained with model (e1)
dist = earthQ$dist; `nrUK>4.5.e1` = earthQ.e1$`nrUK>4.5`
beta0.e1 = sample.e1.df$beta0; beta1.e1 = sample.e1.df$beta1
N = dim(sample.e1.df)[1]
post.pred.e1.1910 = rpois(n = N, lambda = exp(beta0.e1 +
                          beta1.e1*(dist[1] - mean(dist)))*`nrUK>4.5.e1`[1])
post.pred.e1.1910.mean = mean(post.pred.e1.1910)
round(post.pred.e1.1910.mean, 5)
1.35715
post.pred.e1.1940 = rpois(n = N, lambda = exp(beta0.e1 +
                          beta1.e1*(dist[4] - mean(dist)))*`nrUK>4.5.e1`[4])
post.pred.e1.1940.mean = mean(post.pred.e1.1940)
round(post.pred.e1.1940.mean, 5)
1.36325
hist(post.pred.e1.1910, breaks = seq(0, max(post.pred.e1.1910), 1),
     col = "deepskyblue2", xlab = "Number of earthquakes", ylab = "Frequency",
     main = "Posterior predictive distribution for eartrUK\n
             in 1900-1910 obtained with model (e1)")
abline(v = earthQ$eartScot[1], col = "red", lwd = 1.5)
hist(post.pred.e1.1940, breaks = seq(0, max(post.pred.e1.1940), 1),
     col = "deepskyblue2", xlab = "Number of earthquakes", ylab = "Frequency",
     main = "Posterior predictive distribution for eartrUK\n
             in 1930-1940 obtained with model (e1)")
abline(v = earthQ$eartScot[4], col = "red", lwd = 1.5)
```


Posterior predictive distribution for eartrUK in 1900−1910 obtained with model (e1)


Posterior predictive distribution for eartrUK in 1930−1940 obtained with model (e1)

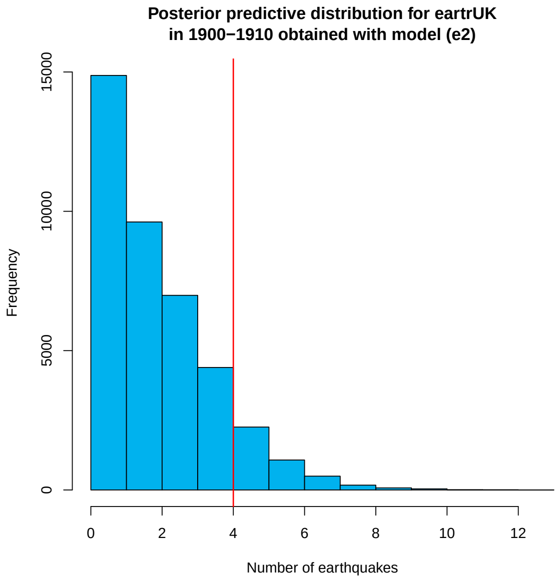
For the model (e2), we also treat the missing values of `nrUK>4.5` as model parameters. Thus, we can take N samples from the posterior distributions of them. For each posterior sample of the parameters `nrUK>4.5[1]`, `nrUK>4.5[4]`, β_0 and β_1, we can obtain a posterior sample of the mean parameter μ_i by the formula:

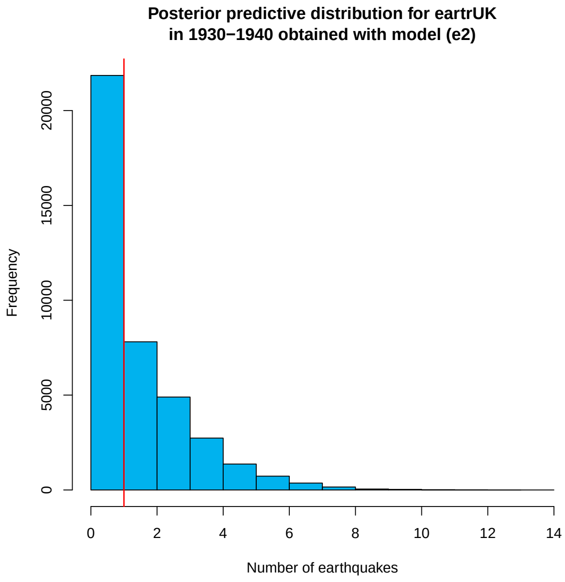
$$\mu_i[k] = \exp(\beta_0[k] + \beta_1[k] * (dist_i - mean(dist)) * \text{`}nrUK > 4.5\text{'}_i[k], \quad i = 1, 4, \quad k = 1, 2, \ldots, N,$$

where $i = 1$ corresponds to the decade 1900-1910 and $i = 4$ corresponds to 1930-1940, and k is the index of each posterior sample. Then, for each posterior sample of μ_i, I take a sample from the Poisson distribution with mean μ_i. Thereby, I can obtain N samples in total from Poisson distribution with mean parameters μ_is, and these are the samples from the posterior predictive distribution of the number of earthquakes in Scotland obtained with model (e2). After the execution of the R code below, we can obtain that the means of the posterior predictive distributions of the response `eartScot` obtained with model (e2) in the decade 1900-1910 and 1930-1940 are 2.28178 and 1.68570 respectively. Also, the two histograms in the following (in the next page) show the posterior predictive distributions of the number of earthquakes in Scotland (`eartScot`) obtained with model (e2) in 1900-1910 and in 1930-1940 respectively, with the red line indicating the observed number of earthquakes in Scotland. Although the results for the decade 1930-1940 of model (e2) are slightly farther from the observed value of this decade than the model (e1), the results for the decade 1900-1910 of model (e2) are much more accurate than those of model (e1). Hence, from the analyses conducted so far, we can conclude that the model (e2) is better than model (e1).

```r
# The posterior predictive distributions for the number of earthquakes in
# Scotland in the decade 1900-1910 and 1930-1940 obtained with model (e2)
N = dim(sample.e2.df)[1]
beta0.e2 = sample.e2.df$beta0; beta1.e2 = sample.e2.df$beta1
`nrUK>4.5[1].e2` = sample.e2.df$`nrUK_g4.5[1]`
`nrUK>4.5[4].e2` = sample.e2.df$`nrUK_g4.5[4]`
post.pred.e2.1910 = rpois(n = N, lambda = exp(beta0.e2 +
                          beta1.e2*(dist[1] - mean(dist)))*`nrUK>4.5[1].e2`)
post.pred.e2.1910.mean = mean(post.pred.e2.1910)
round(post.pred.e2.1910.mean, 5)
```

```
2.28178
post.pred.e2.1940 = rpois(n = N, lambda = exp(beta0.e2 +
                          beta1.e2*(dist[4] - mean(dist)))*`nrUK>4.5[4].e2`)
post.pred.e2.1940.mean = mean(post.pred.e2.1940)
round(post.pred.e2.1940.mean, 5)
1.6857
hist(post.pred.e2.1910, breaks = seq(0, max(post.pred.e2.1910), 1),
     col = "deepskyblue2", xlab = "Number of earthquakes", ylab = "Frequency",
     main = "Posterior predictive distribution for eartrUK\n
             in 1900-1910 obtained with model (e2)")
abline(v = earthQ$eartScot[1], col = "red", lwd = 1.5)
hist(post.pred.e2.1940, breaks = seq(0, max(post.pred.e2.1940), 1),
     col = "deepskyblue2", xlab = "Number of earthquakes", ylab = "Frequency",
     main = "Posterior predictive distribution for eartrUK\n
             in 1930-1940 obtained with model (e2)")
abline(v = earthQ$eartScot[4], col = "red", lwd = 1.5)
```



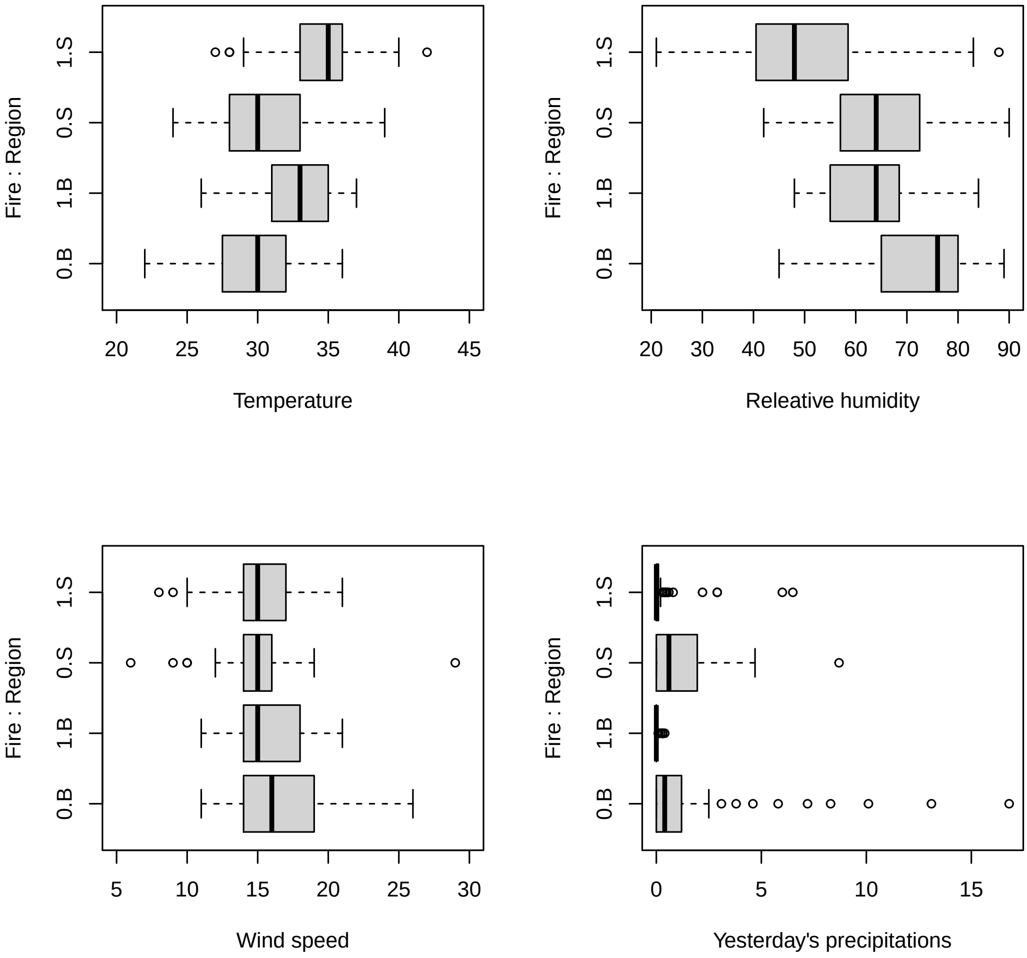


2 Problem 2 - Wildfires in Algerian forests

Question (a)

For this question, I am going to explore the relationship between Fire and the four meterological variables in the two Algerian regions using boxplots. To create a plot with 4 panels, I will use the `par` function to adjust graphical setting, where I will set the argument `mfrow = c(2, 2)` in this function to make the 4 panels form a 2 × 2 matrix. For each panel, I will use the function `boxplot` with the argument setting `formula = X ~ Fire*Region`, where `X` is one of the four meterological variables, `TempCels`, `RH`, `Ws` and `RainYest`, to produce 4 boxplots to show the distribution of the meterological variable `X`, one for each combination of Fire and Region. The R code and the figure are shown below.

```r
par(mfrow = c(2, 2))
boxplot(formula = TempCels ~ Fire*Region, data = AlgForst,
        horizontal = TRUE, xlab = "Temperature", ylim = c(20, 45))
boxplot(formula = RH ~ Fire*Region, data = AlgForst,
        horizontal = TRUE, xlab = "Releative humidity")
boxplot(formula = Ws ~ Fire*Region, data = AlgForst,
        horizontal = TRUE, xlab = "Wind speed", ylim = c(5, 30))
boxplot(formula = RainYest ~ Fire*Region, data = AlgForst,
        horizontal = TRUE, xlab = "Yesterday's precipitations")
graphics.off()
```



From the plot, we can see that the presence of a wildfire in the two Algerian regions has significant relationship with the three meterological variables temperature (`TempCels`), releative humidity (`RH`) and rain (`RainYest`). In both regions of Bejaia and Sidi Bel-abbes, the cases with presence of a wildfire usually have obviously higher temperatures than the cases with absence of a wildfire. Also, in both regions, daily releative humidity and rain levels of the cases with presence of a wildfire are overall significantly less than those of the cases with absence of a wildfire. Moreover, it is worth noting that in both regions the yesterday's precipitations of almost all the cases with presence of a wildfire are almost 0. However, on the other hand, it seems that there is no distinct difference on the daily wind speed between the cases with presence of a wildfire and the cases with absence of a wildfire in both regions. Hence, all in all, the boxplots suggest that in both two Algerian regions, the covariate temperature (`TempCels`) has positive influence on the probability of a fire, and the covariates daily releative humidity (`RH`) and yesterday's precipitations (`RainYest`) have negative influence on the probability of a fire, and there is no evident relationship between the probability of a fire and the variable daily wind speed (`Ws`).

Question (b)

Firstly, let me create a data frame named `AlgForst.center` that has the same variables as the original data set `AlgForst`, where the variables temperature (`TempCels`), relative humidity (`RH`), wind speed (`Ws`), rain (`RainYest`) and Duff Moisture Code (`DMC`) are centered and the other three variables, `month`, `Fire` and `Region`, remain the same.

```
AlgForst.center = data.frame(
  month = AlgForst$month, TempCels = AlgForst$TempCels - mean(AlgForst$TempCels),
  RH = AlgForst$RH - mean(AlgForst$RH), Ws = AlgForst$Ws - mean(AlgForst$Ws),
  AlgForst$DMC - mean(AlgForst$DMC), Fire = AlgForst$Fire,
  Region = AlgForst$Region, RainYest = AlgForst$RainYest - mean(AlgForst$RainYest)
)
```

Then, I use the `glm` function to model the probability of a fire using a Bernoulli logistic model with centered covariates, temperature, relative humidity, wind speed and rain. Because there is not a "Bernoulli" family for the `glm` function, I set the argument `family` of `glm` to be `binomial(link = "logit")`, which means I use a "binomial" family and the *logit* link function, and the effect here is the same with the Bernoulli logistic model. The summary statistics of the model are shown below.

```
glm.model2 = glm(Fire ~ TempCels + RH + Ws + RainYest, data = AlgForst.center,
                 family = binomial(link = "logit"))
summary(glm.model2)
```

```

Call:
glm(formula = Fire ~ TempCels + RH + Ws + RainYest, family = binomial(link = "logit"),
    data = AlgForst.center)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3803  -0.7632   0.3567   0.7718   2.8495

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
```

```
(Intercept)  0.19503    0.17999   1.084 0.278547    
TempCels     0.32031    0.06676   4.798  1.6e-06 ***
RH          -0.03186    0.01461  -2.181 0.029181 *  
Ws           0.07455    0.06474   1.152 0.249475    
RainYest    -0.66590    0.19460  -3.422 0.000622 ***
---
Signif. codes:  
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 334.05  on 243  degrees of freedom
Residual deviance: 232.02  on 239  degrees of freedom
AIC: 242.02

Number of Fisher Scoring iterations: 6
```

Then, I define the inverse function *ilogit* of the *logit* function, i.e. $ilogit(x) = 1/(1+\exp(-x))$. Afterwards, I use the *ilogit* function to transform the intercept β_0 and use exponential transformations for the regressors of this GLM. The results are shown as follows.

```
ilogit = function(x){1/(1 + exp(-x))}
cat("E(ilogit(beta0)) is", round(ilogit(glm.model2$coefficients["(Intercept)"]), 5))
E(ilogit(beta0)) is 0.5486
cat("E(exp(beta1)) is", round(exp(glm.model2$coefficients["TempCels"]), 5))
E(exp(beta1)) is 1.37755
cat("E(exp(beta2)) is", round(exp(glm.model2$coefficients["RH"]), 5))
E(exp(beta2)) is 0.96864
cat("E(exp(beta3)) is", round(exp(glm.model2$coefficients["Ws"]), 5))
E(exp(beta3)) is 1.0774
cat("E(exp(beta4)) is", round(exp(glm.model2$coefficients["RainYest"]), 5))
E(exp(beta4)) is 0.51381
```

Denote p be the probability of the presence of a wildfire in Algerian regions and then the odds of the presence of a wildfire in Algerian regions is defined as $odds = p/(1-p)$. Then, the above results about the transformed regression coefficients (the intercept and the regressors of the GLM) can be interpreted as follows.

- The expected probability of the presence of a wildfire in Algerian regions, when the four meterological conditions of temperature, relative humidity, wind speed and rain are all at average levels, is 0.5486.
- For a one-unit increase in the meterological condition of temperature, the expected odds of the presence of a wildfire in Algerian regions increases of 1.37755 times, i.e. a 37.755% increase, while the other meterological variables remain constant.
- For a one-unit increase in the meterological condition of relative humidity, the expected odds of the presence of a wildfire in Algerian regions decreases of 0.96864 times, i.e. a 3.136% decrease, while the other meterological variables remain constant.

- For a one-unit increase in the meterological condition of wind speed, the expected odds of the presence of a wildfire in Algerian regions increases of 1.0774 times, i.e. a 7.74% increase, while the other meterological variables remain constant.

- For a one-unit increase in the meterological condition of rain, the expected odds of the presence of a wildfire in Algerian regions decreases of 0.51381 times, i.e. a 48.619% decrease, while the other meterological variables remain constant.

To calculate the expected probability of a fire when temperature = 35, relative humidity = 40, wind speed = 18 and rain = 1.0, let me first center these conditions and create a data frame named `newdata` to contain these data. Then, the expected probability at these conditions can be obtained by applying the function `predict`, and we should set the argument `type = "response"` for this function. Finally, according to the result shown below, the expected probability of a fire when temperature = 35, relative humidity = 40, wind speed = 18 and rain = 1.0 is 0.86128.

```
newdata = data.frame(
  TempCels = 35 - mean(AlgForst$TempCels), RH = 40 - mean(AlgForst$RH),
  Ws = 18 - mean(AlgForst$Ws), RainYest = 1.0 - mean(AlgForst$RainYest)
)
round(predict(glm.model2, newdata = newdata, type = "response"), 5)
0.86128
```

Question (c)

The model string that contains the BUGS code for specifying the corresponding Bayesian Bernoulli logistic model about the analyses in (b) is shown as follows. In the BUGS code, I use `dbern` to specify the Bernoulli distributions, and center all the four meterological variables, `TempCels`, `RH`, `Ws` and `RainYest`. Furthermore, there is built-in `logit` function ($logit(x) = \log(x/(1-x))$) in BUGS language.

```
# The Bayesian Bernoulli logistic model in JAGS for Question (c)
P2.jags.model.c = "model{

  # The likelihood
  for (i in 1:n){
    Fire[i] ~ dbern(p[i])
    logit(p[i]) <- beta0 + beta1*(TempCels[i] - mean(TempCels[])) +
                   beta2*(RH[i] - mean(RH[])) + beta3*(Ws[i] - mean(Ws[])) +
                   beta4*(RainYest[i] - mean(RainYest[]))
  }

  # The priors
  beta0 ~ dnorm(0, tau)
  beta1 ~ dnorm(0, tau)
  beta2 ~ dnorm(0, tau)
  beta3 ~ dnorm(0, tau)
  beta4 ~ dnorm(0, tau)

  # The hyperparameters
  tau <- 1/10
}
"
```

Now, I will run the model using JAGS for 3 Markov chains with initial values drawn from the standard normal distribution, and I will run 20000 iterations for each chain after a burn-in period of 5000 iterations. The summary of the model is shown as follows.

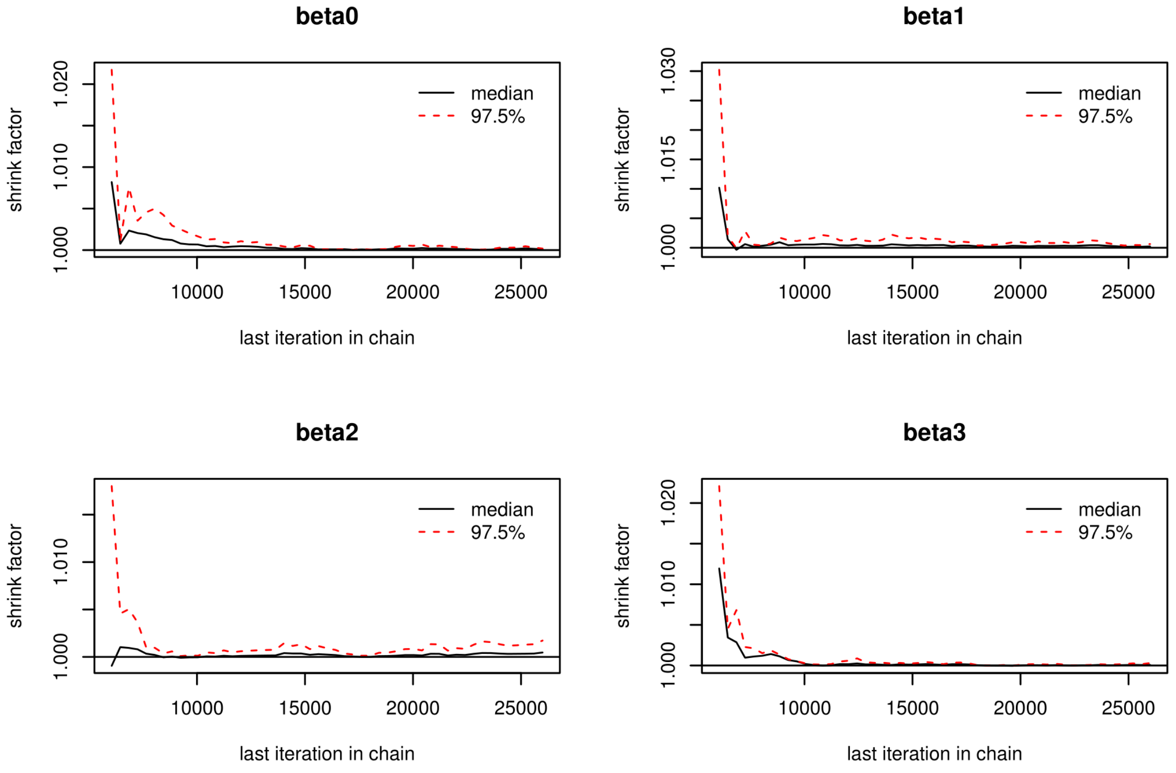
```
# Call from R to JAGS
model.c.data = list(n = dim(AlgForst)[1], Fire = AlgForst$Fire,
                    TempCels = AlgForst$TempCels, RH = AlgForst$RH,
                    Ws = AlgForst$Ws, RainYest = AlgForst$RainYest)
model.c.inits = function(){
  list(beta0 = rnorm(1, 0, 1), beta1 = rnorm(1, 0, 1), beta2 = rnorm(1, 0, 1),
       beta3 = rnorm(1, 0, 1), beta4 = rnorm(1, 0, 1))
}
model.c = jags.model(file = textConnection(P2.jags.model.c), data = model.c.data,
                     inits = model.c.inits, n.chains = 3)
update(model.c, n.iter = 5000)
sample.c = coda.samples(model.c, variable.names = c("beta0", "beta1",
                        "beta2", "beta3", "beta4"), n.iter = 20000)
summary(sample.c)
```

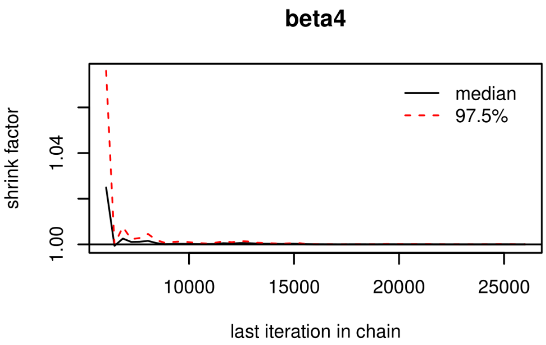
```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 244
   Unobserved stochastic nodes: 5
   Total graph size: 1998

Initializing model


Iterations = 6001:26000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 20000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

          Mean      SD  Naive SE Time-series SE
beta0  0.18085 0.18354 7.493e-04      1.190e-03
beta1  0.33241 0.06816 2.783e-04      4.605e-04
beta2 -0.03247 0.01499 6.119e-05      9.937e-05
beta3  0.07687 0.06575 2.684e-04      3.825e-04
beta4 -0.71243 0.20129 8.218e-04      1.300e-03

2. Quantiles for each variable:
```

```
          2.5%       25%       50%       75%     97.5%
beta0 -0.18151  0.05891  0.18217  0.30490  0.535842
beta1  0.20473  0.28554  0.33061  0.37704  0.471240
beta2 -0.06237 -0.04241 -0.03238 -0.02228 -0.003571
beta3 -0.05144  0.03277  0.07646  0.12053  0.207786
beta4 -1.13675 -0.84079 -0.69916 -0.57268 -0.350734
```

Recall that the estimates of the parameters β_0, β_1, β_2, β_3 and β_4 obtained using the `glm` function in (b) are 0.195, 0.320, -0.032, 0.075, -0.666 respectively. From the summary of the posterior samples of β_0, β_1, β_2, β_3 and β_4, we can see that the posterior means of these parameters are very close to the estimates of these parameters obtained by the frequentist method. Especially for the parameters β_2 and β_3, the posterior means of them are essentially the same with the estimates of them obtained by `glm` function.

Now, let me perform model diagnostics to check convergence and mixing of the chain. I have created the traceplots and the density plots, the plots of Gelman-Rubin-Brooks statistics, the plots of autocorrelation functions for all the parameters of the model, and I also calculate the effective samples sizes of all the parameters. In the trace plots for all the parameters β_0, β_1, β_2, β_3 and β_4, we can see that the plots of all the three Markov chains vibrate in a certain range with no obvious outliers, which indicates good convergence of the chains. The plots of Gelman-Rubin-Brooks statistics for all the parameters are shown as follows. From these figures, we can see that the Gelman-Rubin-Brooks statistics for all the model parameters β_i $(i = 0, 1, 2, 3, 4)$ are extremely close to 1 (much lower than 1.1) after the burn-in period of 6000 iterations. These phenomena also indicate good convergence of the chains. Also, the autocorrelation functions for all the parameters decrease quickly, and the effective sample sizes for all the parameters are above 7000, which indicate good mixing of the chains. Furthermore, I also calculate by the summary statistics of the model that the ratios between the MC error and the posterior standard deviation for the parameters β_0, β_1, β_2, β_3, β_4 are 0.00649, 0.00676, 0.00663, 0.00582, 0.00646 respectively, which are all significantly less than 0.05. Thus, the number of MCMC samples that we collect is sufficient.





Finally, let me compute the posterior expected probability of a fire when Temperature = 35, Relative humidity = 40, Wind Speed = 18 and Rain = 1.0 using the JAGS output (the posterior samples of the parameters β_i $(i = 0, 1, 2, 3, 4)$) obtained in the previous part of this question. Firstly, I center these meterological conditions and store them in a new data frame named `newdata`. In the Bernoulli logistic model of this question, the relationship between the expected probability p of a fire and the covariates, i.e. the four meterological variables, `TempCels`, `RH`, `Ws` and `RainYest`, is

$$\begin{aligned} p =& ilogit(\beta_0 + \beta_1 \text{TempCels} + \beta_2 \text{RH} + \beta_3 \text{Ws} + \beta_4 \text{RainYest}) \\ =& \frac{1}{1 + \exp(-(\beta_0 + \beta_1 \text{TempCels} + \beta_2 \text{RH} + \beta_3 \text{Ws} + \beta_4 \text{RainYest}))}. \end{aligned}$$

I use this relationship to obtain posterior samples of the expected probability p of a fire at these specific meterological conditions given in this question. Then, the posterior mean of the expected probability p of a fire at the given meterological conditions can be computed, and using the R function `quantile`, the 90% credible interval for the expected probability p of a fire can also be obtained. The R code and the results are shown below.

```r
# The posterior distribution of the probability of a fire when
# Temperature = 35, Relative humidity = 40, Wind Speed = 18 and Rain = 1.0
sample.c.df = do.call(rbind.data.frame, sample.c)
ilogit = function(x){1/(1 + exp(-x))}
newdata = data.frame(
  TempCels = 35 - mean(AlgForst$TempCels), RH = 40 - mean(AlgForst$RH),
  Ws = 18 - mean(AlgForst$Ws), RainYest = 1.0 - mean(AlgForst$RainYest)
)
post.samples = ilogit(sample.c.df$beta0 +
  (sample.c.df$beta1)*(newdata$TempCels) + (sample.c.df$beta2)*(newdata$RH) +
  (sample.c.df$beta3)*(newdata$Ws) + (sample.c.df$beta4)*(newdata$RainYest))
# The posterior mean
round(mean(post.samples), 5)
0.85524
# The 90% credible interval
round(quantile(post.samples, 0.05), 5)
5%: 0.75227
round(quantile(post.samples, 0.95), 5)
95%: 0.93167
```

The posterior samples of the expected probability of a fire are stored by the variable `post.samples`. From the results, we can see that the posterior mean of the expected probability of a fire at the given meterological conditions is 0.85524, and the 90% credible interval for the expected probability of a fire is [0.75227, 0.93167].

Question (d)

(1) The analyses of model (d1)

Because the variable Region is a categorical variable, if we want to include it as a covariate in the regression model, we need to create a dummy (indicator) variable. The effect of this dummy variable is to shift the model intercept up or down. For this question, I use a dummy variable named Sidi.Ind to mark different regions, where Sidi.Ind takes value 1 for the Sidi Bel-abbes region and it takes value 0 for the Bejaia region. The priors for the intercept β_0 of the observations from the Bejaia region and the coefficients beta.Sidi of the dummy variable Sidi.Ind and all the other parameters β_i $(i = 1, 2, 3, 4)$ are all the normal distribution with zero mean and precision being equal to 0.1. The model statement in JAGS of model (d1) is shown as follows.

```
# The model d1 in JAGS
P2.jags.model.d1 = "model{

  # The likelihood
  for (i in 1:n){
    Fire[i] ~ dbern(p[i])
    logit(p[i]) <- beta0 + beta1*(TempCels[i] - mean(TempCels[])) +
                   beta2*(RH[i] - mean(RH[])) + beta3*(Ws[i] - mean(Ws[])) +
                   beta4*(RainYest[i] - mean(RainYest[])) + beta.Sidi*Sidi.Ind[i]
  }

  # The priors
  beta0 ~ dnorm(0, tau)
  beta1 ~ dnorm(0, tau)
  beta2 ~ dnorm(0, tau)
  beta3 ~ dnorm(0, tau)
  beta4 ~ dnorm(0, tau)
  beta.Sidi ~ dnorm(0, tau)

  # The hyperparameters
  tau <- 1/10
}
"
```

Then, I will run the model using JAGS and sample from the posterior distributions of all the model parameters. I will run three Markov chains and take initial values from the standard normal distribution for all the model parameters. At first, I run the model for 20000 iterations with no thinning, but I find that the autocorrelation functions for the parameters β_0 and beta.Sidi decrease relatively slow and the effective sample sizes of these two parameters are only 3393.498 and 3747.983 respectively. Therefore, the mixing of the chains for these two parameters are relatively poor. Thus, I increase the number of iterations to 60000 with thin=3 so that the number of the posterior samples is also 20000 for each parameter. After this change, the results has significantly improved. The autocorrelation functions for these two parameters decrease much faster and the effective sample sizes of them are 8874.956 and 9675.913 respectively. The following two figures show the changes before and after increasing the number of iterations and the value of the argument thin.

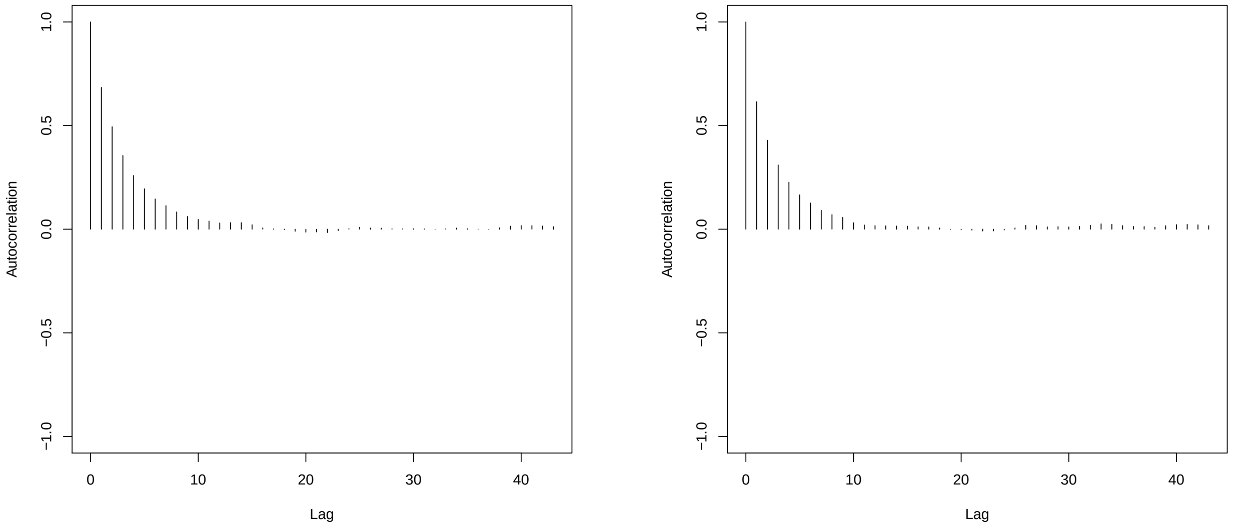


Figure 3: Autocorrelation functions for parameters beta0 and beta.Sidi with 20000 iterations and no thinning

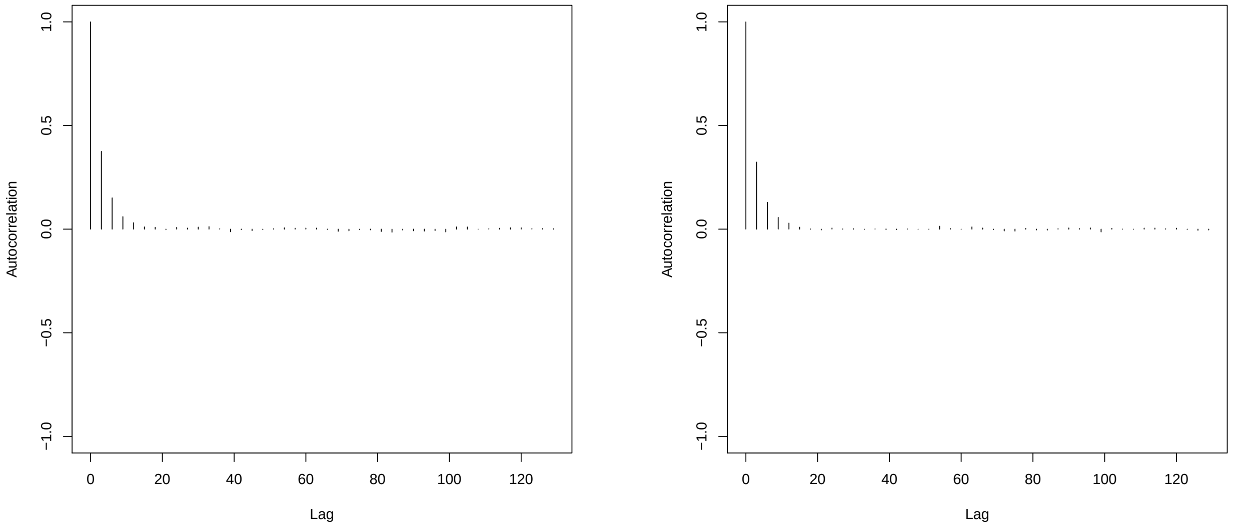


Figure 4: Autocorrelation functions for parameters beta0 and beta.Sidi with 60000 iterations and thin = 3

When running the rest models of this assignment using JAGS, I will take the same model settings to keep consistency, which means I will run 3 chains with 60000 iterations and `thin=3` for each chain. Furthermore, for the model (d1), I also find that the traceplots for all the chains of all the parameters vibrate in a certain range with no obvious outliers and the plots of the Gelman-Rubin-Brooks statistics for all the parameters are extremely close to 1 after the burn-in period of 6000 iterations, which all indicate good convergence of the chains.

The R code for running the model and the summary statistics of the posterior samples of all the model parameters are shown below, and we can see that the ratio between the standard deviation and the MCMC error (Time-series SE) for every parameter of this model is all greater than 20.

```
# Call from R to JAGS
AlgForst.d1 = AlgForst; AlgForst.d1$Sidi.Ind = ifelse(AlgForst$Region == "S", 1, 0)
model.d1.data = list(n = dim(AlgForst.d1)[1], Fire = AlgForst.d1$Fire,
                     TempCels = AlgForst.d1$TempCels, RH = AlgForst.d1$RH,
                     Ws = AlgForst.d1$Ws, RainYest = AlgForst.d1$RainYest,
                     Sidi.Ind = AlgForst.d1$Sidi.Ind)
model.d1.inits = function(){
  list(beta0 = rnorm(1, 0, 1), beta1 = rnorm(1, 0, 1), beta2 = rnorm(1, 0, 1),
       beta3 = rnorm(1, 0, 1), beta4 = rnorm(1, 0, 1), beta.Sidi = rnorm(1, 0, 1))
}
model.d1 = jags.model(file = textConnection(P2.jags.model.d1), data = model.d1.data,
                      inits = model.d1.inits, n.chains = 3)
update(model.d1, n.iter = 5000)
sample.d1 = coda.samples(model.d1, variable.names = c("beta0", "beta1", "beta2",
                         "beta3", "beta4", "beta.Sidi"), n.iter = 60000, thin = 3)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 244
   Unobserved stochastic nodes: 6
   Total graph size: 2245

Initializing model
```

```
summary(sample.d1)
```

```
Iterations = 6003:66000
Thinning interval = 3
Number of chains = 3
Sample size per chain = 20000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

              Mean      SD  Naive SE Time-series SE
beta.Sidi  0.38184 0.35331 1.442e-03      2.050e-03
beta0     -0.01149 0.25691 1.049e-03      1.565e-03
beta1      0.33832 0.06894 2.815e-04      3.119e-04
beta2     -0.02810 0.01550 6.329e-05      7.564e-05
beta3      0.08295 0.06631 2.707e-04      2.792e-04
beta4     -0.76034 0.20864 8.518e-04      1.008e-03

2. Quantiles for each variable:

              2.5%      25%      50%      75%    97.5%
beta.Sidi -0.31701  0.14421  0.38085  0.61923  1.074895
beta0     -0.51596 -0.18408 -0.01097  0.16181  0.488530
```

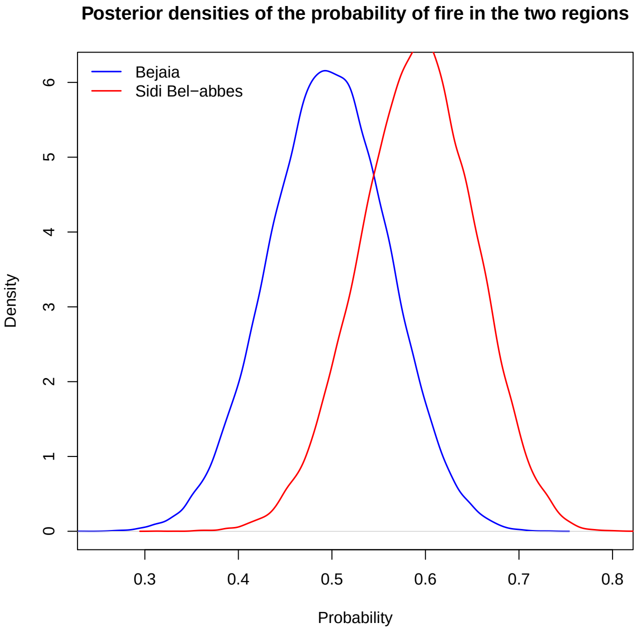
```
beta1       0.20737  0.29124  0.33617  0.38362  0.477757
beta2      -0.05909 -0.03837 -0.02800 -0.01767  0.001789
beta3      -0.04624  0.03815  0.08274  0.12754  0.213916
beta4      -1.20316 -0.89289 -0.74860 -0.61494 -0.384919
```

Average metereological conditions mean that all the four metereological conditions temperature (`TempCels`), relative humidity (`RH`), wind speed (`Ws`) and rain (`RainYest`) should take values of their means, `mean(TempCels)`, `mean(RH)`, `mean(Ws)` and `mean(RainYest)`. Because in all the three Bayesian logistic models of Question (d), model (d1), (d2) and (d3), the four covariates of metereological conditions are centered, we should take the value 0 for them in all the three regression models when at average metereological conditions. The model (d1) at average metereological conditions can be simplified as

$$logit(p) = \beta_0 + beta.Sidi * Sidi.Ind, \tag{1}$$

where p is the probability of a wildfire and $Sidi.Ind$ is the indicator variable of the region Sidi Bel-abbes. After the execution of the R code below, we can obtain that the means of the posterior expected probabilities of fire, i.e. the posterior mean of the probability parameter p, in the Bejaia and Sidi Bel-abbes regions at average metereological conditions are 0.49667 and 0.59067 respectively, and the two densities of the posterior expected probabilities of fire in the two regions are also plotted in the figure below (in the next page).

```
# The posterior expected probabilities of fire (at average metereological conditions)
# in the Bejaia and Sidi Bel-abbes regions
ilogit = function(x){1/(1 + exp(-x))}
sample.d1.df = do.call(rbind.data.frame, sample.d1)
post.d1.B = ilogit(sample.d1.df$beta0)
post.d1.S = ilogit(sample.d1.df$beta0 + sample.d1.df$beta.Sidi)
round(mean(post.d1.B), 5); round(mean(post.d1.S), 5)
0.49667; 0.59067
plot(density(post.d1.B), type = "l", xlab = "Probability", xlim = c(0.25, 0.8),
     ylab = "Density", col = "blue", lwd = 1.5,
     main = "Posterior densities of the probability of fire in the two regions")
lines(density(post.d1.S), col = "red", lwd = 1.5)
legend("topleft", legend = c("Bejaia", "Sidi Bel-abbes"), bty = "n",
       col = c("blue", "red"), lwd = 1.5)
```



From the figure, we can see that the posterior density of the probability of fire in the region Sidi Bel-abbes is significantly right shifted compared with that in the region Bejaia. Consequently, the mean of the posterior expected probability of fire in the region Sidi Bel-abbes is also larger than that in the region Bejaia. All in all, the region Sidi Bel-abbes is much more likely to be prone to a wildfire than the region Bejaia.

(2) The analyses of model (d2)

Because we should add `month` as a categorical variable with 2 levels, July&August vs June&September, with the latter being the reference category to the regression model, I add one more dummy (indicator) variable named `hot.Ind` to implement it. The variable `hot.Ind` takes value 1 for July and August and takes value 0 for June and September. Because the two indicator variables `Sidi.Ind` and `hot.Ind` are all included in the regression model, different effects of the covariate `month` are allowed in the two Algerian regions. The model statement in JAGS of the model (d2) is shown below.

```
# The model d2 in JAGS
P2.jags.model.d2 = "model{

  # The likelihood
  for (i in 1:n){
    Fire[i] ~ dbern(p[i])
    logit(p[i]) <- beta0 + beta1*(TempCels[i] - mean(TempCels[])) +
                   beta2*(RH[i] - mean(RH[])) + beta3*(Ws[i] - mean(Ws[])) +
                   beta4*(RainYest[i] - mean(RainYest[])) +
                   beta.Sidi*Sidi.Ind[i] + beta.hot*hot.Ind[i]
  }

  # The priors
  beta0 ~ dnorm(0, tau)
  beta1 ~ dnorm(0, tau)
  beta2 ~ dnorm(0, tau)
  beta3 ~ dnorm(0, tau)
  beta4 ~ dnorm(0, tau)
  beta.Sidi ~ dnorm(0, tau)
  beta.hot ~ dnorm(0, tau)

  # The hyperparameters
  tau <- 1/10
}
"
```

Then, I also run this model using JAGS for 3 chains, where each chain has 60000 iterations with `thin=3`. Moreover, I also take the normal distribution with zero mean and precision being equal to 0.1 as the prior for the coefficient (`beta.hot`) of the dummy variable `hot.Ind`, and take random variates from standard normal distribution as initial values for the parameter `beta.hot`.

```
# Call from R to JAGS
AlgForst.d2 = AlgForst; AlgForst.d2$Sidi.Ind = ifelse(AlgForst$Region == "S", 1, 0)
AlgForst.d2$hot.Ind = ifelse(AlgForst$month == 7 | AlgForst$month == 8, 1, 0)
model.d2.data = list(n = dim(AlgForst.d2)[1], Fire = AlgForst.d2$Fire,
                     TempCels = AlgForst.d2$TempCels, RH = AlgForst.d2$RH,
                     Ws = AlgForst.d2$Ws, RainYest = AlgForst.d2$RainYest,
                     Sidi.Ind = AlgForst.d2$Sidi.Ind, hot.Ind = AlgForst.d2$hot.Ind)
model.d2.inits = function(){
  list(beta0 = rnorm(1, 0, 1), beta1 = rnorm(1, 0, 1), beta2 = rnorm(1, 0, 1),
       beta3 = rnorm(1, 0, 1), beta4 = rnorm(1, 0, 1), beta.Sidi = rnorm(1, 0, 1),
       beta.hot = rnorm(1, 0, 1))
}
model.d2 = jags.model(file = textConnection(P2.jags.model.d2), data = model.d2.data,
                      inits = model.d2.inits, n.chains = 3)
update(model.d2, n.iter = 5000)
sample.d2 = coda.samples(model.d2, variable.names = c("beta0", "beta1", "beta2",
                         "beta3", "beta4", "beta.Sidi", "beta.hot"),
                         n.iter = 60000, thin = 3)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 244
   Unobserved stochastic nodes: 7
   Total graph size: 2494

Initializing model
```

Afterwards, I perform the model diagnostics to check the convergence and mixing of the chains. I find that, for the model (d2), the trace plots for all the parameters vibrate in a certain range with no obvious outliers and the plots of the Gelman-Rubin-Brooks statistics for all the parameters are also extremely close to 1 (much lower than 1.1) after the burn-in period, which all indicate good convergence of the chains. Moreover, the autocorrelation functions for all the parameters decrease relatively quickly, and the effective sample sizes for all the parameters are above 5000, which indicate good mixing of the chains. The summary statistics for this model are shown as follows. We can see that the ratio between the standard deviation and the MCMC error for every parameter of this model is all greater than 20.

```
# Model diagnostics and summary
plot(sample.d2)
gelman.plot(sample.d2)
autocorr.plot(sample.d2[[1]])
effectiveSize(sample.d2[[1]])
summary(sample.d2)
```

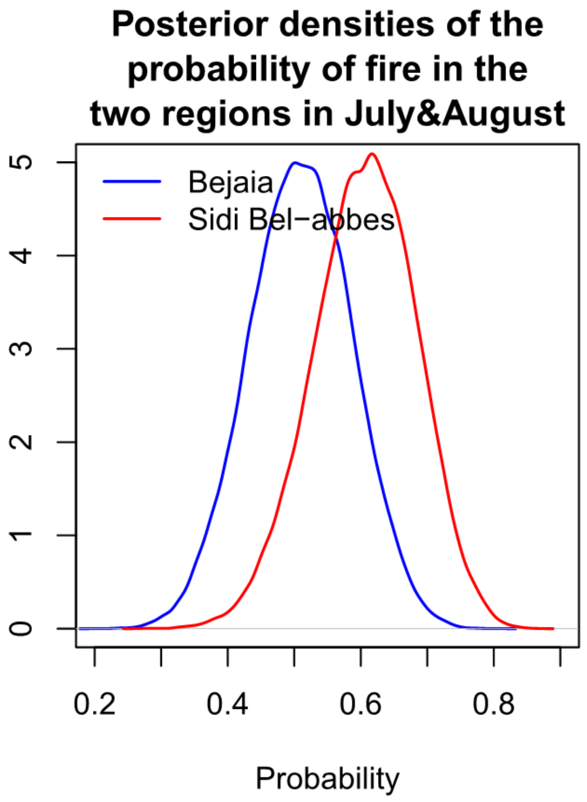
```
Iterations = 6003:66000
Thinning interval = 3
Number of chains = 3
Sample size per chain = 20000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

              Mean      SD  Naive SE Time-series SE
beta.Sidi  0.39988 0.35618 1.454e-03      2.098e-03
beta.hot   0.12051 0.40597 1.657e-03      2.949e-03
beta0     -0.08367 0.34366 1.403e-03      2.676e-03
beta1      0.32589 0.08321 3.397e-04      5.019e-04
beta2     -0.02882 0.01571 6.415e-05      7.724e-05
beta3      0.07752 0.06857 2.800e-04      3.172e-04
beta4     -0.76490 0.20934 8.546e-04      1.005e-03

2. Quantiles for each variable:

              2.5%      25%      50%      75%     97.5%
beta.Sidi -0.30369  0.16003  0.40023  0.63875  1.099062
beta.hot  -0.68413 -0.15117  0.12275  0.39360  0.912560
beta0     -0.76590 -0.31507 -0.08262  0.14813  0.582486
beta1      0.16850  0.26878  0.32390  0.38054  0.495202
beta2     -0.05985 -0.03944 -0.02878 -0.01807  0.001608
beta3     -0.05611  0.03129  0.07753  0.12329  0.212275
beta4     -1.20738 -0.89800 -0.75488 -0.61815 -0.387898
```

Because all the four metereological variables are centered, we should also take the value 0 for all the four metereological covariates, `TempCels`, `RH`, `Ws` and `RainYest` in the regression model (d2) when at average metereological conditions. Therefore, the model (d2) at average metereological conditions can be simplified as

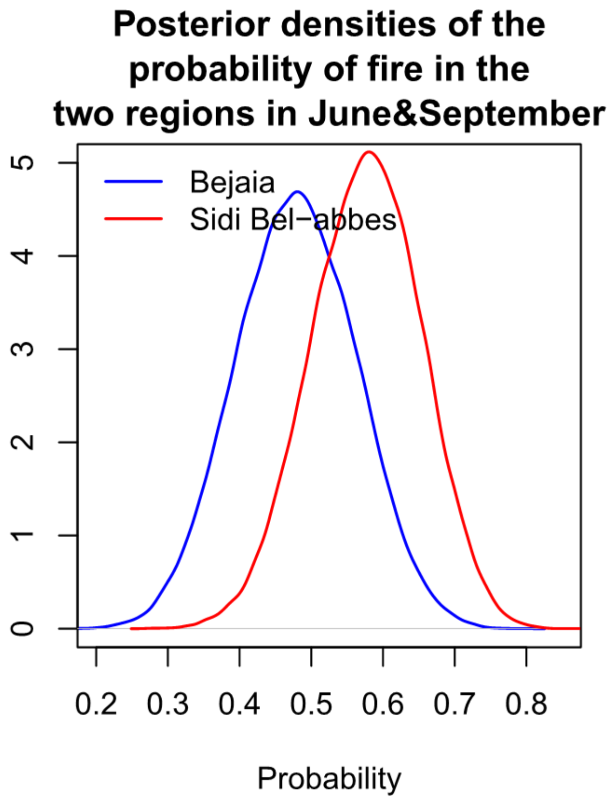
$$logit(p) = \beta_0 + beta.Sidi * Sidi.Ind + beta.hot * hot.Ind, \tag{2}$$

where p is the probability of fire, and *Sidi.Ind* and *hot.Ind* are the indicator variables of the region Sidi Bel-abbes and the months July&August respectively. After the execution of the R code below, we can obtain that the mean of the posterior expected probability of fire, i.e. the posterior mean of the probability parameter p, at average metereological conditions in the Bejaia region in July&August is 0.50948, and in the Sidi Bel-abbes region in July&August is 0.60488, and in the Bejaia region in June&September is 0.48043, and in the Sidi Bel-abbes region in June&September is 0.57683. The densities of the posterior expected probability of fire in the two regions in July&August and in June&September are plotted in the two panels of the figure below (in the next page) separately.

```
# The posterior expected probabilities of fire (at average metereological conditions)
# stratified by the new covariate, and the plot with two panels, each showing the two
# posterior densities for the Bejaia and Sidi Bel-abbes regions
sample.d2.df = do.call(rbind.data.frame, sample.d2)
ilogit = function(x){1/(1 + exp(-x))}
post.B.hot = ilogit(sample.d2.df$beta0 + sample.d2.df$beta.hot)
post.B.cool = ilogit(sample.d2.df$beta0)
post.S.hot = ilogit(sample.d2.df$beta0 + sample.d2.df$beta.Sidi +
  sample.d2.df$beta.hot)
post.S.cool = ilogit(sample.d2.df$beta0 + sample.d2.df$beta.Sidi)
round(mean(post.B.hot), 5); round(mean(post.S.hot), 5)
0.50948; 0.60488
round(mean(post.B.cool), 5); round(mean(post.S.cool), 5)
0.48043; 0.57683
par(mfrow = c(1, 2), pin = c(3, 4))
plot(density(post.B.hot), type = "l", xlab = "Probability", ylab = "Density",
     xlim = c(0.2, 0.9), col = "blue", lwd = 1.5,
     main = "Posterior densities of the\nprobability of fire
             in the\ntwo regions in July&August")
lines(density(post.S.hot), col = "red", lwd = 1.5)
legend("topleft", legend = c("Bejaia", "Sidi Bel-abbes"), bty = "n",
       col = c("blue", "red"), lwd = 1.5)
plot(density(post.B.cool), type = "l", xlab = "Probability", ylab = "Density",
     xlim = c(0.2, 0.85), ylim = c(0, 5), col = "blue", lwd = 1.5,
     main = "Posterior densities of the\nprobability of fire
             in the\ntwo regions in June&September")
lines(density(post.S.cool), col = "red", lwd = 1.5)
legend("topleft", legend = c("Bejaia", "Sidi Bel-abbes"), bty = "n",
       col = c("blue", "red"), lwd = 1.5)
graphics.off()
```



Posterior densities of the probability of fire in the two regions in July&August



Posterior densities of the probability of fire in the two regions in June&September

From the plot above, we can see that, on one hand, in both the period July&August and the period June&September, the posterior densities of the probability of fire in the region Sidi Bel-abbes are all significantly right shifted compared with those in the region Bejaia. Therefore, in both periods, the mean of the posterior expected probability of fire in the region Sidi Bel-abbes is also larger than that in Bejaia, and the region Sidi Bel-abbes is much more likely to be prone to a wildfire than the region Bejaia. On the other hand, in both Bejaia and Sidi Bel-abbes region, the posterior densities of the probability of fire in July&August are also all right shifted compared with those in June&September. Thus, it is more likely to observe a wildfire in July&August than in June&September in both regions. Furthermore, it is also worth noting that the impact of the variable `Region` on the probability of fire is obviously larger than the impact of the variable `hot.Ind`, levels of `month` (2 levels in total), on the probability of fire.

(3) The analyses of model (d3)

For this part, I am going to construct a model (d3) where a region and month-specific extra variance term $\omega[j]$ $(j = 1, 2, \ldots, 8)$ is added to model (d1). It means that we should add an extra variance term $\omega[j]$ to the expression of $logit(p[i])$ where $p[i]$ $(i = 1, 2, \ldots, n = 244)$ are the probability parameters. Then, I set a Gaussian prior for the parameters $\omega[j]$ $(j = 1, 2, \ldots, 8)$ with zero mean and precision with Gamma hyperprior with parameters $a = 0.01$ and $b = 0.01$. The model statement in JAGS of model (d3) is shown as follows.

```
# The model d3 in JAGS
P2.jags.model.d3 = "model{

  # The likelihood
  for (i in 1:n){
    Fire[i] ~ dbern(p[i])
    logit(p[i]) <- mu[i] + omega[j.kind[i]]
    mu[i] <- beta0 + beta1*(TempCels[i] - mean(TempCels[])) +
             beta2*(RH[i] - mean(RH[])) + beta3*(Ws[i] - mean(Ws[])) +
             beta4*(RainYest[i] - mean(RainYest[])) + beta.Sidi*Sidi.Ind[i]
  }

  # The priors
  beta0 ~ dnorm(0, tau)
  beta1 ~ dnorm(0, tau)
  beta2 ~ dnorm(0, tau)
  beta3 ~ dnorm(0, tau)
  beta4 ~ dnorm(0, tau)
  beta.Sidi ~ dnorm(0, tau)

  for (j in 1:8){
  omega[j] ~ dnorm(0, tau.omega)
}

  # The hyperparameters
  tau <- 1/10

  # The hyperpriors
  tau.omega ~ dgamma(a, b)

  # The hyper-hyperparameters
  a <- 0.01
  b <- 0.01
}
"
```

In the model string, note that `j.kind` is the indicator variable to indicate the specific combinations of region and month for the original data set `AlgForst`. It takes a value among $1, 2, \ldots, 8$ for a specific combination of region and month, where it takes value 1 for June in the Bejaia region and takes value 5 for June in the Sidi bel-abbes region, and so on. This indicator variable serves as the argument to the extra variance term $\omega[j]$ to decide which variance term among $\omega[1], \omega[2], \ldots, \omega[8]$ should be added to $logit(p[i])$ according to the combinations of region and month of observations in the data set.

Now, I will run this model using JAGS for 3 chains, where each chain has 60000 iterations with `thin=3`. I first construct the indicator variable `j.kind` as illustrated above for the data set `AlgForst`. Then, I also take random variates from standard normal distribution as initial values for all the model parameters, β_i $i = 0, 1, 2, 3, 4$, `beta.Sidi` and $\omega[j]$ $(j = 1, 2, \ldots, 8)$.

```
# Call from R to JAGS
AlgForst.d3 = AlgForst; AlgForst.d3$Sidi.Ind = ifelse(AlgForst$Region == "S", 1, 0)
AlgForst.d3$j.kind = sapply(1:dim(AlgForst)[1], function(k){
  if(AlgForst[k, ]$Region == "B" & AlgForst[k, ]$month == 6){return(1L)}
  else if(AlgForst[k, ]$Region == "B" & AlgForst[k, ]$month == 7){return(2L)}
  else if(AlgForst[k, ]$Region == "B" & AlgForst[k, ]$month == 8){return(3L)}
  else if(AlgForst[k, ]$Region == "B" & AlgForst[k, ]$month == 9){return(4L)}
  else if(AlgForst[k, ]$Region == "S" & AlgForst[k, ]$month == 6){return(5L)}
  else if(AlgForst[k, ]$Region == "S" & AlgForst[k, ]$month == 7){return(6L)}
  else if(AlgForst[k, ]$Region == "S" & AlgForst[k, ]$month == 8){return(7L)}
  else if(AlgForst[k, ]$Region == "S" & AlgForst[k, ]$month == 9){return(8L)}
})
model.d3.data = list(n = dim(AlgForst.d3)[1], Fire = AlgForst.d3$Fire,
                     TempCels = AlgForst.d3$TempCels, RH = AlgForst.d3$RH,
                     Ws = AlgForst.d3$Ws, RainYest = AlgForst.d3$RainYest,
                     Sidi.Ind = AlgForst.d3$Sidi.Ind, j.kind = AlgForst.d3$j.kind)
model.d3.inits = function(){
  inits = list(beta0 = rnorm(1, 0, 1), beta1 = rnorm(1, 0, 1),
               beta2 = rnorm(1, 0, 1), beta3 = rnorm(1, 0, 1),
               beta4 = rnorm(1, 0, 1), beta.Sidi = rnorm(1, 0, 1))
  for(k in 1:8){inits$omega[k] = rnorm(1, 0, 1)}
  return(inits)
}
model.d3 = jags.model(file = textConnection(P2.jags.model.d3), data = model.d3.data,
                      inits = model.d3.inits, n.chains = 3)
update(model.d3, n.iter = 5000)
sample.d3 = coda.samples(model.d3, variable.names = c("beta0", "beta1", "beta2",
                         "beta3", "beta4", "beta.Sidi", "omega"),
                         n.iter = 60000, thin = 3)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 244
   Unobserved stochastic nodes: 15
   Total graph size: 2745

Initializing model
```

Afterwards, let me perform the model diagnostics to check the convergence and mixing of the chains. For the model (d3), I find that there are no obvious anomalies in the trace plots for all the parameters. Also, the plots of the Gelman-Rubin-Brooks statistics for all the model parameters are also considerably close to 1 (much lower than 1.1). These phenomena all indicate good convergence of the chains. The autocorrelation functions for all the parameters also decrease relatively quickly. The effective sample sizes for all the parameters except β_0 are above 5000 and the effective sample size for β_0 is also above 4500, which indicate sufficient mixing of the chains. The summary statistics for this model are also shown as follows. We can see that the ratio between the standard deviation and the MCMC error for every parameter of this model is all greater than 20.

```
# Model diagnostics and summary
plot(sample.d3)
gelman.plot(sample.d3)
autocorr.plot(sample.d3[[1]])
effectiveSize(sample.d3[[1]])
summary(sample.d3)
```

```
Iterations = 6003:66000
```

```
Thinning interval = 3
Number of chains = 3
Sample size per chain = 20000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

               Mean      SD  Naive SE Time-series SE
beta.Sidi  0.379758 0.44362 0.0018111      0.0034650
beta0     -0.005926 0.31890 0.0013019      0.0025979
beta1      0.342102 0.07304 0.0002982      0.0003518
beta2     -0.028711 0.01575 0.0000643      0.0000761
beta3      0.085689 0.06743 0.0002753      0.0002886
beta4     -0.758443 0.20850 0.0008512      0.0010003
omega[1]   0.099493 0.29311 0.0011966      0.0019359
omega[2]  -0.069612 0.28604 0.0011678      0.0019217
omega[3]   0.035129 0.29428 0.0012014      0.0018556
omega[4]  -0.066561 0.29677 0.0012116      0.0018786
omega[5]  -0.246137 0.34774 0.0014196      0.0026441
omega[6]   0.094053 0.30693 0.0012530      0.0018242
omega[7]   0.004263 0.31192 0.0012734      0.0016778
omega[8]   0.153301 0.32117 0.0013112      0.0020934

2. Quantiles for each variable:

              2.5%      25%       50%      75%     97.5%
beta.Sidi -0.49457  0.09464  0.379717  0.66185  1.258135
beta0     -0.62945 -0.21060 -0.007531  0.20054  0.618156
beta1      0.20356  0.29210  0.340255  0.39044  0.489412
beta2     -0.05997 -0.03923 -0.028523 -0.01814  0.001979
beta3     -0.04602  0.04030  0.085066  0.13114  0.217446
beta4     -1.19892 -0.89124 -0.748111 -0.61230 -0.381309
omega[1]  -0.42851 -0.06521  0.066681  0.23728  0.785287
omega[2]  -0.71788 -0.20703 -0.046078  0.08487  0.467973
omega[3]  -0.55389 -0.11677  0.023672  0.17826  0.680398
omega[4]  -0.74006 -0.20596 -0.043473  0.09252  0.495676
omega[5]  -1.11908 -0.40755 -0.170870 -0.02019  0.254723
omega[6]  -0.46285 -0.07527  0.060929  0.23371  0.815704
omega[7]  -0.64686 -0.14518  0.003210  0.15313  0.675371
omega[8]  -0.37768 -0.03531  0.101383  0.29836  0.934194
```

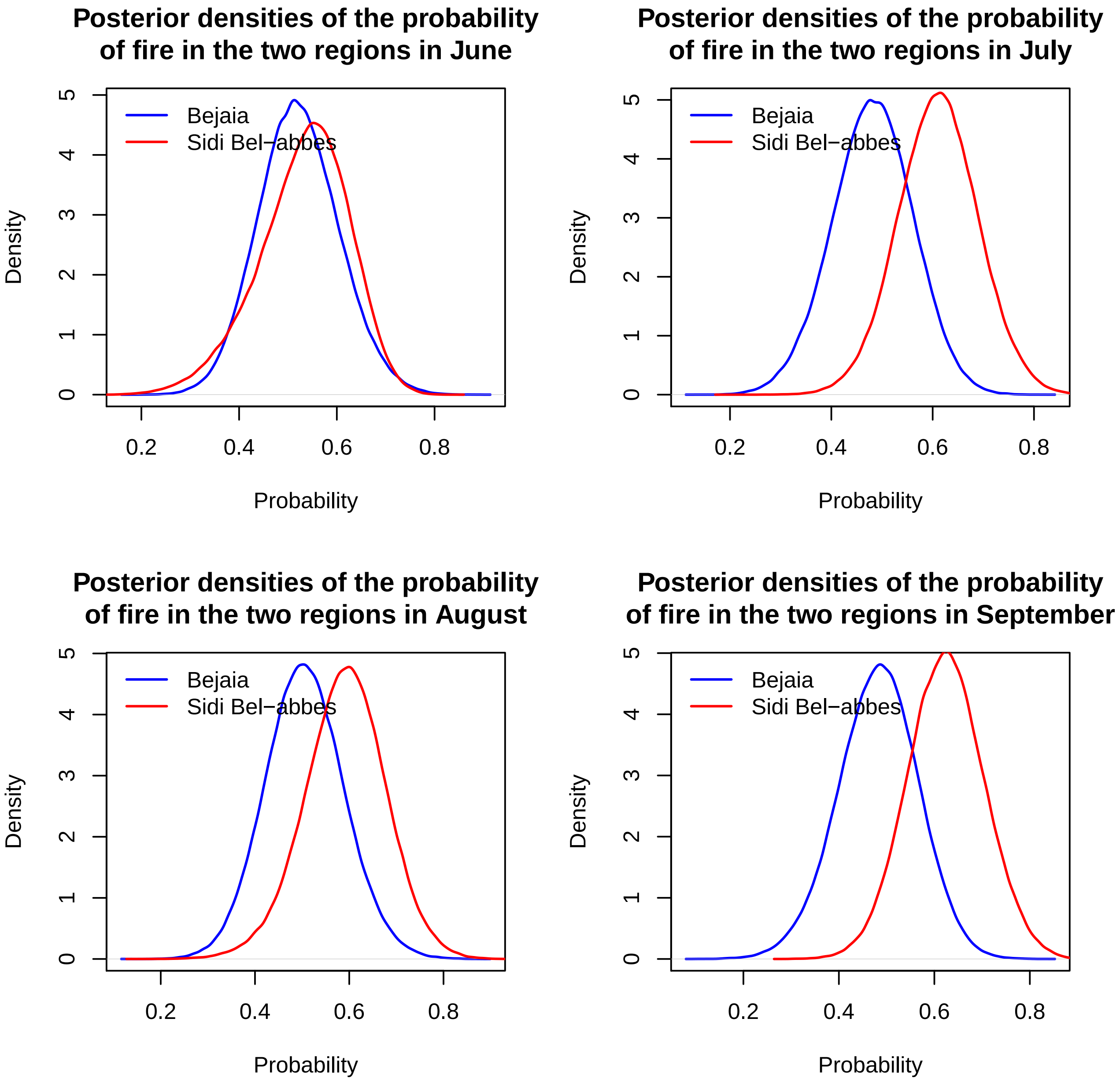
Because all the four metereological variables are also centered for this model, we should also take the value 0 for all the four metereological covariates in the regression model (d3) when at average metereological conditions. Therefore, the model (d3) at average metereological conditions can be simplified as

$$logit(p) = \beta_0 + beta.Sidi * Sidi.Ind + \omega, \tag{3}$$

where p is the probability of fire, and $Sidi.Ind$ is the indicator variable of the region Sidi Bel-abbes, and ω is the region and month-specific extra variance term. After running the R code below, we can obtain that the mean of the posterior expected probability of fire at average metereological conditions in the Bejaia region in June is 0.52259, and in the Sidi Bel-abbes region in June is 0.53123, and in the Bejaia region in July is 0.48170, and in the Sidi Bel-abbes region in July is 0.61162, and in the Bejaia region in August is 0.50702, and in the Sidi Bel-abbes region in August is 0.59042, and in the Bejaia region in September is 0.48257, and in the Sidi Bel-abbes region in September is 0.62500. The posterior densities of the probability of fire in the two regions stratified by the covariate `month` are also plotted in the four panels (one for each month) of the figure below separately.

```
# The posterior expected probabilities of fire (at average metereological
# conditions) in the Bejaia and Sidi Bel-abbes regions stratified by `month`
ilogit = function(x){1/(1 + exp(-x))}
sample.d3.df = do.call(rbind.data.frame, sample.d3)
post.B.6 = ilogit(sample.d3.df$beta0 + sample.d3.df$`omega[1]`)
post.B.7 = ilogit(sample.d3.df$beta0 + sample.d3.df$`omega[2]`)
post.B.8 = ilogit(sample.d3.df$beta0 + sample.d3.df$`omega[3]`)
post.B.9 = ilogit(sample.d3.df$beta0 + sample.d3.df$`omega[4]`)
post.S.6 = ilogit(sample.d3.df$beta0 + sample.d3.df$beta.Sidi +
                  sample.d3.df$`omega[5]`)
post.S.7 = ilogit(sample.d3.df$beta0 + sample.d3.df$beta.Sidi +
                  sample.d3.df$`omega[6]`)
post.S.8 = ilogit(sample.d3.df$beta0 + sample.d3.df$beta.Sidi +
                  sample.d3.df$`omega[7]`)
post.S.9 = ilogit(sample.d3.df$beta0 + sample.d3.df$beta.Sidi +
                  sample.d3.df$`omega[8]`)
round(mean(post.B.6), 5); round(mean(post.S.6), 5)
0.52259; 0.53123
round(mean(post.B.7), 5); round(mean(post.S.7), 5)
0.4817; 0.61162
round(mean(post.B.8), 5); round(mean(post.S.8), 5)
0.50702; 0.59042
round(mean(post.B.9), 5); round(mean(post.S.9), 5)
0.48257; 0.625
par(mfrow = c(2, 2))
plot(density(post.B.6), type = "l", xlab = "Probability",
     ylab = "Density", col = "blue", lwd = 1.5,
     main = "Posterior densities of the probability\nof fire
             in the two regions in June")
lines(density(post.S.6), col = "red", lwd = 1.5)
legend("topleft", legend = c("Bejaia", "Sidi Bel-abbes"), bty = "n",
       col = c("blue", "red"), lwd = 1.5)
plot(density(post.B.7), type = "l", xlab = "Probability",
     ylab = "Density", col = "blue", lwd = 1.5,
     main = "Posterior densities of the probability\nof fire
             in the two regions in July")
lines(density(post.S.7), col = "red", lwd = 1.5)
legend("topleft", legend = c("Bejaia", "Sidi Bel-abbes"), bty = "n",
       col = c("blue", "red"), lwd = 1.5)
plot(density(post.B.8), type = "l", xlab = "Probability",
     ylab = "Density", col = "blue", lwd = 1.5,
     main = "Posterior densities of the probability\nof fire
```

```
             in the two regions in August")
lines(density(post.S.8), col = "red", lwd = 1.5)
legend("topleft", legend = c("Bejaia", "Sidi Bel-abbes"), bty = "n",
       col = c("blue", "red"), lwd = 1.5)
plot(density(post.B.9), type = "l", xlab = "Probability", xlim = c(0.1, 1),
     ylab = "Density", col = "blue", lwd = 1.5,
     main = "Posterior densities of the probability\nof fire
             in the two regions in September")
lines(density(post.S.9), col = "red", lwd = 1.5)
legend("topleft", legend = c("Bejaia", "Sidi Bel-abbes"), bty = "n",
       col = c("blue", "red"), lwd = 1.5)
graphics.off()
```



Posterior densities of the probability of fire in the two regions in June

Posterior densities of the probability of fire in the two regions in July

Posterior densities of the probability of fire in the two regions in August

Posterior densities of the probability of fire in the two regions in September

There are both similarities and differences between the plot of the model (d3), each panel of which shows the relative two posterior densities for the Bejaia and Sidi Bel-abbes regions in each month, and the plots of model (d1) and (d2). For the similarities, on the hand, in all the months from June to September, the posterior densities of the probability of fire in the region Sidi Bel-abbes are also all right shifted compared with those in the region Bejaia, which means in every month the region Sidi Bel-abbes has larger posterior mean of expected probability of fire and is more likely to be prone to a wildfire than the region Bejaia. On the other hand, overall, the impact of the variable `Region` on the probability of fire is also evidently larger than the impact of the variable `month` on that. For the differences, firstly, the plot of model (d3) displays that the extent of the influence on the probability of fire brought by the covariate `Region` is different for each month, which is not shown by the plots of other models. The difference on the probability of fire between the two regions Bejaia and Sidi Bel-abbes, i.e. the influence brought by the covariate `Region`, is the largest in September, closely followed by the result of July. Also, the difference is the smallest in June, during which there is only tiny difference on the probability of fire between the two regions. It is very likely that the way to combine different months to be levels of the covariate `month` in model (d2), where the month June (the influence of `Region` is the smallest during it) and September (the influence of `Region` is the largest during it) are combined and the months July&August (the influence of `Region` is relatively moderate in these two months), has masked the differences in the influence of the covariate `Region` in different months. Furthermore, the plot of model (d3) also shows that the region Bejaia is more likely to be prone to wildfires in June and August than in July and September, but the region Sidi Bel-abbes is more likely to be prone to wildfires in July and September than in June and August which is just on the contrary. It is very likely that this result is also masked by the way to combine months in model (d2).

(4) Repeat the analyses of model (d3) with INLA

Now, let me repeat the analyses of model (d3) with INLA. I first construct a new data frame named `AlgForst.d3.inla`, where the four meterological variables temperature (`TempCels`), releative humidity (`RH`), wind speed (`Ws`) and rain (`RainYest`) are centered, and the response variable `Fire` remains the same as in the original data set `AlgForst`, and the indicator variables `Sidi.Ind` and `j.kind` in the data frame `AlgForst.d3` of the previous part are also included in it. Then, let me specify the prior information for model (d3). The priors for the intercept and regressors of model (d3) are all the normal distribution with zero mean and precision being equal to 0.1, and this information is stored by the variable `prior.beta`. The priors for the random effects $\omega[j]$ ($j = 1, 2, \ldots, 8$) are normal distributions with zero mean and precision that follows a Gamma hyperprior with parameters $a = 0.01$ and $b = 0.01$. The hyperprior information for the precision of $\omega[j]$ is stored by the variable `prec.prior.random.eff`. Because INLA stores log-transformed precision parameters, we need to specify priors in terms of $\log(\tau)$. Thus, I specify the log-Gamma distribution, `"loggamma"`, in the variable `prec.prior.random.eff`. Afterwards, I construct the model (d3) using INLA and store it in the variable `model.d3.inla`. The random effects of $\omega[j]$ are added as `f(j.kind, model = "iid", hyper = prec.prior.random.eff)`, where `"iid"` defines a Gaussian random effect. The likelihood for the model, the argument `family` in the `inla` function, is chosen as `"binomial"` with the `"logit"` link function. Furthermore, in order to sample from the posterior distributions of the parameters, we need to set the option `control.compute = list(config = TRUE)`. The R code for specifying the model (d3) using INLA is shown in the following.

```r
# Repeat the analyses of model (d3) with INLA
AlgForst.d3.inla = data.frame(
  Fire = AlgForst$Fire,
  TempCels = AlgForst$TempCels - mean(AlgForst$TempCels),
  RH = AlgForst$RH - mean(AlgForst$RH),
  Ws = AlgForst$Ws - mean(AlgForst$Ws),
  RainYest = AlgForst$RainYest - mean(AlgForst$RainYest),
  Sidi.Ind = AlgForst.d3$Sidi.Ind,
  j.kind = AlgForst.d3$j.kind
)
prior.beta = list(mean.intercept = 0, prec.intercept = 0.1, mean = 0, prec = 0.1)
prec.prior.random.eff = list(prec  = list(prior = "loggamma", param = c(0.01, 0.01)))
model.d3.inla = inla(Fire ~ TempCels + RH + Ws + RainYest + Sidi.Ind +
  f(j.kind, model = "iid", hyper = prec.prior.random.eff),
  data = AlgForst.d3.inla, family = "binomial",
  control.family = list(link = "logit"), control.fixed = prior.beta,
  control.predictor = list(compute = TRUE), control.compute = list(config = TRUE))
```

The summary statistics of this model are shown as follows.

```
# The summary statistics of the model
summary(model.d3.inla)


Call:
   c("inla(formula = Fire ~ TempCels + RH + Ws + RainYest + Sidi.Ind + ",
   " f(j.kind, model = \"iid\", hyper = prec.prior.random.eff), ", "
   family = \"binomial\", data = AlgForst.d3.inla, control.compute =
   list(config = TRUE), ", " control.predictor = list(compute = TRUE),
   control.family = list(link = \"logit\"), ", " control.fixed =
   prior.beta)")
Time used:
    Pre = 0.474, Running = 0.241, Post = 0.0711, Total = 0.786
Fixed effects:
              mean    sd 0.025quant 0.5quant 0.975quant   mode kld
(Intercept) -0.005 0.310     -0.617   -0.005      0.606 -0.004   0
TempCels     0.337 0.071      0.203    0.335      0.482  0.331   0
RH          -0.028 0.015     -0.059   -0.028      0.001 -0.028   0
Ws           0.085 0.066     -0.044    0.084      0.215  0.084   0
RainYest    -0.726 0.201     -1.153   -0.714     -0.364 -0.690   0
Sidi.Ind     0.377 0.431     -0.476    0.377      1.228  0.377   0

Random effects:
  Name   Model
    j.kind IID model

Model hyperparameters:
                      mean    sd 0.025quant 0.5quant 0.975quant mode
Precision for j.kind 34.36 51.46       1.35    15.69     188.42 2.96

Expected number of effective parameters(stdev): 7.59(1.25)
Number of equivalent replicates : 32.15

Marginal log-Likelihood:  -140.11
Posterior marginals for the linear predictor and
 the fitted values are computed
```

The summary statistics are essentially identical to what we got from JAGS once the precision for the variable `j.kind` (`Precision for j.kind`) is back-transformed into standard deviation. The posterior samples of the precision of the random effects ω can be accessed using the command `` model.d3.inla$marginals.hyperpar$`Precision for j.kind` ``. Then, we can use the `inla.tmarginal` function to transform the precision of ω to the standard deviation of ω. After that, we can use the `inla.zmarginal` to show the summary statistics of the posterior distribution of the standard deviation of ω. The R code and the results are shown below.

```
# The summary of the posterior distribution of the omega standard deviation
marginal.tau = model.d3.inla$marginals.hyperpar$`Precision for j.kind`
mariginal.sigma = inla.tmarginal(function(tau){tau^{-1/2}}, marginal.tau)
inla.zmarginal(mariginal.sigma)
```

```
Mean            0.309606
Stdev           0.216287
Quantile  0.025 0.0715465
Quantile  0.25  0.157843
Quantile  0.5   0.250999
Quantile  0.75  0.397231
Quantile  0.975 0.857713
```

To compute the posterior probability that relative humidity (`RH`) has a negative influence on the probability of a wildfire, I firstly take samples from the posterior distributions of the model parameters using the `inla.posterior.sample`. Then, I use the `inla.posterior.sample.eval` function to extract the marginal posterior samples of the regression coefficient β_2 for the covariate `RH`, and store them in the variable named `sample.beta2.inla`. Finally, because relative humidity has a negative influence on the probability of a wildfire, i.e. larger relative humidity results in smaller wildfire probability, if and only if the regression coefficient β_2 for the covariate `RH` is negative, the frequency that the posterior samples of the parameter β_2 are negative is the estimate of such probability. After the operations performed by the following R code, we can obtain that the posterior probability that relative humidity has a negative influence on the probability of a wildfire is 0.9687.

```
# The posterior probability that relative humidity has a negative
# influence on the probability of a wildfire
nsamp = 10000
sample.d3.inla = inla.posterior.sample(n = nsamp, result = model.d3.inla)
sample.beta2.inla = inla.posterior.sample.eval(function(...){RH}, sample.d3.inla)
mean(sample.beta2.inla < 0)
0.9687
```

Question (e)

(1) The analyses of Part 1

For this question, let me first construct the model (e1). The model (e1) has the addition of the covariate `DMC` (centered), and the rest of this model is the same with the model (d3). I let the parameter β_5 be the regression coefficient for the covariate `DMC`, and also use the normal distribution with zero mean and precision being equal to 0.1 as the prior for β_5. The model statement in JAGS of model (e1) is shown as follows.

```
# The model e1 in JAGS
P2.jags.model.e1 = "model{

  # The likelihood
  for (i in 1:n){
    Fire[i] ~ dbern(p[i])
    logit(p[i]) <- mu[i] + omega[j.kind[i]]
    mu[i] <- beta0 + beta1*(TempCels[i] - mean(TempCels[])) +
             beta2*(RH[i] - mean(RH[])) + beta3*(Ws[i] - mean(Ws[])) +
             beta4*(RainYest[i] - mean(RainYest[])) +
             beta5*(DMC[i] - mean(DMC[])) + beta.Sidi*Sidi.Ind[i]
  }

  # The priors
  beta0 ~ dnorm(0, tau)
  beta1 ~ dnorm(0, tau)
  beta2 ~ dnorm(0, tau)
  beta3 ~ dnorm(0, tau)
  beta4 ~ dnorm(0, tau)
  beta5 ~ dnorm(0, tau)
  beta.Sidi ~ dnorm(0, tau)

  for (j in 1:8){
  omega[j] ~ dnorm(0, tau.omega)
}

  # The hyperparameters
  tau <- 1/10

  # The hyperpriors
  tau.omega ~ dgamma(a, b)

  # The hyper-hyperparameters
  a <- 0.01
  b <- 0.01
}
"
```

Then, I will also run this model using JAGS for 3 chains, where each chain has 60000 iterations with `thin=3`. I also take random variates from standard normal distribution as initial values for all the parameters of model (e1), β_i $(i = 0, 1, \ldots, 5)$, $beta.Sidi$ and $\omega[j]$ $(j = 1, 2, \ldots, 8)$.

```
# Call from R to JAGS
AlgForst.e1 = AlgForst.d3
model.e1.data = list(n = dim(AlgForst.e1)[1], Fire = AlgForst.e1$Fire,
                     TempCels = AlgForst.e1$TempCels, RH = AlgForst.e1$RH,
                     Ws = AlgForst.e1$Ws, RainYest = AlgForst.e1$RainYest,
                     DMC = AlgForst.e1$DMC, Sidi.Ind = AlgForst.e1$Sidi.Ind,
                     j.kind = AlgForst.e1$j.kind)
model.e1.inits = function(){
  inits = list(beta0 = rnorm(1, 0, 1), beta1 = rnorm(1, 0, 1), beta2 = rnorm(1, 0, 1),
               beta3 = rnorm(1, 0, 1), beta4 = rnorm(1, 0, 1), beta5 = rnorm(1, 0, 1),
               beta.Sidi = rnorm(1, 0, 1))
  for(k in 1:8){inits$omega[k] = rnorm(1, 0, 1)}
  return(inits)
}
model.e1 = jags.model(file = textConnection(P2.jags.model.e1), data = model.e1.data,
                      inits = model.e1.inits, n.chains = 3)
update(model.e1, n.iter = 5000)
sample.e1 = coda.samples(model.e1, variable.names = c("beta0", "beta1", "beta2",
                         "beta3", "beta4", "beta5", "beta.Sidi", "omega", "tau.omega"),
                         n.iter = 60000, thin = 3)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 244
   Unobserved stochastic nodes: 16
   Total graph size: 3326

Initializing model
```

Afterwards, let me perform the model diagnostics to check the convergence and mixing of the chains. Firstly, for the model (e1), I find that there are no obvious anomalies in the trace plots for all the parameters of this model. Also, the plots of the Gelman-Rubin-Brooks statistics for all the model parameters are also considerably close to 1 (much lower than 1.1). These phenomena indicate good convergence of the chains. Then, the effective sample sizes for all the parameters are above 4000. The summary statistics for this model are also shown below. We can see that the ratio between the standard deviation and the MCMC error for every parameter of this model is all greater than 20. Therefore, the mixing of the chains and the number of MCMC samples that we collect is sufficient.

```
# Model diagnostics and summary
plot(sample.e1)
gelman.plot(sample.e1)
autocorr.plot(sample.e1[[1]])
effectiveSize(sample.e1[[1]])
summary(sample.e1)
```

```
Iterations = 6003:66000
Thinning interval = 3
Number of chains = 3
Sample size per chain = 20000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

               Mean      SD  Naive SE Time-series SE
beta.Sidi -0.001784  0.53468 0.0021828      4.739e-03
```

```
beta0      1.023106  0.43218 0.0017644      4.044e-03
beta1      0.194507  0.08721 0.0003560      5.751e-04
beta2     -0.013231  0.01933 0.0000789      9.876e-05
beta3      0.028117  0.07526 0.0003073      3.307e-04
beta4     -0.258930  0.18296 0.0007469      9.421e-04
beta5      0.229883  0.04292 0.0001752      2.693e-04
omega[1]   0.185125  0.38554 0.0015740      3.407e-03
omega[2]  -0.108561  0.35257 0.0014394      2.841e-03
omega[3]  -0.168145  0.39831 0.0016261      3.418e-03
omega[4]   0.113574  0.37549 0.0015329      3.096e-03
omega[5]  -0.092937  0.35350 0.0014432      2.230e-03
omega[6]   0.005401  0.35777 0.0014606      2.091e-03
omega[7]  -0.165525  0.42227 0.0017239      3.080e-03
omega[8]   0.259353  0.44474 0.0018156      4.019e-03
tau.omega 29.293450 47.09378 0.1922596      4.129e-01

2. Quantiles for each variable:

              2.5%       25%       50%        75%     97.5%
beta.Sidi -1.07458 -0.329017  0.003863  0.3326327   1.03235
beta0      0.20269  0.741942  1.013859  1.2907969   1.90120
beta1      0.03221  0.135476  0.191836  0.2501756   0.37601
beta2     -0.05122 -0.026185 -0.013161 -0.0001856   0.02438
beta3     -0.12031 -0.022385  0.028745  0.0792600   0.17421
beta4     -0.65173 -0.374954 -0.247278 -0.1293454   0.06324
beta5      0.15053  0.200363  0.227977  0.2577231   0.31889
omega[1]  -0.41481 -0.036107  0.111648  0.3417630   1.14262
omega[2]  -0.93554 -0.263405 -0.067650  0.0754147   0.53435
omega[3]  -1.16837 -0.324359 -0.095333  0.0541182   0.46296
omega[4]  -0.53387 -0.085541  0.061489  0.2642338   1.01902
omega[5]  -0.90507 -0.249242 -0.060955  0.0873657   0.57929
omega[6]  -0.75995 -0.159323  0.006007  0.1711398   0.75804
omega[7]  -1.24455 -0.317390 -0.085057  0.0657409   0.49790
omega[8]  -0.34991 -0.008888  0.148095  0.4266814   1.43142
tau.omega  0.73091  4.117927 11.644079 33.6185551 163.37864
```

Now, let me perform posterior predictive checks using JAGS. I am going to take random samples from the predictive distributions for replicates of the data set `AlgForst` (at the same metereological conditions) for models (d2), (d3) and (e1). For the operations for each model, I firstly attach the data frame containing the posterior samples of model parameters and the data frame (`AlgForst.d2`, `AlgForst.d3`, or `AlgForst.e1`) containing all the covariates of the model using `with` function to simplify notations of extracting columns of data frames and avoid collisions of variable names. Then, I use the `sapply` function to loop over all the observations in the data set. For each observation, I compute the posterior samples of the probability parameter p according to the regression equation of each model. Afterwards, for each posterior sample of the probability parameter p, I take a sample from the Binomial distribution with the argument `size` being equal to 1 and the argument `prob` being equal to the value of this posterior sample of p. Note that for the random effects ω of the model (d3) and (e1), we can use the formula `[, j.kind[k] + 6]` to index the corresponding random effect for each observation. After the operations of the same type on all the observations, we can obtain the samples from the predictive distributions for replicates of the data set for each model. Finally, I transpose the output of the `sapply` function and convert it to the type of data frame, each column of which is a sample of the replicate for the response `Fire`. The R code for implementing this process is shown in the following.

```
# The random samples of the predictive distributions for replicates of the data set
# (at the same metereological conditions) for models (d2), (d3) and (e1)
ilogit = function(x){1/(1 + exp(-x))}
n = dim(AlgForst)[1]

# The model (d2)
m.d2 = dim(sample.d2.df)[1]
pred.repli.d2 = as.data.frame(with(sample.d2.df, with(AlgForst.d2,
  t(sapply(1:n, function(k){rbinom(n = m.d2, size = 1,
  prob = ilogit(beta0 + beta1*(TempCels[k] - mean(TempCels)) +
  beta2*(RH[k] - mean(RH)) + beta3*(Ws[k] - mean(Ws)) +
  beta4*(RainYest[k] - mean(RainYest)) +
  beta.Sidi*Sidi.Ind[k] + beta.hot*hot.Ind[k]))})))))

# The model (d3)
m.d3 = dim(sample.d3.df)[1]
pred.repli.d3 = as.data.frame(with(sample.d3.df, with(AlgForst.d3,
  t(sapply(1:n, function(k){rbinom(n = m.d3, size = 1,
  prob = ilogit(beta0 + beta1*(TempCels[k] - mean(TempCels)) +
  beta2*(RH[k] - mean(RH)) + beta3*(Ws[k] - mean(Ws)) +
  beta4*(RainYest[k] - mean(RainYest)) + beta.Sidi*Sidi.Ind[k] +
  as.numeric(sample.d3.df[, j.kind[k] + 6])))})))))

# The model (e1)
sample.e1.df = do.call(rbind.data.frame, sample.e1)
m.e1 = dim(sample.e1.df)[1]
pred.repli.e1 = as.data.frame(with(sample.e1.df, with(AlgForst.e1,
  t(sapply(1:n, function(k){rbinom(n = m.e1, size = 1,
  prob = ilogit(beta0 + beta1*(TempCels[k] - mean(TempCels)) +
  beta2*(RH[k] - mean(RH)) + beta3*(Ws[k] - mean(Ws)) +
  beta4*(RainYest[k] - mean(RainYest)) +
  beta5*(DMC[k] - mean(DMC)) + beta.Sidi*Sidi.Ind[k] +
  as.numeric(sample.e1.df[, j.kind[k] + 6])))})))))
```

Then, using these samples, I will compute the Bayesian p-value, the probability of observing a proportion of wildfires that is more extreme than those observed in the actual data set, for each region and month and for each model. For each region and month, I first calculate the proportion of wildfires observed in the actual data set `AlgForst` and store this value by the variable `prop.origin`. Note that when we search for the samples that have a proportion of wildfires that is more extreme than those observed in the actual data set, we need to consider the two extreme sides. More precisely, for each region and month and for each model, if the observed proportion of wildfires (`prop.origin`) is greater than or equal to 0.5, the p-value is the proportion of samples that have a proportion of fire greater than `prop.origin` or less than `1 - prop.origin`, and if the observed proportion of fire (`prop.origin`) is less than 0.5, the p-value is the proportion of samples that have a proportion of fire less than `prop.origin` or greater than `1 - prop.origin`. The R code for computing the

Bayesian p-values and the results of the Bayesian p-value for each region and month and for each model are shown as follows.

```
# The computation of the Bayesian p-values
p.values = data.frame(
  Region = c(rep("B", 4), rep("S", 4)),
  month = rep(6:9, 2),
  d2.p.value = numeric(8),
  d3.p.value = numeric(8),
  e1.p.value = numeric(8)
)
for (Region in c("B", "S")){
  for (month in 6:9){
    prop.origin = mean(AlgForst[(AlgForst$Region == Region &
                                 AlgForst$month == month), "Fire"])
    prop.d2 = apply(pred.repli.d2[(AlgForst$Region == Region &
                    AlgForst$month == month), ], MARGIN = 2, FUN = mean)
    prop.d3 = apply(pred.repli.d3[(AlgForst$Region == Region &
                    AlgForst$month == month), ], MARGIN = 2, FUN = mean)
    prop.e1 = apply(pred.repli.e1[(AlgForst$Region == Region &
                    AlgForst$month == month), ], MARGIN = 2, FUN = mean)
    if (prop.origin >= 0.5){
      p.values[(p.values$Region == Region & p.values$month == month), 3:5] =
        c(mean((prop.d2 > prop.origin) | (prop.d2 < (1 - prop.origin))),
          mean((prop.d3 > prop.origin) | (prop.d3 < (1 - prop.origin))),
          mean((prop.e1 > prop.origin) | (prop.e1 < (1 - prop.origin))))
    }
    else{
      p.values[(p.values$Region == Region & p.values$month == month), 3:5] =
        c(mean((prop.d2 < prop.origin) | (prop.d2 > (1 - prop.origin))),
          mean((prop.d3 < prop.origin) | (prop.d3 > (1 - prop.origin))),
          mean((prop.e1 < prop.origin) | (prop.e1 > (1 - prop.origin))))
    }
  }
}
cbind.data.frame(p.values[, 1:2], round(p.values[, 3:5], 5))
```

```
A data.frame: 8 × 5
Region   month   d2.p.value   d3.p.value   e1.p.value
<fct>    <int>      <dbl>        <dbl>        <dbl>
  B        6      0.71412      0.62171      0.63765
  B        7      0.76030      0.74921      0.77162
  B        8      0.37517      0.39057      0.66898
  B        9      0.29628      0.31851      0.70070
  S        6      0.16360      0.18496      0.26742
  S        7      0.24888      0.28141      0.35315
  S        8      0.43302      0.41495      0.74093
  S        9      0.64053      0.61593      0.69275
```

From the results of the Bayesian p-values, we can see that, in most cases, the model (e1) has the largest Bayesian p-values among the three models, and the Bayesian p-value of model (e1) is often much larger than that of the other two models. Therefore, the samples from the predictive distribution for replicates of model (e1) fit the observed data much better than those of model (d2) and model (d3). Finally, let me compute the Deviance Information Criterion (DIC) of these three models. The DIC of the JAGS models can be computed using the `dic.samples` function. The R code for computing DIC for these three models and the results of the DIC are shown below.

```
# The Deviance Information Criterion (DIC) of the three models
DIC.d2 = dic.samples(model = model.d2, n.iter = 10000, type = "pD")
DIC.d2
Mean deviance:  237.8
penalty 6.931
Penalized deviance: 244.7
DIC.d3 = dic.samples(model = model.d3, n.iter = 10000, type = "pD")
DIC.d3
Mean deviance:  235.8
penalty 7.765
Penalized deviance: 243.6
DIC.e1 = dic.samples(model = model.e1, n.iter = 10000, type = "pD")
DIC.e1
Mean deviance:  187.5
penalty 8.885
Penalized deviance: 196.4
```

From the results of the DIC of these three models, we can see that the DIC of the model (e1) is the smallest among the three models and it is much smaller than the DIC of the other two models. Moreover, the values of the DIC for the model (d2) and (d3) are very close, with the DIC for the model (d2) being the largest.

Hence, all in all, according to the results of both the posterior predictive checks and the computation of DIC, the model (e1) all has the best performance among the three models and it behaves much better than the other two models. Thus, I consider that the model (e1) is better for modelling the observed data.

(2) The analyses of Part 2

For this part, I will use model (e1) implemented in JAGS to predict the probabilities of a fire in the two regions during the reported day. Because the reported metereological conditions in the two regions were registered in October which is a new month that is not recorded in the original data set `AlgForst`, we need manage to take posterior samples of the random effect ω for observations from a new group.

Suppose that $\psi^{(s)}$ ($s = 1, 2, \ldots, N$) are the posterior samples of the parameters of model (e1) obtained according to the original data set `AlgForst` using JAGS. Then, to obtain samples from the posterior distribution of the random effect ω for a new month "October", we need to draw a sample from $p(\omega \mid \psi^{(s)})$ for each $s \in \{1, 2, \ldots, N\}$. The R code for this process is shown below. Note that the variable `sample.e1.df` is the data frame that stores all the posterior samples of all the parameters of model (e1) and `tau.omega` is one of the columns of the data frame `sample.e1.df` that stores the posterior samples of the precision parameter of the random effect ω.

```
# Obtain samples from the posterior of the random effect `omega` for a new group
```

```
N = dim(sample.e1.df)[1]
omega.new = rnorm(n = N, mean = 0, sd = 1/sqrt(sample.e1.df$tau.omega))
```

After obtaining the posterior samples of the random effect ω for the new group, the posterior samples of the probability parameter p for the two new observations can be acquired by plugging the posterior samples of ω and other model parameters in the expression for p. Let me first perform the probability prediction for a fire under the following observed metereological conditions (in the region Bejaia):

$$\text{Bejaia: Temp} = 24, \text{ RH} = 40, \text{ Ws} = 15, \text{ RainYest} = 0.5, \text{ DMC} = 8.$$

The R code is shown as follows. The metereological conditions of the observation should be centered at first. Finally, the posterior samples of the probability parameter p for this observation are stored by the variable `post.prob.B`. The posterior mean of the probability parameter p for this observation is 0.19561, which is the prediction for the probability of a fire under these metereological conditions.

```
# The prediction of the probability of a fire in Bejaia during 15th October 2012
ilogit = function(x){1/(1 + exp(-x))}
data.Bejaia = data.frame(
  TempCels = 24 - mean(AlgForst$TempCels), RH = 40 - mean(AlgForst$RH),
  Ws = 15 - mean(AlgForst$Ws), RainYest = 0.5 - mean(AlgForst$RainYest),
  DMC = 8 - mean(AlgForst$DMC), Sidi.Ind = 0
)
post.prob.B = with(sample.e1.df, with(data.Bejaia,
  ilogit(beta0 + beta1*TempCels + beta2*RH + beta3*Ws +
  beta4*RainYest + beta5*DMC + beta.Sidi*Sidi.Ind + omega.new)))
predict.prob.B = mean(post.prob.B)
round(predict.prob.B, 5)
0.19561
```

Then, let me perform the probability prediction for a fire under the next observed metereological conditions shown as follows (in the region Sidi Bel-abbes):

$$\text{Sidi: Temp} = 26, \text{ RH} = 20, \text{ Ws} = 13, \text{ RainYest} = 0.0, \text{ DMC} = 12.$$

The R code is shown below. The metereological conditions of this observation should also be centered at first. The posterior samples of the probability parameter p for this observation are stored by the variable `post.prob.S` finally. The posterior mean of the probability parameter p for this observation is 0.48336, which is the prediction for the probability of a fire under these metereological conditions.

```
# The prediction of the probability of a fire in Sidi during 15th October 2012
data.Sidi = data.frame(
  TempCels = 26 - mean(AlgForst$TempCels), RH = 20 - mean(AlgForst$RH),
  Ws = 13 - mean(AlgForst$Ws), RainYest = 0.0 - mean(AlgForst$RainYest),
  DMC = 12 - mean(AlgForst$DMC), Sidi.Ind = 1
)
post.prob.S = with(sample.e1.df, with(data.Sidi,
  ilogit(beta0 + beta1*TempCels + beta2*RH + beta3*Ws +
  beta4*RainYest + beta5*DMC + beta.Sidi*Sidi.Ind + omega.new)))
```

```
predict.prob.S = mean(post.prob.S)
round(predict.prob.S, 5)
0.48336
```

(3) The analyses of Part 3

In this part, under the background of the model described in this part, I am going to set the precision of zero-centered Gaussian priors for the model parameters β_0 and β_{DMC} using the given prior information. Denote the covariate `DMC` by x. Then, when setting the precision parameters for the priors, we first need to consider the minimum distance in x (`DMC`), $x_u - x_l$, for which we think the change of probability on the inverse-logit curve can happen. We then set a prior that ensures that $|\beta_{DMC}|$ can be larger than 10 over that minimum change distance, which means the standard deviation σ_{DMC} of the Gaussian prior for β_{DMC} must be at least larger than 10 over the minimum change distance $x_u - x_l$. Because the expert has affirmed that it is unlikely to find a wildfire when $DMC < 7$ while it is very likely to see a wildfire when $DMC > 20$, we can determine that the interval for the covariate `DMC` on which the change of probability can happen is $[7, 20]$, which means $x_u = 20$ and $x_l = 7$ under this background. Therefore, the standard deviation σ_{DMC} must be at least larger than $10/(20-7) = 10/13$. Hence, the precision of the zero-centered Gaussian prior for β_{DMC} should at most be

$$\tau_{DMC} = \frac{1}{\sigma_{DMC}^2} = \frac{13^2}{10^2} = 1.69.$$

Then, for the priors of the parameter β_0, we need to ensure that $|\beta_0|$ can be larger than $|\bar{x} - x_m||\beta_{DMC}|$, which means the standard deviation σ_0 of the Gaussian prior for β_0 must be at least larger than $|\bar{x}-x_m||\beta_{DMC}|$, where x_m is the point on x-axis that corresponds to the probability center on the inverse-logit curve (the point on x-axis that lets the probability p of fire take value 0.5). Because in our background, the expert also suggests that a 50/50 chance of a fire should be around $DMC = 10$ and $DMC = 12$, we can see that the value of x_m is about 10 or 12. Moreover, $\bar{x}$ is the mean of the observed data of the covariate `DMC`, and the value of $\bar{x}$ is `mean(AlgForst$DMC)=14.67336`. Therefore, the standard deviation σ_0 of the Gaussian prior for β_0 must at least be

$$\sigma_0 = |\bar{x} - x_m||\beta_{DMC}| = \max(|14.67336 - 10|, |14.67336 - 12|) * \frac{10}{13} = \frac{46.7336}{13} = 3.59489.$$

Hence, the precision of the zero-centered Gaussian prior for the parameter β_0 should at most be

$$\tau_0 = \frac{1}{\sigma_0^2} = \frac{1}{3.59489^2} = 0.07738.$$

The R code for the calculation of this process and the results are shown as follows.

```
# Set the precision of zero-centered Gaussian priors for the regression
# parameters beta0 and beta_DMC using the given prior information
x_u = 20; x_l = 7
sigma.beta.DMC = 10/(x_u - x_l)
tau.beta.DMC = 1/(sigma.beta.DMC^2)
x.bar = mean(AlgForst$DMC)
sigma.beta0 = max(abs(x.bar - 10), abs(x.bar - 12))*sigma.beta.DMC
tau.beta0 = 1/(sigma.beta0^2)
round(tau.beta0, 5)
0.07738
round(tau.beta.DMC, 5)
1.69
```