Bayesian Data Analysis Assignment 2

Johnny Lee (s1687781)

11th Apr 2022

Problem 1 - Fishing in a park

Our dataset from UCLA OARC has data on 250 groups that went to a park.They were questioned about how many fish they caught (count), how many people were in the group (persons), how many children were in the group (child), whether or not they brought a camper van to the park (camper), and whether or not they used live bait (livebait).

We are going to apply several regression models on this dataset. The first step is to load the dataset and JAGS.

```
fish <- read.csv("fish.csv")
head(fish)
```

```
##   livebait camper persons child count
```

```
## 1        0      0       1     0     0
## 2        1      1       1     0     0
## 3        1      0       1     0     0
## 4        1      1       2     1     0
## 5        1      0       1     0     1
## 6        1      1       4     2     0
```

a)[10 marks]

Create a new column in the dataset for proportion of children in the group (prop.child). Fit a Poisson GLM with log link function on the number of fish caught (count) as response, and the covariates camper, livebait, and prop.child as covariates. Use the offset feature of the Poisson GLM to take the number of persons into account. Interpret the results and discuss the meaning of the regression coefficients for the 3 covariates.

Explanation: We fitted a poisson GLM with log linke function using `glm()` where the `count` is in log scale with an offset, `persons`. With that we obtained the regression equation as below.

$$\log(count) = -1.36776 + 0.93438 \times camper + 1.77831 \times livebait - 4.89927 \times prop.child + \log(persons)$$

We have the intercept value of -1.36776. This indicates that the baseline which then there is only one person fishing, the person can catch $\exp(-1.36776)$ fish.

For the coefficient of `camper` is 0.93438. This indicates the number of fish caught by group of people with a camper van is $\exp(0.93438)$ times the number of fish caught by group of people without a camper van.

For the coefficient of `livebait` is 1.77831. This indicates the number of fish caught by group of people using live baits us $\exp(1.77831)$ times the number of fish caught by group of people without using live baits.

For the coefficient of `prop.child` is -4.89927. Since it has a negative value this means that a group of people with higher proportion of children will less likely to catch the fish and to be precise, $\exp(-4.89927 \times 0.01)$ times for 1% increase in `prop.child`.

```
#defining prop.child
fish$prop.child <- fish$child / fish$persons
#fitting poisson GLM with log link
fish.log <- glm(count ~ camper + livebait + prop.child + offset(log(persons)),
                family=poisson(link="log"), data = fish)
summary(fish.log)
```

```
##
## Call:
## glm(formula = count ~ camper + livebait + prop.child + offset(log(persons)),
##     family = poisson(link = "log"), data = fish)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.7987  -1.7770  -1.0204  -0.4346  20.2478
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.36776    0.23646  -5.784 7.28e-09 ***
## camper       0.93438    0.08901  10.498  < 2e-16 ***
## livebait     1.77831    0.23273   7.641 2.16e-14 ***
## prop.child  -4.89927    0.25407 -19.283  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 2557.5  on 249  degrees of freedom
## Residual deviance: 1580.9  on 246  degrees of freedom
## AIC: 1926
##
## Number of Fisher Scoring iterations: 6
```

b)[10 marks] Using JAGS, implement a Bayesian version of the model in part a), i.e. a Bayesian Poisson GLM with log link function and the number of persons acting as the offset variable. Explain how did you construct your model. Use a Gaussian prior with mean 0 and variance 100 for the intercept, and independent Gaussian priors with mean 0 and variance 1 for the other 3 regression coefficients. Do 5000 burn-in iterations, and obtain 10000 samples from all regression coefficients from this model using coda.samples.

Explanation:

Now we proceed to the Bayesian analysis of the model in Q1 a). We specified the priors for β_i as given in the question. We ran with 3 chains with 50000 samples after 5000 burn-in iterations, starting from an initial position `camper`, `livebait`, `prop.child` and `persons`.

```
#defining model data
fish.data <- list(n=nrow(fish), count = fish$count, livebait = fish$livebait,
                  camper = fish$camper, prop.child = fish$prop.child,
                  persons = fish$persons)
#setting number of chians
num.chains <- 3

#defining model string.
fish.model.string <- "model{
#defining prior
beta0 ~ dnorm(0, 1/100)
beta1 ~ dnorm(0, 1)
beta2 ~ dnorm(0, 1)
beta3 ~ dnorm(0, 1)

for(i in 1:n){
  #loop through all data to compute the likelihood
  log(mu[i]) <- beta0 + beta1*camper[i] + beta2*livebait[i] +
  beta3*prop.child[i] + log(persons[i])
  count[i] ~ dpois(mu[i])}
}"
```

```
results.fish.origin <- jags.model(file = textConnection(fish.model.string),
                           data = fish.data,
                           n.chains = 3, quiet = TRUE)
#
update(results.fish.origin, n.iter = 5000)
#
fish.jags.origin <- coda.samples(results.fish.origin, n.iter = 50000,
                          variable.names = c("beta0","beta1","beta2","beta3"))
```

c)[10 marks] Based on your MCMC samples, compute the Gelman-Rubin convergence diagnostics (Hint: you need to run multiple chains in parallel for this by setting the n.chains parameter). Discuss how well has the chain converged to the stationary distribution based on the results.

Compute and print out the effective sample sizes (ESS) for the intercept and 3 regression coefficients.

If the ESS is below 1000 for any of these 4 parameters, increase the sample size/number of chains until the ESS is above 1000 for all of them.

Print out the summary of the fitted JAGS model. Compare this with the results from the GLM model of a).

Check the sensitivity of the summary statistics with respect to the priors.

Explanation:

As we can see from both `gelman.plot()` and `gelman.diag()` the Gelman-Rubin diagnostics values are 1 for each of the β parameters, indicating that we have approximated the stationary distribution well.

We plotted the autocorrelation functions for the *beta* parameters using the `acf()`. We set the `lag.max` parameter to `150` to clearly display the plots. We can see that the samples from β_0 and β_2 suffer from the most autocorrelation, but we still observe that the autocorrelation is decreasing to 0. For β_1 and β_3, we can see that the autocorrelation quickly shrinks and diminishes to zero before lag 25.

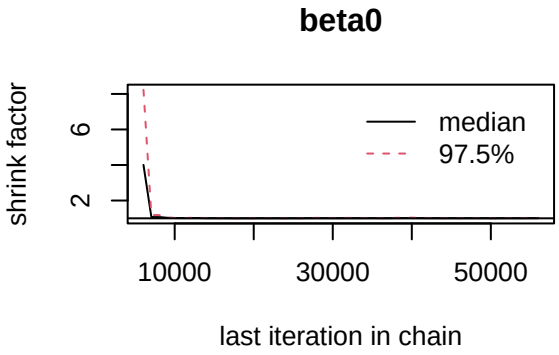
Looking at the effective size, we can confirm that the effective size of all β parameters are above 1000 so no further sample is needed.

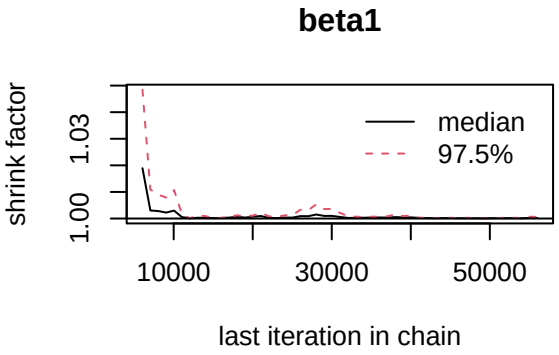
We printed out the posterior summaries for the model β parameters using `summary()`. We can see that the standard deviations are typically much smaller than the means in absolute value, indicating that the the amount of data is sufficient to fit the model parameters with some degree of confidence. The `time series SE` is much smaller than the posterior means (in absolute value), showing that our estimates are accurate and the chain was mixing reasonably well. Also, the coefficients of the covariates are similar to the regression coefficients from `glm()`. Hence, this also indicates good mixing of the JAGS model.

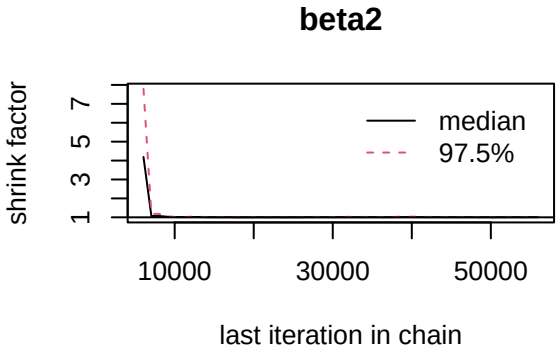
Lastly, we plotted the traceplot and the density of each β parameters. The traceplots of all the components indicated good mixing as there are no significant patterns sticking out of the trace plots.

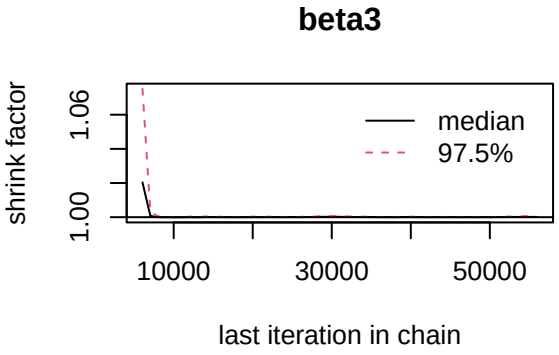
Now we perform sensitivity checks with different values of variance. Our initial model had a variance of 100 for β_0 and 1 for β_n. Now we attempted with two different cases. For the first case where the variance is increased to 1000 for β_0 and 10 for β_n we observed the minor increase in the absolute value of the posterior mean but we still see that the standard deviation and `time series SE` are smaller than the mean. Similarity observed with the second case where the variance is decreased to 10 for β_0 and 0.1 for β_n. Opposite to the previous case we see the decrease in the absolute value of the posterior mean but we still see that the standard deviation and `time series SE` are smaller than the mean. Rest of the statistics and plots also indicate similar results to the original JAGS model. Therefore, we conclude that there is no strong sensitivity to the priors through the 2 additional experiments conducted.

```
gelman.plot(fish.jags.origin)
```





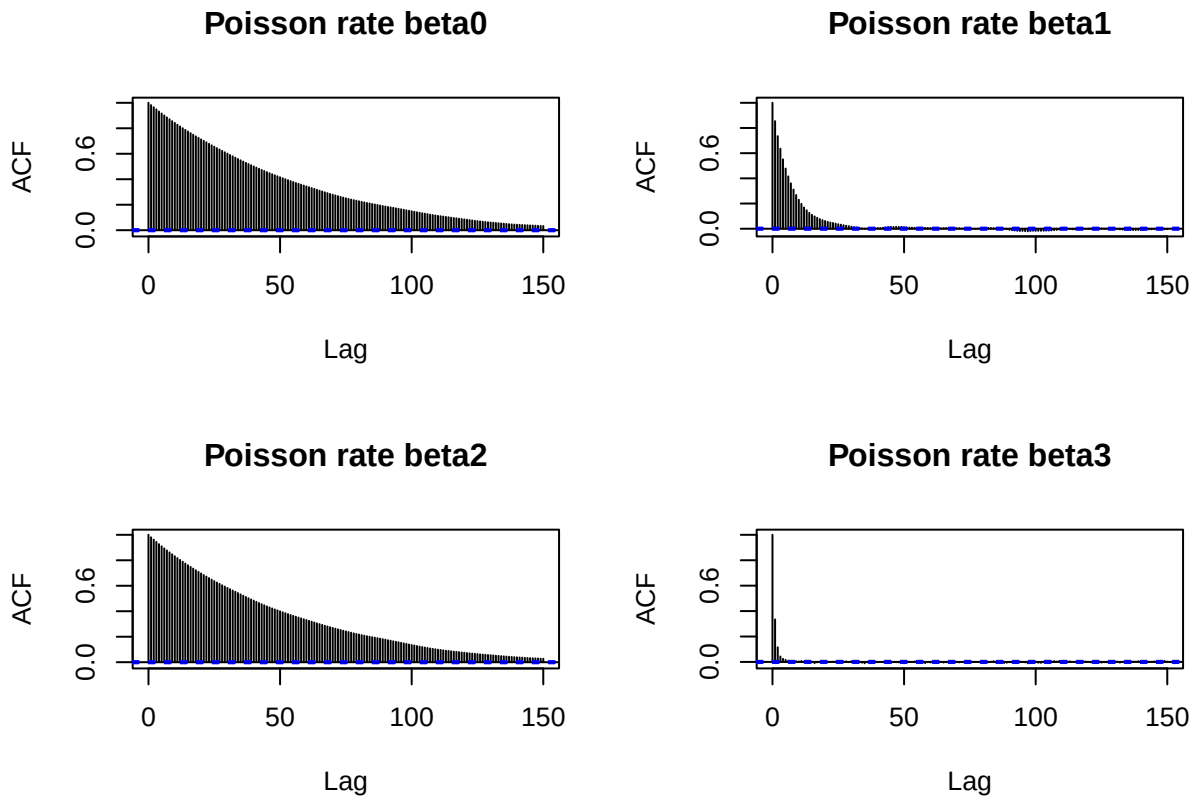





```
gelman.diag(fish.jags.origin)
```

```
## Potential scale reduction factors:
##
##       Point est. Upper C.I.
## beta0       1.01       1.02
## beta1       1.00       1.00
## beta2       1.01       1.03
## beta3       1.00       1.00
##
## Multivariate psrf
##
## 1.01
```

```
par(mfrow=c(2,2))
acf(fish.jags.origin[[1]][,"beta0"], lag.max = 150, main = "Poisson rate beta0")
acf(fish.jags.origin[[1]][,"beta1"], lag.max = 150, main = "Poisson rate beta1")
acf(fish.jags.origin[[1]][,"beta2"], lag.max = 150, main = "Poisson rate beta2")
acf(fish.jags.origin[[1]][,"beta3"], lag.max = 150, main = "Poisson rate beta3")
```




```
effectiveSize(fish.jags.origin)
```

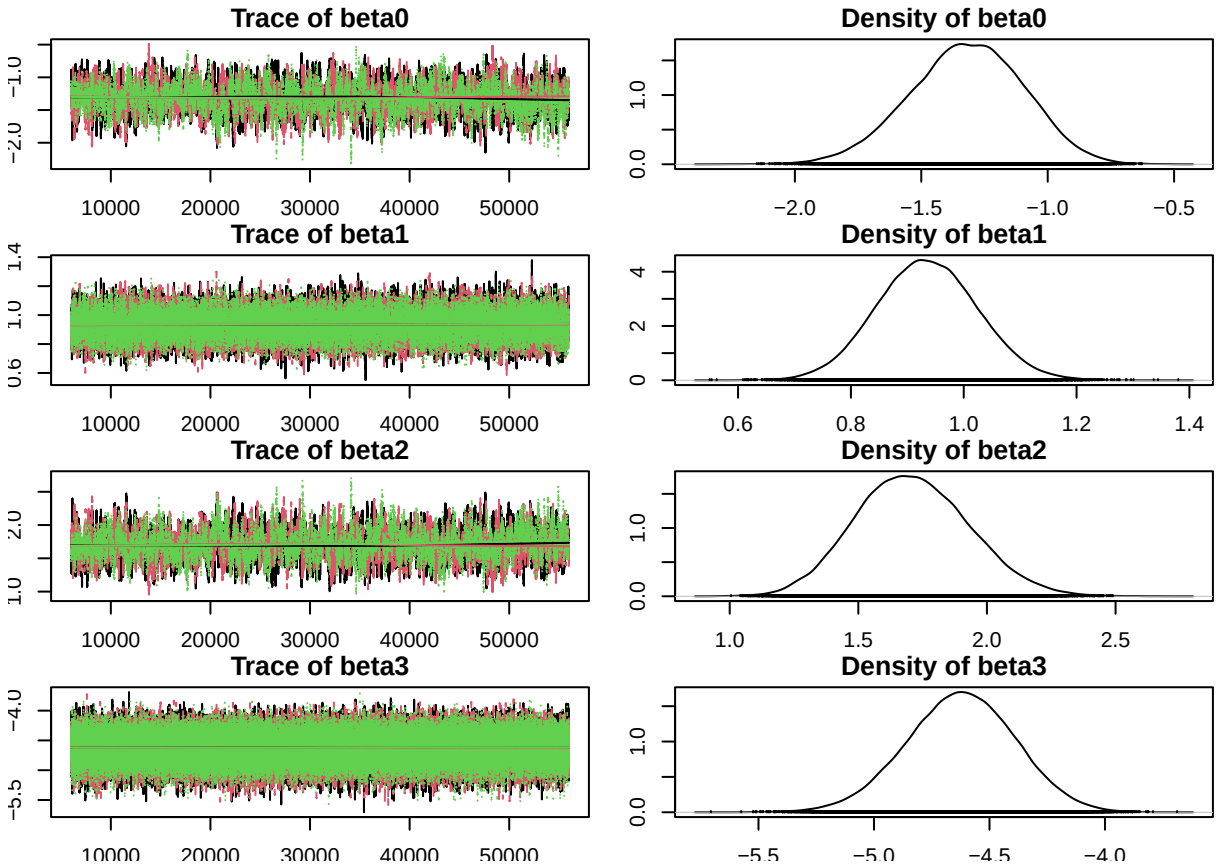
```
##     beta0     beta1     beta2     beta3
##  1339.271 11435.209  1355.991 70907.899
```

```
summary(fish.jags.origin)
```

```
##
## Iterations = 6001:56000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 50000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean      SD  Naive SE Time-series SE
## beta0 -1.3198 0.22475 0.0005803      0.0061515
## beta1  0.9353 0.08902 0.0002299      0.0008328
## beta2  1.7100 0.22071 0.0005699      0.0060125
## beta3 -4.6249 0.23536 0.0006077      0.0008848
##
## 2. Quantiles for each variable:
##
##          2.5%     25%     50%    75%   97.5%
## beta0 -1.7740 -1.4682 -1.3146 -1.163 -0.8989
## beta1  0.7634  0.8746  0.9344  0.995  1.1114
```

```
## beta2  1.2961  1.5555  1.7031  1.858  2.1586
## beta3 -5.0958 -4.7817 -4.6217 -4.464 -4.1730
```

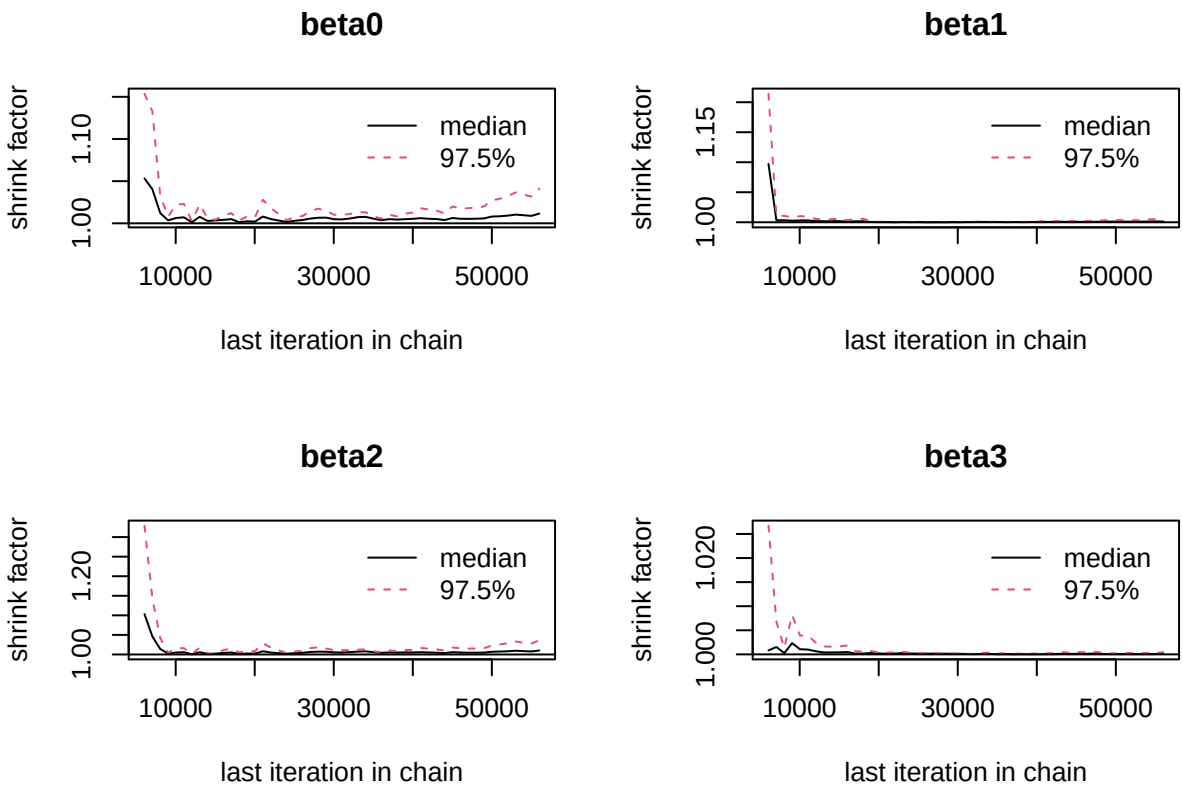
```
par(mar=c(1.7,1.7,1.7,1.7))
plot(fish.jags.origin)
```




```
#perform sensitivity check for priors
fish.data <- list(n=nrow(fish), count = fish$count, livebait = fish$livebait,
                  camper = fish$camper, prop.child = fish$prop.child,
                  persons = fish$persons)
num.chains <- 3

fish.model.string <- "model{
#defining prior
beta0 ~ dnorm(0, 1/1000)
beta1 ~ dnorm(0, 1/10)
beta2 ~ dnorm(0, 1/10)
beta3 ~ dnorm(0, 1/10)

#loop through all data to compute the likelihood
for(i in 1:n){
  log(mu[i]) <- beta0+beta1*camper[i]+beta2*livebait[i]+beta3*prop.child[i] + log(persons[i])
  count[i]~dpois(mu[i])}
}"

results.fish <- jags.model(file = textConnection(fish.model.string),
                           data = fish.data,
```

```
                            n.chains = 3, quiet = TRUE)
#
update(results.fish, n.iter = 5000)
#
fish.jags <- coda.samples(results.fish, n.iter = 50000,
                          variable.names = c("beta0","beta1","beta2","beta3"))
```

```
gelman.plot(fish.jags)
```

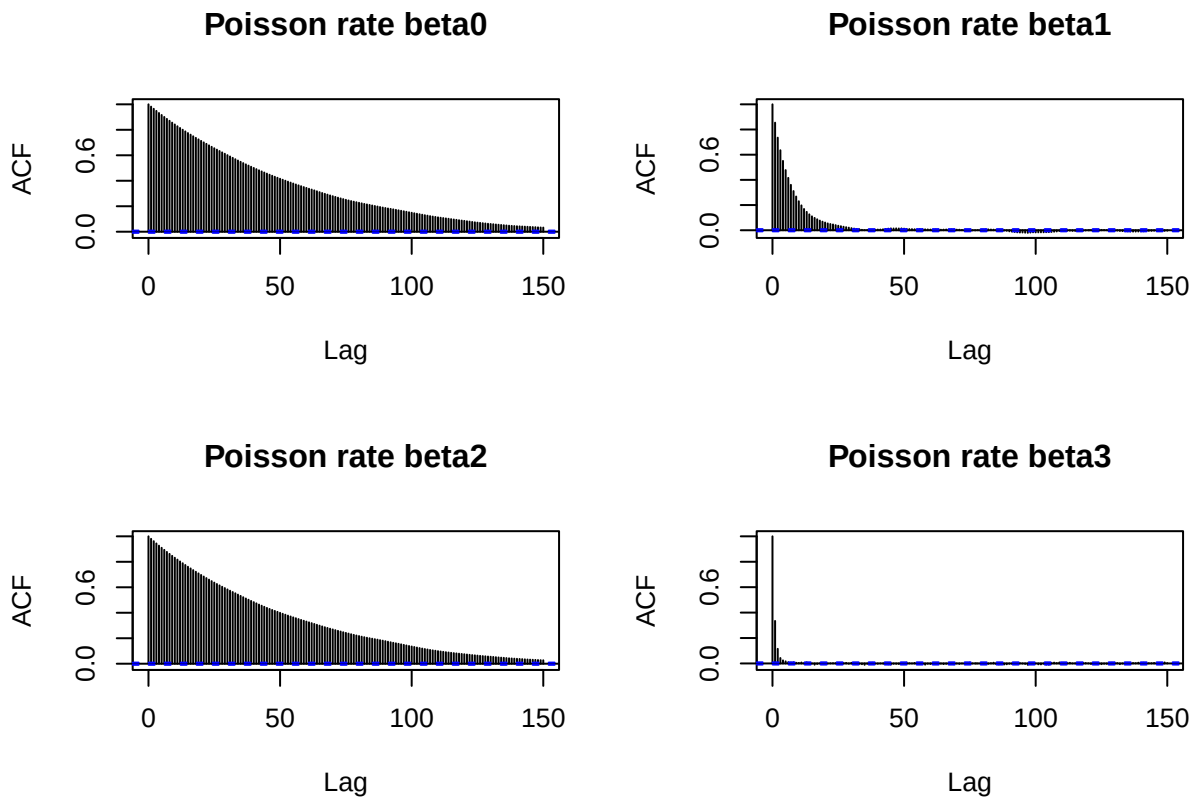



```
gelman.diag(fish.jags)
```

```
## Potential scale reduction factors:
##
##       Point est. Upper C.I.
## beta0       1.01       1.04
## beta1       1.00       1.00
## beta2       1.01       1.04
## beta3       1.00       1.00
##
## Multivariate psrf
##
## 1.01
```

```
par(mfrow=c(2,2))
acf(fish.jags.origin[[1]][,"beta0"], lag.max = 150, main = "Poisson rate beta0")
acf(fish.jags.origin[[1]][,"beta1"], lag.max = 150, main = "Poisson rate beta1")
acf(fish.jags.origin[[1]][,"beta2"], lag.max = 150, main = "Poisson rate beta2")
```

```
acf(fish.jags.origin[[1]][,"beta3"], lag.max = 150, main = "Poisson rate beta3")
```




```
effectiveSize(fish.jags)
```

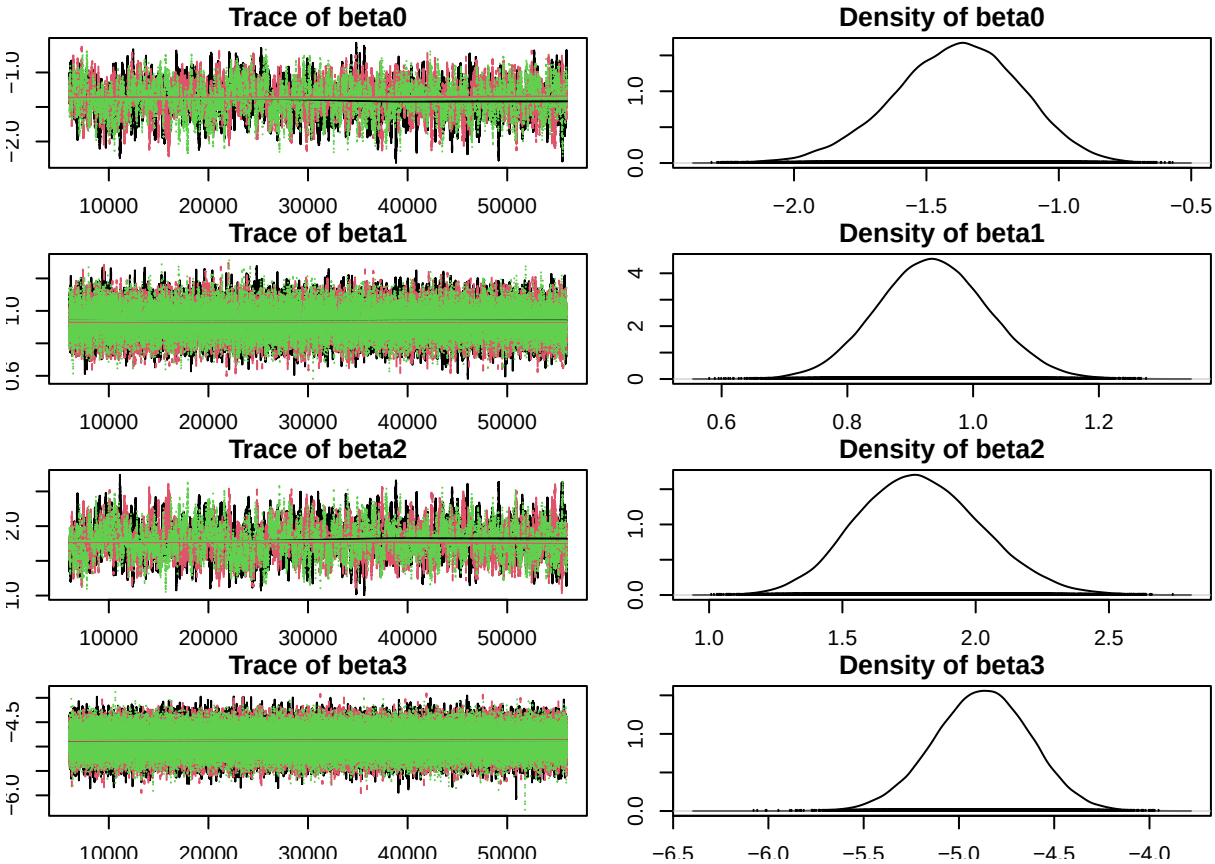
```
##     beta0     beta1     beta2     beta3
##  1105.497 11377.351  1127.395 70833.999
```

```
summary(fish.jags)
```

```
##
## Iterations = 6001:56000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 50000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean      SD  Naive SE Time-series SE
## beta0 -1.3849 0.23774 0.0006139      0.0071423
## beta1  0.9362 0.08798 0.0002272      0.0008249
## beta2  1.7909 0.23306 0.0006018      0.0069378
## beta3 -4.8789 0.25203 0.0006507      0.0009474
##
## 2. Quantiles for each variable:
##
##          2.5%     25%     50%     75%   97.5%
```

```
## beta0 -1.8744 -1.5422 -1.3765 -1.2196 -0.9418
## beta1  0.7666  0.8765  0.9353  0.9949  1.1105
## beta2  1.3570  1.6273  1.7824  1.9450  2.2678
## beta3 -5.3834 -5.0477 -4.8749 -4.7062 -4.3953
```

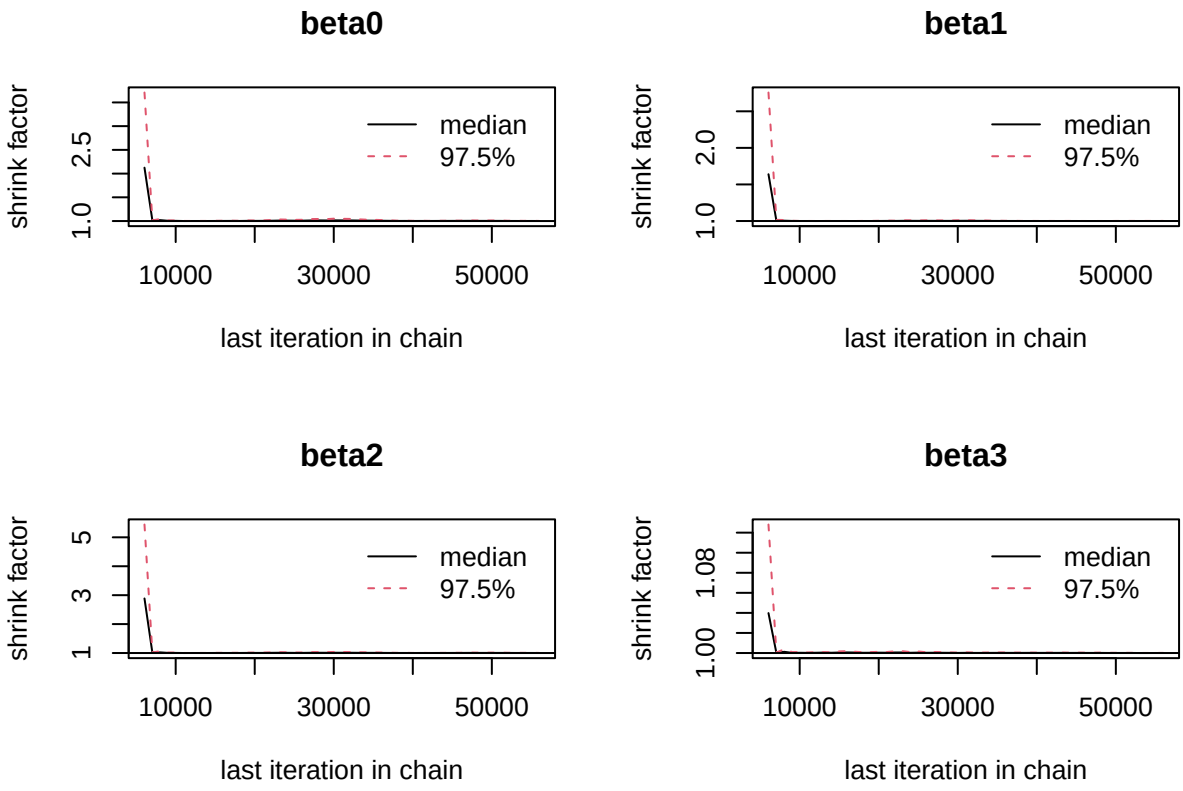
```
par(mar=c(1.7,1.7,1.7,1.7))
plot(fish.jags)
```




```
#perform sensitivity check for priors
fish.data <- list(n=nrow(fish), count = fish$count, livebait = fish$livebait,
                  camper = fish$camper, prop.child = fish$prop.child,
                  persons = fish$persons)
num.chains <- 3

fish.model.string <- "model{
#priors
beta0 ~ dnorm(0, 1/10)
beta1 ~ dnorm(0, 10)
beta2 ~ dnorm(0, 10)
beta3 ~ dnorm(0, 10)

#loop through all data to compute the likelihood
for(i in 1:n){
  log(mu[i]) <- beta0+beta1*camper[i]+beta2*livebait[i]+beta3*prop.child[i] + log(persons[i])
  count[i]~dpois(mu[i])}
}"
```

```
results.fish <- jags.model(file = textConnection(fish.model.string),
                           data = fish.data,
                           n.chains = 3, quiet = TRUE)
#
update(results.fish, n.iter = 5000)
#
fish.jags <- coda.samples(results.fish, n.iter = 50000,
                          variable.names = c("beta0","beta1","beta2","beta3"))
```

```
gelman.plot(fish.jags)
```

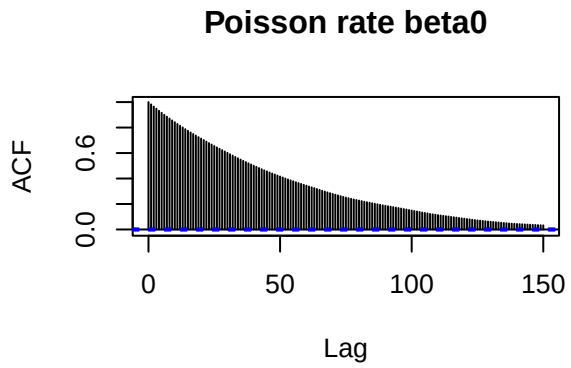


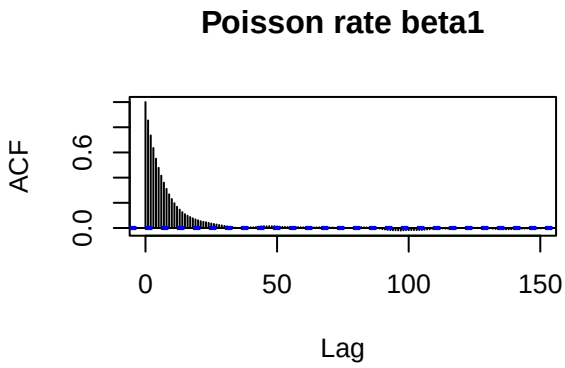

```
gelman.diag(fish.jags)
```

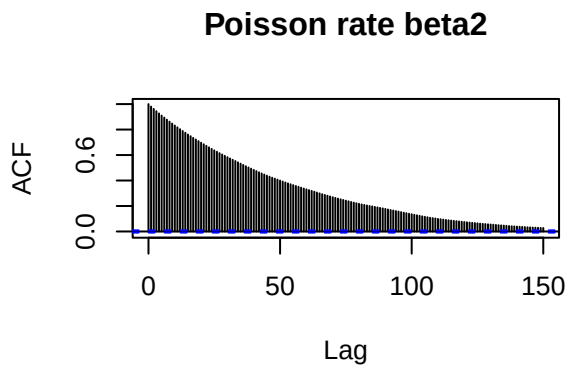
```
## Potential scale reduction factors:
##
##       Point est. Upper C.I.
## beta0          1          1
## beta1          1          1
## beta2          1          1
## beta3          1          1
##
## Multivariate psrf
##
## 1
```

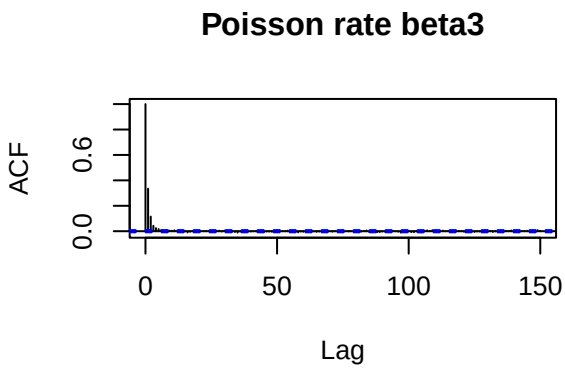
```
par(mfrow=c(2,2))
acf(fish.jags.origin[[1]][,"beta0"], lag.max = 150, main = "Poisson rate beta0")
```

```
acf(fish.jags.origin[[1]][,"beta1"], lag.max = 150, main = "Poisson rate beta1")
acf(fish.jags.origin[[1]][,"beta2"], lag.max = 150, main = "Poisson rate beta2")
acf(fish.jags.origin[[1]][,"beta3"], lag.max = 150, main = "Poisson rate beta3")
```










```
effectiveSize(fish.jags)
```

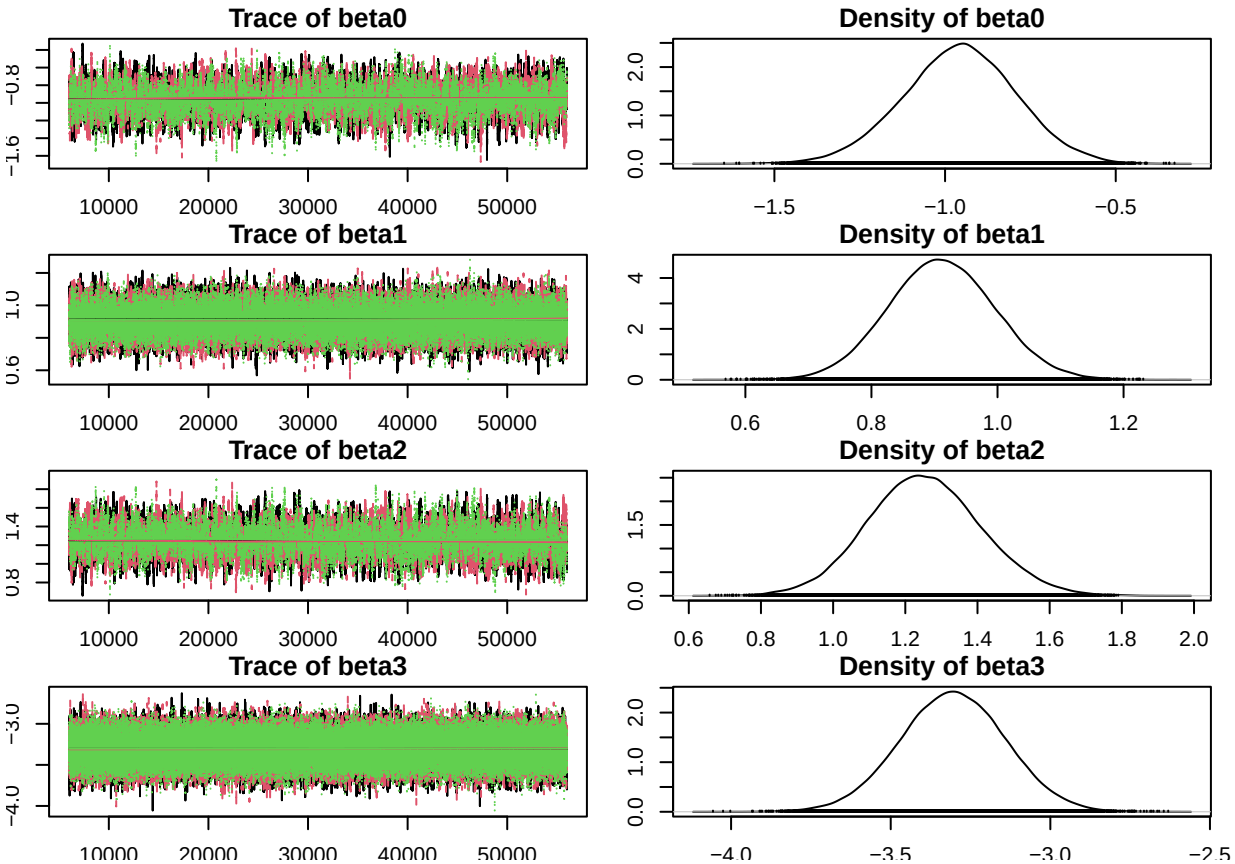
```
##     beta0     beta1     beta2     beta3
##  2431.973 12978.449  2538.772 70253.363
```

```
summary(fish.jags)
```

```
##
## Iterations = 6001:56000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 50000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean      SD  Naive SE Time-series SE
## beta0 -0.9579 0.16625 0.0004293      0.0033716
## beta1  0.9115 0.08401 0.0002169      0.0007375
## beta2  1.2532 0.15729 0.0004061      0.0031198
## beta3 -3.3054 0.16458 0.0004250      0.0006212
##
## 2. Quantiles for each variable:
```

```
##
##           2.5%     25%     50%    75%   97.5%
## beta0 -1.2919 -1.0667 -0.9551 -0.846 -0.6344
## beta1  0.7485  0.8546  0.9107  0.968  1.0778
## beta2  0.9506  1.1463  1.2500  1.357  1.5704
## beta3 -3.6297 -3.4160 -3.3050 -3.194 -2.9833
```

```
par(mar=c(1.7,1.7,1.7,1.7))
plot(fish.jags)
```



d)[10 marks] A closer observation of the number of zeros in the dataset reveals that they are higher than expected. A plausible explanation for this is that some groups went to the park, but did not go fishing at all. To better model this, we consider the zero-inflated Poisson distribution. This is a mixture of a distribution taking 0 with probability 1 with weight w, and a Poisson distribution with weight $1 - w$ (see https://en.wikipedia.org/wiki/Zero-inflated_model for more information).

Implement the zero-inflated version of the model from part c) in JAGS. Use a uniform prior for the weight, and the same priors for the regression coefficients as in part c). [Hint: you can use the dbern or dcat distributions in JAGS as part of this. In order to avoid the incompatible error in JAGS, you may need to approximate Poisson distribution with parameter 0 by a Poisson distribution with a small parameter such as 0.00001.]

Run 5000 burn-in steps, and obtain samples from the weight parameter w **and the regression coefficients, including the intercept). Compute and print out the effective sample sizes (ESS) for these 5 parameters.**

If the ESS is below 1000 for any of these 5 parameters, increase the sample size/number of chains until the ESS is above 1000 for all of them.

Print out the summary of the model. Interpret the posterior summaries of the model parameters.

Explanation:

Now we proceed to the Bayesian analysis of the zero-inflated poisson model different to Q1 b). We specified the priors for β_i as given in the question. Also we included the weight, $1 - w$ that follows a uniform distribution. In case the values are 0 which will return an error, we added 0.00001 as well. We ran with 3 chains with 50000 samples after 5000 burn-in iterations, starting from an initial position `camper`, `livebait`, `prop.child` and `persons`. The weight plays a role in which taking the data into account when some groups went to the park, but did not go fishing at all.

If we now look at the effective sample size using `effectivesize()`, we can see that all parameters have ESS above 1000. Looking at the summary, we can see that the standard deviation is higher compared to the previous poisson model we had. However, the absolute values are still smaller than the absolution value of the posterior mean. This is also observed with the `time series SE`. Now, we look at the coefficients of the covariates. We can see that the coefficients of `camper` and `livebaits` decreased compared to Q1 a). Thus, these covariates have smaller influence in determining the number of fish being caught. This again shows that our estimates are accurate and the chain was mixed well.

```
#defining model data
fish.data <- list(n=nrow(fish), count = fish$count, livebait = fish$livebait,
                  camper = fish$camper, prop.child = fish$prop.child,
                  persons = fish$persons)
num.chains <- 3
#defining model string for zero-inflated poisson distribution
fish.model.string <- "model{
#priors
beta0 ~ dnorm(0, 1/100)
beta1 ~ dnorm(0, 1)
beta2 ~ dnorm(0, 1)
beta3 ~ dnorm(0, 1)
weight ~ dunif(0,1)

#loop through all data to compute the likelihood
for(i in 1:n){
  log(lambda[i]) <- beta0 + beta1*camper[i] + beta2*livebait[i] +
  beta3*prop.child[i] + log(persons[i])
  #forming the weight with bernoulli distribution
```

```
  z[i] ~ dbern(weight)
  #applying the weights
  mu[i] <- lambda[i]*z[i] + 0.00001
  count[i] ~ dpois(mu[i])}
}"
```

```
results.zeroinflate <- jags.model(file = textConnection(fish.model.string),
                          data = fish.data,
                          n.chains = 3, quiet = TRUE)
#burn-in for 5000 values.
update(results.zeroinflate, n.iter = 5000)
#running the JAGS model for 50000 variables monitoring the parameters
zero.inflate.chain <- coda.samples(results.zeroinflate, n.iter = 50000,
                         variable.names = c("beta0", "beta1", "beta2",
                                            "beta3", "weight"))
```

```
effectiveSize(zero.inflate.chain)
```

```
##     beta0     beta1     beta2     beta3    weight
##  1056.360 10061.617  1089.034 40175.715 48661.472
```

```
summary(zero.inflate.chain)
```

```
##
## Iterations = 6001:56000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 50000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean      SD  Naive SE Time-series SE
## beta0  -0.7962 0.23911 0.0006174      0.0073951
## beta1   0.6844 0.09493 0.0002451      0.0009464
## beta2   1.6631 0.23483 0.0006063      0.0071466
## beta3  -3.6294 0.30030 0.0007754      0.0014982
## weight  0.5323 0.03939 0.0001017      0.0001792
##
## 2. Quantiles for each variable:
##
##           2.5%     25%     50%     75%   97.5%
## beta0  -1.2883 -0.9512 -0.7883 -0.6327 -0.3477
## beta1   0.5013  0.6196  0.6833  0.7482  0.8719
## beta2   1.2236  1.5026  1.6559  1.8148  2.1480
## beta3  -4.2218 -3.8313 -3.6268 -3.4261 -3.0442
## weight  0.4557  0.5055  0.5319  0.5587  0.6101
```

e)[10 marks]

Compare the quality of the model fit for the two models via DIC scores.

Perform posterior predictive checks for both models for 3 functions:

1. number of groups catching no fish,

2. number of groups catching more than 5 fish,

3. average number of fish caught among groups who brought a camper van.

Discuss your results.

Explanation:

We compared the DIC values between the two models that we defined in the previous parts using `dic.samples()`. From this, we can see that zero-inflated model with more informative prior has a lower penalised deviance with 1643. Therefore, this indicates a better fit on the data for the zero-inflated poisson model.

Now we perform the posterior predictive check for poisson model. We can see that the posterior predictive checks did not detect any issues only for the average number of fish caught among groups who brought a camper van. For the other two functions, number of groups catching no fish, and number of groups catching more than 5 fish, we detected great abnormally where the true value is outside the typical range of the replicates.

Similarly, we conducted the posterior predictive check for the zero-inflated model. For this time, we can see that the check did not detect any issue for the average number of fish caught among groups who brought a camper van and the number of groups catching no fish. The values of the functions were in the typical range of the realizations of the replicates. Especially, the zero-inflated model well represents the case for first function. However, we still faced an abnormally where the true value is outside the typical range of the replicates for the function of number of groups catching more than 5 fish. This is because the total count of the second function is only 29 which only 10% of the total dataset. As a result, we do not get an accurate result to represent the case which only has small sample size.

```
#DIC value for poisson model
dic.samples(results.fish.origin, n.iter = 10000, progress.bar = "none")
```

```
## Mean deviance:  1923
## penalty 3.897
## Penalized deviance: 1927
```

```
#DIC values for zero-inflated poisson model
dic.samples(results.zeroinflate, n.iter = 10000, progress.bar = "none")
```

```
## Mean deviance:  1382
## penalty 260.7
## Penalized deviance: 1643
```

```
#defining model string
fish.data <- list(n = nrow(fish), count = fish$count, livebait = fish$livebait,
                  camper = fish$camper, prop.child = fish$prop.child,
                  persons = fish$persons)
num.chains <- 3
#defining separate model string for posterior probability check for
#poisson model
model.string.replicate <- "model{
beta0 ~ dnorm(0, 1/100)
beta1 ~ dnorm(0, 1)
beta2 ~ dnorm(0, 1)
```

```
beta3 ~ dnorm(0, 1)

for(i in 1:n){
  log(mu[i]) <- beta0+beta1*camper[i]+beta2*livebait[i]+beta3*prop.child[i] + log(persons[i])
  count[i] ~ dpois(mu[i])
  replicate[i] ~ dpois(mu[i])
  }
}"
```

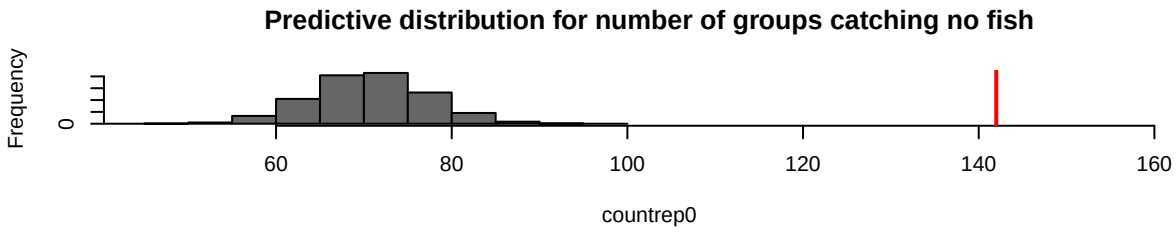
```
model.replicate.1 <- jags.model(textConnection(model.string.replicate),
                    data = fish.data, n.chains = 3)
```

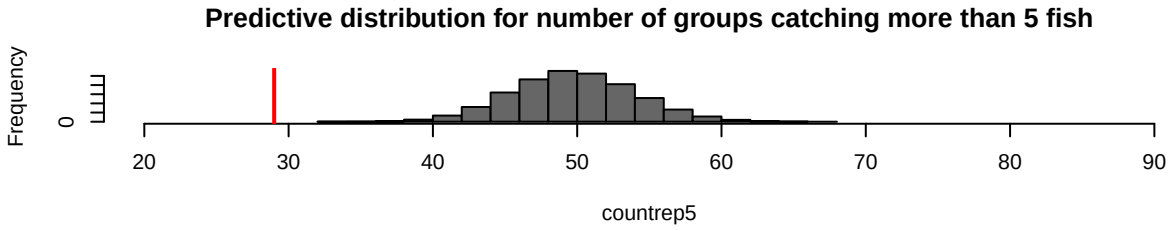
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 250
##    Unobserved stochastic nodes: 254
##    Total graph size: 1597
##
## Initializing model
```

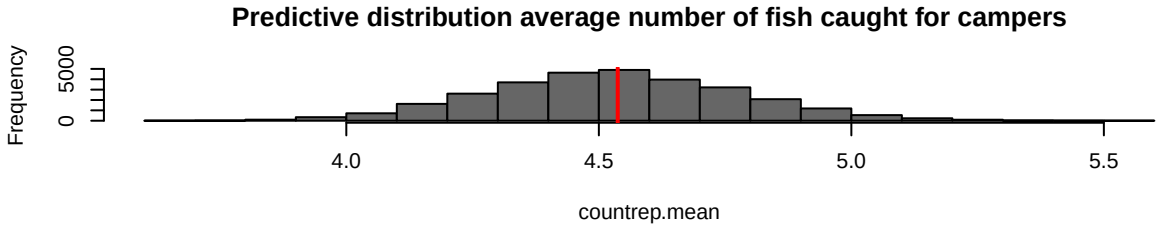
```
#MCMC Sampling with burnin 10000 and 10000 iterations to perform
#posterior predictive checks
update(model.replicate.1, 1000, progress.bar = "none")
res.replicate.1 <- coda.samples(model.replicate.1,
                        variable.names = c("replicate"),
                        n.iter = 10000, progress.bar = "none")
```

```
#Compute posterior predictive distribution for 3 functions
countrep <- t(as.matrix(res.replicate.1))
countrep0 <- apply(countrep, 2, function(x){length(which(x==0))})
countrep5 <- apply(countrep, 2, function(x){length(which(x>5))})
campervan.idx <- which(fish$camper == 1)
countrep.mean <- apply(countrep[campervan.idx,], 2, mean)
```

```
par(mfrow=c(3,1))
hist(countrep0, col = "gray40", xlim=c(45, 160),
     main = "Predictive distribution for number of groups catching no fish")
abline(v = sum(fish$count==0), col = "red", lwd = 2)
hist(countrep5, col = "gray40", xlim=c(20,90),
     main = "Predictive distribution for number of groups catching more than 5 fish")
abline(v = nrow(fish[fish$count>5,]), col = "red", lwd = 2)
hist(countrep.mean, col = "gray40",
     main = "Predictive distribution average number of fish caught for campers")
abline(v = mean(fish$count[campervan.idx], na.rm = TRUE), col = "red", lwd = 2)
```








```
#defining model data
fish.data <- list(n=nrow(fish), count = fish$count, livebait = fish$livebait,
                  camper = fish$camper, prop.child = fish$prop.child,
                  persons = fish$persons)
num.chains <- 3

#defining separate model string for posterior probability check for
#zero-inflated poisson model
model.string.replicate <- "model{
beta0 ~ dnorm(0, 1/100)
beta1 ~ dnorm(0, 1)
beta2 ~ dnorm(0, 1)
beta3 ~ dnorm(0, 1)
weight ~ dunif(0,1)

for(i in 1:n){
  log(lambda[i]) <- beta0 + beta1*camper[i] + beta2*livebait[i] +
  beta3*prop.child[i] + log(persons[i])
  z[i] ~ dbern(weight)
  mu[i] <- lambda[i]*z[i] + 0.00001
  count[i] ~ dpois(mu[i])
  replicate[i] ~ dpois(mu[i])
  }
}"
```

```
model.replicate.2 <- jags.model(textConnection(model.string.replicate),
```

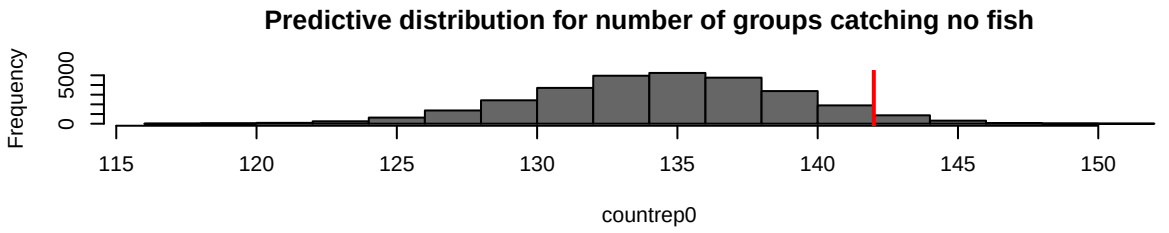
```
                    data = fish.data, n.chains = 3)
```

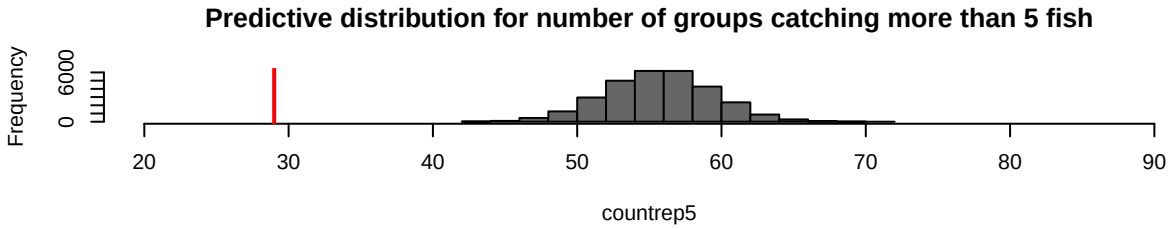
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 250
##    Unobserved stochastic nodes: 505
##    Total graph size: 2349
##
## Initializing model
```

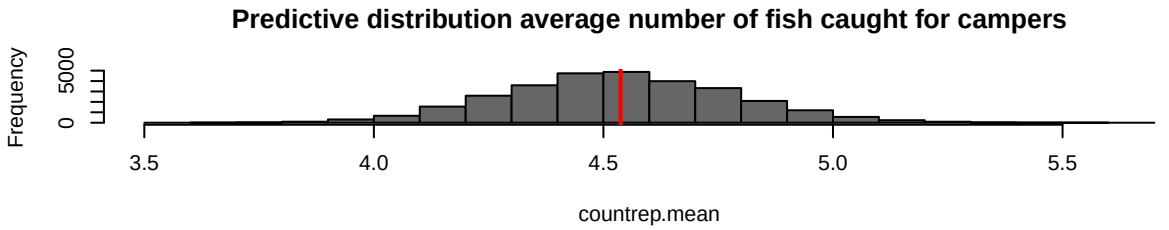
```
#MCMC Sampling with burnin 10000 and 10000 iterations to perform
#posterior predictive checks
update(model.replicate.2, 1000, progress.bar = "none")
res.replicate.2 <- coda.samples(model.replicate.2,
                        variable.names = c("replicate"),
                        n.iter = 10000, progress.bar = "none")
```

```
#Compute posterior predictive distribution for 3 functions
countrep <- t(as.matrix(res.replicate.2))
countrep0 <- apply(countrep, 2, function(x){length(which(x==0))})
countrep5 <- apply(countrep, 2, function(x){length(which(x>5))})
campervan.idx <- which(fish$camper == 1)
countrep.mean <- apply(countrep[campervan.idx,], 2, mean)
```

```
par(mfrow=c(3,1))
hist(countrep0, col = "gray40",
     main = "Predictive distribution for number of groups catching no fish")
abline(v = sum(fish$count==0), col = "red", lwd = 2)
hist(countrep5, col = "gray40", xlim=c(20,90),
     main = "Predictive distribution for number of groups catching more than 5 fish")
abline(v = nrow(fish[fish$count>5,]), col = "red", lwd = 2)
hist(countrep.mean, col = "gray40",
     main = "Predictive distribution average number of fish caught for campers")
abline(v = mean(fish$count[campervan.idx], na.rm = TRUE), col = "red", lwd = 2)
```


Predictive distribution for number of groups catching no fish
Frequency
0
5000
115
120
125
130
135
140
145
150
countrep0


Predictive distribution for number of groups catching more than 5 fish
Frequency
0
6000
20
30
40
50
60
70
80
90
countrep5


Predictive distribution average number of fish caught for campers
Frequency
0
5000
3.5
4.0
4.5
5.0
5.5
countrep.mean

Problem 2 - Personal loans data

In this problem, we are going to analyze some personal loans data from a US bank named LendingClub.

The dataset contains information about 188181 personal loans in the US whose term has already completed.

The column that we aim to model is loan.status, which is either ''Fully Paid", or "Charged Off". In our response variable y, value 1 will to "Charged Off", while value 0 corresponds to "Fully paid". Hence we are modelling whether or not the client defaulted on their loan.

The other columns in the dataset are

loan.amnt: The amount of principle given in the loan in USD

interest.rate: The interest rate assigned to the loan by LendingClub

sub.grade: A grade assigned to the loan internally by LendingClub

annual.income: The lendee's annual income in USD

debt.to.income: The lendee's debt-to-income ratio. This explains what percent of the client's income is spent on loans each month before taking out this new personal loan (such as mortgage, rent, car loan, etc., see https://www.lendingclub.com/loans/resource-center/calculating-debt-to-income)

state, the US state in which the borrower lives (in a 2 letter abbreviated format)

term, the term of the loan, i.e. the number of months in which it has to be paid off

loan.status: The final status, either "Paid" or "Charged Off", of the loan

We are going to use INLA to fit several different logistic regression models to this dataset. First, we load ILNA and the dataset and display the first few rows.

```
lending <- read.csv(file = 'lending.csv')
```

a)[10 marks]

In this question, we are going to fit a logistic GLM on the dataset. First, we need to do some data manipulation.

Create a response column y such that $y = 1$ corresponds to loan.status being "Charged Off", and $y = 0$ corresponds to loan.status being "Paid".

Convert the interest.rate and debt.to.income columns from strings into numbers [Hint: you can use the str_replace_all function from the stringr library to get rid of the % signs].

Create new columns for the logarithm of the loan amount, the logarithm of the interest rate, the logarithm of the annual income.

Scale these 3 columns (i.e. center them and divide them by their standard deviation). Also scale the debt-to-income column (but do not apply log transformation).

Fit a logistic GLM on the response y as a function of the scaled log interest rate, scaled log annual income, scaled log loan amount, scaled debt-to-income, sub.grade, and term (sub.grade and term can be kept categorical). Interpret the results.

Explanation:

We fitted a logistic GLM using `glm()` where y is the response variable and `scaled log interest rate`, `scaled log annual income`, `scaled log loan amount`, `scaled debt-to-income`, `sub,grade` and `term` as the explanatory variable using. Note that for efficient memory allocation purpose we only employed the first $100,000$ rows and we will maintain this consistently through this question.

We then computed the summary statistics using `summary()`. `sub.grade(A1)` and `term(36 months)` are included in the intercept. The rest of the MLE of the covariates are the log odds ratios adjusting the other covariates. The summary states that most of the variables are significant. However, we have some variables such as `scaled log interest rate` is not significant as the p-value is 0.173336 which is greater than 0.05.

```
lending$interest.rate <- as.numeric(gsub('%', '', lending$interest.rate))/100
lending$debt.to.income <- as.numeric(gsub('%', '', lending$debt.to.income))/100
lending$y <- ifelse(lending$loan.status == "Fully Paid", 0, 1)
lending$sub.grade <- as.factor(lending$sub.grade)
lending$term <- as.factor(lending$term)
head(lending)
```

```
##   loan.amount interest.rate sub.grade annual.income debt.to.income loan.status
## 1       12000        0.1199        B3        130000         0.1303  Fully Paid
## 2       15000        0.0890        A5         63000         0.1651  Fully Paid
## 3       24000        0.1353        B5        100000         0.2218  Fully Paid
## 4       12000        0.1353        B5         40000         0.1694  Fully Paid
## 5       10000        0.0967        B1        102000         0.1555  Fully Paid
## 6       11100        0.1498        C3         90000         0.0373  Fully Paid
##   state       term y
## 1    CO  36 months 0
## 2    FL  36 months 0
## 3    MI  36 months 0
## 4    NM  36 months 0
## 5    MA  36 months 0
## 6    NY  36 months 0
```

```
# taking log for the selected variables
lending$log.loan.amount.noscale <- log(lending$loan.amount)
```

```
lending$log.interest.rate.noscale <- log(lending$interest.rate)
lending$log.annual.income.noscale <- log(lending$annual.income)
lending$debt.to.income.noscale <- lending$debt.to.income
# scaling the selected variables
lending$log.loan.amount <- scale(lending$log.loan.amount.noscale)[, 1]
lending$log.interest.rate <- scale(lending$log.interest.rate.noscale)[, 1]
lending$log.annual.income <- scale(lending$log.annual.income.noscale)[, 1]
lending$debt.to.income <- scale(lending$debt.to.income)[, 1]
head(lending)
```

```
##   loan.amount interest.rate sub.grade annual.income debt.to.income loan.status
## 1       12000        0.1199        B3        130000    -0.53045087  Fully Paid
## 2       15000        0.0890        A5         63000    -0.07241352  Fully Paid
## 3       24000        0.1353        B5        100000     0.67387148  Fully Paid
## 4       12000        0.1353        B5         40000    -0.01581695  Fully Paid
## 5       10000        0.0967        B1        102000    -0.19876865  Fully Paid
## 6       11100        0.1498        C3         90000    -1.75451621  Fully Paid
##   state       term y log.loan.amount.noscale log.interest.rate.noscale
## 1    CO  36 months 0                9.392662                 -2.121097
## 2    FL  36 months 0                9.615805                 -2.419119
## 3    MI  36 months 0               10.085809                 -2.000261
## 4    NM  36 months 0                9.392662                 -2.000261
## 5    MA  36 months 0                9.210340                 -2.336142
## 6    NY  36 months 0                9.314700                 -1.898454
##   log.annual.income.noscale debt.to.income.noscale log.loan.amount
## 1                  11.77529                 0.1303     0.009162849
## 2                  11.05089                 0.1651     0.349190565
## 3                  11.51293                 0.2218     1.065385408
## 4                  10.59663                 0.1694     0.009162849
## 5                  11.53273                 0.1555    -0.268660025
## 6                  11.40756                 0.0373    -0.109635498
##   log.interest.rate log.annual.income
## 1      -0.369590574      1.4283392699
## 2      -1.267983279     -0.0007421113
## 3      -0.005326466      0.9107522623
## 4      -0.005326466     -0.8968878859
## 5      -1.017847256      0.9498184920
## 6       0.301571494      0.7028991307
```

```
fit <- glm(y ~ log.interest.rate + log.annual.income +
                log.loan.amount + debt.to.income + sub.grade + term,
           data = lending[1:100000,], family="binomial")
```

```
summary(fit)
```

```
##
## Call:
## glm(formula = y ~ log.interest.rate + log.annual.income + log.loan.amount +
##     debt.to.income + sub.grade + term, family = "binomial", data = lending[1:1e+05,
##     ])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2604  -0.6339  -0.4806  -0.3312   3.0078
##
```

```
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -3.309874   0.347332  -9.529  < 2e-16 ***
## log.interest.rate  0.170062   0.124902   1.362 0.173336
## log.annual.income -0.303997   0.011558 -26.302  < 2e-16 ***
## log.loan.amount    0.184846   0.012214  15.133  < 2e-16 ***
## debt.to.income     0.085634   0.009379   9.131  < 2e-16 ***
## sub.gradeA2        0.352289   0.212417   1.658 0.097220 .
## sub.gradeA3        0.419032   0.217384   1.928 0.053904 .
## sub.gradeA4        0.781450   0.213674   3.657 0.000255 ***
## sub.gradeA5        0.903606   0.232323   3.889 0.000100 ***
## sub.gradeB1        0.929445   0.253229   3.670 0.000242 ***
## sub.gradeB2        1.065111   0.281323   3.786 0.000153 ***
## sub.gradeB3        1.106781   0.306898   3.606 0.000311 ***
## sub.gradeB4        1.229316   0.330132   3.724 0.000196 ***
## sub.gradeB5        1.320525   0.350137   3.771 0.000162 ***
## sub.gradeC1        1.343557   0.361864   3.713 0.000205 ***
## sub.gradeC2        1.367265   0.379046   3.607 0.000310 ***
## sub.gradeC3        1.536332   0.393092   3.908 9.29e-05 ***
## sub.gradeC4        1.603986   0.406080   3.950 7.82e-05 ***
## sub.gradeC5        1.611017   0.424588   3.794 0.000148 ***
## sub.gradeD1        1.667653   0.438383   3.804 0.000142 ***
##  [ reached getOption("max.print") -- omitted 20 rows ]
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 86653  on 99999  degrees of freedom
## Residual deviance: 80456  on 99960  degrees of freedom
## AIC: 80536
##
## Number of Fisher Scoring iterations: 6
```

b)[10 marks] (Identical Model)

Implement a Bayesian logistic GLM of the same structure as in a) in INLA [Hint: use need to use the "binomial" family with Ntrials=1 to indicate logistic regression in INLA, see https://rpubs.com/corey_sparks/431920 for an example).

Propose your own priors for the model parameters (you can take into account slides 45-46 of Lecture 4).

Print out the model summary. Check the sensitivity of the results with respect to the prior choices.

Compute DIC and negative log-CPO (NLSCPO).

Do posterior predictive checks with the functions being evaluated chosen as the proportion of defaults among clients in each of the 49 states contained in the dataset [Hint: plot these in a 7 x 7 format using par(mfrow=c(7,7)). Make sure to write par(mfrow=c(1,1)) after your plots to change back to the default setting]. Discuss the model fit.

Explanation:

Now we implement a Bayesian logistic GLM using `inla()`. Referring to slides 45-46 of Lecture 4, we choose the variance of the parameters to be 100 as well as the intercept. With this, we computed the summary statistics as shown below.

We then performed sensitivity check on the prior values. In this case, we used different precision values. Therefore, we tried 0.001 and 1. As we can see the summary statistics, we do not observe the difference in the mean and standard deviation values for `prec=0.001`. There is a slight decrease in the mean and standard deviation values when we increased the precision values to `prec=1`. However this change is not crucial and we can conclude that these show that the prior is not sensitive and an sufficiently noninformative prior.

We computed the DIC and negative log-CPO and the values are 80535.79 and 40267.95 respectively.

For the posterior predictive checks with the functions being evaluated chosen as the proportion of defaults among clients in each of the 49 states, we plotted the histogram of posterior probability proportion according to each state. Before we proceed, we could not plot for 2 states which are `Iowa, IA` and `Mississippi, MS`. This is because, since we only selected the first $100,000$ rows, we only have 1 data point for these two states. The red vertical line is the true frequency observed from the data. We can observe that states such as `Colorado, CO`, `New York, NY`, `New Jersey, NJ`, `South Carolina, SC` have the true frequency out of the range. However, we still managed to retrieve some good results such as `New Mexico, NM`, `Ohio, OH`, `California, CA`, `Arizona, AZ` and many more. Thus we can say that the identical model is moderately well represented but we expect better performances with independent and hierarchical model.

```
#defining the prior for each beta.
prior.beta.1 <- list(mean.intercept = 0, prec.intercept = 0.01,
                     mean = 0, prec = 0.01)

#INLA logistic regression model
fit.inla.1 <- inla(y ~ log.interest.rate + log.annual.income +
                   log.loan.amount + debt.to.income + sub.grade + term,
                 data = lending[1:100000,], family = "binomial", Ntrials = 1,
                 control.compute = list(config = TRUE, dic = TRUE, cpo = TRUE),
                 control.fixed = prior.beta.1)
summary(fit.inla.1)
```

```
##
## Call:
##    c("inla(formula = y ~ log.interest.rate + log.annual.income +
##    log.loan.amount + ", " debt.to.income + sub.grade + term, family =
##    \"binomial\", data = lending[1:1e+05, ", " ], Ntrials = 1,
```

```
##    control.compute = list(config = TRUE, dic = TRUE, ", " cpo = TRUE),
##    control.fixed = prior.beta.1)")
## Time used:
##     Pre = 1.16, Running = 20.9, Post = 0.663, Total = 22.7
## Fixed effects:
##                     mean    sd 0.025quant 0.5quant 0.975quant   mode kld
## (Intercept)       -3.230 0.339     -3.885   -3.234     -2.556 -3.241   0
## log.interest.rate  0.199 0.123     -0.034    0.197      0.448  0.192   0
## log.annual.income -0.304 0.012     -0.327   -0.304     -0.281 -0.304   0
## log.loan.amount    0.185 0.012      0.161    0.185      0.209  0.185   0
## debt.to.income     0.086 0.009      0.067    0.086      0.104  0.086   0
## sub.gradeA2        0.327 0.210     -0.080    0.325      0.744  0.321   0
## sub.gradeA3        0.381 0.214     -0.032    0.379      0.806  0.375   0
## sub.gradeA4        0.740 0.209      0.336    0.738      1.158  0.733   0
## sub.gradeA5        0.854 0.227      0.412    0.853      1.302  0.850   0
## sub.gradeB1        0.873 0.247      0.388    0.873      1.358  0.873   0
## sub.gradeB2        1.002 0.274      0.460    1.003      1.537  1.005   0
## sub.gradeB3        1.037 0.299      0.443    1.039      1.619  1.043   0
## sub.gradeB4        1.154 0.322      0.513    1.156      1.778  1.162   0
## sub.gradeB5        1.240 0.342      0.559    1.243      1.901  1.250   0
##  [ reached getOption("max.print") -- omitted 26 rows ]
##
## Deviance Information Criterion (DIC) ...............: 80535.79
## Deviance Information Criterion (DIC, saturated) ....: -1485038.77
## Effective number of parameters .....................: 39.94
##
## Marginal log-Likelihood:  -40434.59
## CPO and PIT are computed
##
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

```r
#sensitivity check 2 with prec = 0.001
prior.beta.2 <- list(mean.intercept = 0, prec.intercept = 0.001,
                     mean = 0, prec = 0.001)

# INLA logistic regression model
fit.inla.2 <- inla(y ~ log.interest.rate + log.annual.income +
                log.loan.amount + debt.to.income + sub.grade + term,
              data = lending[1:100000,], family = "binomial", Ntrials = 1,
              control.compute = list(config = TRUE, dic = TRUE, cpo = TRUE),
              control.fixed = prior.beta.2)
summary(fit.inla.2)
```

```
##
## Call:
##    c("inla(formula = y ~ log.interest.rate + log.annual.income +
##    log.loan.amount + ", " debt.to.income + sub.grade + term, family =
##    \"binomial\", data = lending[1:1e+05, ", " ], Ntrials = 1,
##    control.compute = list(config = TRUE, dic = TRUE, ", " cpo = TRUE),
##    control.fixed = prior.beta.2)")
## Time used:
##     Pre = 1.05, Running = 21.1, Post = 0.581, Total = 22.8
## Fixed effects:
##                     mean    sd 0.025quant 0.5quant 0.975quant   mode kld
```

```
## (Intercept)       -3.307 0.346     -3.975   -3.311     -2.616 -3.319   0
## log.interest.rate  0.173 0.125     -0.064    0.171      0.426  0.165   0
## log.annual.income -0.304 0.012     -0.327   -0.304     -0.281 -0.304   0
## log.loan.amount    0.185 0.012      0.161    0.185      0.209  0.185   0
## debt.to.income     0.086 0.009      0.067    0.086      0.104  0.086   0
## sub.gradeA2        0.348 0.212     -0.062    0.346      0.770  0.342   0
## sub.gradeA3        0.412 0.217     -0.007    0.410      0.844  0.406   0
## sub.gradeA4        0.775 0.213      0.364    0.773      1.200  0.768   0
## sub.gradeA5        0.898 0.232      0.446    0.897      1.356  0.894   0
## sub.gradeB1        0.925 0.253      0.429    0.925      1.420  0.925   0
## sub.gradeB2        1.061 0.281      0.507    1.063      1.609  1.065   0
## sub.gradeB3        1.103 0.306      0.495    1.106      1.698  1.110   0
## sub.gradeB4        1.226 0.329      0.570    1.229      1.863  1.235   0
## sub.gradeB5        1.317 0.349      0.620    1.321      1.992  1.328   0
##  [ reached getOption("max.print") -- omitted 26 rows ]
##
## Deviance Information Criterion (DIC) ...............: 80535.86
## Deviance Information Criterion (DIC, saturated) ....: -1485038.71
## Effective number of parameters .....................: 40.00
##
## Marginal log-Likelihood:  -40480.17
## CPO and PIT are computed
##
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

```
#sensitivity check 3 with prec = 1
prior.beta.3 <- list(mean.intercept = 0, prec.intercept = 1,
                     mean = 0, prec = 1)

# INLA logistic regression model
fit.inla.3 <- inla(y ~ log.interest.rate + log.annual.income +
                     log.loan.amount + debt.to.income + sub.grade + term,
                   data = lending[1:100000,], family = "binomial", Ntrials = 1,
                   control.compute = list(config = TRUE, dic = TRUE, cpo = TRUE),
                   control.fixed = prior.beta.3)
summary(fit.inla.3)
```

```
##
## Call:
##    c("inla(formula = y ~ log.interest.rate + log.annual.income +
##    log.loan.amount + ", " debt.to.income + sub.grade + term, family =
##    \"binomial\", data = lending[1:1e+05, ", " ], Ntrials = 1,
##    control.compute = list(config = TRUE, dic = TRUE, ", " cpo = TRUE),
##    control.fixed = prior.beta.3)")
## Time used:
##     Pre = 0.89, Running = 21.1, Post = 0.623, Total = 22.6
## Fixed effects:
##                     mean    sd 0.025quant 0.5quant 0.975quant   mode kld
## (Intercept)       -2.162 0.130     -2.417   -2.162     -1.908 -2.161   0
## log.interest.rate  0.565 0.060      0.449    0.565      0.683  0.564   0
## log.annual.income -0.304 0.012     -0.327   -0.304     -0.281 -0.304   0
## log.loan.amount    0.185 0.012      0.161    0.185      0.209  0.185   0
## debt.to.income     0.086 0.009      0.067    0.086      0.104  0.086   0
## sub.gradeA2        0.039 0.179     -0.312    0.039      0.391  0.039   0
```

```
## sub.gradeA3       -0.047 0.161     -0.361   -0.047      0.270 -0.048   0
## sub.gradeA4        0.256 0.143     -0.021    0.255      0.538  0.253   0
## sub.gradeA5        0.245 0.130     -0.008    0.244      0.504  0.242   0
## sub.gradeB1        0.159 0.123     -0.080    0.158      0.403  0.156   0
## sub.gradeB2        0.175 0.123     -0.065    0.174      0.418  0.173   0
## sub.gradeB3        0.116 0.126     -0.131    0.115      0.364  0.114   0
## sub.gradeB4        0.150 0.130     -0.105    0.150      0.406  0.149   0
## sub.gradeB5        0.171 0.137     -0.098    0.171      0.441  0.171   0
##   [ reached getOption("max.print") -- omitted 26 rows ]
##
## Deviance Information Criterion (DIC) ...............: 80543.62
## Deviance Information Criterion (DIC, saturated) ....: -1485030.94
## Effective number of parameters .....................: 38.72
##
## Marginal log-Likelihood:  -40351.64
## CPO and PIT are computed
##
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

```
cat("NSLCPO of the model:",-sum(log(fit.inla.1$cpo$cpo)),"\n")
```

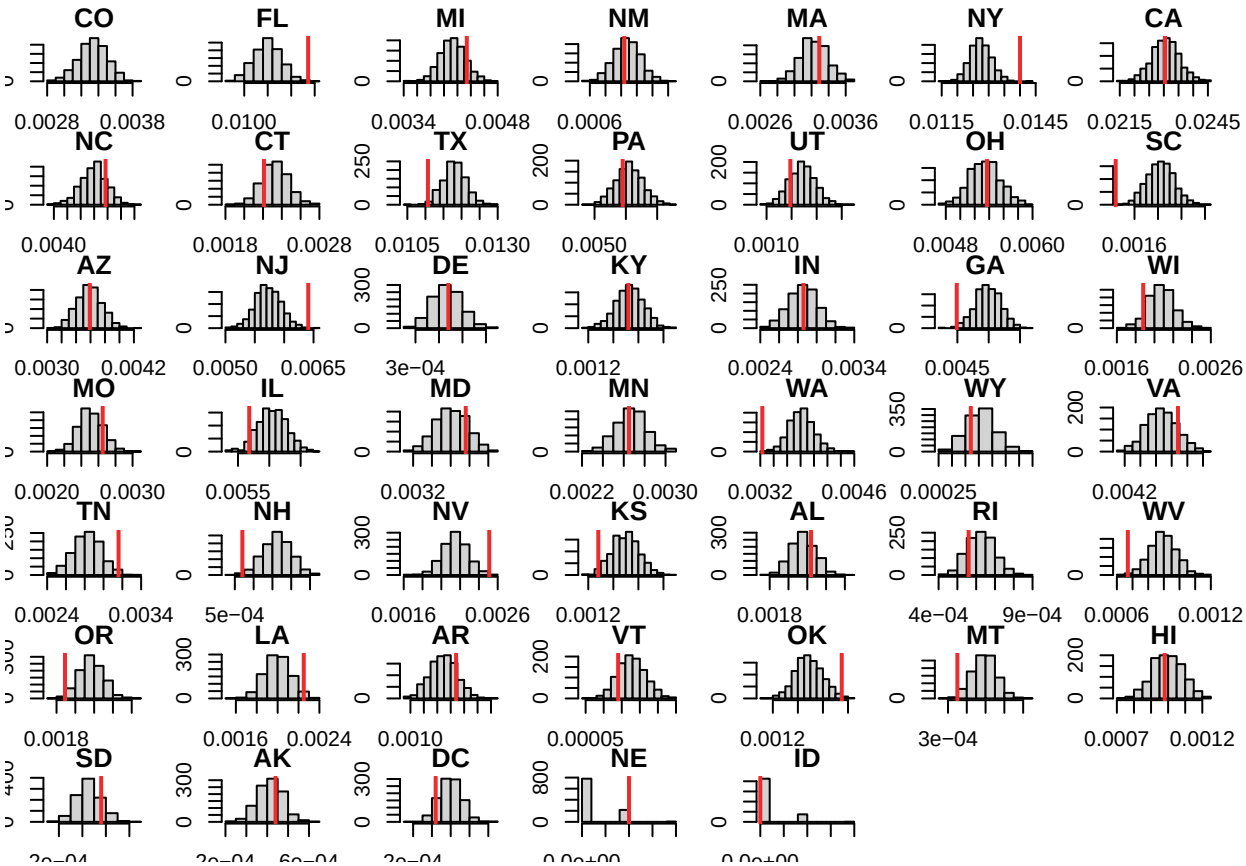
```
## NSLCPO of the model: 40267.95
```

```
cat("DIC of the model:", fit.inla.1$dic$dic,"\n")
```

```
## DIC of the model: 80535.79
```

```
#defining inverse logit function
inv.logit <- function(x) {
  return(1/(1+exp(-x)))
}

nbsamp <- 1000
m.samples <- inla.posterior.sample(n = nbsamp, result = fit.inla.1)
pred.samples <- inla.posterior.sample.eval(function(...) {Predictor}, m.samples)

yrep1 <- matrix(0, nrow = 100000, ncol = nbsamp)
#producing the replicates
for (i in 1:100000){
  yrep1[i,] <- rbinom(n = nbsamp, size = 1, prob = inv.logit(pred.samples[i,]))
}
```

```
par(mfrow=c(7,7), mai=c(0.2,0.2,0.2,0.2))
listofstates <- unique(lending$state)
for(i in 1:49){
  #matching index for each state
  idx <- which(lending$state[1:100000]==listofstates[i])
  if(length(idx)>1){
    #computing the proportion for frequency
    prop.rep <- apply(yrep1[idx,],2,sum) / 100000
    hist(prop.rep, main = paste(listofstates[i]))
    abline(v = sum(lending$y[idx])/100000, col = "firebrick2", lwd = 2)
  }
}
```


CO
FL
MI
NM
MA
NY
CA
NC
CT
TX
PA
UT
OH
SC
AZ
NJ
DE
KY
IN
GA
WI
MO
IL
MD
MN
WA
WY
VA
TN
NH
NV
KS
AL
RI
WV
OR
LA
AR
VT
OK
MT
HI
SD
AK
DC
NE
ID

c)[10 marks] (Independent model)

Using INLA, implement a Bayesian logistic GLM with y as a function of the scaled log interest rate, scaled log annual income, scaled log loan amount, scaled debt-to-income, and sub.grade, and an independent intercept for the state variable (see the Radon levels example on page 38 of Lecture 5).

Propose your own priors for the model parameters (you can take into account slides 57-58 of Lecture 4).

Print out the model summary.

Compute DIC and negative log-CPO (NLSCPO).

Do posterior predictive checks with the functions being evaluated chosen as the proportion of defaults among clients in each of the 49 states contained in the dataset. Discuss the model fit.

Explanation:

Similar to Q2 b), we perform the Bayesian analysis using `inla()` but this time we will have an independent model where an independent intercept for the state variable is introduced. Hence, we defined a new prior based on the slides 57-58 of Lecture 4. To do that, we considered the minimum distance of each continuous variable and set a prior that ensures that $|\beta_n$ can be larger than 10 over that minimum change distances.

$$-5 \approx \ln(0.005/(1-0.005)) = \beta_0 + \sum_{i=1}^{n} \beta_i(x_{il} - \bar{x})$$

$$5 \approx \ln(0.995/(1-0.995)) = \beta_0 + \sum_{i=1}^{n} \beta_i(x_{iu} - \bar{x})$$

Computing all of this, we computed the maximum variance as the prior is noninformative and choose the largest variance. The summary statistics is then shown below.

We computed the DIC and negative log-CPO and the values are 80449.23 and 40224.50 respectively. These values are slightly smaller than the ones we obtained in Q2 b). Hence, independent model is a better model than the identical model.

We conducted the same posterior probability check. We can observe that all the states have the true frequency in the range of the replicates. Unlike in Q2 b) we have the states `Colorado, CO`, `New York, NY`, `New Jersey, NJ` and `South Carolina, SC` have their true frequency lying within the range of the distribution. This is an improvement compared to the identical model. Thus, we can say that the independent model is a well represented model using `inla()` and a better fit than identical model.

```
df <- lending[1:100000,]
#computing the variance for each continuous variable
ir.prec <- (10/(max(df$log.interest.rate)-min(df$log.interest.rate)))^2
ai.prec <- (10/(max(df$log.annual.income)-min(df$log.annual.income)))^2
la.prec <- (10/(max(df$log.loan.amount)-min(df$log.loan.amount)))^2
dti.prec <- (10/(max(df$debt.to.income)-min(df$debt.to.income)))^2
df.prec <- max(ir.prec, ai.prec, la.prec, dti.prec)

prior.independent <- list(mean.intercept = 0, prec.intercept = 0.01,
                          mean = 0, prec = 1/df.prec)

# INLA logistic regression model
prec.prior.random.eff <- list(prec = list(initial = log(0.01), fixed = TRUE))
```

```
fit.inla.independent <- inla(y ~ 0 + log.interest.rate + log.annual.income +
                               log.loan.amount + debt.to.income + sub.grade +
                               term + f(state, model = "iid"),
                             data = lending[1:100000,], family = "binomial",
                             Ntrials = 1, control.compute = list(config = TRUE,
                                                                 dic = TRUE,
                                                                 cpo = TRUE),
                             control.fixed = prior.independent)
summary(fit.inla.independent)
```

```
##
## Call:
##    c("inla(formula = y ~ 0 + log.interest.rate + log.annual.income + ", "
##    log.loan.amount + debt.to.income + sub.grade + term + f(state, ", "
##    model = \"iid\"), family = \"binomial\", data = lending[1:1e+05, ", "
##    ], Ntrials = 1, control.compute = list(config = TRUE, dic = TRUE, ", "
##    cpo = TRUE), control.fixed = prior.independent)")
## Time used:
##     Pre = 1.27, Running = 65.7, Post = 0.99, Total = 67.9
## Fixed effects:
##                       mean    sd 0.025quant 0.5quant 0.975quant   mode kld
## log.interest.rate  0.135 0.112     -0.080    0.133      0.360  0.129   0
## log.annual.income -0.304 0.012     -0.327   -0.304     -0.281 -0.304   0
## log.loan.amount    0.186 0.012      0.162    0.186      0.210  0.186   0
## debt.to.income     0.089 0.009      0.071    0.089      0.108  0.089   0
## sub.gradeA1       -3.402 0.319     -4.022   -3.404     -2.772 -3.408   0
## sub.gradeA2       -3.046 0.270     -3.569   -3.049     -2.511 -3.053   0
## sub.gradeA3       -2.965 0.223     -3.397   -2.967     -2.522 -2.970   0
## sub.gradeA4       -2.601 0.197     -2.981   -2.603     -2.210 -2.608   0
## sub.gradeA5       -2.467 0.155     -2.766   -2.468     -2.157 -2.472   0
## sub.gradeB1       -2.431 0.120     -2.661   -2.432     -2.192 -2.435   0
## sub.gradeB2       -2.288 0.087     -2.457   -2.289     -2.115 -2.290   0
## sub.gradeB3       -2.237 0.063     -2.359   -2.237     -2.114 -2.238   0
## sub.gradeB4       -2.108 0.046     -2.198   -2.107     -2.018 -2.107   0
## sub.gradeB5       -2.009 0.049     -2.107   -2.009     -1.913 -2.008   0
##  [ reached getOption("max.print") -- omitted 26 rows ]
##
## Random effects:
##   Name     Model
##     state IID model
##
## Model hyperparameters:
##                       mean    sd 0.025quant 0.5quant 0.975quant  mode
## Precision for state 96.07 34.99      46.37    89.89     181.89 79.26
##
## Deviance Information Criterion (DIC) ...............: 80449.23
## Deviance Information Criterion (DIC, saturated) ....: -1485125.34
## Effective number of parameters .....................: 67.52
##
## Marginal log-Likelihood:  -40360.79
## CPO and PIT are computed
##
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

```r
cat("NSLCPO of the model:",-sum(log(fit.inla.independent$cpo$cpo)),"\n")
```

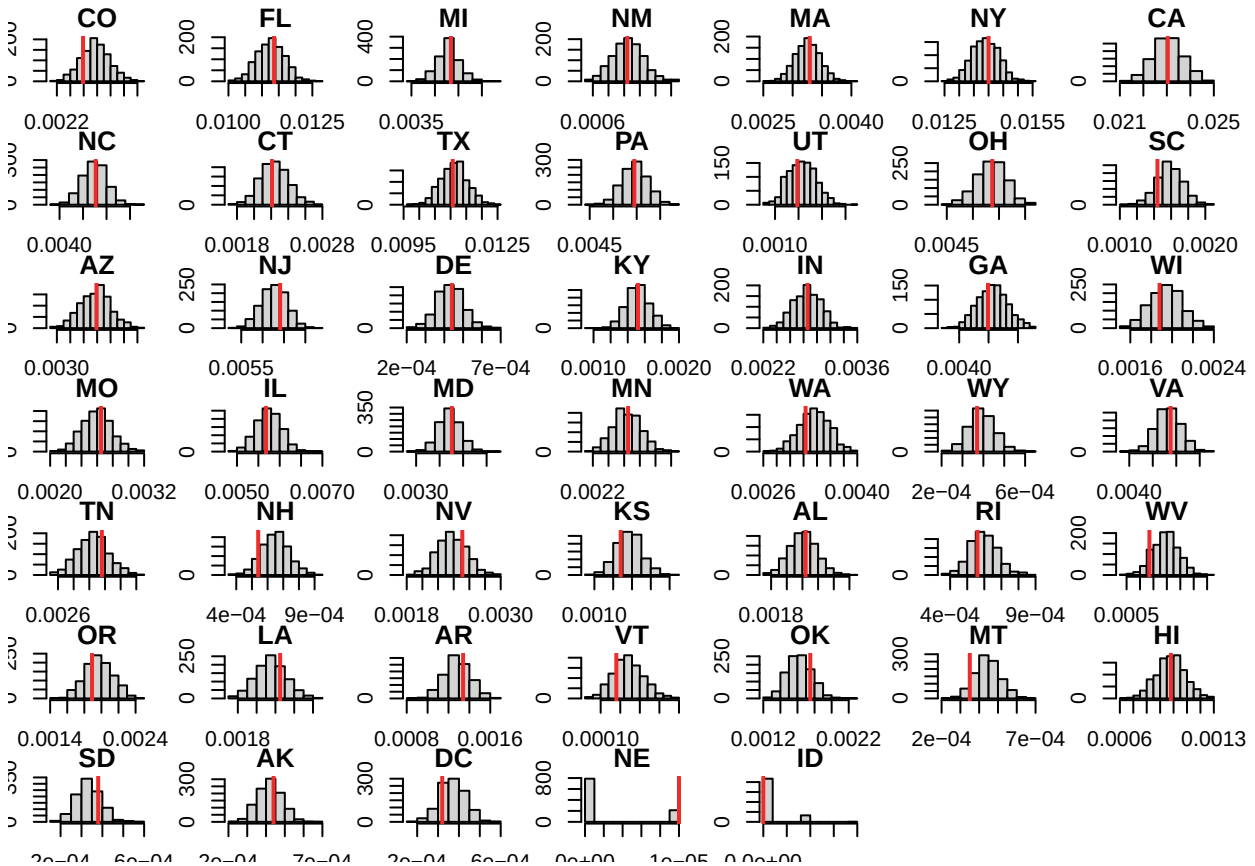
```
## NSLCPO of the model: 40224.5
```

```r
cat("DIC of the model:", fit.inla.independent$dic$dic,"\n")
```

```
## DIC of the model: 80449.23
```

```r
#defining inverse logit function
inv.logit <- function(x) {
  return(1/(1+exp(-x)))
}

nbsamp <- 1000
m.samples <- inla.posterior.sample(n = nbsamp, result = fit.inla.independent)
pred.samples <- inla.posterior.sample.eval(function(...) {Predictor}, m.samples)

yrep1 <- matrix(0, nrow = 100000, ncol = nbsamp)
#producing the replicates
for (i in 1:100000){
  yrep1[i,] <- rbinom(n = nbsamp, size = 1, prob = inv.logit(pred.samples[i,]))
}
```

```r
par(mfrow=c(7,7), mai=c(0.2,0.2,0.2,0.2))
listofstates <- unique(lending$state)
for(i in 1:49){
  #matching the index for each state
  idx <- which(lending$state[1:100000]==listofstates[i])
  if(length(idx)>1){
    #computing the proporiton for frequency
    prop.rep <- apply(yrep1[idx,],2,sum) / 100000
    hist(prop.rep, main = paste(listofstates[i]))
    abline(v = sum(lending$y[idx])/100000, col = "firebrick2", lwd = 2)
  }
 }
```


CO
FL
MI
NM
MA
NY
CA
NC
CT
TX
PA
UT
OH
SC
AZ
NJ
DE
KY
IN
GA
WI
MO
IL
MD
MN
WA
WY
VA
TN
NH
NV
KS
AL
RI
WV
OR
LA
AR
VT
OK
MT
HI
SD
AK
DC
NE
ID

d)[10 marks] (Hierarchical model)

Using INLA, implement a Bayesian logistic GLM with y as a function of the scaled log interest rate, scaled log annual income, scaled log loan amount, scaled debt-to-income, and sub.grade, and an hierarchical model for the state variable (each state has a different intercept but the intercepts are random draws from the same distribution, see the Radon levels example on page 38 of Lecture 5).

Propose your own priors for the model parameters.

Print out the model summary.

Compute DIC and negative log-CPO (NLSCPO).

Do posterior predictive checks with the functions being evaluated chosen as the proportion of defaults among clients in each of the 49 states contained in the dataset. Discuss the model fit.

Explanation:

Similar to Q2 b) and c), we perform the Bayesian analysis using `inla()` but this time we will have an hierarchical model. Hence, we defined a the same prior used in Q2 c) based on the slides 57-58 of Lecture 4. To do that, we considered the minimum distance of each column and set a prior that ensures that $|\beta_n$ can be larger than 10 over that minimum change distances.

$$-5 \approx \ln(0.005/(1-0.005)) = \beta_0 + \sum_{i=1}^{n} \beta_i(x_{il} - \bar{x})$$

$$5 \approx \ln(0.995/(1-0.995)) = \beta_0 + \sum_{i=1}^{n} \beta_i(x_{iu} - \bar{x})$$

Computing all of this, we computed the maximum variance as the prior is noninformative and choose the largest variance. In addition, we defined the random effect of the prior distribution for the hyperparameter of `state`. To do this we referred to slide 62, Lecture 5. We set the log-transformed precision parameter to follow a uniform distribution which $\sigma \sim U(0, b)$ where $b = 20$. The summary statistics is then shown below.

We computed the DIC and negative log-CPO of the hierarchical model and the values are 80451.53 and 40225.78 respectively. These values are slightly higher than the ones we obtained in Q2 b) and c).

We conducted the same posterior probability check. We can observe that all the states have the true frequency in the range of the replicates. Unlike in Q2 b) we have the states `Colorado, CO`, `New York, NY`, `New Jersey, NJ` and `South Carolina, SC` have their true frequency lying within the range of the distribution. This is an improvement compared to the identical model. However, we can observe that the independent model has lower DIC and NLSCPO values which indicating a better fit than hierarchical model. Thus we can say that the hierarchical model is a well represented model using `inla()` but not the best model.

```
df <- lending[1:100000,]
#computing the variance for each continuous variable
ir.prec <- (10/(max(df$log.interest.rate)-min(df$log.interest.rate)))^2
ai.prec <- (10/(max(df$log.annual.income)-min(df$log.annual.income)))^2
la.prec <- (10/(max(df$log.loan.amount)-min(df$log.loan.amount)))^2
dti.prec <- (10/(max(df$debt.to.income)-min(df$debt.to.income)))^2
df.prec <- max(ir.prec, ai.prec, la.prec, dti.prec)

prior.hierarchical <- list(mean.intercept = 0, prec.intercept = 0.01,
                           mean = 0, prec = 1/df.prec)

# INLA logistic regression model
#defining the random effect of state
```

```
sigma.unif.prior.random.eff <- "expression:
  b = 20;
  log_dens = (theta>=(-2*log(b)))*(-log(b)-theta/2-log(2)) + (theta<(-2*log(b)))*(-Inf);
  return(log_dens);"

b <- 20
prec.prior.random.eff <- list(prec = list(prior = sigma.unif.prior.random.eff,
                                          initial = (-2*log(b)+1), fixed = FALSE))

fit.inla.hierarchical <- inla(y ~ 1 + log.interest.rate + log.annual.income +
                                log.loan.amount + debt.to.income + sub.grade +
                                term + f(state, model = "iid",
                                  hyper = prec.prior.random.eff),
                              data = lending[1:100000,], family = "binomial",
                              Ntrials = 1, control.compute = list(config = TRUE,
                                                                  dic = TRUE,
                                                                  cpo = TRUE),
                              control.fixed = prior.hierarchical)
summary(fit.inla.hierarchical)
```

```
##
## Call:
##    c("inla(formula = y ~ 1 + log.interest.rate + log.annual.income + ", "
##    log.loan.amount + debt.to.income + sub.grade + term + f(state, ", "
##    model = \"iid\", hyper = prec.prior.random.eff), family = \"binomial\",
##    ", " data = lending[1:1e+05, ], Ntrials = 1, control.compute =
##    list(config = TRUE, ", " dic = TRUE, cpo = TRUE), control.fixed =
##    prior.hierarchical)" )
## Time used:
##     Pre = 1.39, Running = 58.6, Post = 0.848, Total = 60.8
## Fixed effects:
##                       mean    sd 0.025quant 0.5quant 0.975quant   mode kld
## (Intercept)       -2.591 0.236     -3.054   -2.591     -2.127 -2.592   0
## log.interest.rate  0.431 0.092      0.251    0.430      0.614  0.429   0
## log.annual.income -0.304 0.012     -0.327   -0.304     -0.282 -0.304   0
## log.loan.amount    0.186 0.012      0.162    0.186      0.210  0.186   0
## debt.to.income     0.089 0.009      0.071    0.089      0.108  0.089   0
## sub.gradeA2        0.155 0.193     -0.222    0.154      0.536  0.152   0
## sub.gradeA3        0.123 0.183     -0.234    0.122      0.485  0.119   0
## sub.gradeA4        0.445 0.171      0.114    0.444      0.786  0.440   0
## sub.gradeA5        0.481 0.174      0.144    0.480      0.826  0.477   0
## sub.gradeB1        0.434 0.181      0.082    0.433      0.792  0.431   0
## sub.gradeB2        0.487 0.196      0.104    0.487      0.873  0.485   0
## sub.gradeB3        0.463 0.211      0.048    0.462      0.877  0.462   0
## sub.gradeB4        0.527 0.226      0.084    0.527      0.969  0.527   0
## sub.gradeB5        0.574 0.240      0.103    0.574      1.043  0.575   0
##  [ reached getOption("max.print") -- omitted 26 rows ]
##
## Random effects:
##   Name     Model
##     state IID model
##
## Model hyperparameters:
##                       mean    sd 0.025quant 0.5quant 0.975quant  mode
```

```
## Precision for state 78.92 27.22      38.84    74.39     145.41 66.17
##
## Deviance Information Criterion (DIC) ...............: 80451.53
## Deviance Information Criterion (DIC, saturated) ....: -1485123.03
## Effective number of parameters .....................: 68.45
##
## Marginal log-Likelihood:  -40354.45
## CPO and PIT are computed
##
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

```
cat("NSLCPO of the model:",-sum(log(fit.inla.hierarchical$cpo$cpo)),"\n")
```

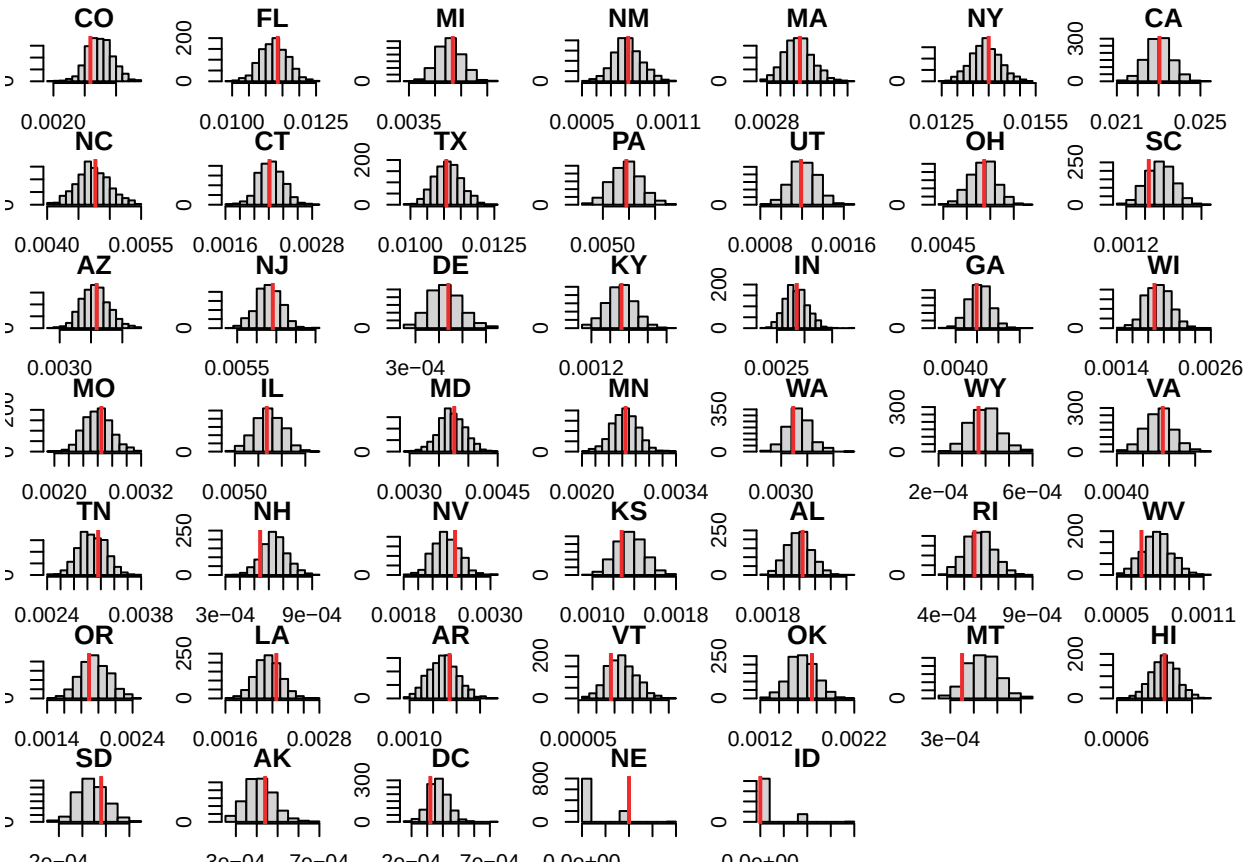
```
## NSLCPO of the model: 40225.78
```

```
cat("DIC of the model:", fit.inla.hierarchical$dic$dic,"\n")
```

```
## DIC of the model: 80451.53
```

```
#defining inverse logit function
inv.logit <- function(x) {
  return(1/(1+exp(-x)))
}

nbsamp <- 1000

m.samples <- inla.posterior.sample(n = nbsamp, result = fit.inla.hierarchical)
pred.samples <- inla.posterior.sample.eval(function(...) {Predictor}, m.samples)

yrep1 <- matrix(0, nrow = 100000, ncol = nbsamp)
#producing the replicates
for (i in 1:100000){
  yrep1[i,] <- rbinom(n = nbsamp, size = 1, prob = inv.logit(pred.samples[i,]))
}
```

```
par(mfrow=c(7,7), mai=c(0.2,0.2,0.2,0.2))
listofstates <- unique(lending$state)
for(i in 1:49){
  #matching index for each state
  idx <- which(lending$state[1:100000]==listofstates[i])
  if(length(idx)>1){
    #computing the proportion for frequency
    prop.rep <- apply(yrep1[idx,],2,sum) / 100000
    hist(prop.rep, main = paste(listofstates[i]))
    abline(v = sum(lending$y[idx])/100000, col = "firebrick2", lwd = 2)
  }
 }
```


CO
FL
MI
NM
MA
NY
CA
NC
CT
TX
PA
UT
OH
SC
AZ
NJ
DE
KY
IN
GA
WI
MO
IL
MD
MN
WA
WY
VA
TN
NH
NV
KS
AL
RI
WV
OR
LA
AR
VT
OK
MT
HI
SD
AK
DC
NE
ID

e)[10 marks] Prediction of probability of default based on salary

Consider a set of clients with loan amount of 20000$, interest rate of 15%, sub grade of B1, debt to income of 10%, state of "CA" (California), term of 60 months, and annual incomes of 40000, 50000, 60000, ... , 300000$.

Plot the posterior means of the probabilities of default on the loans in function of the annual incomes based on the hierarchical model from part d).

Explanation:

We predicted 27 values in total and computed the posterior means of the probabilities depending on the amount of annual income using hierarchical model. If we look at the plot below, we can see the the decreasing trend from $40000. The gradient is sharp from $40000 to $120000 and we can observe a change in magnitude and sign of the gradients where it becomes more gentle. Therefore we can then conclude that the probability of default on the loans in function of the annual incomes is low with higher annual incomes.

```
#creating a new dataframe that contains the attributed for prediction
new.df <- lending[1:100000, c(3,5,7,8,9,14,15,16)]
new.loan.amount <- data.frame(rep((log(20000)-mean(lending$log.loan.amount.noscale)) /
                      sd(lending$log.loan.amount.noscale),27))
new.interest.rate <- data.frame(rep((log(0.15)-
                                     mean(lending$log.interest.rate.noscale)) /
                                  sd(lending$log.interest.rate.noscale),27))
new.state <-  data.frame(rep("CA",27))
new.sub.grade <- data.frame(rep("B1", 27))
new.debt.to.income <- data.frame(rep((0.1-mean(lending$debt.to.income.noscale)) /
                         sd(lending$debt.to.income.noscale),27))
new.term <- data.frame(rep(" 60 months", 27))
new.y <- data.frame(rep(NA,27))
new.annual.income <- (log(seq(40000, 300000, by=10000)) - mean(lending$log.annual.income.noscale)) /
  sd(lending$log.annual.income.noscale)
add.df <- cbind(new.sub.grade, new.debt.to.income, new.state, new.term,
                new.y, new.loan.amount, new.interest.rate, new.annual.income)
colnames(add.df) <- colnames(new.df)
#combing the original data with the new data
new.df <- rbind(new.df, add.df)
```

```
#fitting new hierarchical model with the new data
fit.inla.hierarchical <- inla(y ~ 1 + log.interest.rate + log.annual.income +
                                log.loan.amount + debt.to.income + sub.grade +
                                term + f(state, model = "iid",
                                  hyper = prec.prior.random.eff),
                              data = new.df, family = "binomial",
                              Ntrials = 1, control.compute = list(config = TRUE,
                                                                  dic = TRUE,
                                                                  cpo = TRUE),
                              control.fixed = prior.hierarchical)
```

```
nbsamp <- 1000
fit.predict.sample <- inla.posterior.sample(nbsamp, fit.inla.hierarchical,
                                            selection=
                                              list(Predictor = 100001:100027))

predictor.samples <- inla.posterior.sample.eval(function(...) {Predictor},
                                                fit.predict.sample)
```

```r
post.pred.samples <- matrix(0, nrow = 27, ncol = nbsamp)
#predicting for the 27 different values
for(it in 1:27){
  post.pred.samples[it,] <- rbinom(n = nbsamp, size = 1,
                                   prob = inv.logit(predictor.samples[it]))
}
```

```r
plot(seq(40000, 300000, by=10000), apply(post.pred.samples, 1, mean),
     type = "l", xlab = "Annual Income", ylab = "Probabilities",
     main = "Posterior Means of the Probabilities", col = "firebrick2")
```