University of Edinburgh

School of Mathematics

Bayesian Data Analysis, 2022/2023, Semester 2

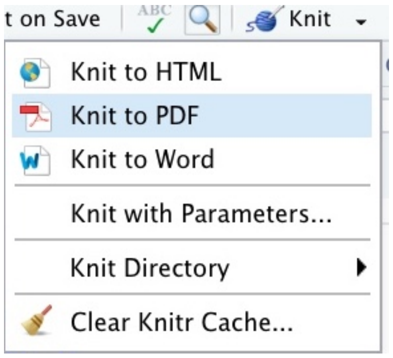
Assignment 1

IMPORTANT INFORMATION ABOUT THE ASSIGNMENT

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

1) Deadline: The deadline for submitting your solutions to this assignment is the 6 March 12:00 noon Edinburgh time.

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



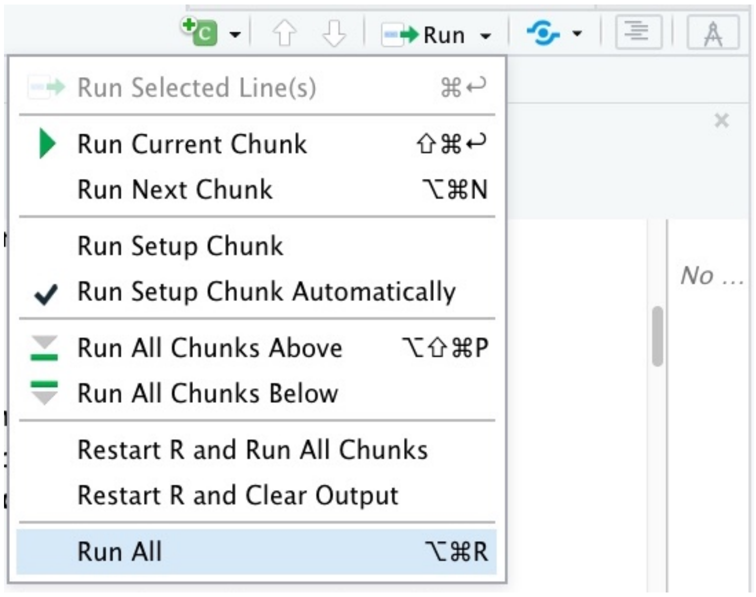
The compiled PDF needs to contain everything in this notebook, with your code sections clearly visible (not hidden), and the output of your code included. Reports without the code displayed in the PDF, or without the output of your code included in the PDF will be marked as 0, with the only feedback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

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

Some key points that are different from other courses:

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

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



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

We illustrate these rules by some examples:

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

Bob has spent a lot of time and effort on his assignment for BDA. Unfortunately he forgot to submit his code. - Bob will get no personal reminder to submit his code. Bob will get 0 for the whole assignment, with the only feedback "Results are not reproducible from the code, as the code was not submitted."

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

Denise has spent a lot of time and effort on her assignment for BDA. She has submitted her report in the correct format, but thought that she can include her code as a link in the report, and upload it online (such as Github, or Dropbox). - Denise will get 0 for the whole assignment, with the only feedback "Code was not uploaded on Learn."

3) Group work: This is an INDIVIDUAL ASSIGNMENT, like a 2 week exam for the course. Communication between students about the assignment questions is not permitted. Students who submit work that has not been done individually will be reported for Academic Misconduct, that can lead to serious consequences. Each problem will be marked by a single instructor, so we will be able to spot students who copy.

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

Only questions regarding clarification of the statement of the problems will be answered by the instructors. The instructors will not give you any information related to the solution of the problems, such questions will be simply answered as "This is not about the statement of the problem so we cannot answer your question."

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

5) Office hours: There will be two office hours per week (Monday 14:00-15:00, and Wednesdays 15:00-16:00) during the 2 weeks for this assignment. The links are available on Learn / Course Information. I will be happy to discuss the course/workshop materials. However, I will only answer questions about the assignment that require clarifying the statement of the problems, and will not give you any information about the solutions. Students who ask for feedback on their assignment solutions during office hours will be removed from the meeting.

6) Late submissions and extensions: NO EXTENSIONS ARE ALLOWED FOR THIS ASSIGNMENT, AND THERE IS NO SUCH OPTION PROVIDED IN THE ESC SYSTEM. Students who have existing Learning Adjustments in Euclid will be allowed to have the same adjustments applied to this course as well, but they need to apply for this BEFORE THE DEADLINE on the website

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

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

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

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

Problem 1

In this problem, we study a dataset about currency exchange rates. The exrates dataset of the stochvol package contains the daily average exchange rates of 24 currencies versus the EUR, from 2000-01-03 until 2012-04-04.

```
require(stochvol)
```

```
## Loading required package: stochvol
```

```
data("exrates")
#You may need to set the working directory first before loading the dataset
#setwd("location of Assignment 1")
#The first 6 rows of the dataframe
print.data.frame(exrates[1:6,])
```

```
##                AUD    CAD    CHF    CZK    DKK    GBP    HKD     IDR    JPY
## 2000/01/03 1.5346 1.4577 1.6043 36.063 7.4404 0.6246 7.8624 7133.32 102.75
## 2000/01/04 1.5677 1.4936 1.6053 36.270 7.4429 0.6296 8.0201 7265.16 105.88
## 2000/01/05 1.5773 1.5065 1.6060 36.337 7.4444 0.6324 8.0629 7437.97 107.34
## 2000/01/06 1.5828 1.5091 1.6068 36.243 7.4441 0.6302 8.0843 7495.52 108.72
## 2000/01/07 1.5738 1.5010 1.6079 36.027 7.4436 0.6262 8.0030 7398.88 108.09
## 2000/01/10 1.5587 1.4873 1.6089 35.988 7.4448 0.6249 7.9471 7320.49 107.26
##                KRW    MXN    MYR    NOK    NZD    PHP    PLN    RON     RUB
## 2000/01/03 1140.02 9.6105 3.8422 8.0620 1.9331 40.424 4.1835 1.8273 27.7548
## 2000/01/04 1157.32 9.7453 3.9188 8.1500 1.9745 40.992 4.2423 1.8858 28.3594
## 2000/01/05 1176.08 9.8969 3.9393 8.2060 1.9956 41.637 4.2627 1.8979 28.3006
## 2000/01/06 1191.90 9.9751 3.9498 8.2030 2.0064 42.148 4.2593 1.9000 28.5111
## 2000/01/07 1169.52 9.8368 3.9100 8.1945 1.9942 41.493 4.1897 1.8822 28.2526
## 2000/01/10 1157.84 9.6449 3.8824 8.1900 1.9783 41.226 4.1567 1.8729 28.7609
##               SEK    SGD     THB      TRY    USD       date
## 2000/01/03 8.5520 1.6769 37.2793 0.546131 1.0090 2000-01-03
## 2000/01/04 8.6215 1.7047 38.2078 0.552354 1.0305 2000-01-04
## 2000/01/05 8.6415 1.7159 38.5375 0.555329 1.0368 2000-01-05
## 2000/01/06 8.6445 1.7291 39.0734 0.555674 1.0388 2000-01-06
## 2000/01/07 8.6450 1.7096 38.4299 0.554980 1.0284 2000-01-07
## 2000/01/10 8.6570 1.6975 38.0328 0.552469 1.0229 2000-01-10
```

```
cat(paste("Data from ", min(exrates$date)," until ",max(exrates$date)))
```

```
## Data from  2000-01-03  until  2012-04-04
```

As we can see, not all dates are included in the dataset. Some are missing, such as weekends, and public holidays.

In this problem, we are going to fit a various stochastic volatility models on this dataset (see e.g. https://www.jstor.org/stable/1392251).

a)[10 marks] Consider the following leveraged Stochastic Volatility (SV) model.

$$y_t = \beta_0 + \beta_1 y_{t-1} + \exp(h_t/2)\epsilon_t \quad \text{for} \quad 1 \le t \le T,$$
$$h_{t+1} = \mu + \phi(h_t - \mu) + \sigma\eta_t \quad \text{for} \quad 0 \le t \le T, \quad h_0 \sim N(\mu, \sigma^2/(1-\phi^2)),$$
$$(\epsilon_t, \eta_t) \sim N(0, \Sigma_\rho) \quad \text{for} \quad \Sigma_\rho = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}.$$

Here $t$ is the time index, y_t are the observations (such as daily USD/EUR rate), h_t are the log-variance process, ϵ_t is the observation noise, and η_t is the log-variance process noise (which are correlated, but independent for different values of \$t\$). The hyperparameters are $\beta_0, \beta_1, \mu, \phi, \sigma, \rho$.

For stability, it is necessary to have $\phi \in (-1, 1)$, and by the definition of correlation matrices, we have $\rho \in [-1, 1]$.

Implement this model in JAGS or Stan on the first 3 months of USD/EUR data from the dataset, i.e. from dates 2000-01-03 until 2000-04-02.

Explain how did you choose priors for all parameters. Explain how did you take into account the days without observation in your model.

Fit the model, do convergence diagnostics, print out the summary of the results, and discuss them.

Make sure that the Effective Sample Size is at least 1000 for all 6 hyperparameters (you need to choose burn-in and number of steps appropriately for this).

```
library(rjags)
```

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

```
## Linked to JAGS 4.3.1
```

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

```r
sdate <- as.Date("2000-01-03")
edate <- as.Date("2000-04-02")

# Generating a sequence of dates
dates<- seq(sdate, as.Date("2012-04-04"), by = "day")

## Preparing a date frame for dates
seq_dates <- data.frame(date = dates)

# Merge the data frames to get all the dates with NAs for missing values
exrates_new <- merge(seq_dates, exrates, by = "date", all.x = TRUE)

## Getting data for USD
exratesUSD <- exrates_new$USD[exrates_new$date >= sdate & exrates_new$date <= edate]


## Getting the mean
mean.USD <- mean(exratesUSD[which(!is.na(exratesUSD))])
```

Explanation: Since, the dataset doesn't contain the dates on which data is missing. To fix this, we have generated a sequence of dates for the analysis and introduced the missingness by merging the sequence into original data set. After doing this, we have extracted the values of USD with NA's on every Saturday and Sunday. JAGS is able to handle these NA values automatically by creating stochastic nodes for them.

```r
model_string01 <- "
  ## Declaring the size of the variables before the model using the var command
  var epsilon_eta[n,2], Sigma_rho[2,2], beta[2];

  model{
  # Priors for hyperparameters
  mu ~ dnorm(0, 1.0E-1)
  phi ~ dunif(-1, 1)
  rho ~ dunif(-1, 1)
  tau ~ dgamma(1, 1)


  for (i in 1:2){
    beta[i] ~ dnorm(0, 1.0E-2)
  }
  # Priors for h[1] and y[1]
  h[1] ~ dnorm(mu, sqrt(1-phi*phi)/sigma)
  y[1] ~ dnorm(mu.y0,0.01)

  # Covariance matrix for Sigma_rho
  Sigma_rho[1, 1] <- 1
  Sigma_rho[2, 2] <- 1
  Sigma_rho[1, 2] <- rho
  Sigma_rho[2, 1] <- rho

  ## for epsilon and eta
  for (t in 1:n) {
    epsilon_eta[t,1:2] ~ dmnorm(c(0,0), inverse(Sigma_rho))
    eps[t] <- epsilon_eta[t,1]
    eta[t] <- epsilon_eta[t,2]
  }

  ## Likelihood
  for (t in 2:n) {
    mu.y[t] <- beta[1] + beta[2] * (y[t-1]-mean.USD)
    #observations
    y[t] ~ dnorm(mu.y[t], exp(h[t]/2) * sqrt(1-rho^2))
    # Replicates
    y.rep[t] ~ dnorm(mu.y[t], exp(h[t]/2) * sqrt(1-rho^2))


  }

  for(t in 1:n){
    h[t+1] ~ dnorm(mu + phi * (h[t] - mu)+ sigma * eta[t], tau.h)
  }

  tau.h ~ dgamma(2,1)

  # Precision for likelihood
  sigma <- 1 / sqrt(tau)
}"

## Intializing the parameters
n <- length(exratesUSD)
y <- exratesUSD
mean.USD <- mean.USD
mu.y0 <- exratesUSD[1]

## compiling model the model using jags
model01 <- jags.model(textConnection(model_string01), data = list(y=y, n=n,mu.y0 = mu.y0, mean.USD=mean
```

```
## Compiling model graph
##    Declaring variables
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 65
##    Unobserved stochastic nodes: 306
##    Total graph size: 1469
##
## Initializing model
```

```
## Burnin for 10000 samples
update(model01,10000,progress.bar="none")

## Collecting sample from the model
samples01 <- coda.samples(model01, variable.names = c("beta[1]", "beta[2]", "mu", "phi", "sigma", "rho")
```

Explanation for Choice of priors:

μ ~ dnorm(0, 1.0E-1): This gives the parameter μ a normal prior distribution with mean 0 and a low precision . This prior suggests that the model has limited knowledge of the actual value of μ, which might be anywhere within a range close to zero.

ϕ and ρ ~ dunif(-1, 1): This gives the parameter ϕ and ρ uniform prior distribution between -1 and 1. These are specified in question itself for the stability of the model.

For beta[i] ~ dnorm(0, 1.0E-2): This assigns a normal prior distribution with mean 0 and low precision for beta[1] and beta[2]. This prior indicates that the model has little information about the true values of beta.

In this model, the priors for ϕ , ρ, and beta are also non-informative because they are uniform or normal distributions with small variances. These priors do not express any strong prior belief about the values of these parameters.

The priors for μ and τ are somewhat informative but not strongly so. The normal prior for mu with mean 0 and variance 0.01 indicates that the model has a weak prior belief that μ is likely to be near zero, but this prior is not strongly informative. The gamma prior for τ with shape parameter 1 and rate parameter 1 is a weakly informative prior.

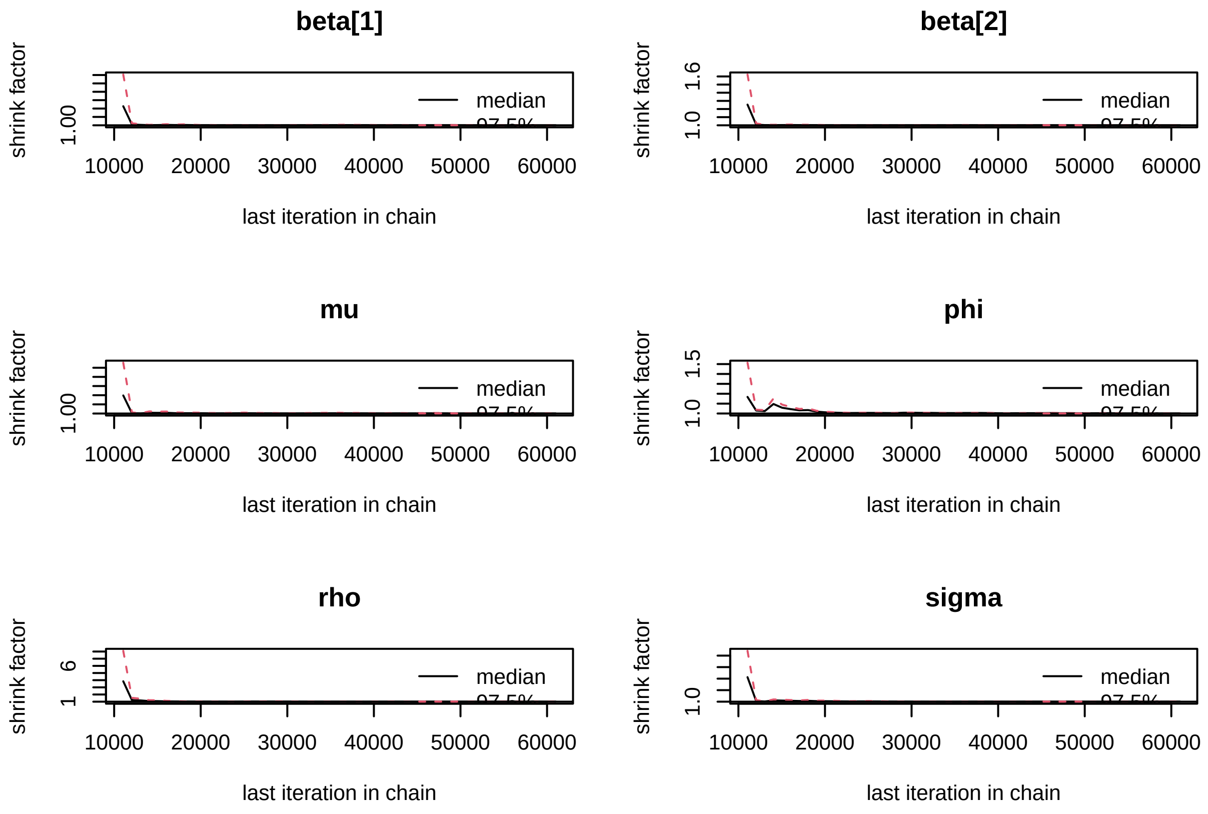
The prior for h[1] is informative because it depends on the values of μ, ϕ, and σ .This prior expresses a prior belief that h[1] is likely to be close to μ but allows for some variability depending on the values of ϕ and σ.

The prior for y[1] is also somewhat informative because it has a small variance of 0.01. This prior expresses a weak prior belief that y[1] is likely to be close to mu.y0 i.e. first value of USD on starting date but allows for some variability. It is used to initialize the first node.

These can help to prevent over fitting and improve the generalization of the model to new data. Hence, we selected these priors.

We are also centering the data around the mean because centering data can improve the convergence. This is because centering the data can make the objective function smoother and more well-behaved, which can make optimization more efficient. We ran 5 chains with 50000 samples after 10000 burn-in iterations.

```
gelman.plot(samples01)
```



Explaination: These plots show the Gelman-Rubin statistics computed from increasing number of samples. The red dotted line shows 97.5% confidence interval for these statistics, indicating that a burnin of 10000 would be sufficient.

```
gelman.diag(samples01)
```

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

Explaination: The Gelman-Rubin diagnostic can be used to evaluate the convergence of MCMC algorithms and guarantee the validity of the output of Bayesian inference. Before making decisions or drawing inferences from the data, as we will be doing in the upcoming questions, it is crucial to ensure that the MCMC algorithm have reached convergence.

As, we can see from the gelman.diag or gelman.plot results that Gelman-Rubin diagnostics values are below 1.05 for each of the parameters, indicating that we have approximated the stationary distribution quite well.

```r
summary(samples01)
```

```
##
## Iterations = 11001:61000
## Thinning interval = 1
## Number of chains = 5
## 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
## beta[1] 0.98560 0.0006458 1.292e-06      4.967e-06
## beta[2] 0.94987 0.0280110 5.602e-05      1.909e-04
## mu      4.26892 3.7265415 7.453e-03      1.810e-02
## phi     0.99799 0.0046984 9.397e-06      4.031e-05
## rho     0.03467 0.5067805 1.014e-03      1.311e-02
## sigma   0.92031 0.2798680 5.597e-04      4.717e-03
##
## 2. Quantiles for each variable:
##
##             2.5%     25%     50%    75%   97.5%
## beta[1]  0.9843  0.9852 0.98561 0.9860  0.9869
## beta[2]  0.8940  0.9315 0.95011 0.9686  1.0043
## mu      -2.9176  1.7217 4.23129 6.7949 11.6271
## phi      0.9850  0.9984 0.99962 0.9999  1.0000
## rho     -0.8649 -0.3761 0.04324 0.4528  0.8923
## sigma    0.5079  0.7186 0.87683 1.0742  1.5830
```

Explanation: The summary (samples01) command is used to print the posterior summaries for the model parameters beta[1], beta[2], mu, phi, rho and sigma. The standard deviations obtained are much smaller in magnitude than the means except for ρ, suggesting that the amount of data available was enough to obtain reasonably confident estimates of the model parameters. Moreover, the time series SE was considerably smaller than the posterior means (in absolute value), which implies that our estimates are quite accurate and the chain are mixing reasonably well.

```r
## Effective sample size
effectiveSize(samples01)
```

```
##   beta[1]   beta[2]        mu       phi       rho     sigma
## 17016.511 21615.209 42430.198 14105.562  1581.541  3555.508
```

Explanation : We have computed the ESS(Effective sample size) for the model parameters beta[1], beta[2], sigma, phi, rho. By using the effective Size function. The ESS is above 1000 for each of them, so we don't require more samples.

b)[10 marks] In practice, one often encounters outliers in exchange rates. These can be sometimes modeled by assuming Student's t distribution in the observation errors (i.e. ϵ_t). The robust leveraged SV model can be expressed as

$$
\begin{aligned}
y_t &= \beta_0 + \beta_1 y_{t-1} + \exp(h_t/2)\epsilon_t \quad \text{for} \quad 1 \le t \le T, \\
h_{t+1} &= \mu + \phi(h_t - \mu) + \sigma\eta_t \quad \text{for} \quad 0 \le t \le T, \quad h_0 \sim N(\mu, \sigma^2/(1-\phi^2)), \\
\eta_t &\sim N(0,1) \\
\epsilon_t|\eta_t &\sim t_\nu(\rho\eta_t, 1).
\end{aligned}
$$

Here ν is the degrees of freedom parameter (unknown).

Implement this model in JAGS or Stan on the first 3 months of USD/EUR data from the dataset.

Explain how did you choose priors for all parameters. Explain how did you take into account the days without observation in your model.

Fit the model, do convergence diagnostics, print out the summary of the results, and discuss them.

Make sure that the Effective Sample Size is at least 1000 for all 6 hyperparameters (you need to choose burn-in and number of steps appropriately for this).

Explanation: Since, the dataset doesn't contain the dates on which data is missing. To fix this, we have generated a sequence of dates for the analysis and introduced the missingness by merging the sequence into original data set. After doing this, we have extracted the values of USD with NA's on every Saturday and Sunday. JAGS is able to handle these NA values automatically by creating stochastic nodes for them.

```
model_string02 <- "
  ## Declaring the size of the variables before the model using the var command
  var epsilon_eta[n,2], Sigma_rho[2,2], beta[2];

  model{
  # Priors for hyperparameters
  mu ~ dnorm(0, 1.0E-1)
  phi ~ dunif(-1, 1)
  rho ~ dunif(-1, 1)
  tau ~ dgamma(1, 1)
  v ~ dunif(1.0E-5, 1.0E+5)

  for (i in 1:2){
    beta[i] ~ dnorm(0, 1.0E-2)
  }
  # Priors for h[1] and y[1]
  h[1] ~ dnorm(mu, sqrt(1-phi*phi)/sigma)
  y[1] ~ dnorm(mu.y0,0.01)

  # Covariance matrix for Sigma_rho
  Sigma_rho[1, 1] <- 1
  Sigma_rho[2, 2] <- 1
  Sigma_rho[1, 2] <- rho
  Sigma_rho[2, 1] <- rho

  ## for epsilon and eta
  for (t in 1:n) {
    eta[t] ~ dnorm(0,1)
    epsilon[t] ~ dt(rho * eta[t], 1, v)
  }

  ## Likelihood
  for (t in 2:n) {
    mu.y[t] <- beta[1] + beta[2] * (y[t-1]-mean.USD)
    #observations
    y[t] ~ dnorm(mu.y[t], exp(h[t]/2) * sqrt(1-rho^2))
    # Replicates
    y.rep[t] ~ dnorm(mu.y[t], exp(h[t]/2) * sqrt(1-rho^2))
  }

  for(t in 1:n){
    h[t+1] ~ dnorm(mu + phi * (h[t] - mu)+ sigma * eta[t], tau.h)
  }

  tau.h ~ dgamma(2,1)


  # Precision for likelihood
  sigma <- 1 / sqrt(tau)
}"

## Intializing the parameters
n <- length(exratesUSD)
y <- exratesUSD
mean.USD <- mean.USD
mu.y0 <- exratesUSD[1]

## compiling model the model using jags
model02 <- jags.model(textConnection(model_string02), data = list(y=y, n=n,mu.y0 = mu.y0, mean.USD=mean
```

```
## Compiling model graph
##    Declaring variables
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 65
##    Unobserved stochastic nodes: 398
##    Total graph size: 1469
##
## Initializing model
```

```
## Burnin for 10000 samples
update(model02,10000,progress.bar="none")

## Collecting sample from the model02
samples02 <- coda.samples(model02, variable.names = c("beta[1]", "beta[2]", "mu", "phi", "sigma", "rho"
```

The choice of priors for all six hyper parameters is same as the previous question but in this we are taking a conditional distribution i.e. $\epsilon_t \mid \eta_t$ that follows a student's t distribution, which is robust to outliers.

μ ~ dnorm(0, 1.0E-1): This gives the parameter μ a normal prior distribution with mean 0 and a low precision . This prior suggests that the model has limited knowledge of the actual value of μ, which might be anywhere within a range close to zero.

ϕ and ρ ~ dunif(-1, 1): This gives the parameter ϕ and ρ uniform prior distribution between -1 and 1. The model does not favour any particular values between -1 and 1, according to this prior.

ν ~ dunif(1.0E-5, 1.0E+5): We used uniform prior for the degrees of freedom. The model does not favour any particular values between the given range, according to this prior.

For beta[i] ~ dnorm(0, 1.0E-2): This assigns a normal prior distribution with mean 0 and a small variance of 0.0001 for the parameters beta[1] and beta[2]. This prior indicates that the model has little information about the true values of beta.

In this model, the priors for ϕ , ρ, ν and beta are non-informative because they are uniform or normal distributions with very low precision. These priors do not express any strong prior belief about the values of these parameters.

The priors for μ and τ are somewhat informative but not strongly so. The normal prior for mu with mean 0 and variance 0.01 indicates that the model has a weak prior belief that μ is likely to be near zero, but this prior is not strongly informative. The gamma prior for τ with shape parameter 1 and rate parameter 1 is a weakly informative prior that is sometimes used for variance parameters.

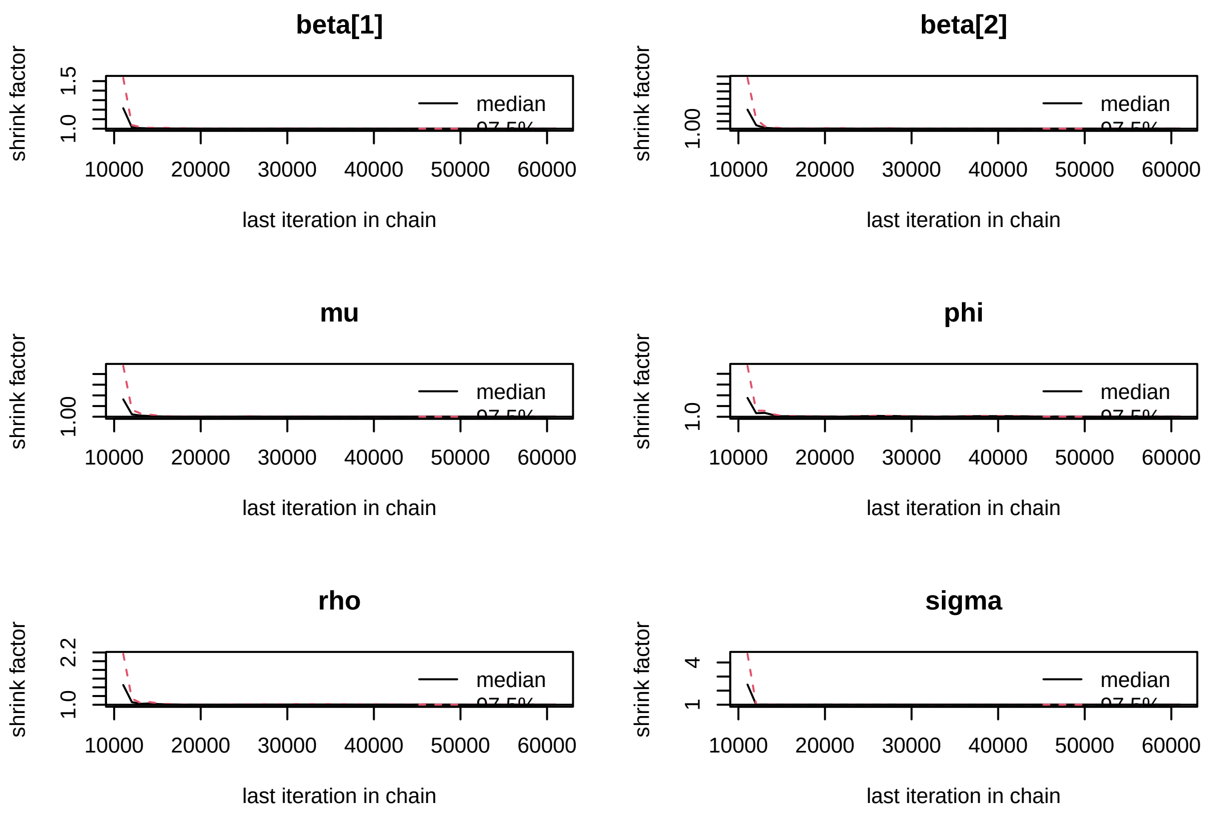
The prior for h[1] is informative because it depends on the values of μ, ϕ, and σ. This prior expresses a prior belief that h[1] is likely to be close to μ but allows for some variability depending on the values of ϕ and σ.

The prior for y[1] is somewhat informative because it has a small variance of 0.01. This prior expresses a weak prior belief that y[1] is likely to be close to mu.y0 (i.e. first value of USD) on starting date but allows for some variability.

The main reason to choose these priors is that these can help to prevent over fitting and improve the generalization of the model to new data. Hence, we selected these priors.

We are also centering the data around the mean because centering data can improve the convergence. This is because centering the data can make the objective function smoother and more well-behaved, which can make optimization more efficient. We ran 5 chains with 50000 samples after 10000 burn-in iterations.

```
gelman.plot(samples02)
```



Explanation: These plots show the Gelman-Rubin statistics computed from increasing number of samples. The red dotted line shows 97.5% confidence interval for these statistics, indicating that a burnin of 10000 would be sufficient.

```
gelman.diag(samples02)
```

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

Explanation : As the Gelman-Rubin diagnostics values are less than 1.05 for each of the parameters, as can be seen from the gelman.diag or gelman.plot findings, we have done a good job of approximating the stationary distribution.

```
summary(samples02)
```

```
##
## Iterations = 11001:61000
## Thinning interval = 1
## Number of chains = 5
## 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
## beta[1]  0.9856 0.0006389 1.278e-06      4.658e-06
## beta[2]  0.9498 0.0279934 5.599e-05      2.027e-04
## mu       4.2924 3.7287011 7.457e-03      1.834e-02
## phi      0.9979 0.0048364 9.673e-06      4.040e-05
## rho     -0.0029 0.4957033 9.914e-04      7.135e-03
## sigma    0.9414 0.2969151 5.938e-04      5.268e-03
##
## 2. Quantiles for each variable:
##
##             2.5%     25%       50%    75%   97.5%
## beta[1]  0.9843  0.9852  0.985607 0.9860  0.9868
## beta[2]  0.8936  0.9314  0.950133 0.9685  1.0038
## mu      -2.8798  1.7274  4.247547 6.8205 11.6715
## phi      0.9842  0.9982  0.999597 0.9999  1.0000
## rho     -0.8693 -0.4058 -0.002143 0.3990  0.8666
## sigma    0.5097  0.7270  0.892343 1.1023  1.6460
```

Explanation: The summary (samples02) command is used to print the posterior summaries for the model parameters beta[1], beta[2], mu, phi, rho and sigma. The standard deviations obtained are much smaller in magnitude than the means, suggesting that the amount of data available was enough to obtain reasonably confident estimates of the model parameters. Moreover, the time series SE was considerably smaller than the posterior means (in absolute value), which implies that our estimates are quite accurate and the chain are mixing reasonably well.

```r
## Getting the effective sample size
effectiveSize(samples02)
```

```
##   beta[1]   beta[2]        mu       phi       rho     sigma
## 19146.599 19296.465 41377.097 14842.552  4984.116  3286.250
```

Explanation: We have computed the ESS for the model parameters beta[1], beta[2], sigma, phi, rho, by using the effective Size function. And we can see that sample size is about 1000 for all 6 hyperparamters so we don't need more samples.

c)[10 marks]

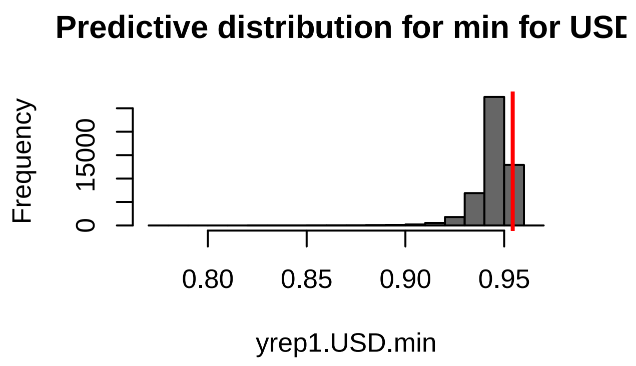
Perform posterior predictive checks on both models a) and b). Explain how did you choose the test functions.

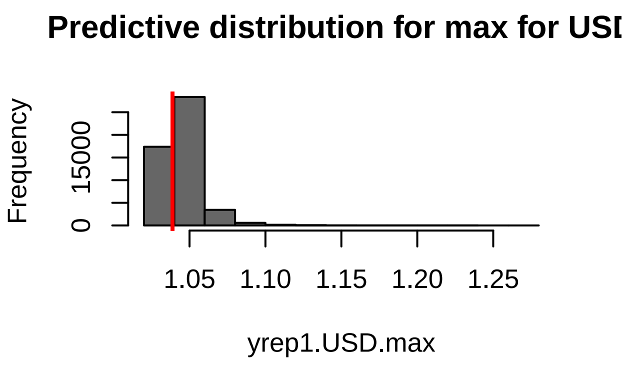
Discuss the results.

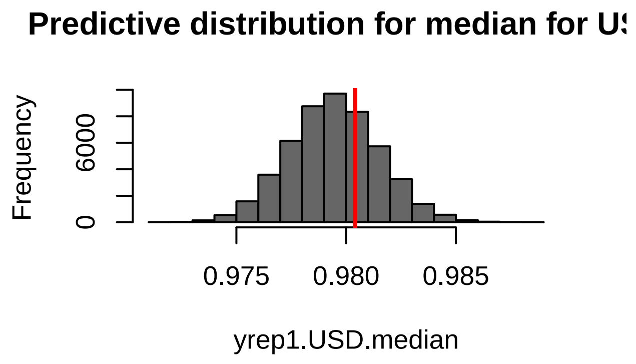
Explanation: To perform posterior predictive checking, we are using minimum, maximum, median test function. Because, this helps in comparing the observed minimum, maximum, median of observed data to the min, max, median of the simulated samples. If the observed min, max and median doesn't lie in the range of the minimum, maximum, median of simulated samples, then it may indicate potential shortcoming of our model. We can say that observed data not plausible under predictive distribution and model is not a good fit for the data. if min, max and median of the simulated samples does lie in the range that it may indicate that model is good fit for the data.

```r
## Posterior predictive checking for model1
y.replicates = coda.samples(model01,variable.names=c("y.rep"),n.iter=10000, progress.bar="none", n.thin

yrep1 =as.matrix(y.replicates)

yrep.USD1 <- yrep1[,1:(n-1)]
y.not.NA <- which(!is.na(y))

exratesUSD.not.NA <- exratesUSD[y.not.NA]

yrep1.not.NA <-yrep.USD1[,y.not.NA]

yrep1.USD.min <- apply(yrep1, 1, min)
yrep1.USD.max <- apply(yrep1, 1, max)
yrep1.USD.median <- apply(yrep1, 1, median)

par(mfrow=c(2,2))
hist(yrep1.USD.min,col="gray40",main="Predictive distribution for min for USD")
abline(v=min(exratesUSD.not.NA),col="red",lwd=2)

hist(yrep1.USD.max,col="gray40",main="Predictive distribution for max for USD")
abline(v=max(exratesUSD.not.NA),col="red",lwd=2)
```

```r
hist(yrep1.USD.median,col="gray40",main="Predictive distribution for median for USD")
abline(v=median(exratesUSD.not.NA), col="red",lwd=2)
```



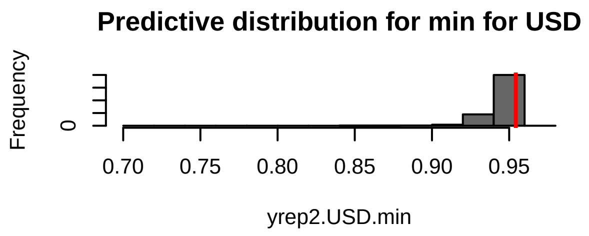


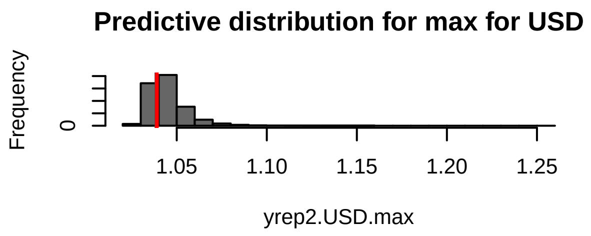


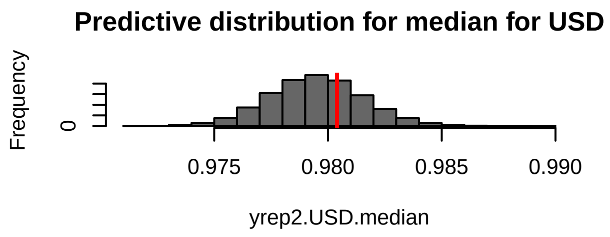
As we can see, the posterior predictive checks did not detect any issues with the models in 1(a). The values of the functions min, max and median applied on the original data for USD's were in the typical range of the realizations of the replicates

```r
## Posterior predictive checking for model 2
y2.replicates = coda.samples(model02,variable.names=c("y.rep"),n.iter=10000, progress.bar="none", n.thi

yrep2 =as.matrix(y2.replicates)

yrep.USD2 <- yrep2[,1:(n-1)]
y.not.NA <- which(!is.na(y))

exratesUSD.not.NA <- exratesUSD[y.not.NA]

yrep2.not.NA <-yrep.USD2[,y.not.NA]

yrep2.USD.min <- apply(yrep2, 1, min)
yrep2.USD.max <- apply(yrep2, 1, max)
yrep2.USD.median <- apply(yrep2, 1, median)

par(mfrow=c(3,2))
hist(yrep2.USD.min,col="gray40",main="Predictive distribution for min for USD")
abline(v=min(exratesUSD.not.NA),col="red",lwd=2)
```

```
hist(yrep2.USD.max,col="gray40",main="Predictive distribution for max for USD")
abline(v=max(exratesUSD.not.NA),col="red",lwd=2)

hist(yrep2.USD.median,col="gray40",main="Predictive distribution for median for USD")
abline(v=median(exratesUSD.not.NA), col="red",lwd=2)
```







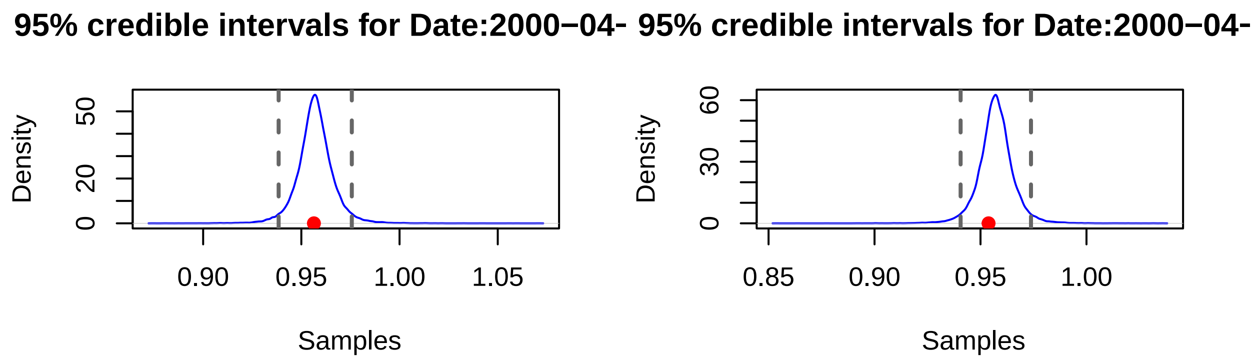
Explanation: As we can see, the posterior predictive checks did not detect any issues with both the models. The values of the functions min, max and median applied on the original data for USD's were in the typical range of the realizations of the replicates.

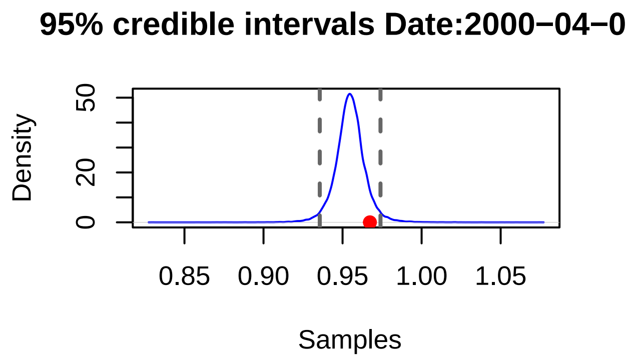
d)[10 marks]

Based on your models a) and b), plot the posterior predictive densities of the USD/EUR rate on the dates 2000-04-03, 2000-04-04 and 2000-04-05 (the next 3 days after the period considered). Compute the posterior means and 95% credible intervals. Discuss the results.

```
## For model (a)
## Extarcting the Data
sdate <- as.Date("2000-01-03")
edate <- as.Date("2000-04-05")
exratesUSD2 <- exrates_new$USD[exrates_new$date >= sdate & exrates_new$date <= edate]

m.y0 <- exratesUSD2[1]
n <- length(exratesUSD2)
model01.d <- jags.model(textConnection(model_string01), data = list(y=exratesUSD2, n=n,mu.y0 = m.y0, mea
## Compiling model graph
##    Declaring variables
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 68
##    Unobserved stochastic nodes: 315
##    Total graph size: 1517
##
## Initializing model
```

```
ypost1.replicates <- coda.samples(model01.d,variable.names=c("y.rep"),n.iter=10000, progress.bar="none"

ypost1_mat <- as.matrix(ypost1.replicates)
ypost1 <- as.matrix(ypost1_mat[,(ncol(ypost1_mat)-2):ncol(ypost1_mat)])

post1.mean <- apply(ypost1, 2, mean)
post1.sd <- apply(ypost1, 2, sd)

cred_intdate1 <- post1.mean[1] + 1.96 * post1.sd[1]
cred_n_intdate1 <- post1.mean[1] - 1.96 * post1.sd[1]


cred_intdate2 <- post1.mean[2] + 1.96 * post1.sd[2]
cred_n_intdate2 <- post1.mean[2] - 1.96 * post1.sd[2]


cred_intdate3 <- post1.mean[3] + 1.96 * post1.sd[3]
cred_n_intdate3 <- post1.mean[3] - 1.96 * post1.sd[3]

par(mfrow=c(2,2))
plot(density(ypost1[, 1]), main="95% credible intervals for Date:2000-04-03 ",col='blue', xlab=" Samples
points(exratesUSD2[length(exratesUSD2)-2], 0, col = "red", pch = 19) ## True value
abline(v=c(cred_intdate1, cred_n_intdate1),col="gray40",lwd=2, lty = 2)

plot(density(ypost1[, 2]), main="95% credible intervals for Date:2000-04-04 ", col='blue', xlab=" Sampl
points(exratesUSD2[length(exratesUSD2)-1], 0, col = "red", pch = 19) ## True value
abline(v=c(cred_intdate2, cred_n_intdate2),col="gray40",lwd=2, lty = 2)

plot(density(ypost1[, 3]), main="95% credible intervals Date:2000-04-05 ", col="blue", xlab=" Samples")
points(exratesUSD2[length(exratesUSD2)], 0, col = "red", pch = 19) ## True value
abline(v=c(cred_intdate3, cred_n_intdate3),col="gray40",lwd=2, lty = 2)
```





Above posterior predictive density plot shows the distribution of predicted values based on the observed data and the posterior distribution of the model parameters. The plot can help us to evaluate whether the model is a good fit for the data, and whether the predicted values are consistent with the observed values. We can see from above plot that our observed values(red point) lie well within the range of 95% credible interval(gray intervals) for all of the three dates.

```
cat("Posterior means for samples of dates 2000-04-03, 2000-04-04, 2000-04-05 are:", post1.mean[1:3],"re
```

```
## Posterior means for samples of dates 2000-04-03, 2000-04-04, 2000-04-05 are: 0.9570686 0.9572142 0.9
```

```
cat("95% credible intervals for date 2000-04-03 are:","[",cred_n_intdate1, cred_intdate1,"]","\n")
```

```
## 95% credible intervals for date 2000-04-03 are: [ 0.9384435 0.9756937 ]
```

```
cat("95% credible intervals for date 2000-04-04 are:","[",cred_n_intdate2, cred_intdate2,"]","\n")
```

```
## 95% credible intervals for date 2000-04-04 are: [ 0.940609 0.9738193 ]
```

```
cat("95% credible intervals for date 2000-04-05 are:","[",cred_n_intdate3, cred_intdate3,"]","\n")
```

```
## 95% credible intervals for date 2000-04-05 are: [ 0.9355916 0.9739953 ]
```

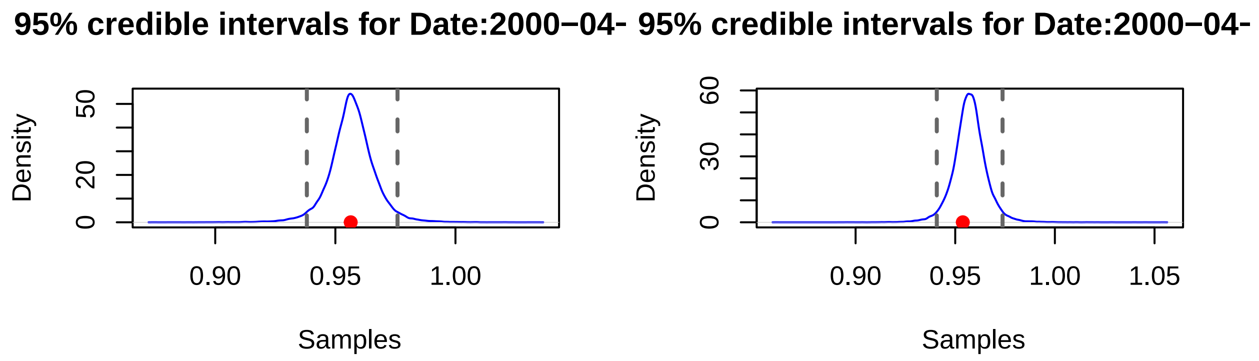
Explanation: Given the observed data and any prior knowledge, the posterior mean of samples informs us of the typical value of a parameter in the posterior distribution. The predicted value of the parameter under the posterior distribution is known as the posterior mean. We have computed the posterior means for three given dates, and their 95% credible interval. This shows that there is a 95% probability that the true value of the parameter falls within the interval which can be seen from plots plotted above for the model fitted in 1(a).

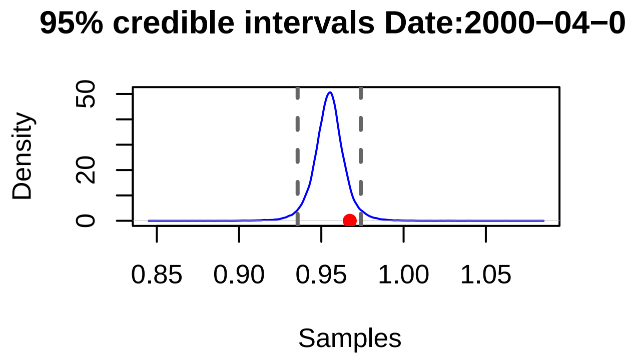
```
## Similarly for 2nd model i.e 1(b)

## Intializing the data
m.y0 <- exratesUSD2[1]
n <- length(exratesUSD2)

model02.d <- jags.model(textConnection(model_string02), data = list(y = exratesUSD2, n=n,mu.y0 = m.y0, 
```

```
## Compiling model graph
##    Declaring variables
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 68
##    Unobserved stochastic nodes: 410
##    Total graph size: 1517
##
## Initializing model
```

```
ypost2.replicates <- coda.samples(model02.d,variable.names=c("y.rep"),n.iter=10000, progress.bar="none"

ypost2_mat <- as.matrix(ypost2.replicates)
ypost2 <- as.matrix(ypost2_mat[,(ncol(ypost2_mat)-2):ncol(ypost2_mat)])

post2.mean <- apply(ypost2, 2, mean)
post2.sd <- apply(ypost2, 2, sd)

cred2_intdate1 <- post2.mean[1] + 1.96 * post2.sd[1]
cred2_n_intdate1 <- post2.mean[1] - 1.96 * post2.sd[1]


cred2_intdate2 <- post2.mean[2] + 1.96 * post2.sd[2]
cred2_n_intdate2 <- post2.mean[2] - 1.96 * post2.sd[2]


cred2_intdate3 <- post1.mean[3] + 1.96 * post1.sd[3]
cred2_n_intdate3 <- post1.mean[3] - 1.96 * post1.sd[3]

par(mfrow=c(2,2))
plot(density(ypost2[, 1]), main="95% credible intervals for Date:2000-04-03 ",col='blue', xlab=" Sample
points(exratesUSD2[length(exratesUSD2)-2], 0, col = "red", pch = 19)
abline(v=c(cred2_intdate1, cred2_n_intdate1),col="gray40",lwd=2, lty = 2)

plot(density(ypost2[, 2]), main="95% credible intervals for Date:2000-04-04 ", col='blue', xlab=" Sampl
points(exratesUSD2[length(exratesUSD2)-1], 0, col = "red", pch = 19)
abline(v=c(cred2_intdate2, cred2_n_intdate2),col="gray40",lwd=2, lty = 2)
```

```
plot(density(ypost2[, 3]), main="95% credible intervals Date:2000-04-05 ", col="blue", xlab=" Samples")
points(exratesUSD2[length(exratesUSD2)], 0, col = "red", pch = 19)
abline(v=c(cred2_intdate3, cred2_n_intdate3),col="gray40",lwd=2, lty = 2)
```





Explanation: Above posterior predictive density plot shows the distribution of predicted values based on the observed data and the posterior distribution of the model parameters. The plot can help us to evaluate whether the model is a good fit for the data, and whether the predicted values are consistent with the observed values. We can see from above plot that our observed values(red point) lie well within the range of 95% credible interval(gray intervals) for all of the three dates for model 2(b).

```
cat("Posterior means for samples of dates 2000-04-03, 2000-04-04, 2000-04-05 are:", post2.mean[1:3],"re
```

```
## Posterior means for samples of dates 2000-04-03, 2000-04-04, 2000-04-05 are: 0.9570123 0.9572471 0.9
```

```
cat("95% credible intervals for date 2000-04-03 are:","[",cred2_n_intdate1, cred2_intdate1,"]","\n")
```

```
## 95% credible intervals for date 2000-04-03 are: [ 0.9381392 0.9758854 ]
```

```
cat("95% credible intervals for date 2000-04-04 are:","[",cred2_n_intdate2, cred2_intdate2,"]","\n")
```

```
## 95% credible intervals for date 2000-04-04 are: [ 0.9407526 0.9737416 ]
```

```
cat("95% credible intervals for date 2000-04-05 are:","[",cred2_n_intdate3, cred2_intdate3,"]","\n")
```

```
## 95% credible intervals for date 2000-04-05 are: [ 0.9355916 0.9739953 ]
```

Explanation: Given the observed data and any prior knowledge, the posterior mean of samples informs us of the typical value of a parameter in the posterior distribution. The predicted value of the parameter under the posterior distribution is known as the posterior mean. We have computed the posterior means for three given dates, and their 95% credible interval. This shows that there is a 95% probability that the true value of the parameter falls within the interval which can be seen from plots plotted above for model 1(b)

e)[10 marks]

In this question, we are going to look use a multivariate stochastic volatility model with leverage to study the USD/EUR and GBP/EUR exchange rates jointly. The model is described as follows,

$$\boldsymbol{y}_t = \boldsymbol{\beta}_0 + \boldsymbol{\beta}_1 \boldsymbol{y}_{t-1} + \exp(h_t/2)\boldsymbol{\epsilon}_t \quad \text{for} \quad 1 \le t \le T,$$
$$\boldsymbol{h}_{t+1} = \boldsymbol{\phi}(\boldsymbol{h}_t) + \boldsymbol{\eta}_t \quad \text{for} \quad 0 \le t \le T, \quad h_0 \sim N(0, I),$$
$$(\epsilon_t, \eta_t) \sim N(0, \Sigma).$$

Here I denotes the 2 x 2 identity matrix, $\boldsymbol{y}_t, \boldsymbol{\beta}_0, \boldsymbol{h}_t, \boldsymbol{\eta}_t, \boldsymbol{\epsilon}_t$ are 2 dimensional vectors, $\boldsymbol{\beta}_1$ and $\boldsymbol{\phi}$ are 2 x 2 matrices, $\boldsymbol{\Sigma}$ is a 4 x 4 covariance matrix. At each time step t, the two components of y_t will be used to model the USD/EUR and GBP/EUR exchange rates, respectively.

Implement this model in JAGS or Stan.

Discuss your choices for priors for every parameter [Hint: you can use Wishart or scaled Wishart priors for** $\boldsymbol{\Sigma}$, **see https://www.stats.ox.ac.uk/~nicholls/MScMCMC15/jags_user_manual.pdf , https://mc-stan.org/docs/2_19/functions-reference/wishart-distribution.html].

Fit the model, do convergence diagnostics, print out the summary of the results, and discuss them.

Problem 2 - NBA data

In this problem, we are going to construct a predictive model for NBA games.

We start by loading the dataset.

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

```
##   GAME_DATE_EST  GAME_ID GAME_STATUS_TEXT HOME_TEAM_ID VISITOR_TEAM_ID SEASON
## 1    2022-12-22 22200477            Final   1610612740      1610612759   2022
## 2    2022-12-22 22200478            Final   1610612762      1610612764   2022
## 3    2022-12-21 22200466            Final   1610612739      1610612749   2022
## 4    2022-12-21 22200467            Final   1610612755      1610612765   2022
## 5    2022-12-21 22200468            Final   1610612737      1610612741   2022
## 6    2022-12-21 22200469            Final   1610612738      1610612754   2022
##   TEAM_ID_home PTS_home FG_PCT_home FT_PCT_home FG3_PCT_home AST_home REB_home
## 1   1610612740      126       0.484       0.926        0.382       25       46
## 2   1610612762      120       0.488       0.952        0.457       16       40
## 3   1610612739      114       0.482       0.786        0.313       22       37
## 4   1610612755      113       0.441       0.909        0.297       27       49
## 5   1610612737      108       0.429       1.000        0.378       22       47
## 6   1610612738      112       0.386       0.840        0.317       26       62
##   TEAM_ID_away PTS_away FG_PCT_away FT_PCT_away FG3_PCT_away AST_away REB_away
## 1   1610612759      117       0.478       0.815        0.321       23       44
## 2   1610612764      112       0.561       0.765        0.333       20       37
## 3   1610612749      106       0.470       0.682        0.433       20       46
## 4   1610612765       93       0.392       0.735        0.261       15       46
## 5   1610612741      110       0.500       0.773        0.292       20        47
## 6   1610612754      117       0.469       0.778        0.462       27        47
##   HOME_TEAM_WINS
## 1              1
## 2              1
## 3              1
## 4              1
## 5              0
## 6              0
```

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

```
##   LEADUE_ID    TEAM_ID MIN_YEAR MAX_YEAR ABBREVIATION  NICKNAME YEARFOUNDED
## 1         0 1610612737     1949     2019          ATL     Hawks        1949
## 2         0 1610612738     1946     2019          BOS   Celtics        1946
## 3         0 1610612740     2002     2019          NOP  Pelicans        2002
## 4         0 1610612741     1966     2019          CHI     Bulls        1966
## 5         0 1610612742     1980     2019          DAL Mavericks        1980
## 6         0 1610612743     1976     2019          DEN   Nuggets        1976
##          CITY                    ARENA ARENACAPACITY          OWNER
## 1     Atlanta         State Farm Arena         18729    Tony Ressler
## 2      Boston                TD Garden         18624   Wyc Grousbeck
## 3 New Orleans     Smoothie King Center            NA      Tom Benson
## 4     Chicago           United Center         21711 Jerry Reinsdorf
## 5      Dallas American Airlines Center         19200      Mark Cuban
## 6      Denver             Pepsi Center         19099    Stan Kroenke
##   GENERALMANAGER      HEADCOACH DLEAGUEAFFILIATION
## 1 Travis Schlenk   Lloyd Pierce      Erie Bayhawks
## 2    Danny Ainge   Brad Stevens    Maine Red Claws
## 3 Trajan Langdon   Alvin Gentry       No Affiliate
## 4     Gar Forman     Jim Boylen   Windy City Bulls
## 5  Donnie Nelson  Rick Carlisle      Texas Legends
## 6   Tim Connelly Michael Malone       No Affiliate
```

games.csv contains the information about games such as GAME_DATE, SEASON, HOME_TEAM_ID, VISITOR_TEAM_ID, PTS_home (final score for home team) and PTS_away (final score for away team).

teams.csv contains the names of each team, i.e. the names corresponding to each team ID.

We are going to fit some Bayesian linear regression models on the scores of each team.

You can use either INLA, JAGS or Stan.

a)[10 marks]

The dataset contains data from 20 seasons, but we are going to focus on only one, the 2021 season.
Please only keep games where SEASON is 2021 in the dataset, and remove all other seasons.
Please order the games according to the date of occurrence (they are not ordered like that in the dataset).

The scores are going to be assumed to follow a linear Gaussian model,

$$S_g^H \sim N(\mu_g^H, \sigma^2), \quad S_g^A \sim N(\mu_g^A, \sigma^2).$$

Here S_g^H denotes the final score of the home team in game g, and S_g^A denotes the final score of the away team in game g.

Note that the true scores can only take non-negative integer values, so the Gaussian distribution is not perfect, but it can still be used nevertheless.

The means for the scores are going to be modeled as a combination of three terms: attacking strength, defending ability, and whether the team is playing at home, or away. For each team, we denote their attacking strength parameter by a_{team}, their defending strength parameter by d_{team}, and the effect of playing at home as h. This quantifies the effect of playing at home on the expected number of goals scored. Our model is the following (μ_g^H is for the goals scored by the home team, and is μ_g^A is for the away team):

$\mu_g^H = \beta_0 + a_{home.team} + d_{away.team} + h$
$\mu_g^A = \beta_0 + a_{away.team} + d_{home.team}$

Implement this model. Select your own prior distributions for the parameters, and discuss the reason for using those priors.

Obtain the summary statistics for the posterior distribution of the model parameters.

Evaluate the root mean square error (RMSE) of your posterior means versus the true scores.

Interpret the results.

Games data set consists for 20 seasons that contains information about Final scores, date of the game and other information. Since, we only have perform analysis on 2021 season, we are only extracting that particular season from the games data and ordering the data by dates column i.e (GAME_DATE_EST). Teams data set contains the id of the team and their name and other information.

```r
## loading the dplyr package
library(dplyr)
```

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

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

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

```r
## loading the INLA package
library(INLA)
```

```
## Loading required package: Matrix

## Loading required package: foreach

## Loading required package: parallel

## Loading required package: sp
## This is INLA_22.12.16 built 2022-12-23 13:24:10 UTC.
##  - See www.r-inla.org/contact-us for how to get help.
```

```
## Getting the data for 2021 season
games.2021 <- games[games$SEASON ==2021,]

## Odering the 2021 season by dates
games.2021 <- games.2021[order(games.2021$GAME_DATE_EST),]
```

Now, mean score will depend on attacking strength, defending ability, and whether the team is playing at home, or away. We have to extract all of this information from the games dataset.

Choice of prior: We are choosing non-informative prior log-gamma function. This is often used as a prior distribution for the precision parameter of a Gaussian distribution when we don't have no prior knowledge or information about the parameter(s) of interest. It ensures that prior doesn't dominates the posterior. In other words, non-informative priors can help to prevent overfitting and improve the generalization of the model to new data. Hence, we selected these priors.

```
## Final scores of home as well as away games
Scores <- c(games.2021$PTS_home, games.2021$PTS_away)

Ng <- nrow(games.2021)

## Getting Home and visitor teams id
HT.ID <- as.character(games.2021$HOME_TEAM_ID)
AT.ID <- as.character(games.2021$VISITOR_TEAM_ID)

## Converting Home and visitor id's inot categorical variables
attack.id <- as.factor(c(HT.ID, AT.ID))
defense.id <- as.factor(c(AT.ID ,HT.ID))

## Getting at home effect
at.home=c(rep(1,Ng), rep(0,Ng))

## Creating a data Frame of the required variables
df <- data.frame(Scores, attack.id, defense.id, at.home)

## Setting the priors
prec.prior <- list(prec=list(prior = "loggamma", param = c(0.01, 0.01)))
prior.beta <- list(mean.intercept = 0, prec.intercept = 0.001, mean = 0, prec = 0.001)

## Fitting the model
model.NBA1 <- inla(Scores ~ 1 + attack.id + defense.id + at.home , data = df, family="gaussian", control

options(max.print=50)

## Summary of the model
summary(model.NBA1)
```

```
##
## Call:
##    c("inla.core(formula = formula, family = family, contrasts = contrasts,
##    ", " data = data, quantiles = quantiles, E = E, offset = offset, ", "
##    scale = scale, weights = weights, Ntrials = Ntrials, strata = strata,
##    ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose =
##    verbose, ", " lincomb = lincomb, selection = selection, control.compute
##    = control.compute, ", " control.predictor = control.predictor,
##    control.family = control.family, ", " control.inla = control.inla,
##    control.fixed = control.fixed, ", " control.mode = control.mode,
##    control.expert = control.expert, ", " control.hazard = control.hazard,
##    control.lincomb = control.lincomb, ", " control.update =
##    control.update, control.lp.scale = control.lp.scale, ", "
##    control.pardiso = control.pardiso, only.hyperparam = only.hyperparam,
##    ", " inla.call = inla.call, inla.arg = inla.arg, num.threads =
##    num.threads, ", " blas.num.threads = blas.num.threads, keep = keep,
##    working.directory = working.directory, ", " silent = silent, inla.mode
##    = inla.mode, safe = FALSE, debug = debug, ", " .parent.frame =
##    .parent.frame)")
## Time used:
##     Pre = 0.847, Running = 0.695, Post = 0.29, Total = 1.83
## Fixed effects:
##                        mean    sd 0.025quant 0.5quant 0.975quant    mode kld
## (Intercept)         113.394 1.714    110.032  113.394    116.755 113.394   0
## attack.id1610612738   -2.660 1.634     -5.865   -2.660      0.546  -2.660   0
## attack.id1610612739   -6.195 1.723     -9.574   -6.195     -2.816  -6.195   0
## attack.id1610612740   -4.171 1.704     -7.513   -4.171     -0.828  -4.171   0
## attack.id1610612741   -1.854 1.711     -5.210   -1.854      1.503  -1.854   0
## attack.id1610612742   -4.740 1.666     -8.008   -4.740     -1.472  -4.741   0
## attack.id1610612743   -0.582 1.714     -3.943   -0.582      2.780  -0.582   0
##  [ reached getOption("max.print") -- omitted 53 rows ]
##
## Model hyperparameters:
##                                          mean   sd 0.025quant 0.5quant
## Precision for the Gaussian observations 0.007 0.00      0.007    0.007
##                                         0.975quant  mode
## Precision for the Gaussian observations      0.008 0.007
##
## Deviance Information Criterion (DIC) ...............: 21615.74
## Deviance Information Criterion (DIC, saturated) ....: 2844.03
## Effective number of parameters .....................: 61.01
##
## Marginal log-Likelihood:  -10955.96
## CPO, PIT is computed
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

Explanation: As, we have used only 3 covariates : attack.id(attacking strength), defense.id(defending strength), at.home(home effect) but there are more regression coefficients, including the intercept mainly because covariates are categorical e.g. attack.id and defense.id. and each different category has a separate contribution, and so a different regression coefficient. Our model is fitting an intercept term and a series of attack-related predictor variables (attack.id1610612738, etc.). Each predictor's mean and standard deviation provide the predicted effect size and variability, while the credible intervals give a range of possible effect size values. All predictors have KLD values of 0, which means that there is no need to do the more computationally intensive "full" Laplace approximation because small values (as in the current case) show that the posterior distribution is well approximated by a Gaussian distribution.

```r
predicted <- model.NBA1$summary.fitted.values[,1]
## Calculating the RMSE
RMSE = sqrt(mean((predicted- df$Scores)^2))
cat("RMSE for 2(a) is:",RMSE)
```

```
## RMSE for 2(a) is: 11.58316
```

```r
cat("DIC of model 2(a):",model.NBA1$dic$dic,"\n")
```

```
## DIC of model 2(a): 21615.74
```

```r
cat("NSLCPO of model 2(a):",-sum(log(model.NBA1$cpo$cpo)),"\n")
```

```
## NSLCPO of model 2(a): 10808.26
```

```r
cat("Standard deviation of mean residuals for this model:",sd(Scores-model.NBA1$summary.fitted.values$me
```

```
## Standard deviation of mean residuals for this model: 11.58524
```

Explanation: The calculated RMSE for our model is 11.58316.

We have also computed the DIC, NLSCPO and the standard deviation of the mean residuals for the fitted model. DIC stands for Deviance Information Criterion, and it is a measure of model fit. In general, lower DIC values indicate better model fit and parsimony. Therefore, when comparing multiple models fit with INLA, the model with the lowest DIC value is preferred and same is the case with NLSCPO values. These number can be useful to compare with other models that we are going to fit and check which model is best.

b)[10 marks] In part a), the model assumed that the home effect is the same for each team. In this part, we consider a team-specific home effect $h_{home.team}$,

$\mu_g^H = \beta_0 + a_{home.team} + d_{away.team} + h_{home.team}$

$\mu_g^A = \beta_0 + a_{away.team} + d_{home.team}$

Implement this model. Select your own prior distributions for the parameters, and discuss the reason for using those priors.

Obtain the summary statistics for the posterior distribution of the model parameters.

Evaluate the root mean square error (RMSE) of your posterior means versus the true scores.

Interpret the results.

In this question, we are considering team specific home effect. Hence, we modified our data set accordingly.

Choice of prior: We are choosing non-informative prior log-gamma function. It ensures that prior doesn't dominates the posterior. In other words, non-informative priors can help to prevent overfitting and improve the generalization of the model to new data. Hence, we selected these priors.

```r
## Getting the Home team ID
HT.id <- as.character(games.2021$HOME_TEAM_ID)
## Modifying the home effect
at.home <- c(HT.id, rep(0,Ng))

## Creating the data for modified home effect
```

```r
df2 <- data.frame(Scores, attack.id, defense.id, at.home)

## Priors
prec.prior <- list(prec=list(prior = "loggamma", param = c(0.01, 0.01)))
prior.beta <- list(mean.intercept = 0, prec.intercept = 0.0001, mean = 0, prec = 0.0001)

## Fitting the model using INLA
model.NBA2 <- inla(Scores ~ 1+attack.id + defense.id + at.home, data = df2, family ="gaussian", control

options(max.print=50)

## Printing the summary of the model
summary(model.NBA2)
```

```
##
## Call:
##    c("inla.core(formula = formula, family = family, contrasts = contrasts,
##    ", " data = data, quantiles = quantiles, E = E, offset = offset, ", "
##    scale = scale, weights = weights, Ntrials = Ntrials, strata = strata,
##    ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose =
##    verbose, ", " lincomb = lincomb, selection = selection, control.compute
##    = control.compute, ", " control.predictor = control.predictor,
##    control.family = control.family, ", " control.inla = control.inla,
##    control.fixed = control.fixed, ", " control.mode = control.mode,
##    control.expert = control.expert, ", " control.hazard = control.hazard,
##    control.lincomb = control.lincomb, ", " control.update =
##    control.update, control.lp.scale = control.lp.scale, ", "
##    control.pardiso = control.pardiso, only.hyperparam = only.hyperparam,
##    ", " inla.call = inla.call, inla.arg = inla.arg, num.threads =
##    num.threads, ", " blas.num.threads = blas.num.threads, keep = keep,
##    working.directory = working.directory, ", " silent = silent, inla.mode
##    = inla.mode, safe = FALSE, debug = debug, ", " .parent.frame =
##    .parent.frame)")
## Time used:
##     Pre = 0.627, Running = 0.816, Post = 0.335, Total = 1.78
## Fixed effects:
##                        mean    sd 0.025quant 0.5quant 0.975quant    mode kld
## (Intercept)         111.730 2.098    107.614  111.730    115.845 111.730   0
## attack.id1610612738   1.096 2.314     -3.443    1.096      5.635   1.096   0
## attack.id1610612739  -3.625 2.431     -8.393   -3.625      1.143  -3.625   0
## attack.id1610612740  -3.746 2.396     -8.446   -3.746      0.954  -3.746   0
## attack.id1610612741  -1.688 2.431     -6.455   -1.688      3.080  -1.688   0
## attack.id1610612742  -2.211 2.342     -6.805   -2.211      2.383  -2.211   0
## attack.id1610612743   0.123 2.398     -4.579    0.123      4.826   0.123   0
##  [ reached getOption("max.print") -- omitted 82 rows ]
##
## Model hyperparameters:
##                                          mean   sd 0.025quant 0.5quant
## Precision for the Gaussian observations 0.007 0.00      0.007    0.007
##                                         0.975quant  mode
## Precision for the Gaussian observations      0.008 0.007
##
## Deviance Information Criterion (DIC) ...............: 21622.67
## Deviance Information Criterion (DIC, saturated) ....: 2873.16
## Effective number of parameters .....................: 90.13
##
## Marginal log-Likelihood:  -11099.44
## CPO, PIT is computed
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

Explanation: Again we can see that, each predictor's mean and standard deviation provide the predicted effect size and variability, while the credible intervals give a range of possible effect size values and all predictors have KLD values of 0, which means that there is no need to do the more computationally intensive "full" Laplace approximation because small values (as in the current case) show that the posterior distribution is well approximated by a Gaussian distribution.

```
## Getting the predictions
predicted <- model.NBA2$summary.fitted.values[,1]

## Calculating the RMSE
RMSE = sqrt(mean((df2$Scores - predicted)^2))
cat("RMSE for 2(b) is:",RMSE)
```

```
## RMSE for 2(b) is: 11.47512
```

```
cat("DIC of model 2(b):",model.NBA2$dic$dic,"\n")
```

```
## DIC of model 2(b): 21622.67
```

```
cat("NSLCPO of model 2(b):",-sum(log(model.NBA2$cpo$cpo)),"\n")
```

```
## NSLCPO of model 2(b): 10812.14
```

```
cat("Standard deviation of mean residuals for this model:",sd(df2$Scores-model.NBA2$summary.fitted.valu
```

```
## Standard deviation of mean residuals for this model: 11.47718
```

Explanation: By comparing the RMSE i.e. 11.58 ,11.47 for 2(a) and 2(b) respectively, we can say that model fitted in 2(b) predicted better than 2(a) by considering the team specific home effect.

We know that DIC (Deviance Information Criterion) is a measure of the goodness of fit of a Bayesian model and by comparing it with 2(a), model fitted in 2(b) has slightly larger value and the NSLPCO value is too slightly larger. But Root means square errors for prediction is quite good than the previous model.

c)[10 marks] Propose an improved linear model using the information in the dataset before the game (you cannot use any information in the same row as the game, as this is only available after the game). Hint: you can try incorporating running averages of some covariates specific to each team, by doing some pre-processing.

Implement your model. Select your own prior distributions for the parameters, and discuss the reason for using those priors.

Obtain the summary statistics for the posterior distribution of the model parameters.

Evaluate the root mean square error (RMSE) of your posterior means versus the true scores.

Interpret the results.

Proposed Model: In this model, we are introducing two more covariates i.e. assists and rebounds done by each team. These assists and rebounds can have a great influence on the final scores. By taking this intuition, we are calculating the rolling averages for both in the previous 4 games. This can be done by doing some preprocessing of the given data set.

First, we are extracting assists and rebounds from the games that happened in season of 2021. After doing that, we are creating a temporary data frame to store them and ordering teams according to dates and their ID's to calculates rolling average for the that specific team. Again, reordering the resultant data frame by the dates. All this manipulation is done by using "dplyr" package and "zoo" package is used for calculating rolling averages.

Choice of prior: We are choosing non-informative prior log-gamma function as same as the previous models. It ensures that prior doesn't dominates the posterior. It can help to prevent over fitting and improve the generalization of the model to new data. Hence, we selected these priors.

```
## Loading the required packages
library(dplyr)
library(zoo)
```

```
##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## No. of home games
Ng <- nrow(games.2021)
## home Team ID
HT.id <- as.character(games.2021$HOME_TEAM_ID)

## Extarcting the ID's
HT <- as.character(games.2021$HOME_TEAM_ID)
AT <- as.character(games.2021$VISITOR_TEAM_ID)

## getting attack ID's and defense ID's
attack.id <- as.factor(c(HT, AT))
defense.id <- as.factor(c(AT ,HT))

## getting Data for Assists and rebounds
Assists <- c(games.2021$AST_home, games.2021$AST_away)
Rebounds <- c(games.2021$REB_away, games.2021$REB_home)

at.home <- c(HT.id, rep(0,Ng))

## Getting NickNmaes for the teams
team.nick <- teams$NICKNAME[attack.id]

## Getting the dates
Date <- games.2021$GAME_DATE_EST

## creating a temporary dataframe to do rolling average on
tempdf <- data.frame(Date, Scores, attack.id, Assists, defense.id, Rebounds,
```

```r
                      at.home, team.nick)
## Arranging by date and attack.id
tempdf <- tempdf %>% arrange(attack.id,Date)

## grouping by attack.id and calculating rolling avg for attack and defense for a specific ## team
df_rolling <- tempdf %>% group_by(attack.id) %>%
  mutate(
    roll.avg.assi = rollmean(Assists, k = 4, fill = NA,align="right"),
    roll.avg.re = rollmean(Rebounds, k = 4, fill = NA,align="right"))

## Again arranging the date in order of dates
df_rolling <- df_rolling %>% arrange(Date)

## Priors
prec.prior <- list(prec=list(prior = "loggamma", param = c(0.01, 0.01)))
prior.beta <- list(mean.intercept = 0, prec.intercept = 0.001, mean = 0, prec = 0.001)

## fiiting the model Using INLA
model.NBA3 <- inla(Scores ~ 1 + attack.id + defense.id + roll.avg.assi+ roll.avg.re + at.home, data = d

options(max.print=50)
## Summary of the model
summary(model.NBA3)
```

```
##
## Call:
##    c("inla.core(formula = formula, family = family, contrasts = contrasts,
##    ", " data = data, quantiles = quantiles, E = E, offset = offset, ", "
##    scale = scale, weights = weights, Ntrials = Ntrials, strata = strata,
##    ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose =
##    verbose, ", " lincomb = lincomb, selection = selection, control.compute
##    = control.compute, ", " control.predictor = control.predictor,
##    control.family = control.family, ", " control.inla = control.inla,
##    control.fixed = control.fixed, ", " control.mode = control.mode,
##    control.expert = control.expert, ", " control.hazard = control.hazard,
##    control.lincomb = control.lincomb, ", " control.update =
##    control.update, control.lp.scale = control.lp.scale, ", "
##    control.pardiso = control.pardiso, only.hyperparam = only.hyperparam,
##    ", " inla.call = inla.call, inla.arg = inla.arg, num.threads =
##    num.threads, ", " blas.num.threads = blas.num.threads, keep = keep,
##    working.directory = working.directory, ", " silent = silent, inla.mode
##    = inla.mode, safe = FALSE, debug = debug, ", " .parent.frame =
##    .parent.frame)")
## Time used:
##     Pre = 0.619, Running = 0.835, Post = 0.357, Total = 1.81
## Fixed effects:
##                          mean    sd 0.025quant 0.5quant 0.975quant    mode kld
## (Intercept)           107.982 2.167    103.731  107.982    112.231 107.983   0
## attack.id1610612738     0.227 2.111     -3.913    0.227      4.368   0.227   0
## attack.id1610612739    -4.677 2.220     -9.031   -4.677     -0.323  -4.678   0
## attack.id1610612740    -5.748 2.189    -10.042   -5.748     -1.454  -5.748   0
## attack.id1610612741    -0.716 2.219     -5.068   -0.716      3.636  -0.717   0
## attack.id1610612742    -0.056 2.138     -4.248   -0.056      4.137  -0.056   0
## attack.id1610612743   -4.632 2.201     -8.949   -4.632     -0.315  -4.633   0
##  [ reached getOption("max.print") -- omitted 84 rows ]
##
## Model hyperparameters:
##                                           mean   sd 0.025quant 0.5quant
## Precision for the Gaussian observations 0.009 0.00      0.008    0.008
##                                         0.975quant  mode
## Precision for the Gaussian observations      0.009 0.009
##
## Deviance Information Criterion (DIC) ...............: 21221.68
## Deviance Information Criterion (DIC, saturated) ....: 2878.76
## Effective number of parameters .....................: 91.98
##
## Marginal log-Likelihood:  -10820.37
## CPO, PIT is computed
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

Explanation: We can again see that, each predictor's mean and standard deviation provide the predicted effect size and variability, while the credible intervals give a range of possible effect size values and all predictors have KLD values of 0, which means that there is no need to do the more computationally intensive "full" Laplace approximation because small values (as in the current case) show that the posterior distribution is well approximated by a Gaussian distribution.

```r
## Getting the predictions
predicted <- model.NBA3$summary.fitted.values[,1]

## Calculating the RMSE
RMSE = sqrt(mean((df_rolling$Scores - predicted)^2))
cat("RMSE for 2(c) is:",RMSE)
```

```
## RMSE for 2(c) is: 10.66945
```

```r
cat("DIC of model 2(b):",model.NBA3$dic$dic,"\n")
```

```
## DIC of model 2(b): 21221.68
```

```r
cat("NSLCPO of model 2(b):",-sum(log(model.NBA3$cpo$cpo)),"\n")
```

```
## NSLCPO of model 2(b): 10611.69
```

```r
cat("Standard deviation of mean residuals for this model:",sd(df_rolling$Scores-model.NBA3$summary.fitt
```

```
## Standard deviation of mean residuals for this model: 10.67137
```

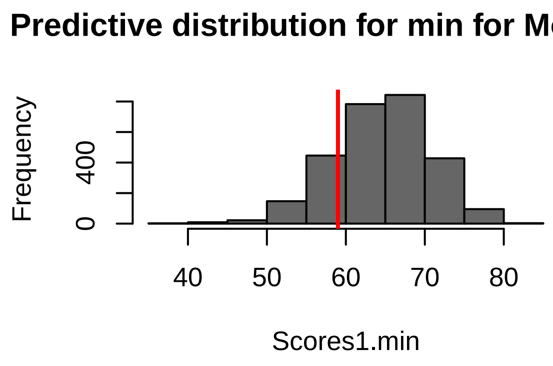
Explanation: The Root mean square error for the improved model is 10.66945, which is best of the three fitted models. We can also check the DIC as well as NSLCPO values which is lower than both model fitted in 2(a) and 2(b) parts. We can conclude that model fitted is the best of three fitted models. Hence, the assumption of taking rolling averages for assists and rebounds turns out to be true.

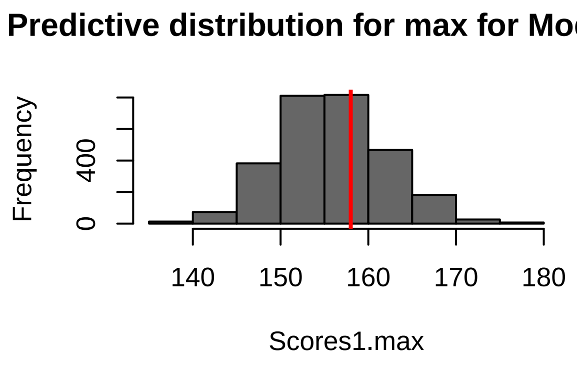
d)[10 marks] Perform posterior predictive checks on all 3 models a), b), and c). Explain how did you choose the test functions.

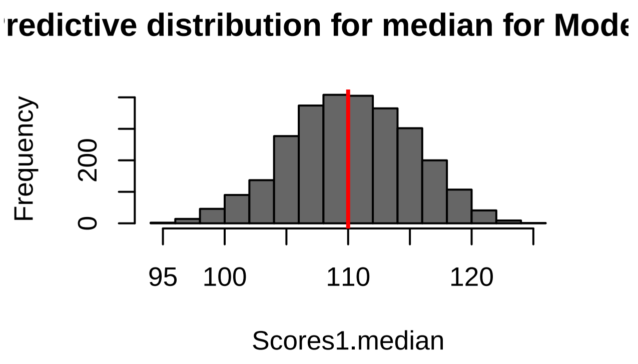
Discuss the results.

Explanation: To perform posterior predictive checking, we are using minimum, maximum, median test function. Because, this helps in comparing the observed minimum, maximum, median of observed data to the min, max, median of the simulated samples. If the observed min, max and median doesn't lie in the range of the minimum, maximum, median of simulated samples, then it may indicate potential shortcoming of our model. We can say that observed data not plausible under predictive distribution and model is not a good fit for the data. if min, max and median of the simulated samples does lie in the range that it may indicate that model is good fit for the data.

```r
## for Model 2(a)
nsamp <- 10000
model01.samples <- inla.posterior.sample(n=nsamp, result=model.NBA1)
predicted.samples1 <- inla.posterior.sample.eval(function(...) {Predictor}, model01.samples)
sigma.samples1 <- 1/sqrt(inla.posterior.sample.eval(function(...) {theta}, model01.samples))

n <- nrow(df)
yNBA1 <- matrix(0,nrow=n,ncol=nsamp)

for(row.num in 1:n){
   yNBA1[row.num, ] <- predicted.samples1[row.num, ] + rnorm(n=nsamp,mean=0,sd=sigma.samples1)
}

Scores1.min <- apply(yNBA1, 1, min)
Scores1.max <- apply(yNBA1, 1, max)
Scores1.median <- apply(yNBA1, 1, median)

## Plotting the plots
par(mfrow=c(2,2))
hist(Scores1.min, col="gray40",main="Predictive distribution for min for Model 2(a)")
abline(v=min(df$Scores),col="red",lwd=2)

hist(Scores1.max,col="gray40",main="Predictive distribution for max for Model 2(a)")
abline(v=max(df$Scores),col="red",lwd=2)

hist(Scores1.median,col="gray40",main="Predictive distribution for median for Model 2(a)")
abline(v=median(df$Scores),col="red",lwd=2)
```



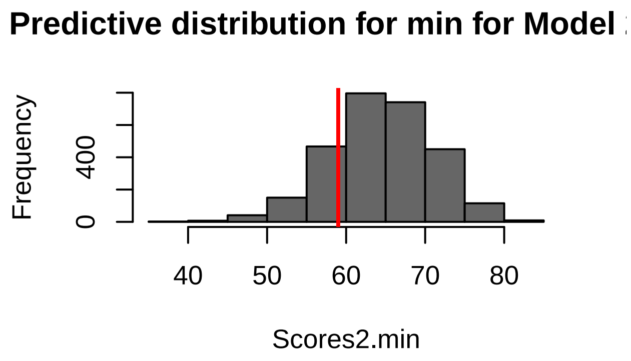


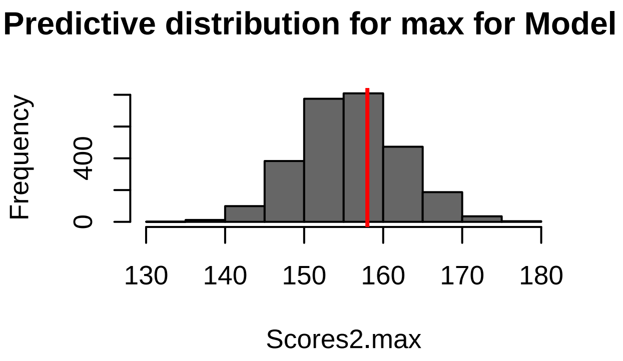


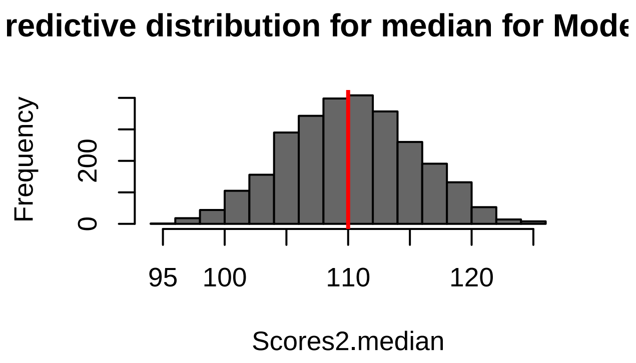
Explanation: As we can see in above plots that for model fitted in 2(a), the min, max, median lie well within the range of observed data. This implies that observed data is plausible under predictive distribution and model 2(a) is a good fit for the data.

```r
## for Model 2(b)
nsamp <- 10000
model02.samples <- inla.posterior.sample(n=nsamp, result=model.NBA2)
predicted.samples2 <- inla.posterior.sample.eval(function(...) {Predictor}, model02.samples)
sigma.samples2 <- 1/sqrt(inla.posterior.sample.eval(function(...) {theta}, model02.samples))

n <- nrow(df2)
yNBA2 <- matrix(0,nrow=n,ncol=nsamp)

for(row.num in 1:n){
   yNBA2[row.num, ] <- predicted.samples2[row.num, ] + rnorm(n=nsamp,mean=0,sd=sigma.samples2)
}

Scores2.min <- apply(yNBA2, 1, min)
Scores2.max <- apply(yNBA2, 1, max)
Scores2.median <- apply(yNBA2, 1, median)

## Plotting the plots
par(mfrow=c(2,2))
hist(Scores2.min, col="gray40",main="Predictive distribution for min for Model 2(b)")
abline(v=min(df2$Scores),col="red",lwd=2)
```

```
hist(Scores2.max,col="gray40",main="Predictive distribution for max for Model 2(b)")
abline(v=max(df2$Scores),col="red",lwd=2)

hist(Scores2.median,col="gray40",main="Predictive distribution for median for Model 2(b)")
abline(v=median(df2$Scores),col="red",lwd=2)
```



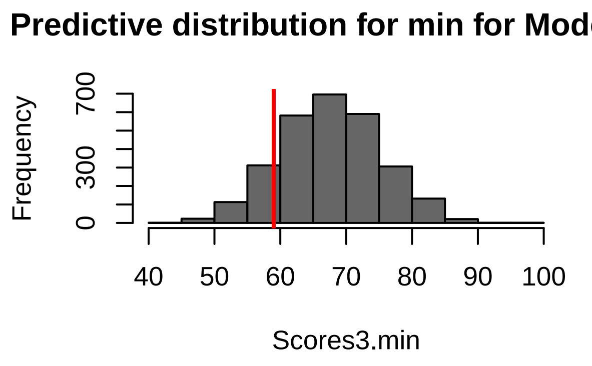


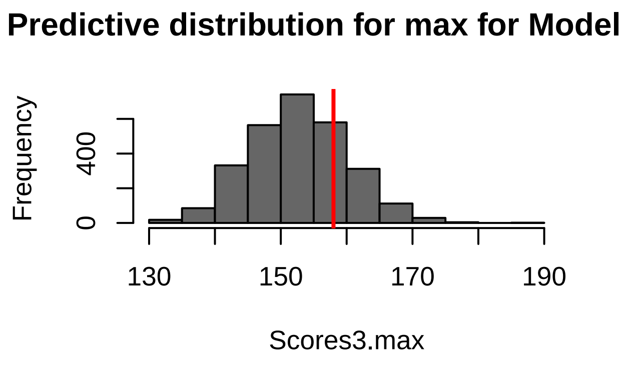


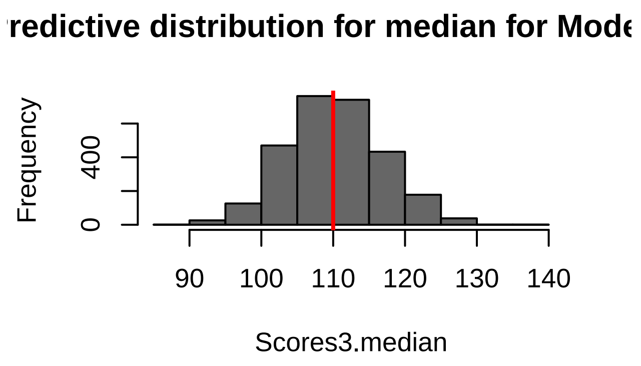
Explanation: As we can see in above plots that for model fitted in 2(b), the min, max, median lie well within the range of observed data. This implies that observed data is plausible under predictive distribution and model 2(b) is also a good fit for the data.

```
## for Model 2(c)
nsamp <- 10000
model03.samples <- inla.posterior.sample(n=nsamp, result=model.NBA3)
predicted.samples3 <- inla.posterior.sample.eval(function(...) {Predictor}, model03.samples)
sigma.samples3 <- 1/sqrt(inla.posterior.sample.eval(function(...) {theta}, model03.samples))

n <- nrow(df_rolling)
yNBA3 <- matrix(0,nrow=n,ncol=nsamp)

for(row.num in 1:n){
   yNBA3[row.num, ] <- predicted.samples3[row.num, ] + rnorm(n=nsamp,mean=0,sd=sigma.samples3)
}

Scores3.min <- apply(yNBA3, 1, min)
Scores3.max <- apply(yNBA3, 1, max)
Scores3.median <- apply(yNBA3, 1, median)
```

```r
## Plotting the plots
par(mfrow=c(2,2))
hist(Scores3.min, col="gray40",main="Predictive distribution for min for Model 2(c)")
abline(v=min(df_rolling$Scores),col="red",lwd=2)

hist(Scores3.max,col="gray40",main="Predictive distribution for max for Model 2(c)")
abline(v=max(df_rolling$Scores),col="red",lwd=2)

hist(Scores3.median,col="gray40",main="Predictive distribution for median for Model 2(c)")
abline(v=median(df_rolling$Scores),col="red",lwd=2)
```







Explanation: As we can see in above plots that for model fitted in 2(c), the min, max, median lie well within the range of observed data. This implies that observed data is plausible under predictive distribution and model 2(c) is also a good fit for the data.

e)[10 marks] In the previous questions, we were assuming a model of the form.

$$S_g^H \sim N(\mu_g^H, \sigma^2), \quad S_g^A \sim N(\mu_g^A, \sigma^2).$$

It is natural to model these two results jointly with a multivariate normal,

$$(S_g^H, S_g^A) \sim N\left(\begin{pmatrix}\mu_g^H \\ \mu_g^A\end{pmatrix}, \Sigma\right),$$

where Σ is a 2 times 2 covariance matrix.

Implement such a model. The definition of μ_g^H and μ_g^A can be either one of a), b), or c), you just need to implement one of them.

Explain how did you choose the prior on Σ [Hint: you can use a Wishart prior, or express this a product of diagonal and correlation matrices and put priors on those terms].

Obtain the summary statistics for the posterior distribution of the model parameters.

Evaluate the root mean square error (RMSE) of your posterior means versus the true scores.

Interpret the results.

Explanation : We are taking wishart distribution for Σ as specified in the hint. Since, the hyperparameters for wishart distribution include a positive definite matrix and degree of freedom. So, I took a 2x2 Identity matrix with a uniform non-informative prior on degrees of freedom. The main reason for choosing is that wishart distribution has a closed-form expression for its normalization constant, which can make it computationally efficient to compute the marginal likelihood.

We are taking a non-informative Normal prior for our attack , defense ID's with very low precision. Because, we have no prior knowledge or information about the parameter(s) of interest. It can help to prevent over fitting and improve the generalization of the model to new data. Hence, we selected these priors.

We are Implementing the definition given in 2(a) .

```r
## Libaray data.table is required for uniqueN function
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```r
# JAGS model string
model_NBA_string <- "
  model {
    # Priors
    beta_0 ~ dnorm(0, 1.0E-6)
    v ~ dunif(1.0E-3, 1.0E+3)

    ## Identity Matrix
    prec_mat[1,1]<-1
    prec_mat[1,2]<-0
    prec_mat[2,1]<-0
    prec_mat[2,2]<-1


    for(j in 1:N.id){
       attack.homeID[j] ~ dnorm(0,1.0E-5)
       attack.awayID[j] ~ dnorm(0,1.0E-5)
       defense.homeID[j] ~ dnorm(0,1.0E-5)
       defense.awayID[j] ~ dnorm(0,1.0E-5)
    }

    ## Likelihood
    for (g in 1:n) {
      # team score
      Score[g,1:2] ~ dmnorm(c(mu.home[g], mu.away[g]), tau)

      ## Replicates
      Scores.rep[g,1:2] ~ dmnorm(c(mu.home[g], mu.away[g]), tau)

      ## For home games
      mu.home[g] <- beta_0 + attack.homeID[attack[g,1]] + defense.awayID[defense[g,2]]
      + 1

      ## For away games
      mu.away[g] <- beta_0 + attack.awayID[attack[g,2]] + defense.homeID[defense[g,1]]
    }
    ## Prior for precision matrix
    tau ~ dwish(prec_mat, v)

  }
"

# Initialize the model and parameters
Scores <- c(games.2021$PTS_home, games.2021$PTS_away)
n <-  Ng ## No. of home games
## first colum consists of home and other one away game data
attack <- cbind(as.numeric(attack.id[1:Ng]), as.numeric(attack.id[Ng+1:length(Scores)]))

defense <- cbind(as.numeric(defense.id[1:Ng]), as.numeric(defense.id[Ng+1:length(Scores)]))

N.id <- uniqueN(attack.id) ## No. of unique ID's

S <- cbind(Scores[1:Ng], Scores[Ng+1:length(Scores)])

modelNBAj <- jags.model(textConnection(model_NBA_string), data = list(attack = attack, defense = defens
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1389
##    Unobserved stochastic nodes: 1512
##    Total graph size: 13563
##
## Initializing model
```

```
## Burnin for 10000 samples
update(modelNBAj,10000,progress.bar="none")

## Collecting sample from the model
samplesNBA <- coda.samples(modelNBAj, variable.names = c("beta_0", "tau", "attack.homeID","attack.awayI
```

```
## Getting posterior summary
summary(samplesNBA)
```

```
##
## Iterations = 11001:31000
## Thinning interval = 1
## Number of chains = 4
## 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
## attack.awayID[1]    7.527e+01 1.942e+02 6.865e-01      4.753e+01
## attack.awayID[2]    5.662e+01 2.358e+02 8.336e-01      5.947e+01
## attack.awayID[3]    2.454e+02 2.207e+02 7.802e-01      3.895e+01
## attack.awayID[4]   -4.120e+01 1.632e+02 5.769e-01      5.295e+01
## attack.awayID[5]   -1.073e+01 1.719e+02 6.077e-01      1.906e+01
## attack.awayID[6]   -6.582e+01 1.790e+02 6.327e-01      3.416e+01
## attack.awayID[7]    1.520e+02 2.117e+02 7.485e-01      4.459e+01
## attack.awayID[8]    1.554e+02 2.364e+02 8.357e-01      2.378e+01
## attack.awayID[9]    1.023e+01 1.995e+02 7.055e-01      4.998e+01
## attack.awayID[10]   4.077e+01 1.590e+02 5.621e-01      4.245e+01
## attack.awayID[11]   3.014e+01 2.675e+02 9.459e-01      4.179e+01
## attack.awayID[12]  -1.724e+01 1.924e+02 6.804e-01      5.243e+01
##  [ reached getOption("max.print") -- omitted 113 rows ]
##
## 2. Quantiles for each variable:
##
##                          2.5%       25%       50%        75%      97.5%
## attack.awayID[1]   -3.668e+02 -4.092e+01  8.726e+01 212.254512 418.631545
## attack.awayID[2]   -3.690e+02 -1.127e+02  4.116e+01 239.449155 482.669015
## attack.awayID[3]   -1.934e+02  1.301e+02  2.539e+02 405.870475 645.111833
## attack.awayID[4]   -3.656e+02 -1.532e+02 -6.464e+01  76.938423 297.400683
## attack.awayID[5]   -2.996e+02 -1.540e+02 -2.277e+01 141.017013 279.241259
## attack.awayID[6]   -3.870e+02 -1.914e+02 -8.669e+01  87.314005 288.270247
## attack.awayID[7]   -1.732e+02 -4.087e+01  1.242e+02 325.590329 562.798236
## attack.awayID[8]   -1.762e+02 -1.207e+01  8.325e+01 285.661956 657.433515
## attack.awayID[9]   -3.599e+02 -1.518e+02  8.826e+00 173.915916 358.961541
## attack.awayID[10]  -2.758e+02 -6.083e+01  4.934e+01 143.634183 355.863342
##  [ reached getOption("max.print") -- omitted 115 rows ]
```

The summary (samplesNBA) command was used to print the posterior summaries for the model parameters "beta_0", "tau", "attack.homeID", "attack.awayID", "defense.homeID", "defense.awayID".

The standard deviations obtained are not much smaller in magnitude than the means(in every case), suggesting that the amount of data available was not enough to obtain reasonably confident estimates of the model parameters. Moreover, the time series SE is also not considerably smaller for every hyperparameter than the posterior means (in absolute value), which implies that our estimates are not quite accurate and the chains are not mixing reasonably well.

Let's calculate the Root mean square error for our estimates by calculating the posterior means for the samples first.

```
## Calculating the RMSE
scores.replicates = coda.samples(modelNBAj,variable.names=c("Scores.rep"),n.iter=2000, progress.bar="no

## Storing Samples in a matrix
```

```
scores.rep <- as.matrix(scores.replicates)
## Getting the predictions for scores by taking m
scores.mean <- apply(scores.rep, 2, mean)
RMSE = sqrt(mean((Scores - scores.mean)^2))
cat("RMSE for 2(e) by using the deinition of 2(a) is:",RMSE)
```

```
## RMSE for 2(e) by using the deinition of 2(a) is: 12.13158
```

Explanation: The Root mean square error is large that the other models fitted above indicating that our model is not a good fit to the observed data as compared to previously fitted model above.