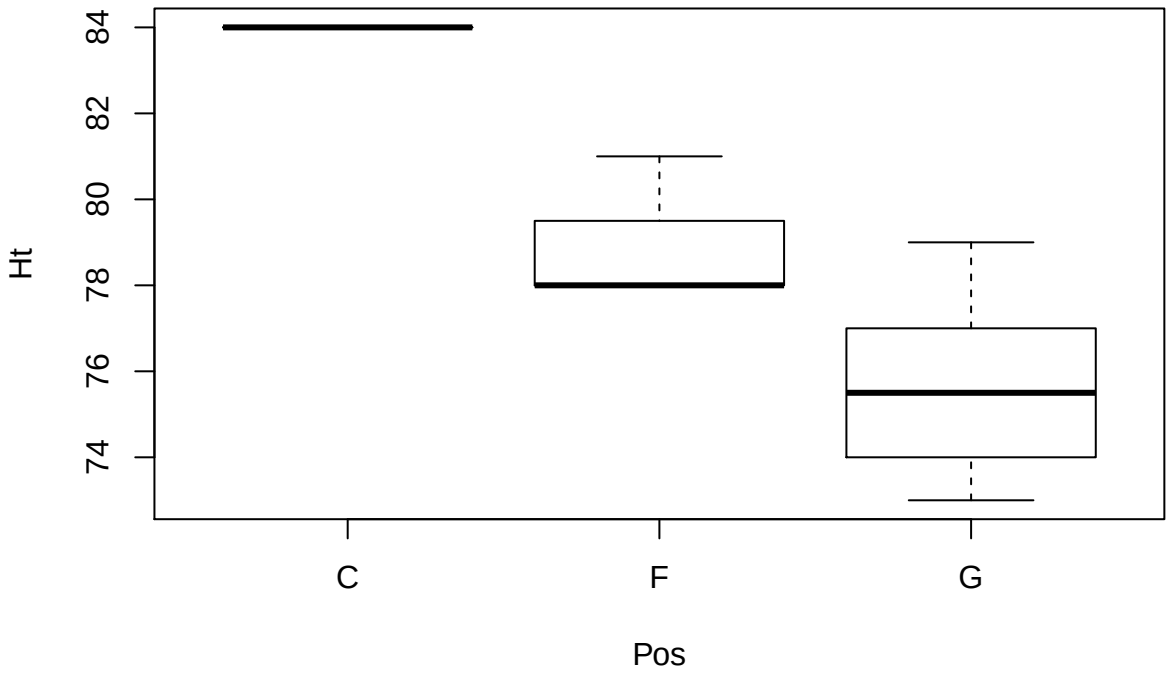
STAT 578 - Advanced Bayesian Modeling - Fall 2019
Assignment 6

Xiaoming Ji

Solution for Problem 1

```
perf_data = read.csv("illinimensbb.csv", header=TRUE)

plot(Ht ~ Pos, data= perf_data)
```



By checking the plot, we do see height and position are highly correlated. *center* has highest mean of height, *forward* has shortest mean of height and *forward* has in between these two. Their value ranges also don't seem to cross each other significantly.

Solution for Problem 2

(a)

```
model {
    for (i in 1:length(FGM)) {
        FGM[i] ~ dbin(prob[i], FGA[i])
```

```
        logit(prob[i]) <- beta_pos[Pos[i]] + beta_ht * Ht_Scaled[i]
        FGM_rep[i] ~ dbin(prob[i], FGA[i])
    }
    for (j in 1:max(Pos)) {
        beta_pos[j] ~ dt(0, 0.01, 1)
    }

    beta_ht ~ dt(0, 0.16, 1)
}
```

```
library(rjags)

df_jags_1 <- list( FGM = perf_data$FGM, FGA = perf_data$FGA,
                   Pos = unclass(perf_data$Pos),
                   Ht_Scaled = as.vector(scale(perf_data$Ht, scale=2*sd(perf_data$Ht))))

initial_vals_1 <- list(list(beta_pos = c(10,10,10), beta_ht=10),
                       list(beta_pos = c(10,10,-10), beta_ht=-10),
                       list(beta_pos = c(10,-10,10), beta_ht=-10),
                       list(beta_pos = c(10,-10,-10), beta_ht=10))

model_1 <- jags.model("perf_1.bug", df_jags_1, initial_vals_1, n.chains = 4,
                      n.adapt = 1000)
update(model_1, 1000)
x1 <- coda.samples(model_1, c("beta_pos","beta_ht","prob","FGM_rep"),
                             n.iter = 1000)
```

```
gelman.diag(x1, autoburnin=FALSE, multivariate = FALSE)
```

```
## Potential scale reduction factors:
##
##             Point est. Upper C.I.
## FGM_rep[1]        1.00       1.00
## FGM_rep[2]        1.00       1.00
## FGM_rep[3]        1.00       1.00
## FGM_rep[4]        1.00       1.01
## FGM_rep[5]        1.00       1.00
## FGM_rep[6]        1.00       1.00
## FGM_rep[7]        1.00       1.00
## FGM_rep[8]        1.00       1.00
## FGM_rep[9]        1.00       1.01
## FGM_rep[10]       1.00       1.00
## FGM_rep[11]       1.00       1.00
## FGM_rep[12]       1.00       1.00
## FGM_rep[13]       1.00       1.00
## FGM_rep[14]       1.00       1.00
## FGM_rep[15]       1.00       1.00
## beta_ht           1.01       1.03
## beta_pos[1]       1.01       1.02
## beta_pos[2]       1.00       1.01
## beta_pos[3]       1.01       1.02
## prob[1]           1.00       1.00
## prob[2]           1.00       1.01
## prob[3]           1.00       1.00
```

```
## prob[4]           1.01       1.02
## prob[5]           1.00       1.01
## prob[6]           1.00       1.01
## prob[7]           1.01       1.02
## prob[8]           1.00       1.01
## prob[9]           1.01       1.03
## prob[10]          1.01       1.02
## prob[11]          1.00       1.00
## prob[12]          1.00       1.01
## prob[13]          1.00       1.00
## prob[14]          1.00       1.01
## prob[15]          1.00       1.00
```

```
coef_sample_1 <- coda.samples(model_1, c("beta_pos","beta_ht","prob","FGM_rep"),
                              n.iter = 10000, thin = 5)
effectiveSize(coef_sample_1[,c("beta_pos[1]", "beta_pos[2]", "beta_pos[3]", "beta_ht")])
```

```
## beta_pos[1] beta_pos[2] beta_pos[3]     beta_ht
##    5620.206    6154.606    5384.465    4514.978
```

(b)

```
summary(coef_sample_1[, c("beta_pos[1]", "beta_pos[2]", "beta_pos[3]", "beta_ht")])
```

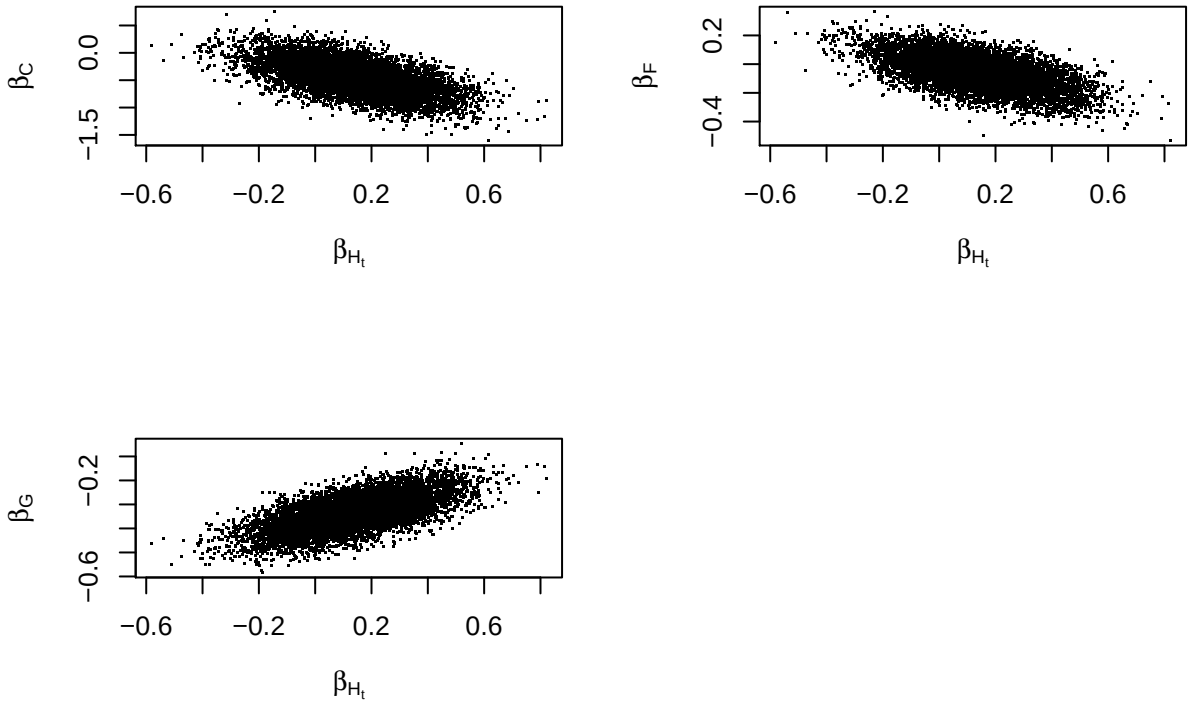
```
##
## Iterations = 3005:13000
## Thinning interval = 5
## Number of chains = 4
## Sample size per chain = 2000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                 Mean     SD  Naive SE Time-series SE
## beta_pos[1] -0.44893 0.2922 0.0032671      0.0039115
## beta_pos[2] -0.06119 0.1115 0.0012472      0.0014231
## beta_pos[3] -0.33594 0.0708 0.0007916      0.0009668
## beta_ht      0.13634 0.1804 0.0020168      0.0026874
##
## 2. Quantiles for each variable:
##
##                2.5%      25%      50%      75%   97.5%
## beta_pos[1] -1.0208 -0.64743 -0.44926 -0.25084  0.1218
## beta_pos[2] -0.2826 -0.13543 -0.05903  0.01311  0.1555
## beta_pos[3] -0.4759 -0.38297 -0.33498 -0.28862 -0.1977
## beta_ht     -0.2115  0.01527  0.13399  0.25912  0.4885
```

(c)

```
par(mfrow=c(2, 2))

plot(as.matrix(coef_sample_1)[,"beta_pos[1]"] ~ as.matrix(coef_sample_1)[,"beta_ht"],
```

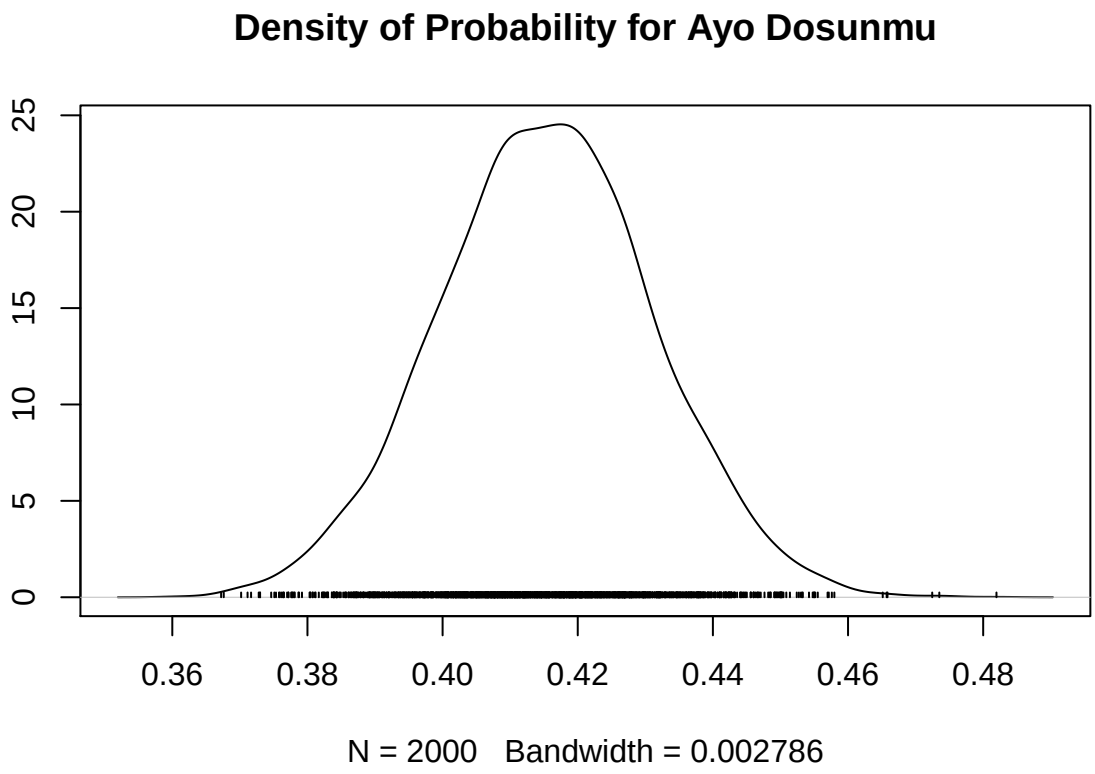
```
     xlab = expression(paste(beta[H[t]])), ylab = expression(paste(beta[C])), pch='.')
plot(as.matrix(coef_sample_1)[,"beta_pos[2]"] ~ as.matrix(coef_sample_1)[,"beta_ht"],
     xlab = expression(paste(beta[H[t]])), ylab = expression(paste(beta[F])), pch='.')
plot(as.matrix(coef_sample_1)[,"beta_pos[3]"] ~ as.matrix(coef_sample_1)[,"beta_ht"],
     xlab = expression(paste(beta[H[t]])), ylab = expression(paste(beta[G])), pch='.')
```



According to the plots, β_C, β_F, β_G are correlated with β_{H_t}.

(d)

```
Dosunmu_index = which(perf_data$X==11)
densplot(coef_sample_1[, paste("prob[",Dosunmu_index,"]",sep="")], main = "Density of Probability for Ay
```



(e)

Probability of $\beta_F > \beta_G$,

```
beta_F = as.matrix(coef_sample_1)[, "beta_pos[2]"]
beta_G = as.matrix(coef_sample_1)[, "beta_pos[3]"]
mean(beta_F > beta_G)
```

```
## [1] 0.961875
```

Bayes factor favoring $\beta_F > \beta_G$ versus $\beta_F < \beta_G$,

```
mean(beta_F > beta_G) / mean(beta_F < beta_G)
```

```
## [1] 25.22951
```

Given the Bayes factor is between 20 to 150, we can say that the data has **Strong** evidence that $\beta_F > \beta_G$.

(f)

```
probs <- as.matrix(coef_sample_1)[, paste("prob[",1:nrow(perf_data),"]", sep="")]
FGM_rep <- as.matrix(coef_sample_1)[, paste("FGM_rep[",1:nrow(perf_data),"]", sep="")]


Tchi <- numeric(nrow(FGM_rep))
Tchirep <- numeric(nrow(FGM_rep))
```

```
for(s in 1:nrow(FGM_rep)){
  Tchi[s] <- sum((perf_data$FGM - perf_data$FGA * probs[s,])^2 /
                   (perf_data$FGA * probs[s,] * (1 - probs[s,])))
  Tchirep[s] <- sum((FGM_rep[s,] - perf_data$FGA * probs[s,])^2 /
                      (perf_data$FGA * probs[s,] * (1 - probs[s,])))
 }

mean(Tchirep >= Tchi)
```

```
## [1] 0.0505
```

The posterior predictive p-value is suspiciously small, although not exceedingly so. Given we don't find any outliers, we conclude that there could be a bit of overdispersion.

(g)

(i)

```
model {
    for (i in 1:length(FGM)) {
        FGM[i] ~ dbin(prob[i], FGA[i])
        logit(prob[i]) <- beta_pos[Pos[i]] + beta_ht * Ht_Scaled[i] + epsilon[i]
        epsilon[i] ~ dnorm(0, 1 / sigma_epsilon^2)
        FGM_rep[i] ~ dbin(prob[i], FGA[i])
    }
    for (j in 1:max(Pos)) {
        beta_pos[j] ~ dt(0, 0.01, 1)
    }

    beta_ht ~ dt(0, 0.16, 1)
    sigma_epsilon ~ dunif(0,10)
}
```

```
df_jags_2 <- list( FGM = perf_data$FGM, FGA = perf_data$FGA,
                   Pos = unclass(perf_data$Pos),
                   Ht_Scaled = as.vector(scale(perf_data$Ht, scale=2*sd(perf_data$Ht))))

initial_vals_2 <- list(list(beta_pos = c(10,10,10), beta_ht=10, sigma_epsilon = 0.01),
                       list(beta_pos = c(10,10,-10), beta_ht=-10, sigma_epsilon = 9),
                       list(beta_pos = c(10,-10,10), beta_ht=-10, sigma_epsilon = 0.01),
                       list(beta_pos = c(10,-10,-10), beta_ht=10, sigma_epsilon = 9))

model_2 <- jags.model("perf_2.bug", df_jags_2, initial_vals_2, n.chains = 4,
                      n.adapt = 1000)
update(model_2, 1000)
x2 <- coda.samples(model_2, c("beta_pos","beta_ht","prob","FGM_rep", "sigma_epsilon"),
                             n.iter = 10000)
```

```
gelman.diag(x2, autoburnin=FALSE, multivariate = FALSE)
```

```
## Potential scale reduction factors:
##
##               Point est. Upper C.I.
## FGM_rep[1]          1.00       1.00
```

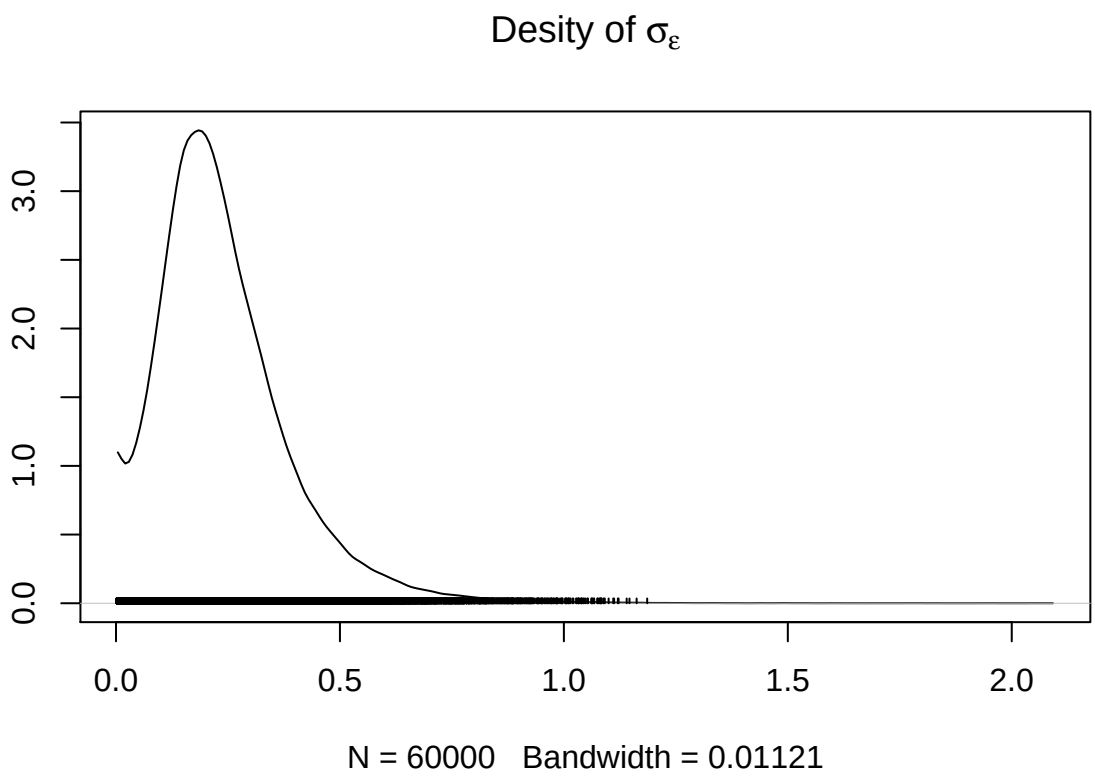
```
## FGM_rep[2]          1.00       1.00
## FGM_rep[3]          1.00       1.00
## FGM_rep[4]          1.00       1.00
## FGM_rep[5]          1.00       1.00
## FGM_rep[6]          1.00       1.00
## FGM_rep[7]          1.00       1.00
## FGM_rep[8]          1.00       1.01
## FGM_rep[9]          1.00       1.00
## FGM_rep[10]         1.00       1.00
## FGM_rep[11]         1.00       1.00
## FGM_rep[12]         1.00       1.00
## FGM_rep[13]         1.00       1.00
## FGM_rep[14]         1.00       1.00
## FGM_rep[15]         1.00       1.00
## beta_ht             1.06       1.07
## beta_pos[1]         1.14       1.25
## beta_pos[2]         1.05       1.05
## beta_pos[3]         1.31       2.32
## prob[1]             1.00       1.00
## prob[2]             1.01       1.01
## prob[3]             1.00       1.00
## prob[4]             1.00       1.00
## prob[5]             1.00       1.00
## prob[6]             1.00       1.00
## prob[7]             1.00       1.00
## prob[8]             1.05       1.09
## prob[9]             1.00       1.00
## prob[10]            1.00       1.00
## prob[11]            1.00       1.01
## prob[12]            1.00       1.00
## prob[13]            1.01       1.01
## prob[14]            1.02       1.03
## prob[15]            1.00       1.00
## sigma_epsilon       1.31       2.33
```

```
coef_sample_2 <- coda.samples(model_2, c("beta_pos","beta_ht","prob","FGM_rep",
                                         "sigma_epsilon"), n.iter = 60000)
effectiveSize(coef_sample_2[,c("beta_pos[1]", "beta_pos[2]", "beta_pos[3]", "beta_ht",
                               "sigma_epsilon")])
```

```
##   beta_pos[1]   beta_pos[2]   beta_pos[3]       beta_ht sigma_epsilon
##      7867.532      6299.644      7446.332      5292.057      4481.706
```

(ii)

```
densplot(coef_sample_2[, "sigma_epsilon"], main = expression(paste("Desity of ", sigma[epsilon])))
```



(iii)

```
beta_F = as.matrix(coef_sample_2)[, "beta_pos[2]"]
beta_G = as.matrix(coef_sample_2)[, "beta_pos[3]"]
mean(beta_F > beta_G)
```

```
## [1] 0.7873792
```

This posterior probability is smaller than previous model.

```
mean(beta_F > beta_G) / mean(beta_F < beta_G)
```

```
## [1] 3.703208
```

This Bayes factor favoring $\beta_F > \beta_G$ versus $\beta_F < \beta_G$ is much smaller than previous model, and we can only say the data has **Positive** (between 3 to 30) evidence that $\beta_F > \beta_G$.

Also Chi-square discrepancy,

```
## [1] 0.3756125
```

Thus we says no overdispersion problems for this model.

Solution for Problem 3

(a)

```
model {
    for (i in 1:length(BLK)) {
        BLK[i] ~ dpois(lambda[i])
        log(lambda[i]) <- log_MIN[i] + beta_pos[Pos[i]] + beta_ht * Ht_Scaled[i]
        BLK_rep[i] ~ dpois(lambda[i])
    }

    for (j in 1:max(Pos)) {
        beta_pos[j] ~ dnorm(0, 0.0001)
    }

    beta_ht ~ dnorm(0, 0.0001)
}
```

```
df_jags_3 <- list( BLK = perf_data$BLK,
                   Pos = unclass(perf_data$Pos),
                   log_MIN = log(perf_data$MIN),
                   Ht_Scaled = as.vector(scale(perf_data$Ht, scale=sd(perf_data$Ht))))

initial_vals_3 <- list(list(beta_pos = c(100,100,100), beta_ht=100),
                       list(beta_pos = c(100,100,-100), beta_ht=-100),
                       list(beta_pos = c(100,-100,100), beta_ht=-100),
                       list(beta_pos = c(100,-100,-100), beta_ht=100))

model_3 <- jags.model("perf_3.bug", df_jags_3, initial_vals_3, n.chains = 4,
                      n.adapt = 1000)
update(model_3, 1000)
x3 <- coda.samples(model_3, c("beta_pos","beta_ht", "lambda","BLK_rep"),
                             n.iter = 2000)
```

```
gelman.diag(x3, autoburnin=FALSE, multivariate = FALSE)
```

```
## Potential scale reduction factors:
##
##             Point est. Upper C.I.
## BLK_rep[1]        1.00       1.00
## BLK_rep[2]        1.00       1.00
## BLK_rep[3]        1.00       1.00
## BLK_rep[4]        1.00       1.00
## BLK_rep[5]        1.00       1.00
## BLK_rep[6]        1.00       1.00
## BLK_rep[7]        1.00       1.00
## BLK_rep[8]        1.00       1.00
## BLK_rep[9]        1.00       1.00
## BLK_rep[10]       1.00       1.00
## BLK_rep[11]       1.00       1.00
## BLK_rep[12]       1.00       1.00
## BLK_rep[13]       1.00       1.00
## BLK_rep[14]       1.00       1.00
## BLK_rep[15]       1.00       1.00
```

```
## beta_ht           1.01       1.02
## beta_pos[1]       1.01       1.02
## beta_pos[2]       1.01       1.02
## beta_pos[3]       1.00       1.01
## lambda[1]         1.00       1.00
## lambda[2]         1.00       1.01
## lambda[3]         1.00       1.00
## lambda[4]         1.00       1.00
## lambda[5]         1.00       1.01
## lambda[6]         1.00       1.01
## lambda[7]         1.00       1.00
## lambda[8]         1.00       1.01
## lambda[9]         1.00       1.01
## lambda[10]        1.00       1.00
## lambda[11]        1.00       1.00
## lambda[12]        1.00       1.01
## lambda[13]        1.00       1.00
## lambda[14]        1.00       1.01
## lambda[15]        1.00       1.01
```

```
coef_sample_3 <- coda.samples(model_3, c("beta_pos","beta_ht", "lambda","BLK_rep"),
                              n.iter = 20000, thin = 5)
effectiveSize(coef_sample_3[,c("beta_pos[1]", "beta_pos[2]", "beta_pos[3]", "beta_ht")])
```

```
## beta_pos[1] beta_pos[2] beta_pos[3]     beta_ht
##    4928.629    5591.952   11216.852    4600.840
```

(b)

```
summary(coef_sample_3[, c("beta_pos[1]", "beta_pos[2]", "beta_pos[3]", "beta_ht")])
```

```
##
## Iterations = 4005:24000
## Thinning interval = 5
## Number of chains = 4
## Sample size per chain = 4000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                Mean     SD Naive SE Time-series SE
## beta_pos[1] -5.2842 0.6055 0.004787       0.008663
## beta_pos[2] -4.5046 0.2848 0.002252       0.003821
## beta_pos[3] -4.4523 0.1802 0.001425       0.001707
## beta_ht      0.9993 0.2738 0.002165       0.004050
##
## 2. Quantiles for each variable:
##
##                2.5%     25%     50%    75%  97.5%
## beta_pos[1] -6.4983 -5.6922 -5.2751 -4.870 -4.128
## beta_pos[2] -5.0794 -4.6943 -4.4973 -4.309 -3.963
## beta_pos[3] -4.8245 -4.5709 -4.4463 -4.328 -4.114
## beta_ht      0.4802  0.8088  0.9967  1.184  1.548
```

(c)

```
beta_ht = as.matrix(coef_sample_3)[, "beta_ht"]
quantile(exp(beta_ht), c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 1.616452 4.703165
```

The values within 95% central posterior credible interval are all greater than 1 and thus we can conclude that greater height is associated with a higher rate of blocking shots.

(d)

```
lambdas <- as.matrix(coef_sample_3)[, paste("lambda[",1:nrow(perf_data),"]", sep="")]
BLK_rep <- as.matrix(coef_sample_3)[, paste("BLK_rep[",1:nrow(perf_data),"]", sep="")]

Tchi <- numeric(nrow(BLK_rep))
Tchirep <- numeric(nrow(BLK_rep))

for(s in 1:nrow(BLK_rep)){
  Tchi[s] <- sum((perf_data$BLK - lambdas[s,])^2 / lambdas[s,])
  Tchirep[s] <- sum((BLK_rep[s,] - lambdas[s,])^2 / lambdas[s,])
 }

mean(Tchirep >= Tchi)
```

```
## [1] 0.007375
```

The posterior predictive p-value is extremely small. Thus this could indicate a problem of overdispersion.

(e)

(i)

```
p_sample <- matrix(FALSE, nrow = nrow(BLK_rep), ncol = nrow(perf_data))
for(s in 1:nrow(BLK_rep)){
  p_sample[s,] <- BLK_rep[s,] > perf_data$BLK
}

p = apply(p_sample, 2, mean)
p_df = data.frame(name=perf_data$Player, p_value=p)
p_df
```

```
##                     name   p_value
## 1  Bezhanishvili, Giorgi 0.5315000
## 2            Cayce, Drew 0.0590000
## 3     De La Rosa, Adonis 0.9868750
## 4           Dosunmu, Ayo 0.7056875
## 5          Feliz, Andres 0.8322500
## 6         Frazier, Trent 0.8228125
## 7          Griffin, Alan 0.0075000
## 8         Griffith, Zach 0.1846250
## 9          Jones, Tevian 0.9021250
## 10         Jordan, Aaron 0.1375625
```

```
## 11           Kane, Samba 0.0016875
## 12       Nichols, Kipper 0.2358125
## 13     Oladimeji, Samson 0.1816875
## 14      Underwood, Tyler 0.2709375
## 15    Williams, Da'Monte 0.0446875
```

(ii)

```
p_df[p_df$p_value < 0.05,]
```

```
##                   name   p_value
## 7       Griffin, Alan 0.0075000
## 11        Kane, Samba 0.0016875
## 15 Williams, Da'Monte 0.0446875
```

(iii)

```
p_df[p_df$p_value > 0.95,]
```

```
##                 name  p_value
## 3 De La Rosa, Adonis 0.986875
```

By looking at the original data, **Adonis** played in center position and was 84 height. He played 225 minutes but blocked only 1 shot. Samba in another hand, also played in center position and was also 84 height. For 86 minutes he played, blocked 10 shots. This makes the model always overestimate the blocks by Adonis. Thus the p-value is very high.