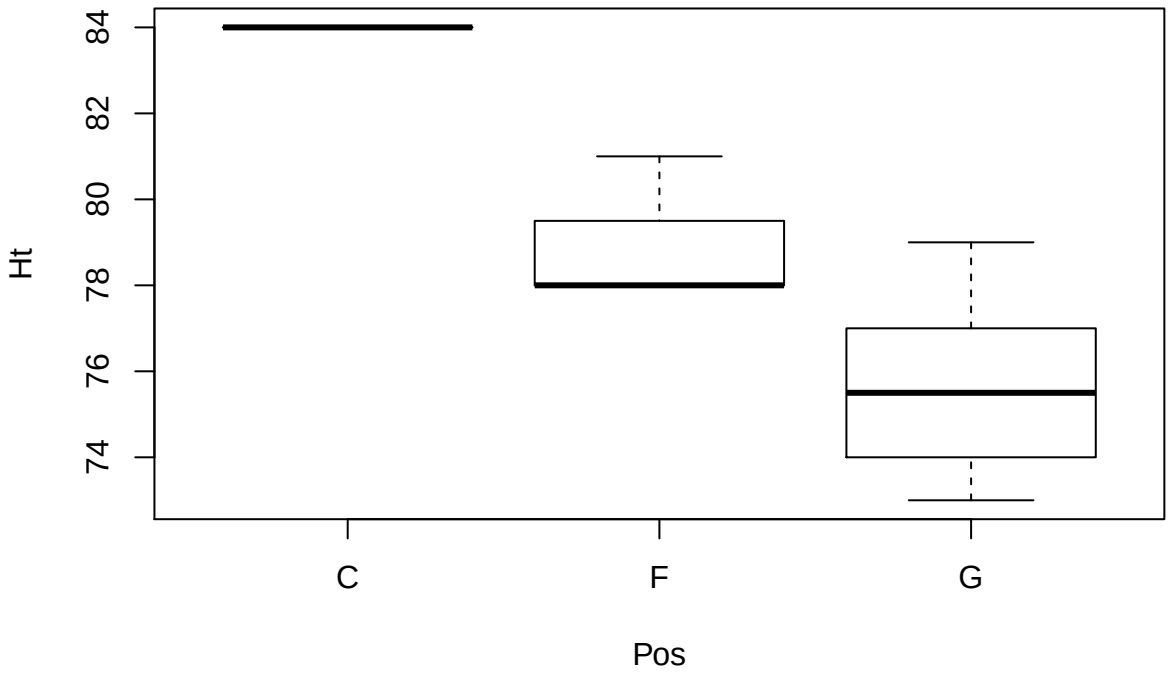
STAT 578 - Advanced Bayesian Modeling - Fall 2019
Assignment 6

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)

#Need only check top-level parameters (in the DAG) for convergence.
x1 <- coda.samples(model_1, c("beta_pos","beta_ht"), n.iter = 2000)
```

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

```
## Potential scale reduction factors:
##
##             Point est. Upper C.I.
## beta_ht              1       1.00
## beta_pos[1]          1       1.01
## beta_pos[2]          1       1.01
## beta_pos[3]          1       1.00
##
## Multivariate psrf
##
## 1
```

```
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
##    6477.117    5828.870    5433.513    5000.387
```

(b)

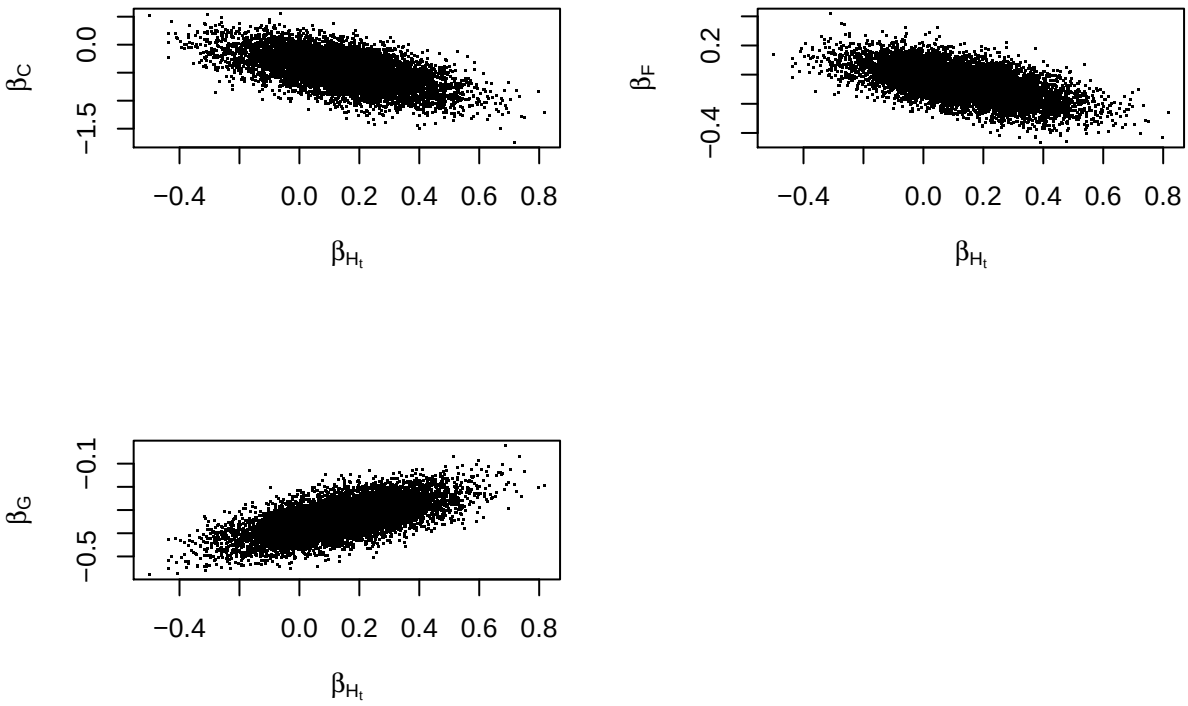
```
summary(coef_sample_1[, c("beta_pos[1]", "beta_pos[2]", "beta_pos[3]", "beta_ht")])
```

```
##
```

```
## Iterations = 4005:14000
## 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] -0.4524 0.2906 0.0032489      0.0036267
## beta_pos[2] -0.0606 0.1112 0.0012436      0.0014579
## beta_pos[3] -0.3346 0.0710 0.0007938      0.0009682
## beta_ht      0.1377 0.1789 0.0019998      0.0025543
##
## 2. Quantiles for each variable:
##
##                2.5%      25%      50%      75%   97.5%
## beta_pos[1] -1.0287 -0.64280 -0.44865 -0.25358  0.1061
## beta_pos[2] -0.2777 -0.13604 -0.05938  0.01606  0.1515
## beta_pos[3] -0.4733 -0.38276 -0.33470 -0.28653 -0.1964
## beta_ht     -0.2152  0.01372  0.13798  0.25776  0.4848
```

(c)

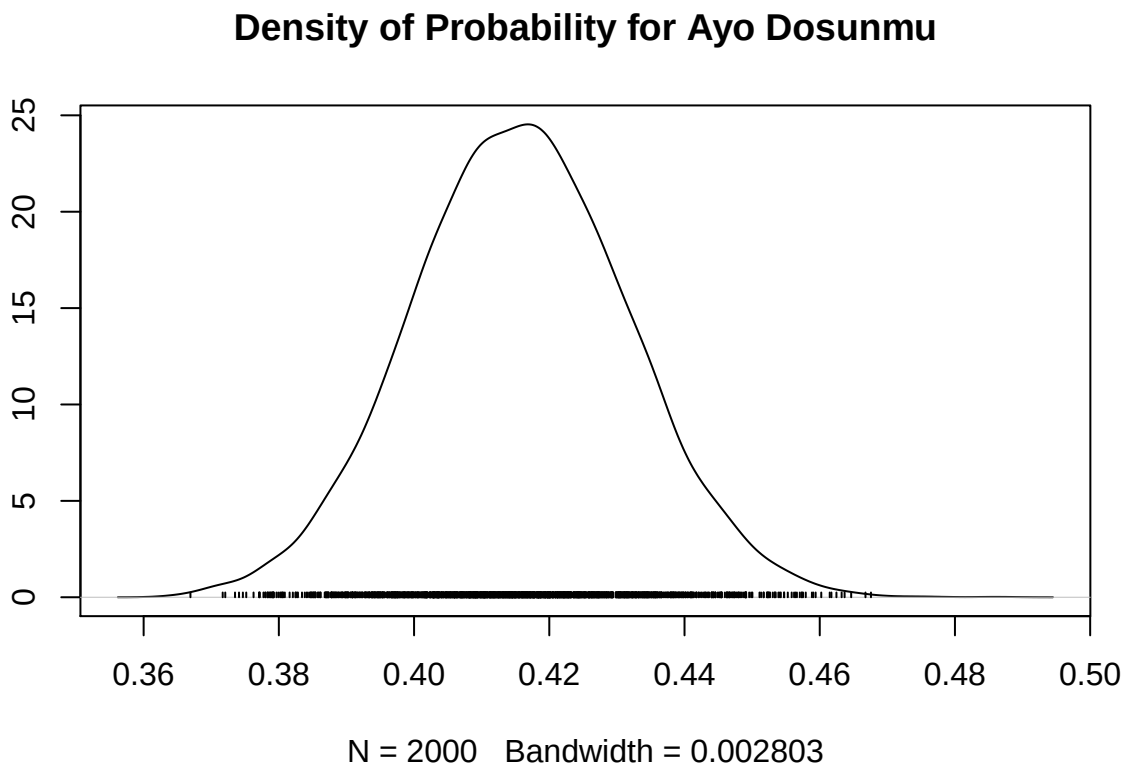
```
par(mfrow=c(2, 2))

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 Ayo Dosunmu")
```



(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] 0.9635
```

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

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

```
## [1] 26.39726
```

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] 0.049375
```

The posterior predictive p-value is small, although not exceedingly so. Given we don't find any outliers, we conclude that there is a problem 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", "sigma_epsilon"), n.iter = 20000)
```

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

```
## Potential scale reduction factors:
##
##               Point est. Upper C.I.
## beta_ht             1.00       1.01
## beta_pos[1]         1.00       1.01
```

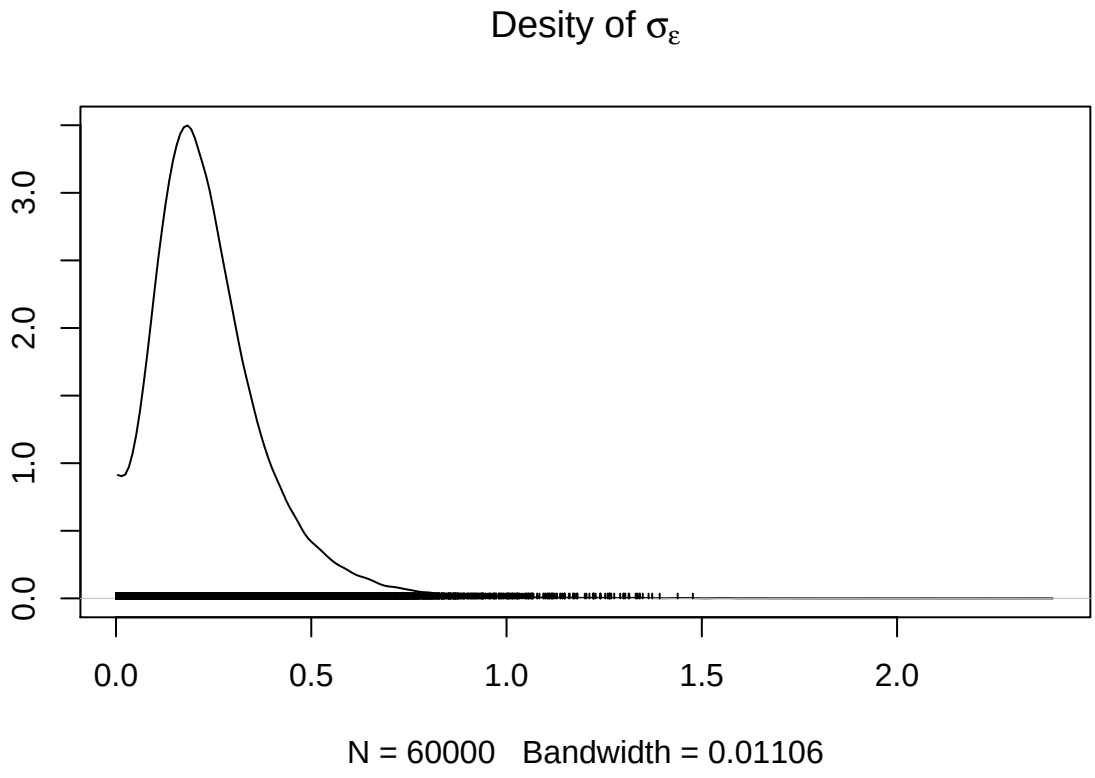
```
## beta_pos[2]          1.00       1.00
## beta_pos[3]          1.01       1.01
## sigma_epsilon        1.00       1.01
##
## Multivariate psrf
##
## 1.01
```

```
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
##      6303.863      5247.294      6967.143      4261.217      4139.574
```

(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] 0.7888375
```

This posterior probability is smaller than previous model.

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

```
## [1] 3.735689
```

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] 0.3781875
```

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"), n.iter = 2000)
```

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

```
## Potential scale reduction factors:
##
##             Point est. Upper C.I.
## beta_ht           1.00       1.01
## beta_pos[1]       1.01       1.01
```

```
## beta_pos[2]       1.00       1.01
## beta_pos[3]       1.00       1.00
##
## Multivariate psrf
##
## 1
```

```
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
##    4978.518    5321.187   10306.882    4715.428
```

(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] -5.304 0.6010 0.004751       0.008587
## beta_pos[2] -4.515 0.2838 0.002243       0.003930
## beta_pos[3] -4.449 0.1785 0.001411       0.001761
## beta_ht      1.011 0.2722 0.002152       0.003998
##
## 2. Quantiles for each variable:
##
##                2.5%    25%    50%    75%  97.5%
## beta_pos[1] -6.5089 -5.704 -5.290 -4.887 -4.153
## beta_pos[2] -5.0990 -4.703 -4.506 -4.323 -3.977
## beta_pos[3] -4.8113 -4.566 -4.445 -4.326 -4.113
## beta_ht      0.4937  0.827  1.005  1.192  1.559
```

(c)

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

```
##     2.5%    97.5%
## 1.638419 4.751957
```

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] 0.0074375
```

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 0.5950625
## 2            Cayce, Drew 1.0000000
## 3     De La Rosa, Adonis 0.9982500
## 4           Dosunmu, Ayo 0.7986250
## 5          Feliz, Andres 0.9553750
## 6         Frazier, Trent 0.9503125
## 7          Griffin, Alan 0.0218125
## 8         Griffith, Zach 1.0000000
## 9          Jones, Tevian 0.9768750
## 10         Jordan, Aaron 0.1902500
## 11           Kane, Samba 0.0043125
## 12       Nichols, Kipper 0.3206250
## 13     Oladimeji, Samson 1.0000000
## 14      Underwood, Tyler 1.0000000
## 15    Williams, Da'Monte 0.0885625
```

(ii)

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

```
##             name   p_value
```

```
## 7  Griffin, Alan 0.0218125
## 11   Kane, Samba 0.0043125
```

(iii)

```
p_df[p_df$p_value == 1,]
```

```
##                 name p_value
## 2        Cayce, Drew       1
## 8     Griffith, Zach       1
## 13 Oladimeji, Samson       1
## 14  Underwood, Tyler       1
```

By looking at the data, these players all got 0 shot blocks, since the y_i^{rep} can't be lower than 0, thus it must be greater or equal to y_i.