STAT 466 HW 9

Brody Anderson

2024-03-15

Question 1

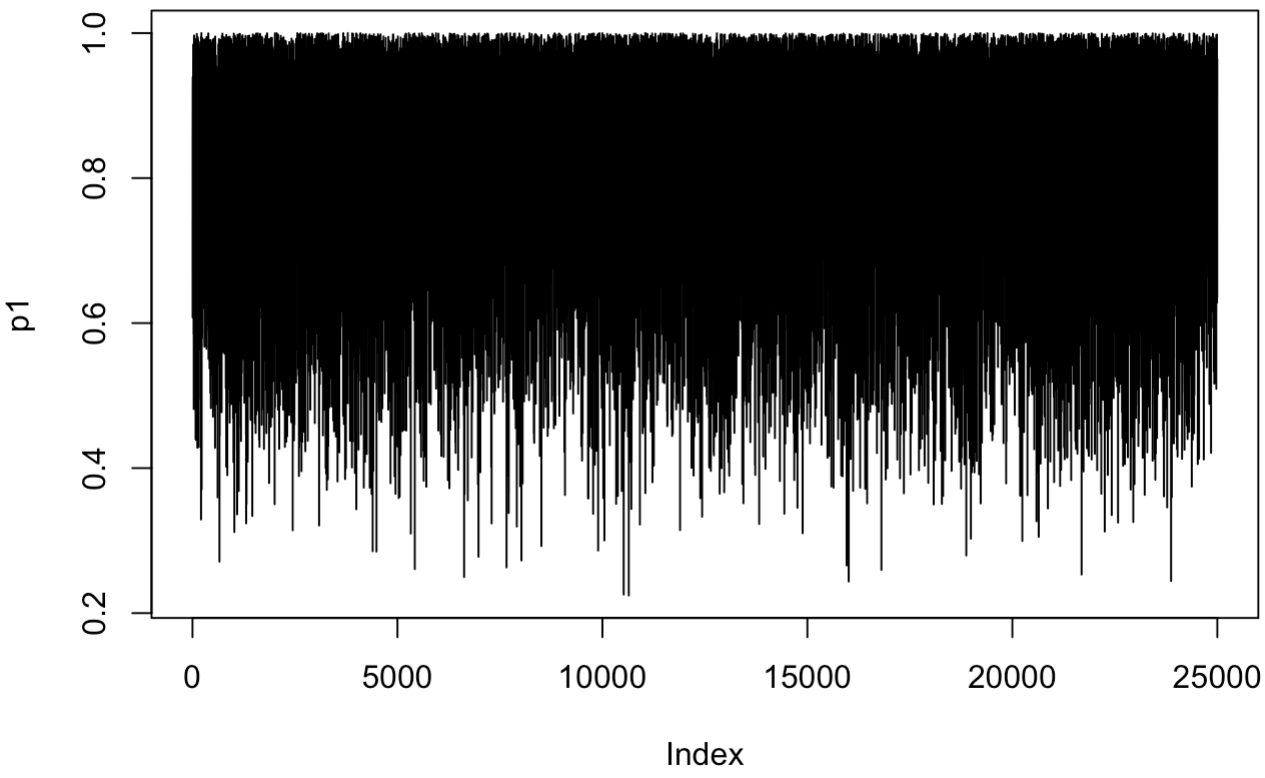
I read all of Chapter 7.

Question 2

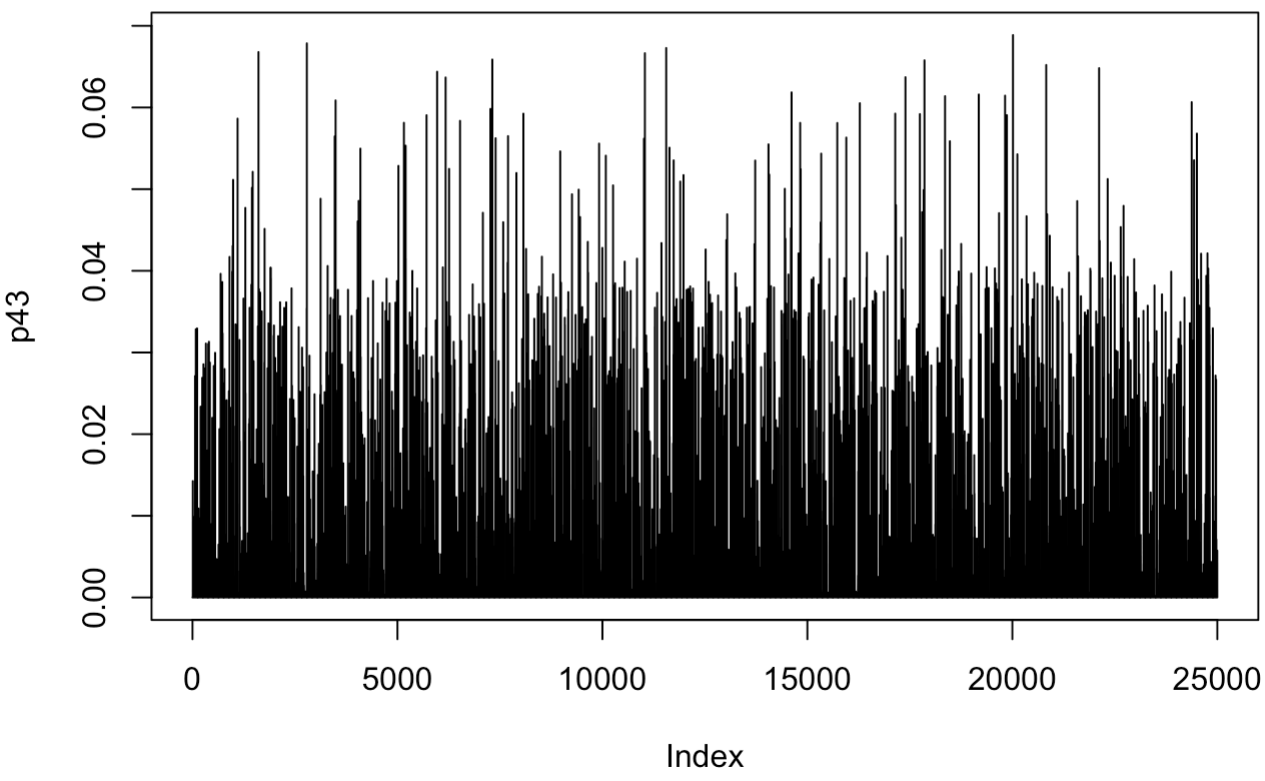
```
logit <- function(x) {log(x/(1-x))}
ilogit <- function(x) {1/(1+exp(-x))}

afw.data <- read.csv('Table10-8.csv')
#head(afw.data)
names(afw.data) <- c('Plant', 'failures', 'n')
#head(afw.data)

failures <- afw.data$failures
n <- afw.data$n

AFWModel <- "model {
  for(i in 1:68){
    failures[i] ~ dbin(p[i], n[i])
    p[i] ~ dbeta(alpha, beta)
  }
  alpha ~ dbeta(1,1)
  beta ~ dbeta(1,1)
}
"

AFWModel.sim <- jags(
  data=c('failures', 'n'),
  parameters.to.save=c('p', 'beta', 'alpha'),
  model.file=textConnection(AFWModel),
  n.iter=12000,
  n.burnin=2000,
  n.chains=5,
  n.thin=2
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 68
##    Unobserved stochastic nodes: 70
##    Total graph size: 207
##
## Initializing model
```

```
#head(AFWModel.sim$BUGSoutput$sims.matrix)
chains <- AFWModel.sim$BUGSoutput$sims.matrix[,c('p[1]', 'p[44]', 'deviance')]
#head(chains)
chains <- as.mcmc(chains)

p1 <- AFWModel.sim$BUGSoutput$sims.matrix[,2]
p43 <- AFWModel.sim$BUGSoutput$sims.matrix[,44]

plot(p1,type='l')
```




```
plot(p43,type='l')
```




```
#acf(p1)
#acf(p2)

effectiveSize(chains)
```

```
##     p[1]    p[44] deviance
## 25000.00 24246.98 25000.00
```

```
gelman.diag(AFWModel.sim$BUGSoutput)
```

```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## alpha          1.48       2.12
## beta           1.01       1.02
## deviance       1.01       1.04
## p[1]           1.13       1.31
## p[10]          1.11       1.23
## p[11]          1.35       3.00
## p[12]          1.16       1.33
## p[13]          1.13       1.22
## p[14]          4.92      25.50
## p[15]          1.47       5.01
## p[16]          1.38       3.64
## p[17]          1.24       1.62
## p[18]          1.00       1.00
## p[19]          1.10       1.14
## p[2]           1.26       1.78
## p[20]          1.15       1.25
## p[21]          1.38       3.62
## p[22]          1.21       1.43
## p[23]          1.23       1.64
## p[24]          1.27       1.75
## p[25]          1.21       1.41
## p[26]          1.00       1.00
## p[27]          1.00       1.00
## p[28]          1.29       2.02
## p[29]          1.11       1.18
## p[3]           1.86       6.50
## p[30]          2.46       9.28
## p[31]          1.22       1.54
## p[32]          1.00       1.00
## p[33]          1.11       1.22
## p[34]          1.25       1.65
## p[35]          1.31       2.36
## p[36]          1.28       2.01
## p[37]          1.30       2.20
## p[38]          1.31       2.19
## p[39]          1.24       1.58
## p[4]           1.16       1.31
## p[40]          1.38       2.84
## p[41]          1.14       1.25
## p[42]          1.20       1.46
## p[43]          1.33       2.69
## p[44]          2.28       5.33
## p[45]          1.31       2.25
## p[46]          1.28       1.70
## p[47]          1.00       1.00
## p[48]          1.28       2.01
## p[49]          1.40       3.55
## p[5]           1.27       1.70
## p[50]          1.32       2.39
## p[51]          1.22       1.58
## p[52]          1.10       1.23
## p[53]          1.25       1.67
## p[54]          1.19       1.28
## p[55]          1.24       1.58
## p[56]          1.20       1.47
## p[57]          1.26       1.69
## p[58]          1.51       4.44
## p[59]          1.35       2.72
## p[6]           1.27       1.86
## p[60]          1.11       1.15
## p[61]          1.23       1.54
## p[62]          1.33       2.56
## p[63]          1.30       2.13
## p[64]          1.35       2.99
## p[65]          1.15       1.25
## p[66]          1.12       1.19
## p[67]          1.32       2.45
## p[68]          1.32       2.39
## p[7]           1.36       2.65
## p[8]           2.83       5.81
## p[9]           1.13       1.21
##
## Multivariate psrf
##
## 5.12
```

```
AFWModel.sim$BUGSoutput$DIC
```

```
## [1] 24.54216
```

```
mcmcChains <- AFWModel.sim$BUGSoutput$sims.matrix[,2:69]

quantile(p1,c(0.025,0.975))
```

```
##      2.5%     97.5%
## 0.4900023 0.9942941
```

```
quantile(p43,c(0.025,0.975))
```

```
##         2.5%        97.5%
## 7.202704e-63 2.571413e-02
```

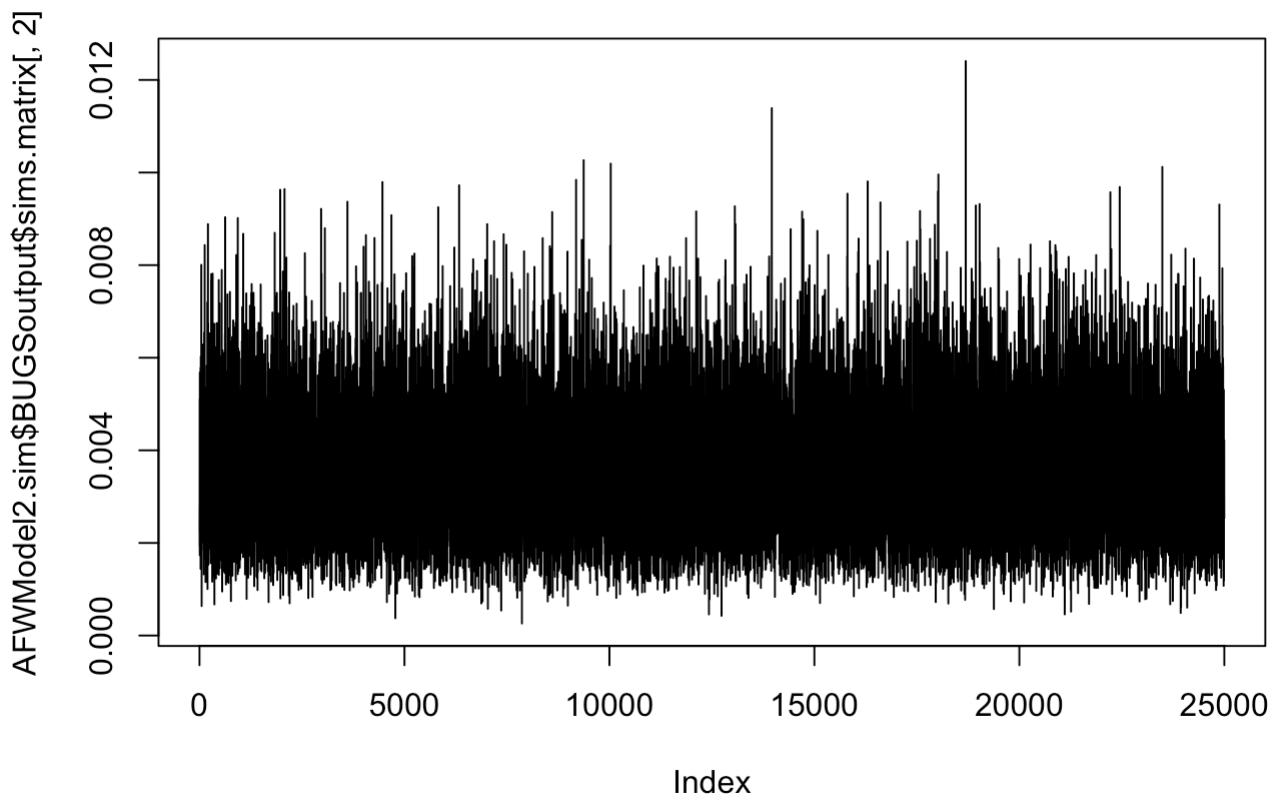
```
GoF <- matrix(NA,ncol=length(failures),nrow=length(mcmcChains[,1]))
for (i in 1:length(mcmcChains[,1])) {
  GoF[i,] <- runif(length(failures),pbinom(failures-1, 68, mcmcChains[i,]),pbinom(failures, 68,mcmcChains[i,]))
}

# Function requires fitted quantiles and returns a p-value
GoF_Test <- function(fitted_quantiles) {
  n <- length(fitted_quantiles)
  K <- round((n)^(0.4))
  mK <- table(cut(fitted_quantiles,(0:K)/K))
  np <- n/K
  RB <- sum(((mK-np)^2)/np)
  return(1-pchisq(RB,K-1))
}

# Calculating the p-values for each posterior model
GoF_Summary <- apply(GoF,1,GoF_Test)

# Histogram of posterior model p-values
#hist(GoF_Summary,xlim=c(0,1))

# Percent of posterior models with p-value less than 0.05
mean(GoF_Summary < 0.05)
```

```
## [1] 0.14624
```

```
#######

AFWModel2 <- "model {
  for(i in 1:68){
    failures[i] ~ dbin(pi, n[i])

  }
  pi ~ dbeta(1,1)
}
"

AFWModel2.sim <- jags(
  data=c('failures', 'n'),
  parameters.to.save=c('pi'),
  model.file=textConnection(AFWModel2),
  n.iter=12000,
  n.burnin=2000,
  n.chains=5,
  n.thin=2
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 68
##    Unobserved stochastic nodes: 1
##    Total graph size: 138
##
## Initializing model
```

```
#head(AFWModel2.sim$BUGSoutput$sims.matrix)
plot(AFWModel2.sim$BUGSoutput$sims.matrix[,2],type='l')
```




```
effectiveSize(AFWModel2.sim$BUGSoutput$sims.matrix)
```

```
## deviance       pi
## 24777.23 25197.89
```

```
AFWModel2.sim$BUGSoutput$DIC
```

```
## [1] 48.07058
```

```
#mcmcChains <- as.mcmc(AFWModel2.sim$BUGSoutput$sims.matrix)

#mean(AFWModel2.sim$BUGSoutput$sims.matrix[,2])

#GoF <- matrix(NA,ncol=length(failures),nrow=length(mcmcChains[,1]))
#for (i in 1:length(mcmcChains[,1])) {
#  GoF[i,] <- runif(length(failures),pbinom(failures-1, 68, mcmcChains[i,2]),pbinom(failures, 68,mcmcChains[i,
2]))
#}

# Function requires fitted quantiles and returns a p-value
GoF_Test <- function(fitted_quantiles) {
  n <- length(fitted_quantiles)
  K <- round((n)^(0.4))
  mK <- table(cut(fitted_quantiles,(0:K)/K))
  np <- n/K
  RB <- sum(((mK-np)^2)/np)
  return(1-pchisq(RB,K-1))
}

# Calculating the p-values for each posterior model
#GoF_Summary <- apply(GoF,1,GoF_Test)

# Histogram of posterior model p-values
#hist(GoF_Summary,xlim=c(0,1))

# Percent of posterior models with p-value less than 0.05
#mean(GoF_Summary < 0.05)
```

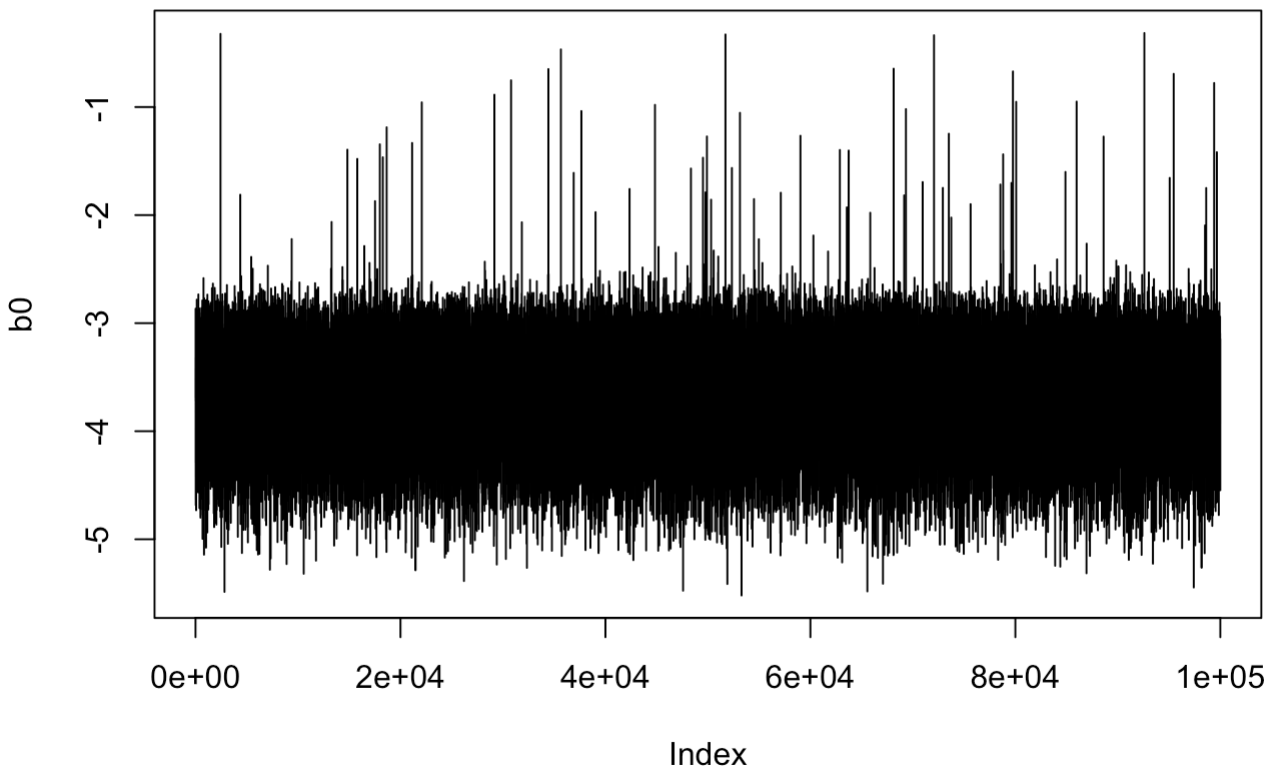
The hierarchical binomial model does not fit the data very well based on a Bayesian χ2 goodness-of-fit test. However, the DIC is still fairly low. Although the DIC really isn't helpful unless we use it compared to another model. So, I created another model and the hierarchical model had a lower DIC.

Question 3

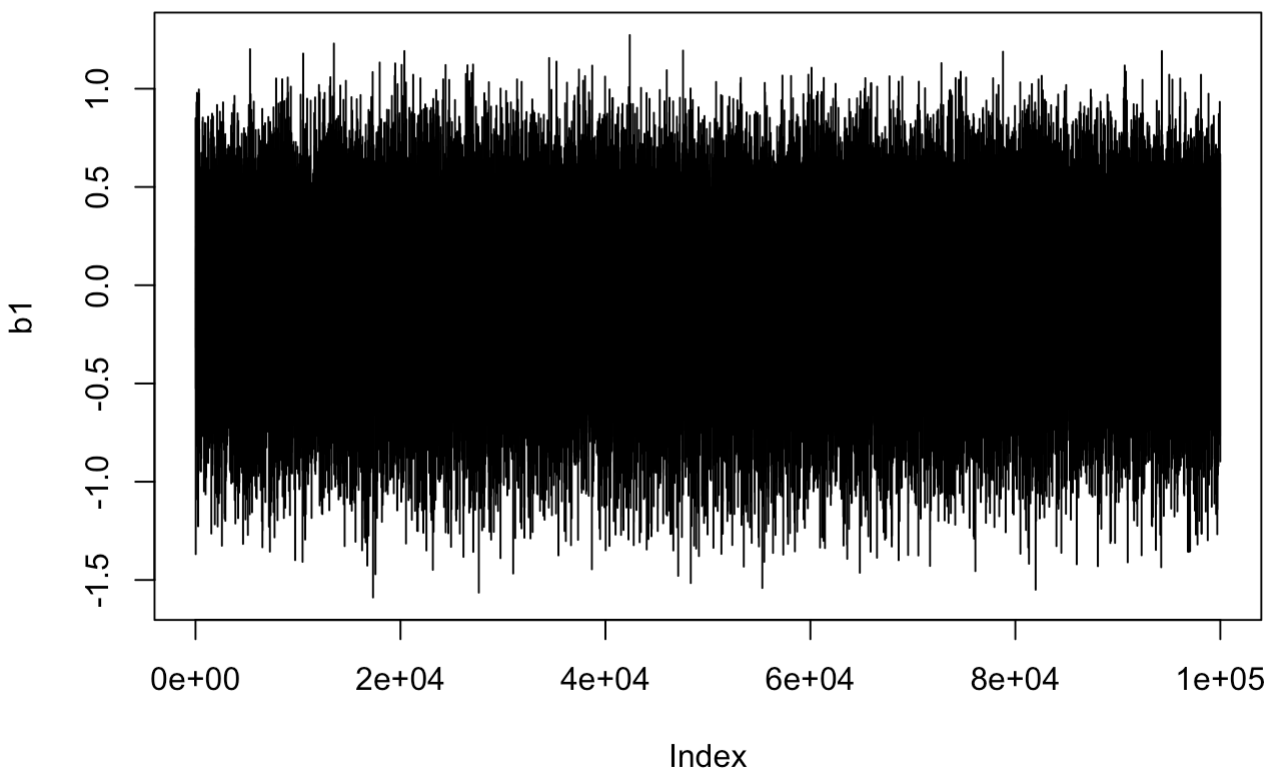
```
########## Centered Model
count <- c(0,0,1,3,0,0,4,0,0,2,0,0,0,0,1,0,0,0,1,0)
total <- c(10,31,56,13,17,43,44,1,7,33,21,1,12,31,22,1,9,19,16,1)
soak_time <- c(1,1,1,1,1.7,1.7,1.7,1.7,2.2,2.2,2.2,2.2,2.8,2.8,2.8,2.8,4,4,4,4)
heat_time <- c(7,14,27,51,7,14,27,51,7,14,27,51,7,14,27,51,7,14,27,51)
prob <- count / total
# similar to non-parametric procedures from STAT 435, use avgs. and data around a 'non-response'
# to inform what would have been there

ingot.data <- cbind(soak_time, heat_time, count, total, prob)
# un-centered model didn't fit well

soak_timec <- soak_time - mean(soak_time)
heat_timec <- heat_time - mean(heat_time)

CIngotModel <- "model {
  for(i in 1:20){
    count[i] ~ dbin(pi[i], total[i])
    logit(pi[i]) <- beta[1] + beta[2]*soak_timec[i] + beta[3]*heat_timec[i]
                        + beta[4]*soak_timec[i]*heat_timec[i]
  }
  beta[1] ~ dnorm(0,1/100)
  beta[2] ~ dnorm(0,1/100)
  beta[3] ~ dnorm(0,1/100)
  beta[4] ~ dnorm(0,1/100)
}
"

CIngotModel.sim <- jags(
  data=c('soak_timec','heat_timec', 'total', 'count'),
  parameters.to.save=c('beta'),
  model.file=textConnection(CIngotModel),
  n.iter=22000,
  n.burnin=2000,
  n.chains=5,
  n.thin=1
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 20
##    Unobserved stochastic nodes: 4
##    Total graph size: 157
##
## Initializing model
```

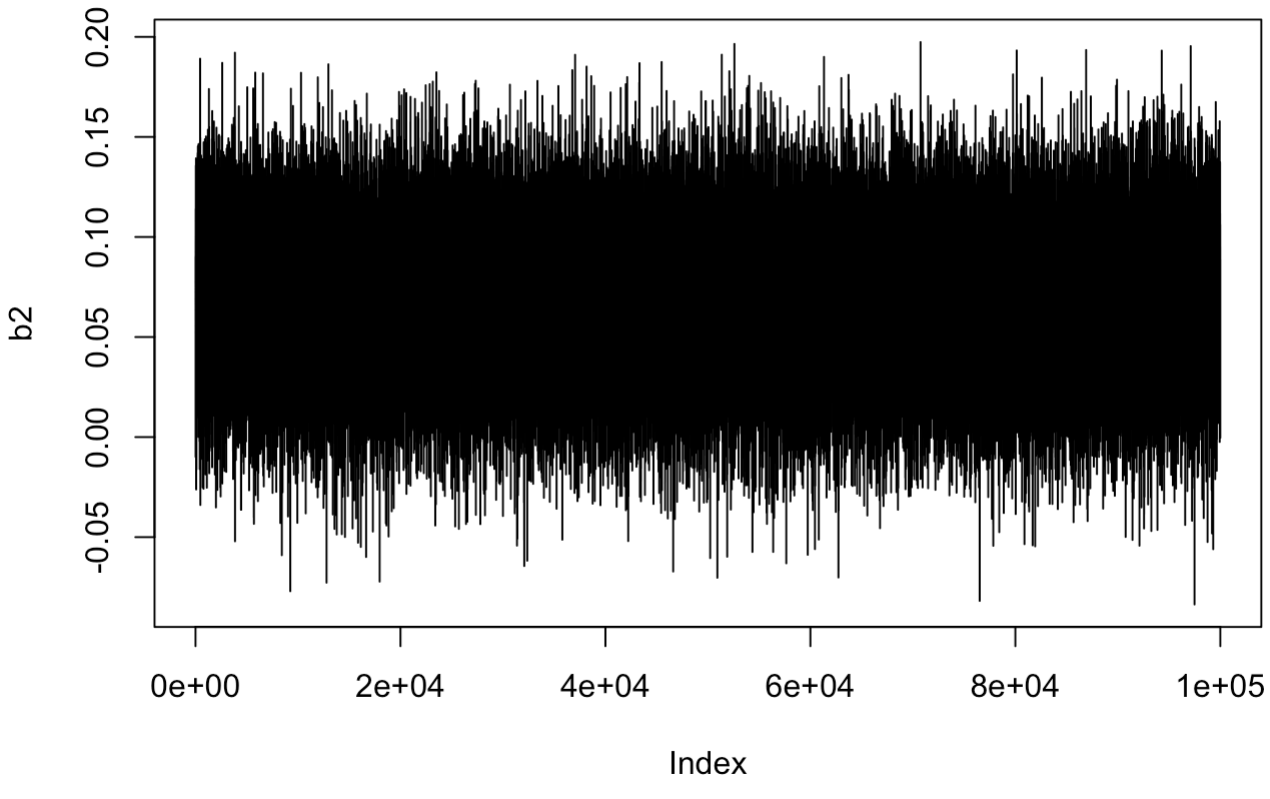
```
#head(CIngotModel.sim$BUGSoutput$sims.matrix)

b0 <- CIngotModel.sim$BUGSoutput$sims.matrix[,1] # intercept
b1 <- CIngotModel.sim$BUGSoutput$sims.matrix[,2] # soak time
b2 <- CIngotModel.sim$BUGSoutput$sims.matrix[,3] # heat time
b3 <- CIngotModel.sim$BUGSoutput$sims.matrix[,4] # interaction

plot(b0,type="l")
```



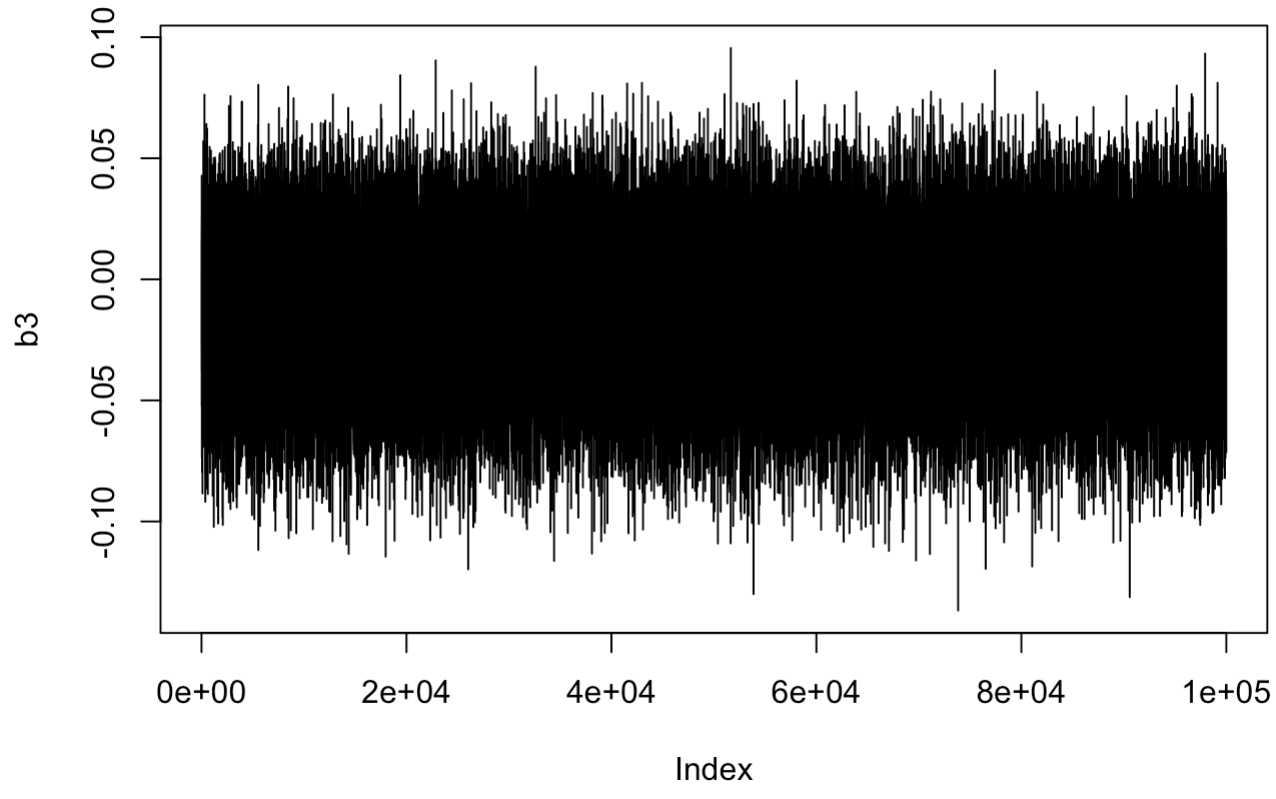

```
plot(b1,type="l")
```




```
plot(b2,type="l")
```



```
plot(b3,type="l")
```




```
#acf(b0)
#acf(b1)
#acf(b2)
#acf(b3)

effectiveSize(CIngotModel.sim)
```

```
##  beta[1]  beta[2]  beta[3]  beta[4] deviance
## 1343.134 1772.622 3299.745 4262.223 4872.826
```

```
gelman.diag(CIngotModel.sim$BUGSoutput)
```

```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## beta[1]        1.01       1.04
## beta[2]        1.00       1.01
## beta[3]        1.00       1.01
## beta[4]        1.00       1.01
## deviance       1.00       1.01
##
## Multivariate psrf
##
## 1.03
```

```
CIngotModel.sim$BUGSoutput$DIC
```

```
## [1] 43.53115
```

```
#plot(b0,b1,pch='.')
#cor(b0,b1)
#plot(b0,b2,pch='.')
#cor(b0,b2)
#plot(b0,b3,pch='.')
#cor(b0,b3)
#plot(b1,b2,pch='.')
#cor(b1,b2)
#plot(b1,b3,pch='.')
#cor(b1,b3)
#plot(b2,b3,pch='.')
#cor(b2,b3)

#plot(b0,b1,pch='.')

GoF <- matrix(NA,ncol=length(count),nrow=length(b0))
for (i in 1:length(b0)) {
  GoF[i,] <- runif(length(count),
                   pbinom(count-1,1,ilogit(b0[i]+ (b1[i]*soak_timec) + (b2[i]*heat_timec) + (b3[i]*soak_timec*hea
t_timec))),
                   pbinom(count,1,ilogit(b0[i]+ (b1[i]*soak_timec) + (b2[i]*heat_timec) + (b3[i]*soak_timec*heat_
timec)))
  )
}

# Function requires fitted quantiles and returns a p-value
GoF_Test <- function(fitted_quantiles) {
  n <- length(fitted_quantiles)
  K <- round((n)^(0.4))
  mK <- table(cut(fitted_quantiles,(0:K)/K))
  np <- n/K
  RB <- sum(((mK-np)^2)/np)
  return(1-pchisq(RB,K-1))
}


# Calculating the p-values for each posterior model
GoF_Summary <- apply(GoF,1,GoF_Test)

# Histogram of posterior model p-values
#hist(GoF_Summary,xlim=c(0,1))

# Percent of posterior models with p-value less than 0.05
mean(GoF_Summary < 0.05)
```

```
## [1] 0.22903
```

```
#beta0 <- rnorm(100000,0,sqrt(1000))
#beta1 <- rnorm(100000,0,sqrt(1000))
#beta2 <- rnorm(100000,0,sqrt(1000))
#beta3 <- rnorm(100000,0,sqrt(1000))
#plot(beta0,beta1,pch='.',ylim=c(-1,.1),xlim=c(-10,55))
#par(new=T)
#plot(b0,b1,pch='.',ylim=c(-10,10),xlim=c(-10,10),col=2,xlab='',ylab='')

#hist(b1)
mean(b1<0)
```

```
## [1] 0.52192
```

```
#hist(b2)
#mean(b2<0)

#hist(b3)
#mean(b3<0)

#hist(b0)
```

What do you conclude about the impact of these factors on the probability of an ingot not being ready for rolling?

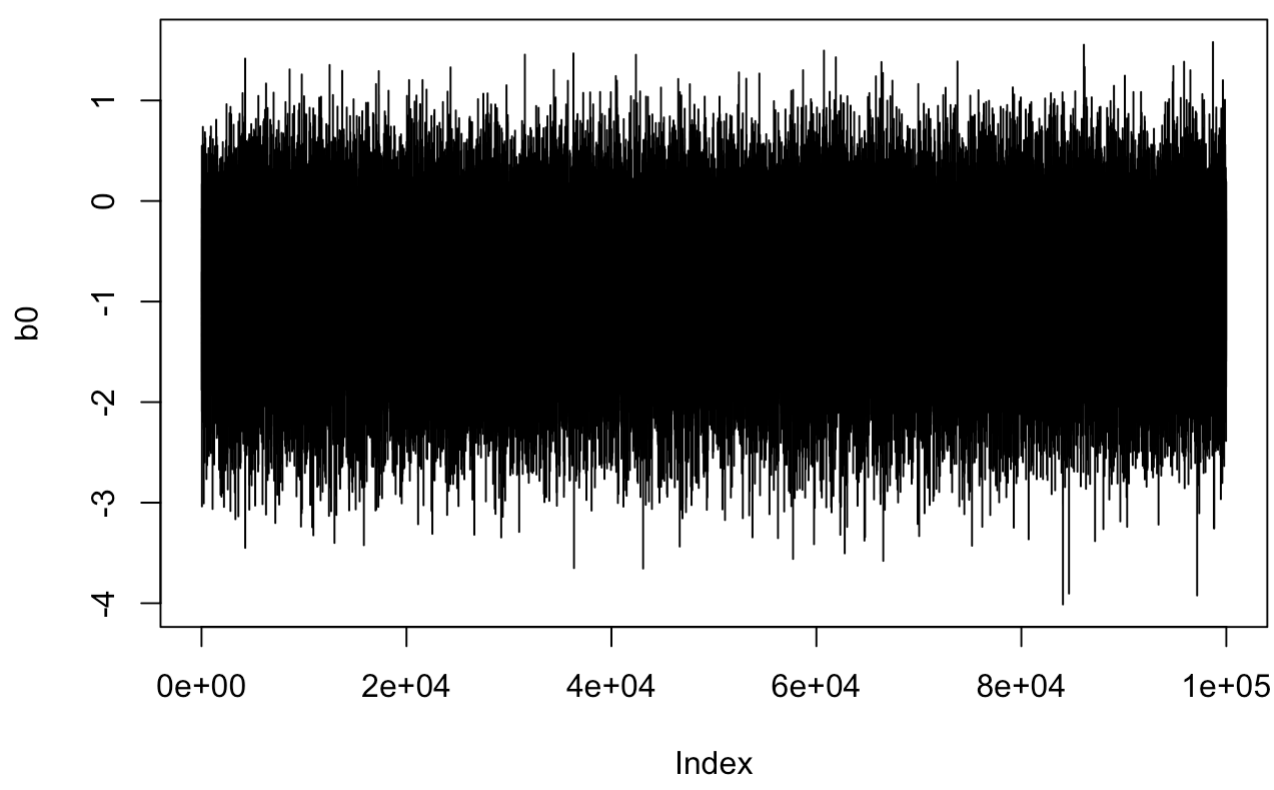
While the model isn't as great as I would have liked, we can see that soak time (centered soak time actually), has the largest impact about the probability of an ingot being ready for rolling. Also, the interaction between the two covariates is impactful as well.

Question 4

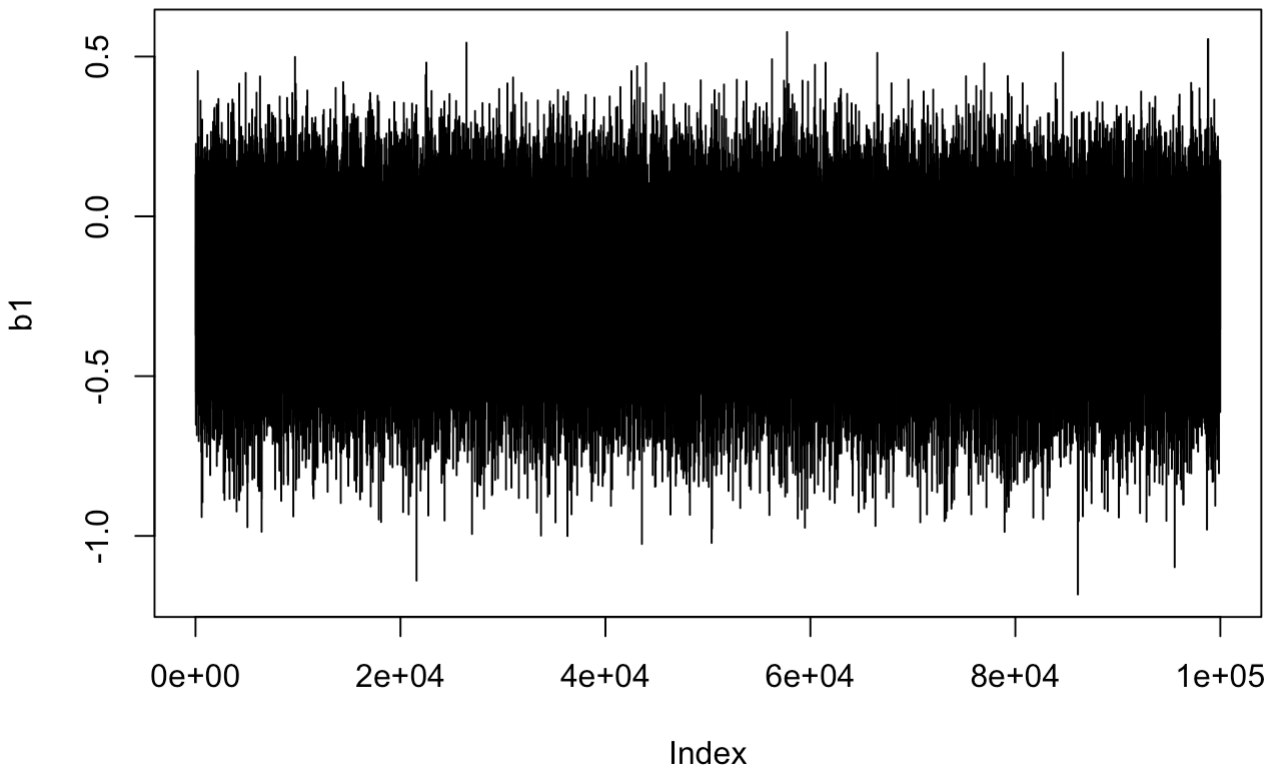
```
failures <- c(4,2,1,3,2,0,0)
demands <- c(16,10,7,13,9,6,2)
time <- c(1,2,3,4,5,6,7)

HPCIModel <- "model {
  for(i in 1:7){
    failures[i] ~ dbin(pi[i], demands[i])
    logit(pi[i]) <- beta[1] + beta[2]*time[i]
  }
  beta[1] ~ dnorm(0,1/100)
  beta[2] ~ dnorm(0,1/100)

}
"

HPCIModel.sim <- jags(
  data=c('failures', 'demands', 'time'),
  parameters.to.save=c('beta', 'pi'),
  model.file=textConnection(HPCIModel),
  n.iter=22000,
  n.burnin=2000,
  n.chains=5,
  n.thin=1
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 7
##    Unobserved stochastic nodes: 2
##    Total graph size: 48
##
## Initializing model
```

```
#head(HPCIModel.sim$BUGSoutput$sims.matrix)

b0 <- HPCIModel.sim$BUGSoutput$sims.matrix[,1] #intercept
b1 <- HPCIModel.sim$BUGSoutput$sims.matrix[,2] #slope
pi1 <- HPCIModel.sim$BUGSoutput$sims.matrix[,2]

plot(b0,type="l")
```




```
plot(b1,type="l")
```




```
#acf(b0)
#acf(b1)

quantile(b0, c(0.025, 0.05, 0.5, 0.095, 0.975))
```

```
##       2.5%         5%        50%       9.5%      97.5% 
## -2.1734638 -1.9432824 -0.8527053 -1.7145827  0.3494690
```

```
quantile(b1, c(0.025, 0.05, 0.5, 0.095, 0.975))
```

```
##       2.5%         5%        50%       9.5%      97.5% 
## -0.6114291 -0.5415150 -0.2077868 -0.4699142  0.1538701
```

```
HPCIModel.sim$BUGSoutput$DIC
```

```
## [1] 19.37929
```

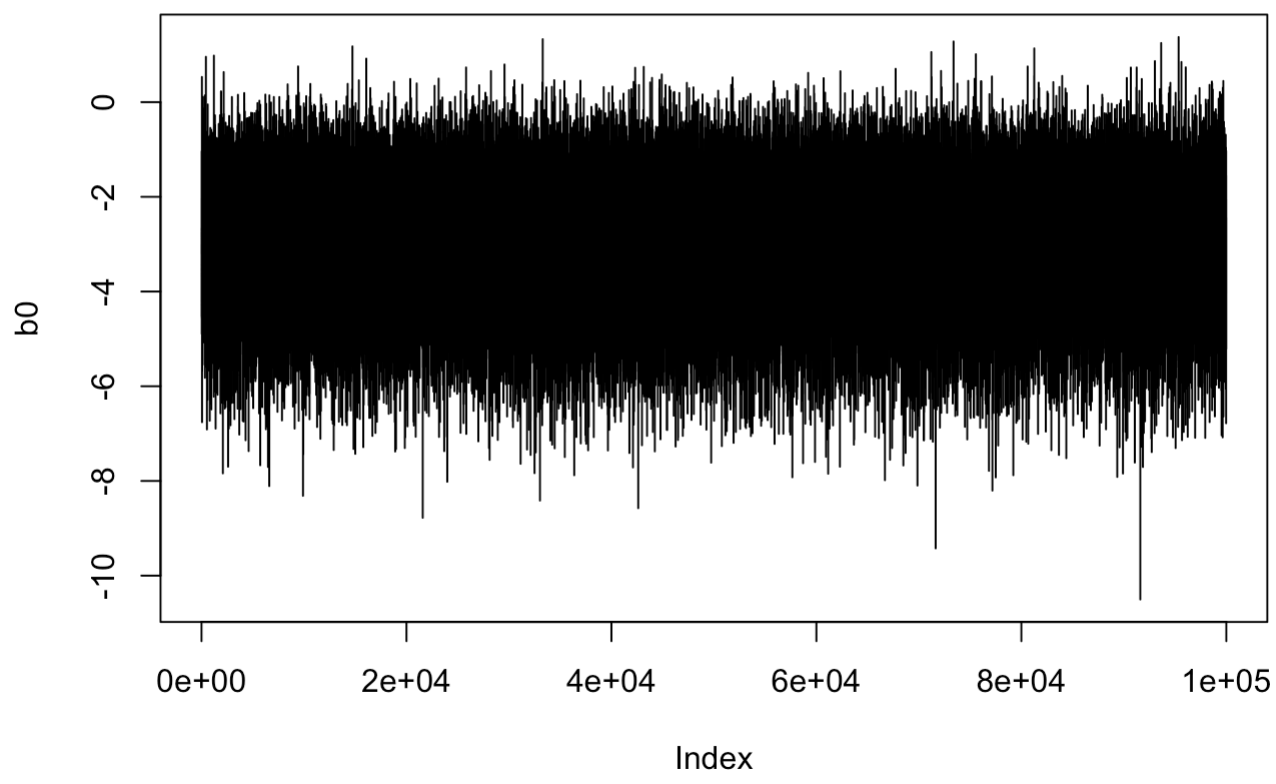
```
#Looking at how b0 and b1 are correlated
#plot(b0,b1,pch='.')
#cor(b0,b1)


GoF <- matrix(NA,ncol=length(failures),nrow=length(b0))
for (i in 1:length(b0)) {
  GoF[i,] <- runif(length(failures),
                   pbinom(failures-1,1,ilogit(b0[i]+b1[i]*(time))),
                   pbinom(failures,1,ilogit(b0[i]+b1[i]*(time)))
  )
}

# Function requires fitted quantiles and returns a p-value
GoF_Test <- function(fitted_quantiles) {
  n <- length(fitted_quantiles)
  K <- round((n)^(0.4))
  mK <- table(cut(fitted_quantiles,(0:K)/K))
  np <- n/K
  RB <- sum(((mK-np)^2)/np)
  return(1-pchisq(RB,K-1))
}

# Calculating the p-values for each posterior model
GoF_Summary <- apply(GoF,1,GoF_Test)

# Histogram of posterior model p-values
#hist(GoF_Summary,xlim=c(0,1))

# Percent of posterior models with p-value less than 0.05
mean(GoF_Summary < 0.05)
```

```
## [1] 0.18659
```

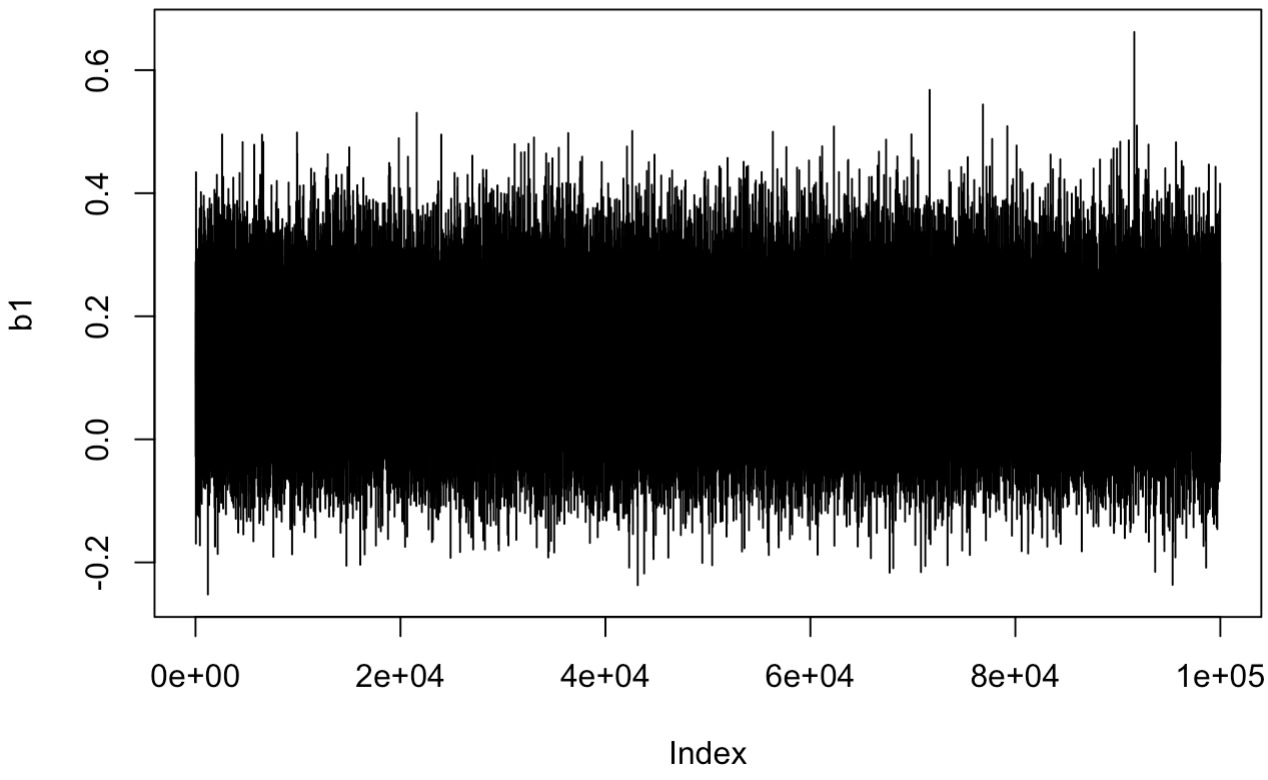
```
####################
HPCIModel <- "model {
  for(i in 1:7){
    failures[i] ~ dbin(pi[i], demands[i])
    logit(pi[i]) <- beta[1] + beta[2]*demands[i]
  }
  beta[1] ~ dnorm(0,1/100)
  beta[2] ~ dnorm(0,1/100)

}
"

HPCIModel.sim <- jags(
  data=c('failures', 'demands'),
  parameters.to.save=c('beta', 'pi'),
  model.file=textConnection(HPCIModel),
  n.iter=22000,
  n.burnin=2000,
  n.chains=5,
  n.thin=1
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 7
##    Unobserved stochastic nodes: 2
##    Total graph size: 41
##
## Initializing model
```

```
#head(HPCIModel.sim$BUGSoutput$sims.matrix)

b0 <- HPCIModel.sim$BUGSoutput$sims.matrix[,1] #intercept
b1 <- HPCIModel.sim$BUGSoutput$sims.matrix[,2] #slope

plot(b0,type="l")
```




```
plot(b1,type="l")
```




```
#acf(b0)
#acf(b1)

quantile(b0, c(0.025, 0.05, 0.5, 0.095, 0.975))
```

```
##       2.5%         5%        50%       9.5%      97.5% 
## -5.4077615 -4.9524466 -2.8721752 -4.4941992 -0.8141048
```

```
quantile(b1, c(0.025, 0.05, 0.5, 0.095, 0.975))
```

```
##         2.5%           5%          50%         9.5%        97.5% 
## -0.051987171 -0.024443068  0.120437231  0.004725935  0.311466401
```

```
HPCIModel.sim$BUGSoutput$DIC
```

```
## [1] 18.67491
```

```
#Looking at how b0 and b1 are correlated
#plot(b0,b1,pch='.')
#cor(b0,b1)


GoF <- matrix(NA,ncol=length(failures),nrow=length(b0))
for (i in 1:length(b0)) {
  GoF[i,] <- runif(length(failures),
                   pbinom(failures-1,1,ilogit(b0[i]+b1[i]*(demands))),
                   pbinom(failures,1,ilogit(b0[i]+b1[i]*(demands)))
  )
}

# Function requires fitted quantiles and returns a p-value
GoF_Test <- function(fitted_quantiles) {
  n <- length(fitted_quantiles)
  K <- round((n)^(0.4))
  mK <- table(cut(fitted_quantiles,(0:K)/K))
  np <- n/K
  RB <- sum(((mK-np)^2)/np)
  return(1-pchisq(RB,K-1))
}

# Calculating the p-values for each posterior model
GoF_Summary <- apply(GoF,1,GoF_Test)

# Histogram of posterior model p-values
#hist(GoF_Summary,xlim=c(0,1))

# Percent of posterior models with p-value less than 0.05
mean(GoF_Summary < 0.05)
```

```
## [1] 0.19628
```

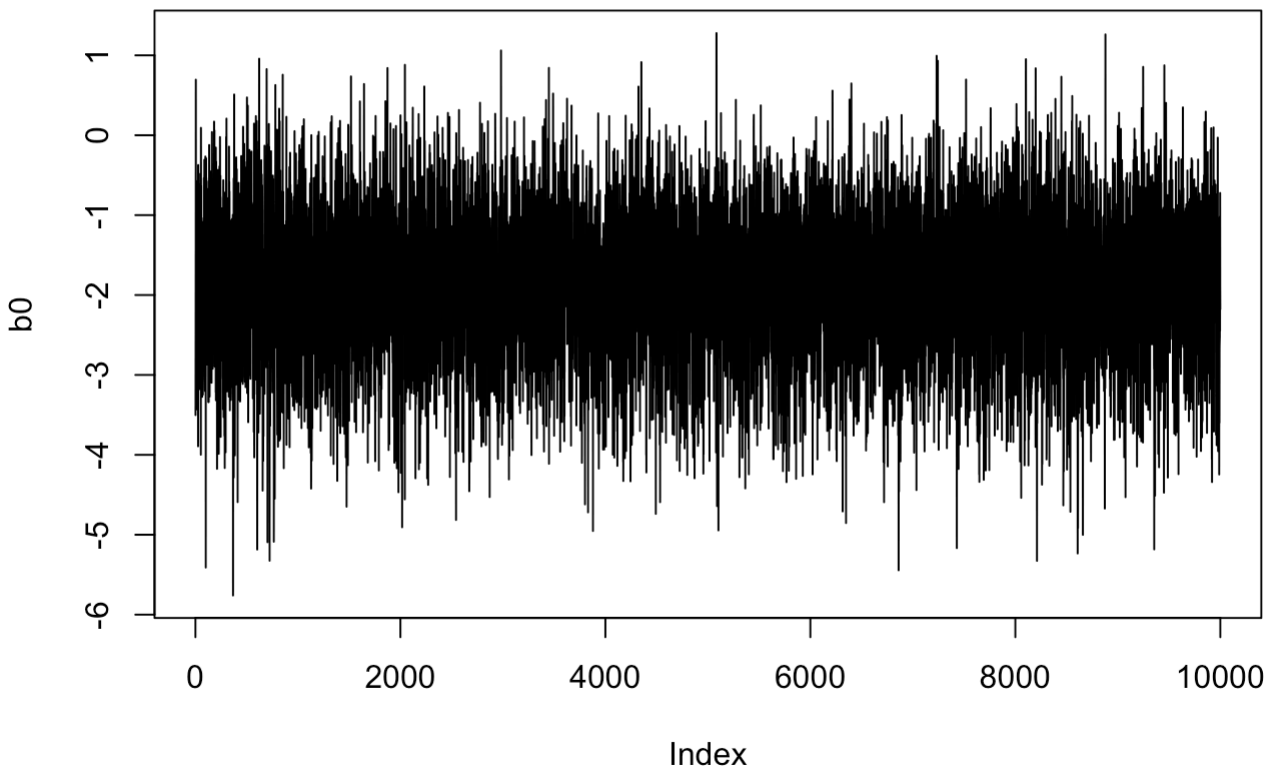
```
# Comparing the posterior stucture with the independent priors
beta0 <- rnorm(100000,0,sqrt(1000))
beta1 <- rnorm(100000,0,sqrt(1000))

#plot(beta0,beta1,pch='.',ylim=c(-1,1),xlim=c(-30,50))
#par(new=T)
#plot(b0,b1,pch='.',ylim=c(-1,1),xlim=c(-30,50),col=2,xlab='',ylab='')




############ Informative Priors
y <- matrix(c(1,1,2,16), ncol=2)
solve(y)
```

```
##             [,1]        [,2]
## [1,]  1.14285714 -0.14285714
## [2,] -0.07142857  0.07142857
```

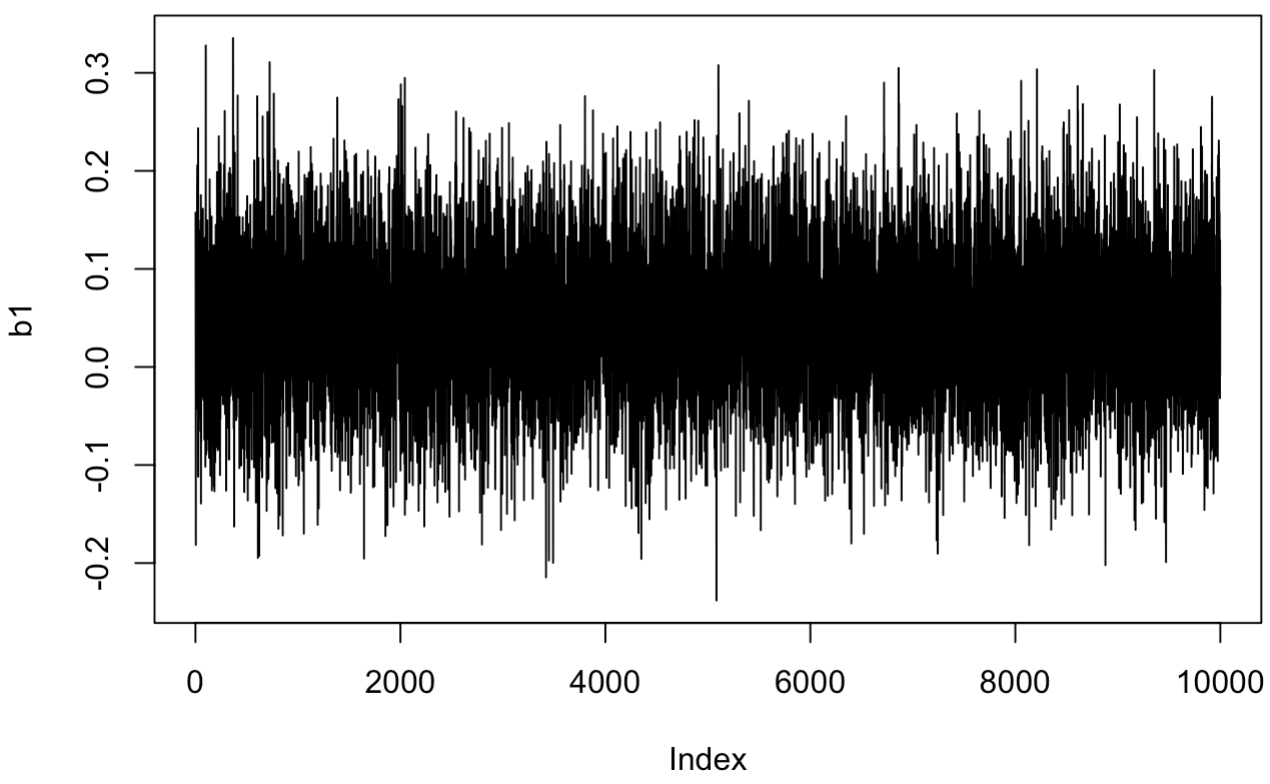
```
pi2 <- rbeta(100000,1.6,1)
pi16 <- rbeta(100000,1,1.6)
beta0 <- (1.14285714)*logit(pi2)-(0.14285714)*logit(pi16)
beta1 <- (0.07142857)*logit(pi16)-(0.07142857)*logit(pi2)

HPCIModel2 <- "model {
  for(i in 1:7){
    failures[i] ~ dbin(pi[i], demands[i])
    logit(pi[i]) <- beta[1] + beta[2]*demands[i]
  }

  beta[1] <- (1.14285714)*logit(pi2)-(0.14285714)*logit(pi16)
  beta[2] <- (0.07142857)*logit(pi16)-(0.07142857)*logit(pi2)
  pi2 ~ dbeta(1.6,1)
  pi16 ~ dbeta(1,1.6)

}
"

HPCIModel2.sim <- jags(
  data=c('failures','demands'),
  parameters.to.save=c('beta'),
  model.file=textConnection(HPCIModel2),
  n.iter=102000,
  n.burnin=2000,
  n.chains=1,
  n.thin=10
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 7
##    Unobserved stochastic nodes: 2
##    Total graph size: 50
##
## Initializing model
```

```
#head(HPCIModel2.sim$BUGSoutput$sims.matrix)

b0 <- HPCIModel2.sim$BUGSoutput$sims.matrix[,1]
b1 <- HPCIModel2.sim$BUGSoutput$sims.matrix[,2]
#plot(b0,b1,pch='.')

plot(b0,type="l")
```




```
plot(b1,type="l")
```




```
#acf(b0)
#acf(b1)

# DIC
HPCIModel2.sim$BUGSoutput$DIC
```

```
## [1] 19.70661
```

```
quantile(b0, c(0.025, 0.05, 0.5, 0.095, 0.975))
```

```
##      2.5%        5%       50%      9.5%     97.5% 
## -3.748747 -3.428982 -1.839662 -3.077050 -0.190692
```

```
quantile(b1, c(0.025, 0.05, 0.5, 0.095, 0.975))
```

```
##        2.5%          5%         50%        9.5%       97.5% 
## -0.09968534 -0.07370814  0.04591646 -0.04963894  0.19800494
```

```
######################
# Check Model Fit w/ Bayesian Chi-Squared Test
GoF <- matrix(NA,ncol=length(failures),nrow=length(b0))
for (i in 1:length(b0)) {
  GoF[i,] <- runif(length(failures),
                   pbinom(failures-1,1,ilogit(b0[i]+b1[i]*(demands))),
                   pbinom(failures,1,ilogit(b0[i]+b1[i]*(demands)))
  )
}

# Function requires fitted quantiles and returns a p-value
GoF_Test <- function(fitted_quantiles) {
  n <- length(fitted_quantiles)
  K <- round((n)^(0.4))
  mK <- table(cut(fitted_quantiles,(0:K)/K))
  np <- n/K
  RB <- sum(((mK-np)^2)/np)
  return(1-pchisq(RB,K-1))
}

# Calculating the p-values for each posterior model
GoF_Summary <- apply(GoF,1,GoF_Test)

# Histogram of posterior model p-values
#hist(GoF_Summary,xlim=c(0,1))

# Percent of posterior models with p-value less than 0.05
mean(GoF_Summary < 0.05)
```

```
## [1] 0.1559
```

Seeing how the results in Example 7.1 said that the model fit will using the Bayesian χ2 goodness-of-fit test, I would say that the results don’t compare well. I find this surprising because in previous classes, the binomial distribution worked about the same, whether data was input in chunks or one at a time.