Final Exam PIBO

Maike Holthuijzen

December 11, 2017

1. Intermediate disturbance hypothesis analysis.

The MLE estimates for β_0, β_1, and β_2 were 2.056, 0.44, and -0.045, respectively. The negative log likelihood was 848.023.

```
library(rjags)
```

```
## Warning: package 'rjags' was built under R version 3.4.2
```

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

```
## Warning: package 'coda' was built under R version 3.4.2
```

```
## Linked to JAGS 4.3.0
```

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

```
library(coda)
srDat<-read.table("http://www.uvm.edu/~bbeckage/Teaching/PBIO_294/Data/idh.csv",sep=',',header=TRUE)

nfires = srDat$nFires
sr = srDat$sr
firesq = srDat$nFires^2

firePoiQuad = function(parVec,nfires,sr,firesq){
  b0 = parVec[1]
  b1 = parVec[2]
  b2 = parVec[3]
  firepreds <- exp(b0 + b1*nfires + b2*firesq)
  nllik = -sum(dpois(x = sr, lambda = firepreds,log=TRUE))
  # cat("nllik= ",nllik,sep=" ",fill=T);cat(" ",sep=" ",fill=T)
  return(nllik)
}

parVec = c(1, 1.0, -.5) # Initial parameter values
MLestimates = optim(par = parVec,fn=firePoiQuad, method="L-BFGS-B", lower=c(-10, -10, -2),
               upper=c(5, 5, 0), nfires = nfires, sr = sr, firesq = firesq)
#parameter estimates for b0, b1, and b2
MLestimates$par
```

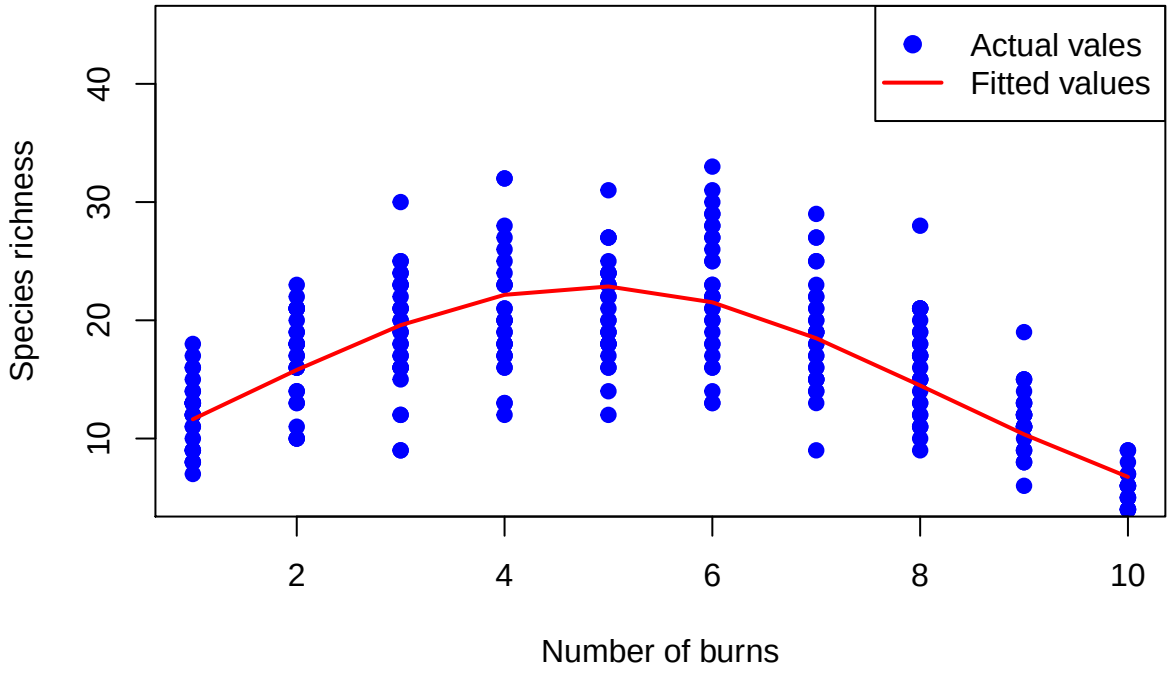
```
## [1]  2.05503116  0.44430099 -0.04588194
```

```
#negative log likelihood value
MLestimates$value
```

```
## [1] 848.023
```

```
#plot
burnpreds  = nfires[order(nfires)]
Species_richness = srDat$sr
plot(burnpreds, Species_richness, pch = 19, col = "blue", ylim = c(5, 45), xlab = "Number of burns", yla
lines(burnpreds, exp(MLestimates$par[1] + MLestimates$par[2]*burnpreds + MLestimates$par[3]*firesq) , co
```

```
legend("topright", legend=c("Actual vales", "Fitted values"),
       col=c("blue", "red"), pch=c(19, NA), lty=c(NA, 1), cex=1, lwd=2)
```



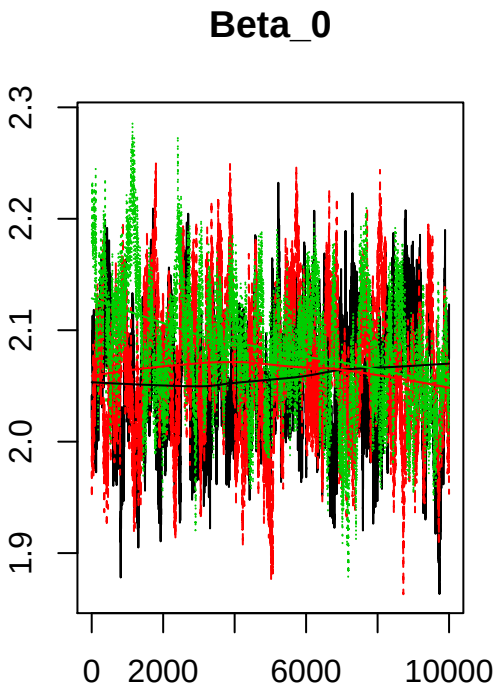
2. (30 points) Repeat exercise 1 except now fitting the model using either Rstan or Rjags. The jags model, along with resulting plots of the posterior distributions for parameters and predicted values is shown below.

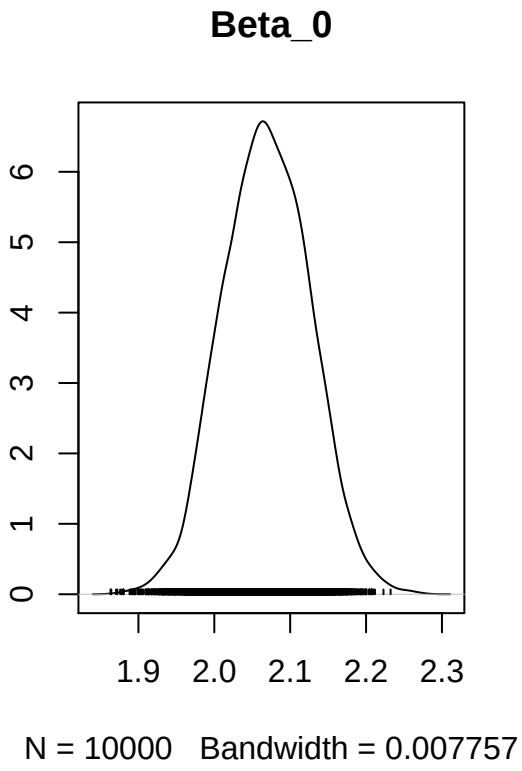
```
library(rjags)
library(coda)
#scaling function for plotting (below)
scale2 <- function(x) {
  sdx <- sqrt(var(x))
  meanx <- mean(x)
  return((x - meanx)/sdx)
}

jags.data = list(nobs = length(nfires), sr = sr, nfires = nfires)

#jags model
modelstring = "
model
{
  # uninformative priors
  beta0 ~ dnorm(0,0.001)
  beta1 ~ dnorm(0,0.001)
  beta2 ~ dnorm(0,0.001)
```

```
  # likelihood
  for(i in 1:nobs)
  {
    sr[i] ~ dpois(lambda[i])
    log(lambda[i]) <- beta0 + beta1*nfires[i] + beta2*pow(nfires[i],2)
    # get predicted values
    prediction[i] ~ dpois(lambda[i])
  }
}
"
writeLines(modelstring, con='poisFire.txt')
params = c("beta0", "beta1", "beta2", "prediction")

jm = jags.model("poisFire.txt", data = jags.data, n.chains = 3, n.adapt = 1000)
```

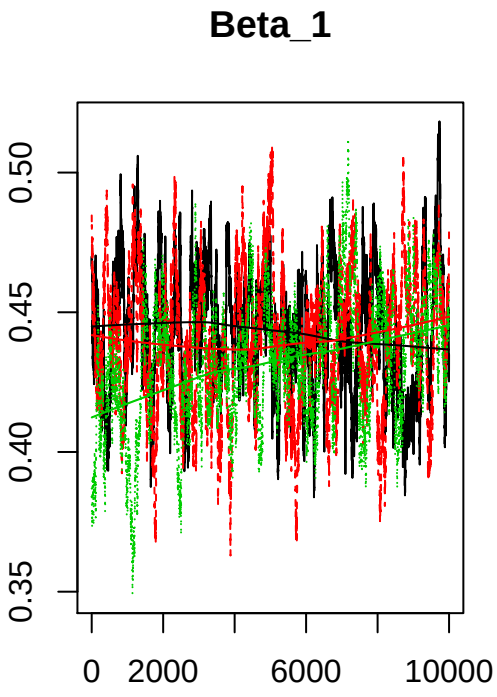
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 300
##    Unobserved stochastic nodes: 303
##    Total graph size: 957
##
## Initializing model
```

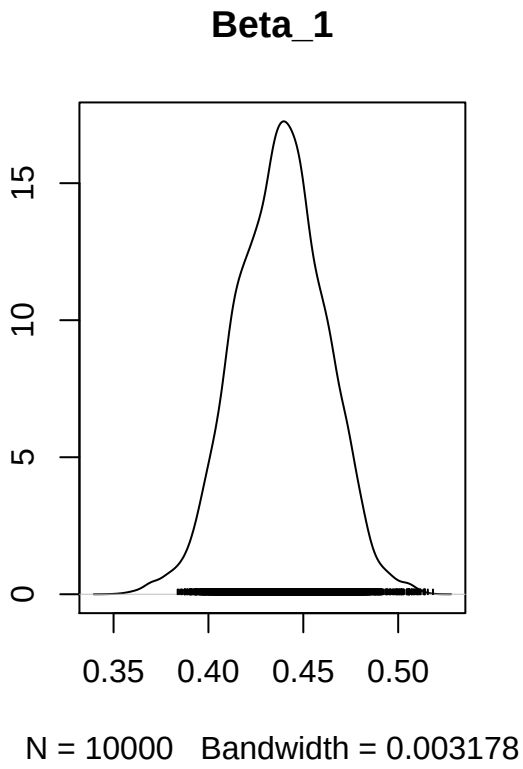
```
update(jm, n.iter = 1000)
jm.sample = jags.samples(jm, variable.names = params, n.iter = 10000, thin = 1)

#plot posterior densities of parameters
plot(as.mcmc.list(jm.sample$beta0), main = "Beta_0")
```



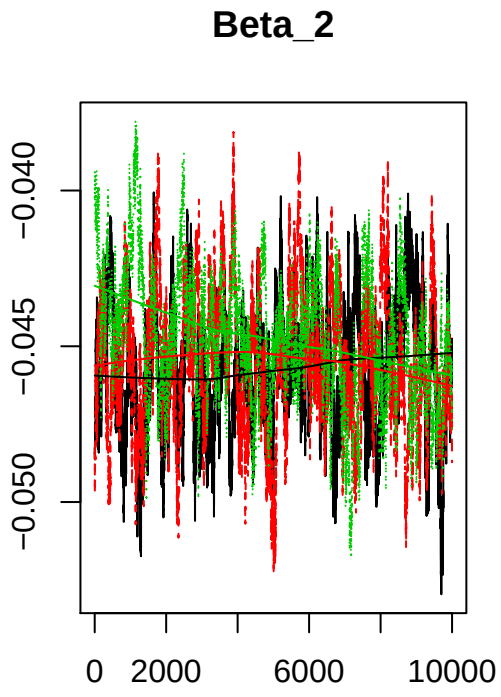


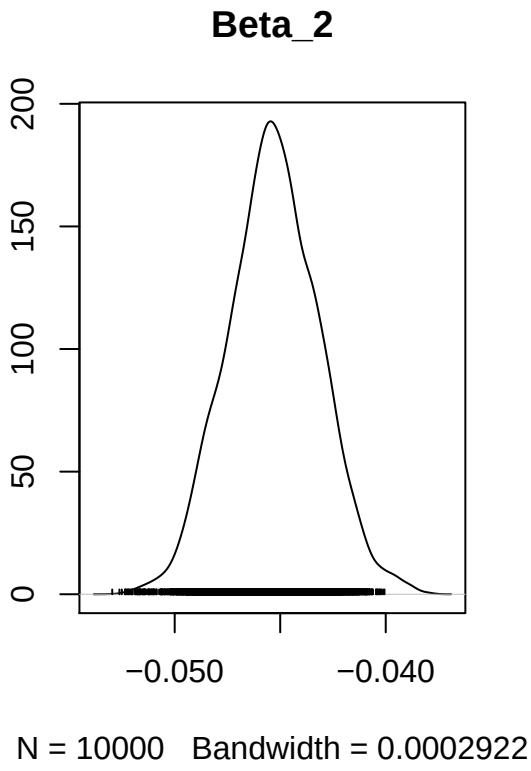

```
plot(as.mcmc.list(jm.sample$beta1), main = "Beta_1")
```






```
plot(as.mcmc.list(jm.sample$beta2), main = "Beta_2")
```






```
#summary of posterior di for parameters
summary(as.mcmc.list(jm.sample$beta0))
```

```
##
## Iterations = 1:10000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean             SD       Naive SE Time-series SE
##      2.0688909      0.0575212      0.0003321      0.0039917
##
## 2. Quantiles for each variable:
##
##  2.5%   25%   50%   75% 97.5%
## 1.960 2.029 2.068 2.109 2.181
```

```
summary(as.mcmc.list(jm.sample$beta1))
```

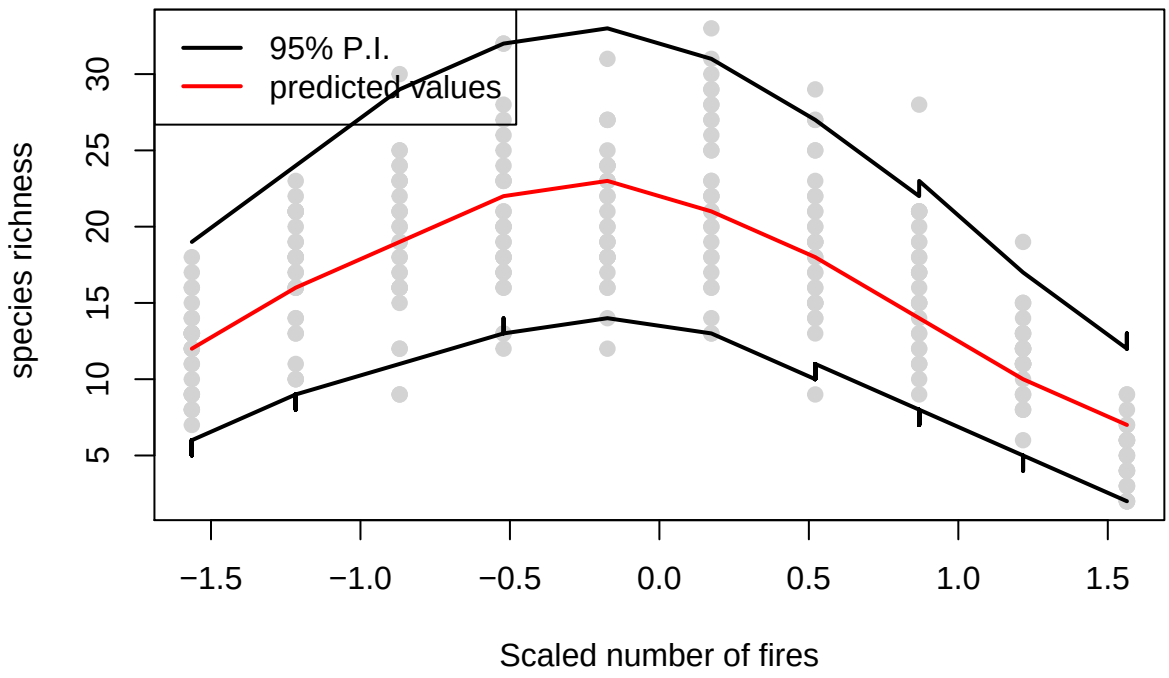
```
##
## Iterations = 1:10000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 10000
```

```
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean             SD       Naive SE Time-series SE
##      0.4380047      0.0235655      0.0001361      0.0018780
##
## 2. Quantiles for each variable:
##
##   2.5%    25%    50%    75%  97.5%
## 0.3923 0.4216 0.4384 0.4539 0.4824
```

```
summary(as.mcmc.list(jm.sample$beta2))
```

```
##
## Iterations = 1:10000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean             SD       Naive SE Time-series SE
##     -4.531e-02      2.167e-03      1.251e-05      1.595e-04
##
## 2. Quantiles for each variable:
##
##     2.5%      25%      50%      75%    97.5%
## -0.04946 -0.04676 -0.04533 -0.04385 -0.04107
```

```
#obtain predicted values for plot
predictions = summary(as.mcmc.list(jm.sample$prediction))
mypreds = data.frame(scfires = scale2(nfires), predictions$quantiles)
mypreds = mypreds[order(mypreds[, 1]), ]

#make plot
plot(scale2(nfires), sr, cex = 1, col = "lightgrey", pch = 19, ylab = " species richness",
     xlab = "Scaled number of fires")
lines(mypreds[, 1], mypreds[, 2], lwd = 2)
lines(mypreds[, 1], mypreds[, 4], lwd = 2, col = "red")
lines(mypreds[, 1], mypreds[, 6], lwd = 2)

legend("topleft", legend = c("95% P.I.", "predicted values"), col = c("black", "red"),
       lwd = c(2, 2))
```



The HPD intervals and Gelman statistic are listed below. The Gelman - Rubin statistics for each of the three parameters are near 1, satisfying the criteria for adequate convergence of the MCMC chains.

```
#make this an "mcmc.list".
beta0Obj <- as.mcmc.list(jm.sample$beta0)
beta1Obj = as.mcmc.list(jm.sample$beta1)
beta2Obj = as.mcmc.list(jm.sample$beta2)

#gelman rubin statistic
gelman.diag(beta0Obj)
```

```
## Potential scale reduction factors:
##
##       Point est. Upper C.I.
## beta0          1       1.01
```

```
gelman.diag(beta1Obj)
```

```
## Potential scale reduction factors:
##
##       Point est. Upper C.I.
## beta1          1       1.01
```

```
gelman.diag(beta2Obj)
```

```
## Potential scale reduction factors:
##
##       Point est. Upper C.I.
```

```
## beta2          1       1.01
```

```
#high density interval
HPDinterval(beta0Obj)
```

```
## [[1]]
##          lower    upper
## beta0 1.957362 2.160352
## attr(,"Probability")
## [1] 0.95
##
## [[2]]
##          lower    upper
## beta0 1.952982 2.181951
## attr(,"Probability")
## [1] 0.95
##
## [[3]]
##          lower    upper
## beta0 1.977862 2.193293
## attr(,"Probability")
## [1] 0.95
```

```
HPDinterval(beta1Obj)
```

```
## [[1]]
##           lower     upper
## beta1 0.4012699 0.4834245
## attr(,"Probability")
## [1] 0.95
##
## [[2]]
##           lower     upper
## beta1 0.3912551 0.4848971
## attr(,"Probability")
## [1] 0.95
##
## [[3]]
##           lower     upper
## beta1 0.3845952 0.4742942
## attr(,"Probability")
## [1] 0.95
```

```
HPDinterval(beta2Obj)
```

```
## [[1]]
##             lower       upper
## beta2 -0.04947694 -0.04174237
## attr(,"Probability")
## [1] 0.95
##
## [[2]]
##             lower       upper
## beta2 -0.04980745 -0.04107952
## attr(,"Probability")
## [1] 0.95
```

```
##
## [[3]]
##             lower       upper
## beta2 -0.04916171 -0.04080664
## attr(,"Probability")
## [1] 0.95
```

3. Seed predation. The final model has the form:

Seed lost ~ Binomial(theta) # 1 if seed is predated, 0 if seed is left alone Logit(theta) = b0i + b1 x gap
b0i ~ gamma(mu$^{2/sd}$2,mu/sd^2)
mui = b2 + b3 x seed mass i - b4 x (seed mass i)^2
sd^2 ~ gamma(0.001,0.001) b1,b2,b3,b4 ~ normal(0,100)

The data can be read using:

```
library(rjags)
library(coda)
seeds<-read.table("http://www.uvm.edu/~bbeckage/Teaching/PBIO_294/Data/seeds.csv",sep=',',header=TRUE)


N = length(seeds$seedLost)
seedsLost = seeds$seedLost
gap = seeds$gap
mass = seeds$mass

data_jags = list(N = N, seedsLost = seedsLost, gap = gap, mass = mass)
```

The jags model is:

```
modelstring3 = "
model {
# N observations
for (i in 1:N) {
seedsLost[i] ~ dbern(theta[i])
logit(theta[i]) <-  beta0[i] + beta1*gap[i]
mu[i] <- beta2 + beta3*mass[i] + beta4*pow(mass[i], 2)
beta0[i] ~ dgamma( pow(m1[i], 2) / b , m1[i] / b )
m1[i] = mu[i] +0.01
}

b  ~ dgamma(.001, .001)

beta1     ~ dnorm(0.0, 100)
beta2   ~ dnorm(0.0, 100)
beta3 ~ dnorm(0.0, 100)
beta4  ~ dnorm(0.0, 100)
}
"


writeLines(modelstring3, con='seeds3.txt')
params = c("beta1", "beta2", "beta3", "beta4")
mod = jags.model('seeds3.txt', data=data_jags, n.chains=3)
```

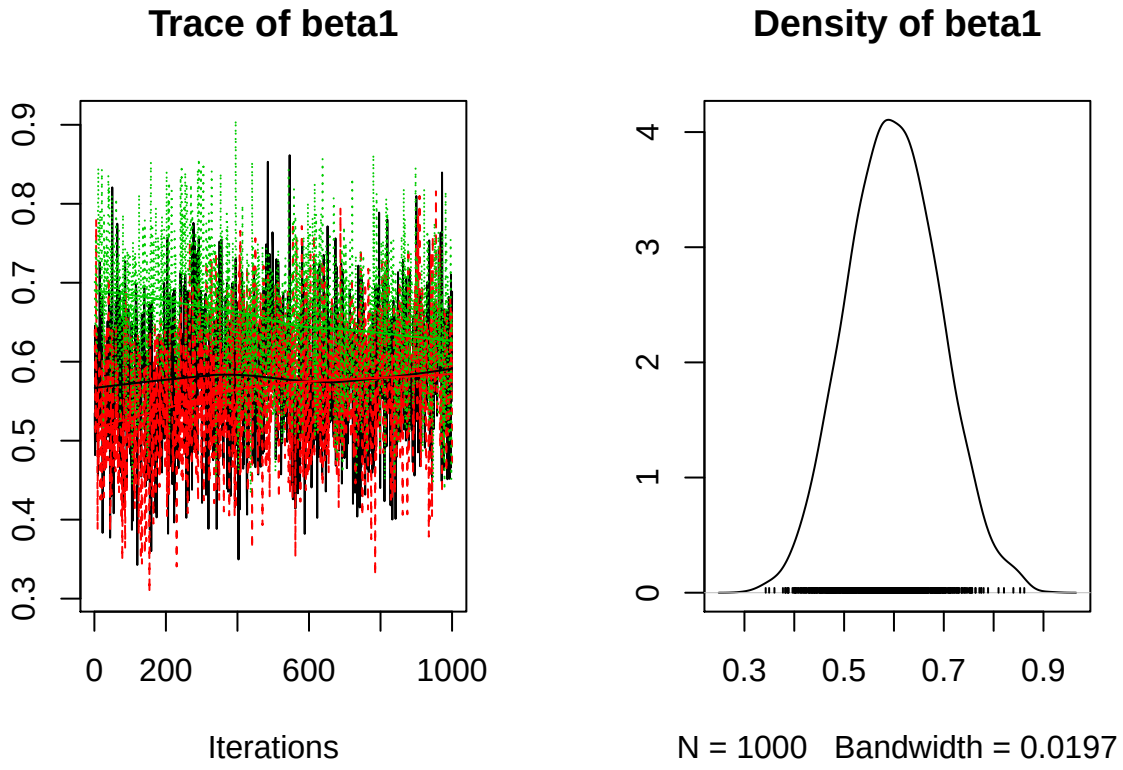
```
## Compiling model graph
##    Resolving undeclared variables
```

```
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1000
##    Unobserved stochastic nodes: 1005
##    Total graph size: 14013
##
## Initializing model
```

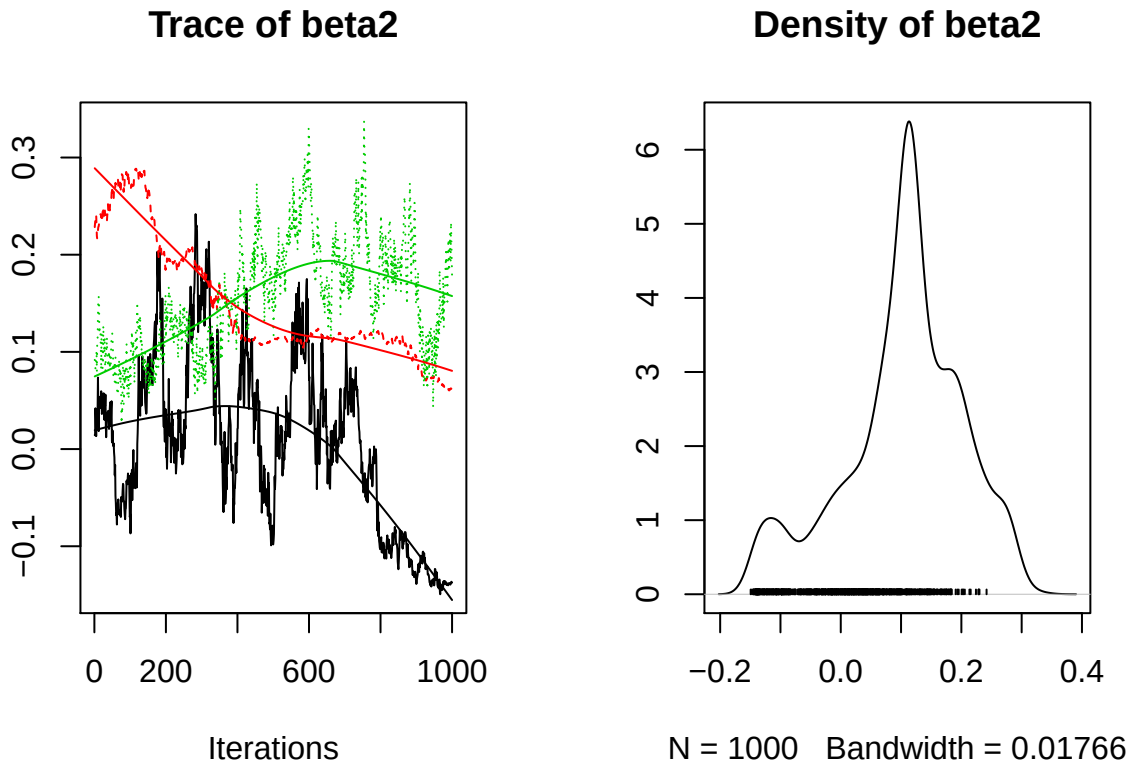
```
update(mod, 10000)
mod_sim <- jags.samples(mod, variable.names = params, n.iter = 10000, thin = 10)
```

Posterior density plots and summaries of posterior densities:

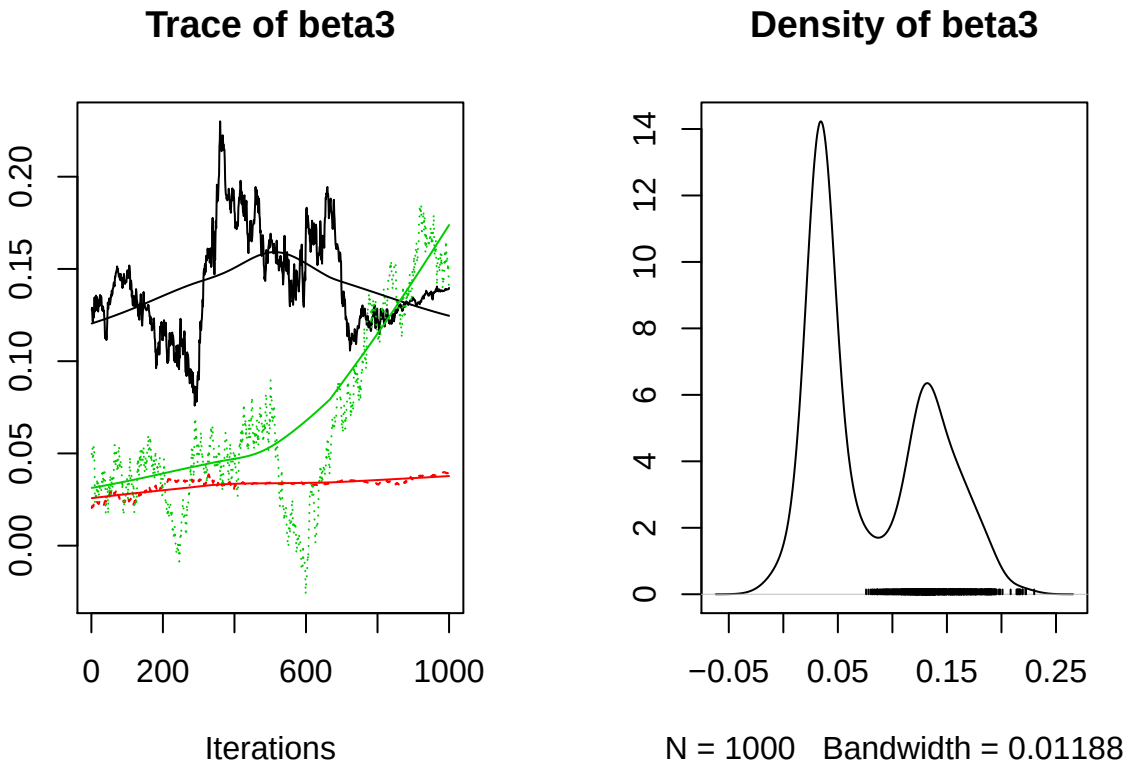
```
plot(as.mcmc.list(mod_sim$beta1))
```



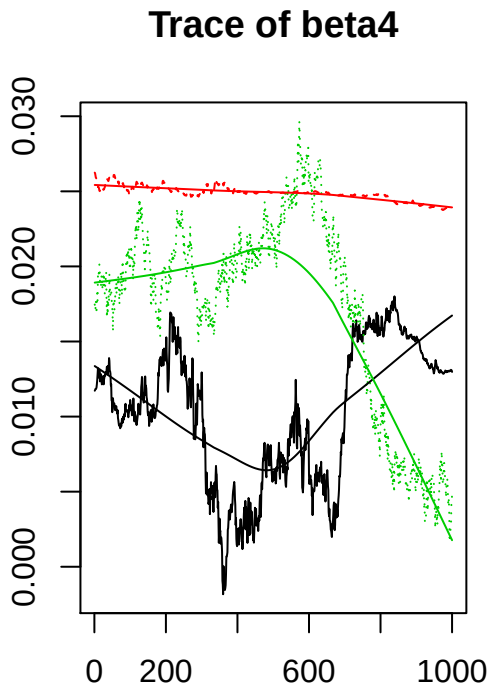

```
plot(as.mcmc.list(mod_sim$beta2))
```

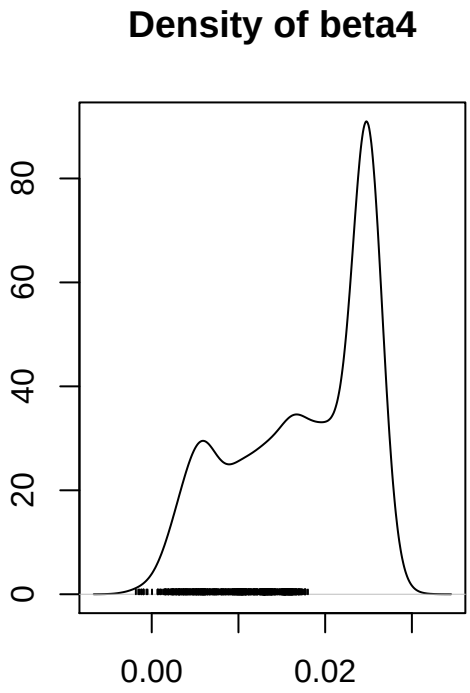



```
plot(as.mcmc.list(mod_sim$beta3))
```




```
plot(as.mcmc.list(mod_sim$beta4))
```






```
summary(as.mcmc.list(mod_sim$beta1))
```

```
##
## Iterations = 1:1000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 1000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean             SD       Naive SE Time-series SE
##       0.599270       0.092173       0.001683       0.003186
##
## 2. Quantiles for each variable:
##
##   2.5%    25%    50%    75%  97.5%
## 0.4256 0.5343 0.5976 0.6623 0.7800
```

```
summary(as.mcmc.list(mod_sim$beta2))
```

```
##
## Iterations = 1:1000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 1000
##
```

```
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean             SD       Naive SE Time-series SE
##       0.103959       0.098272       0.001794       0.020691
##
## 2. Quantiles for each variable:
##
##     2.5%      25%      50%      75%    97.5%
## -0.12765  0.06036  0.11346  0.17109  0.27580
```

```
summary(as.mcmc.list(mod_sim$beta3))
```

```
##
## Iterations = 1:1000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 1000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean             SD       Naive SE Time-series SE
##       0.080349       0.055594       0.001015       0.009741
##
## 2. Quantiles for each variable:
##
##     2.5%      25%      50%      75%    97.5%
## 0.009071 0.033931 0.051734 0.131520 0.186558
```

```
summary(as.mcmc.list(mod_sim$beta4))
```

```
##
## Iterations = 1:1000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 1000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean             SD       Naive SE Time-series SE
##      0.0174490      0.0075751      0.0001383      0.0016831
##
## 2. Quantiles for each variable:
##
##     2.5%      25%      50%      75%    97.5%
## 0.002842 0.011119 0.019245 0.024779 0.025791
```

The Gelman-Rubin statistics for all of the four parameters suggest convergence issues, especially for β_3. Values around 1 of the Gelman-Rubin statistic are desirable. Poor convergence among the four chains is also apparent in the trace plots. Increasing the number of iterations from 5000 to 10000 did help somewhat with convergence, but the computation time increased considerably.

```
#convert to "mcmc.list".
B1 = as.mcmc.list(mod_sim$beta1)
```

```
B2 = as.mcmc.list(mod_sim$beta2)
B3 = as.mcmc.list(mod_sim$beta3)
B4 = as.mcmc.list(mod_sim$beta4)

#gelman rubin statistic
gelman.diag(B1)
```

```
## Potential scale reduction factors:
##
##       Point est. Upper C.I.
## beta1       1.14       1.41
```

```
gelman.diag(B2)
```

```
## Potential scale reduction factors:
##
##       Point est. Upper C.I.
## beta2       2.99       7.75
```

```
gelman.diag(B3)
```

```
## Potential scale reduction factors:
##
##       Point est. Upper C.I.
## beta3        2.4       9.04
```

```
gelman.diag(B4)
```

```
## Potential scale reduction factors:
##
##       Point est. Upper C.I.
## beta4       2.02       5.39
```

```
#high density interval
HPDinterval(B1)
```

```
## [[1]]
##           lower     upper
## beta1 0.4126915 0.7367106
## attr(,"Probability")
## [1] 0.95
##
## [[2]]
##           lower     upper
## beta1 0.4044681 0.7251936
## attr(,"Probability")
## [1] 0.95
##
## [[3]]
##           lower     upper
## beta1 0.4961575 0.8199201
## attr(,"Probability")
## [1] 0.95
```

```
HPDinterval(B2)
```

```
## [[1]]
##            lower     upper
```

```
## beta2 -0.1434403 0.1549307
## attr(,"Probability")
## [1] 0.95
##
## [[2]]
##            lower     upper
## beta2 0.06900726 0.2838184
## attr(,"Probability")
## [1] 0.95
##
## [[3]]
##            lower     upper
## beta2 0.06347949 0.2698997
## attr(,"Probability")
## [1] 0.95
```

```
HPDinterval(B3)
```

```
## [[1]]
##            lower     upper
## beta3 0.09430661 0.1944204
## attr(,"Probability")
## [1] 0.95
##
## [[2]]
##            lower      upper
## beta3 0.02266975 0.03838655
## attr(,"Probability")
## [1] 0.95
##
## [[3]]
##             lower     upper
## beta3 -0.00475143 0.1665386
## attr(,"Probability")
## [1] 0.95
```

```
HPDinterval(B4)
```

```
## [[1]]
##             lower      upper
## beta4 0.002192304 0.01722211
## attr(,"Probability")
## [1] 0.95
##
## [[2]]
##            lower      upper
## beta4 0.02386817 0.02570738
## attr(,"Probability")
## [1] 0.95
##
## [[3]]
##             lower      upper
## beta4 0.004642976 0.02711527
## attr(,"Probability")
## [1] 0.95
```