Bayesian SIR Model

Rob Deardon (Calgary) and Caitlin Ward (Minnesota)

Writing the SIR model

```
library(nimble)
```

```
## nimble version 0.12.2 is loaded.
## For more information on NIMBLE and a User Manual,
## please visit https://R-nimble.org.
```

```
##
## Attaching package: 'nimble'
```

```
## The following object is masked from 'package:stats':
##
##     simulate
```

```
SIR_code <-  nimbleCode({

    S[1] <- N - I0 - R0
    I[1] <- I0
    R[1] <- R0

    probIR <- 1 - exp(-gamma)

    ### loop over time
    for(t in 1:tau) {

        probSI[t] <- 1 - exp(- beta * I[t] / N)

        Istar[t] ~ dbin(probSI[t], S[t])
        Rstar[t] ~ dbin(probIR, I[t])

        # update S, I, R
        S[t + 1] <- S[t] - Istar[t]
        I[t + 1] <- I[t] + Istar[t] - Rstar[t]
        R[t + 1] <- R[t] + Rstar[t]

    }

    # priors
    beta ~ dgamma(1, 1)
    gamma ~ dgamma(aa, bb)

})
```

Simulating epidemics

Here we specify the population size $N = 10,000$, 5 initially infectious individuals, and simulate 100 days of the epidemic.

```
constantsList <- list(N = 10000,
                      I0 = 5,
                      R0 = 0,
                      tau = 100)


sirModel <- nimbleModel(SIR_code,
                        constants = constantsList)
```

```
## Defining model
```

```
## Building model
```

```
## Running calculate on model
##   [Note] Any error reports that follow may simply reflect missing values in model variables.
```

```
## Checking model sizes and dimensions
```

```
##   [Note] This model is not fully initialized. This is not an error.
##          To see which variables are not initialized, use model$initializeInfo().
##          For more information on model initialization, see help(modelInitialization).
```

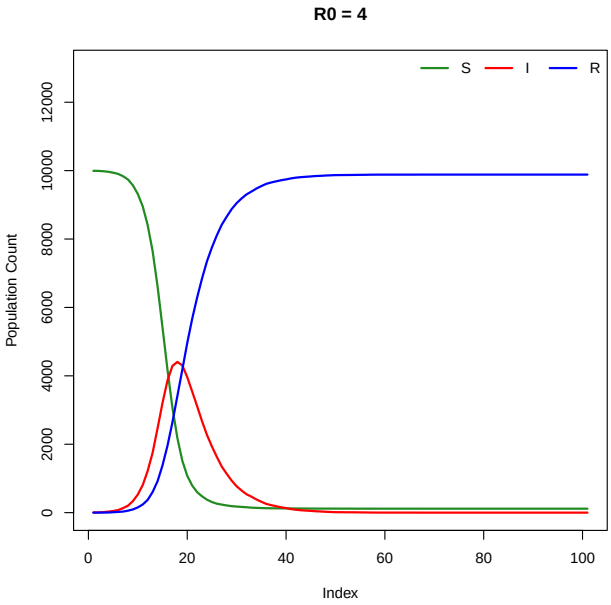
```
# exclude data from parent nodes
dataNodes <- c('Istar', 'Rstar')
dataNodes <- sirModel$expandNodeNames(dataNodes, returnScalarComponents = TRUE)
parentNodes <- sirModel$getParents(dataNodes, stochOnly = TRUE)
parentNodes <- parentNodes[-which(parentNodes %in% dataNodes)]
parentNodes <- sirModel$expandNodeNames(parentNodes, returnScalarComponents = TRUE)
nodesToSim <- sirModel$getDependencies(parentNodes, self = FALSE, downstream = T)
```

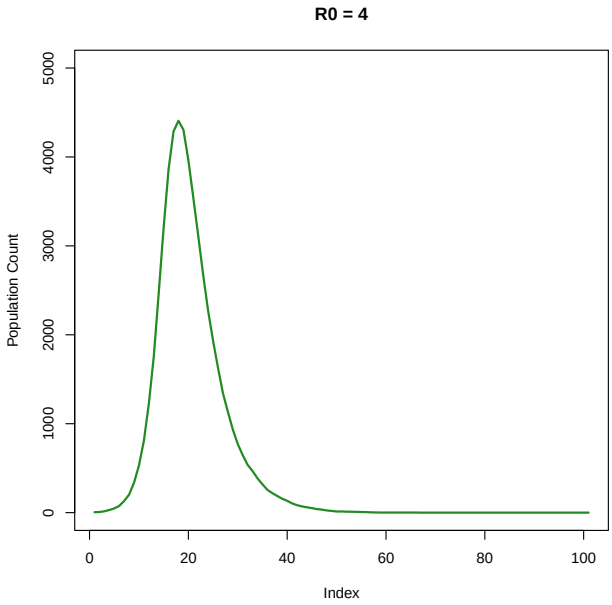
We can simulate using various values of β and γ to specify various reproductive numbers.

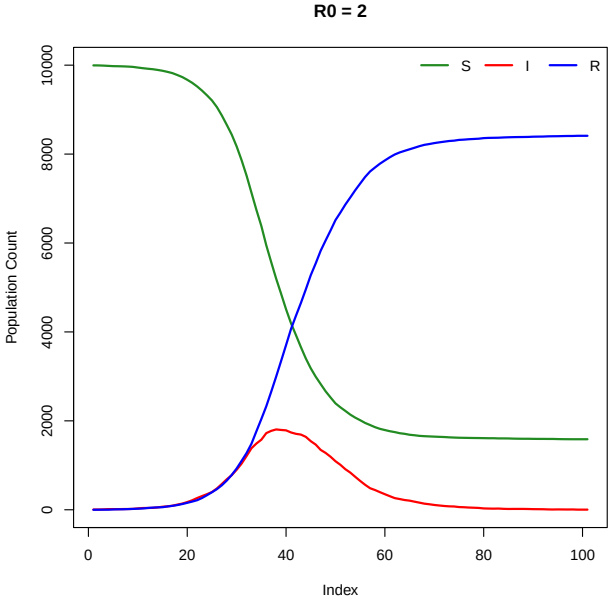
In all simulations the mean infectious period is 5 days.

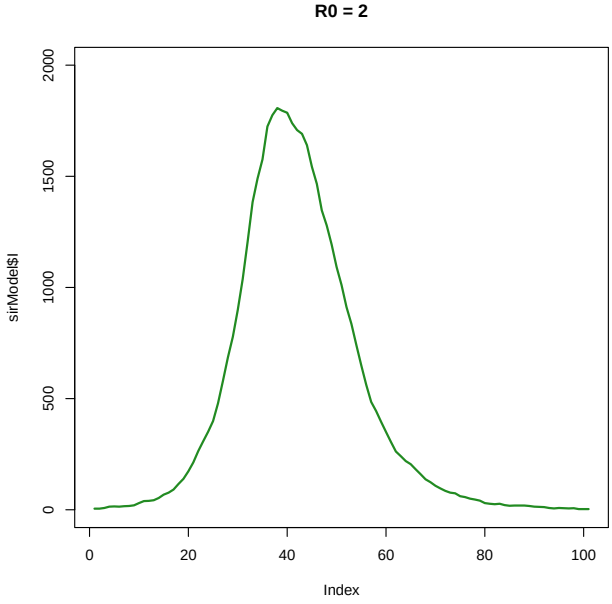
```
pal <- c('forestgreen', 'red', 'blue')


par(mfrow = c(2,2))

# simulation 1
initsList <- list(beta = 0.8,
                  gamma = 0.2)
sirModel$setInits(initsList)

set.seed(1)
sirModel$simulate(nodesToSim, includeData = TRUE)

plot(sirModel$S, type = 'l', col = pal[1], ylim = c(0, 1.3e4),
     main = paste0('R0 = ', sirModel$beta / sirModel$gamma), lwd = 2, ylab = "Population Count")
lines(sirModel$I, col = pal[2], lwd = 2)
lines(sirModel$R, col = pal[3], lwd = 2)
legend('topright', c('S', 'I', 'R'), col = pal, lwd = 2,
       bty = 'n', horiz = T)
```

```r
plot(sirModel$I, type = 'l', col = pal[1], ylim = c(0, 5000),
     main = paste0('R0 = ', sirModel$beta / sirModel$gamma), lwd = 2, ylab = "Population Count")




# simulation 2
initsList <- list(beta = 0.4,
                  gamma = 0.2)
sirModel$setInits(initsList)

set.seed(1)
sirModel$simulate(nodesToSim, includeData = TRUE)

plot(sirModel$S, type = 'l', col = pal[1], ylim = c(0, 10000),
     main = paste0('R0 = ', sirModel$beta / sirModel$gamma), lwd = 2, ylab = "Population Count")
lines(sirModel$I, col = pal[2], lwd = 2)
lines(sirModel$R, col = pal[3], lwd = 2)
legend('topright', c('S', 'I', 'R'), col = pal, lwd = 2,
       bty = 'n', horiz = T)


plot(sirModel$I, type = 'l', col = pal[1], ylim = c(0, 2000),
     main = paste0('R0 = ', sirModel$beta / sirModel$gamma), lwd = 2)
```


R0 = 4
S
I
R
Population Count
0
2000
4000
6000
8000
10000
12000
0
20
40
60
80
100
Index


R0 = 4
Population Count
0
1000
2000
3000
4000
5000
0
20
40
60
80
100
Index

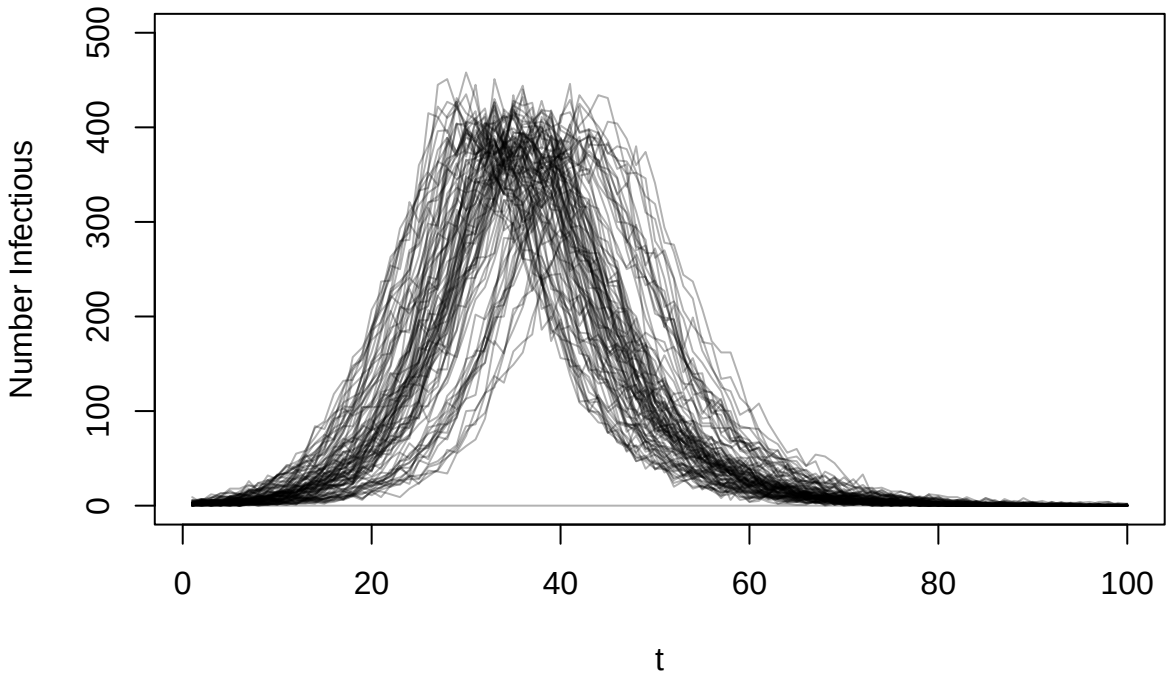

R0 = 2
S
I
R
Population Count
0
2000
4000
6000
8000
10000
0
20
40
60
80
100
Index


R0 = 2
sirModel$I
0
500
1000
1500
2000
0
20
40
60
80
100
Index

Epidemics are Stochastic

Here we simulate 100 epidemics from the same parameter values and plot the observed incidence curve from each simulation.

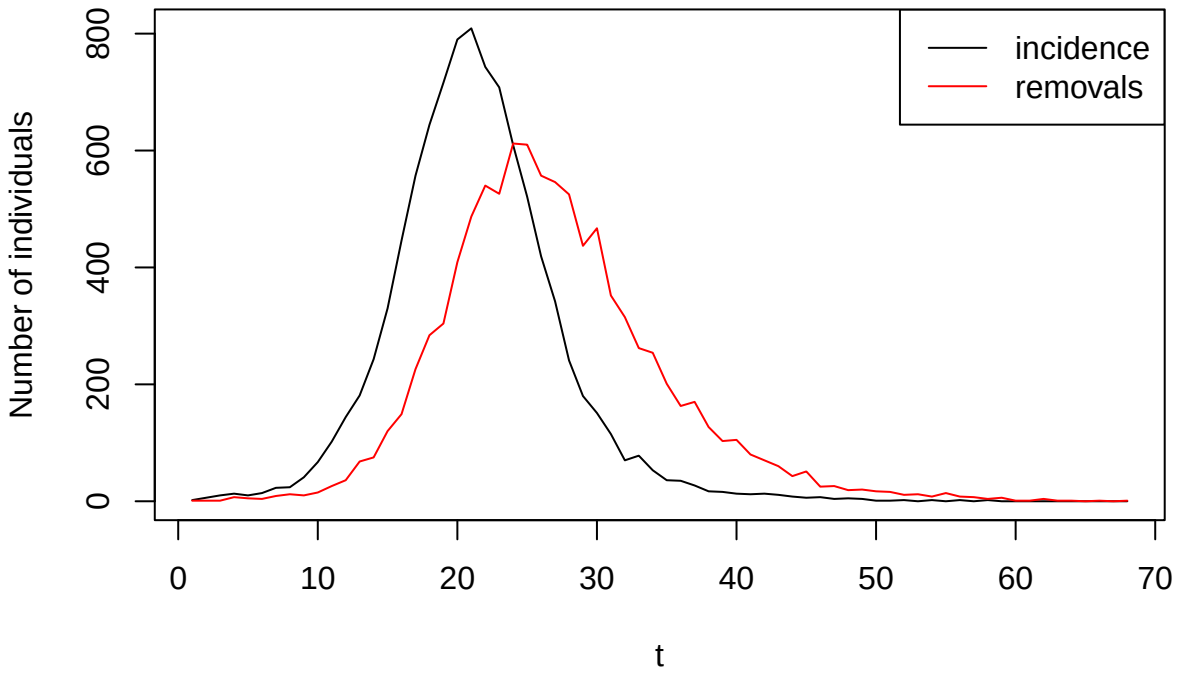
```
initsList <- list(beta = 0.4,
                  gamma = 0.2)
sirModel$setInits(initsList)

nSim <- 100

set.seed(1)
epiCurve <- matrix(NA, nrow = length(sirModel$Istar), ncol = nSim)
for (i in 1:nSim) {
    sirModel$simulate(nodesToSim, includeData = TRUE)
    epiCurve[,i] <- sirModel$Istar
}

plot(epiCurve[,1], type = 'l', col = adjustcolor('black', alpha = 0.3),
     ylim = c(0, 500), ylab = "Number Infectious", xlab='t')
for (i in 2:nSim) {
    lines(epiCurve[,i], col = adjustcolor('black', alpha = 0.3))
}
```



Model fitting to simulated data

Simulate data, then use it to fit the model.

```
initsList <- list(beta = 0.6,
                  gamma = 0.2)
sirModel$setInits(initsList)

set.seed(1)
sirModel$simulate(nodesToSim, includeData = TRUE)

trueIstar <- sirModel$Istar
trueRstar <- sirModel$Rstar

endTime <- max(which(trueIstar > 0)) + 10
trueIstar <- trueIstar[1:endTime]
trueRstar <- trueRstar[1:endTime]

plot(trueIstar, type = 'l', ylab='Number of individuals', xlab='t')
lines(trueRstar, col = 'red')
legend('topright', c('incidence', 'removals'), col = c('black', 'red'), lwd = 1)
```



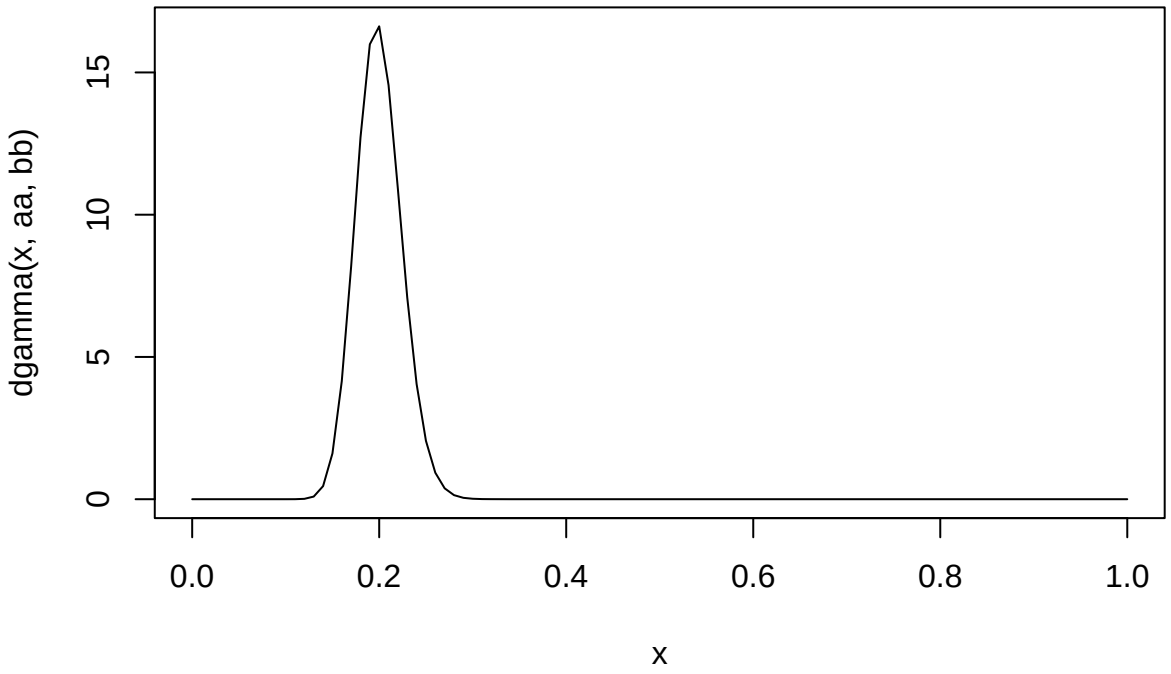
Model Specifications

Before fitting the model, we need to determine a reasonable prior for γ. The true value corresponds to a mean infectious period of 5 days, so we choose a prior that puts 90% probability on the mean infectious period between 4 and 6 days and is centered on 5 days.

```
bb <- 348
aa <- 0.2 * bb
pgamma(1/4, aa, bb) - pgamma(1/6, aa, bb)
```

```
## [1] 0.900183
```

```
curve(dgamma(x, aa, bb))
```




```
dataList <- list(Istar = trueIstar,
                 Rstar = trueRstar)

constantsList <- list(N = 10000,
                      I0 = 5,
                      R0 = 0,
                      tau = length(dataList$Istar),
                      aa = aa,
                      bb = bb)

set.seed(2)
initsList <- list(beta = runif(1, 0, 1),
                  gamma = rgamma(1, aa, bb))


sirModelFit <- nimbleModel(SIR_code,
                           constants = constantsList,
                           data = dataList,
                           inits = initsList)
```

```
## Defining model
```

```
## Building model

## Setting data and initial values

## Running calculate on model
##   [Note] Any error reports that follow may simply reflect missing values in model variables.

## Checking model sizes and dimensions
```

NIMBLE automatically calculates S, I, and R from Istar and Rstar, so these do not need to be inputs to the model

```
with(sirModelFit, cbind(S, Istar, I, Rstar, R))[1:20,]
```

```
## Warning in cbind(S, Istar, I, Rstar, R): number of rows of result is not a
## multiple of vector length (arg 2)
```

```
##          S Istar    I Rstar    R
##  [1,] 9995     2    5     1    0
##  [2,] 9993     6    6     1    1
##  [3,] 9987    10   11     1    2
##  [4,] 9977    13   20     7    3
##  [5,] 9964    10   26     5   10
##  [6,] 9954    14   31     4   15
##  [7,] 9940    23   41     9   19
##  [8,] 9917    24   55    12   28
##  [9,] 9893    41   67    10   40
## [10,] 9852    67   98    15   50
## [11,] 9785   102  150    26   65
## [12,] 9683   144  226    36   91
## [13,] 9539   181  334    68  127
## [14,] 9358   243  447    75  195
## [15,] 9115   330  615   120  270
## [16,] 8785   446  825   149  390
## [17,] 8339   557 1122   226  539
## [18,] 7782   644 1453   284  765
## [19,] 7138   716 1813   304 1049
## [20,] 6422   790 2225   409 1353
```

Use Default Configurations and Obtain Samples

Plotted with burn-in included here

```
myConfig <- configureMCMC(sirModelFit)
```

```
## ===== Monitors =====
## thin = 1: beta, gamma
## ===== Samplers =====
## RW sampler (2)
##   - beta
##   - gamma
```

```
myMCMC <- buildMCMC(myConfig)

system.time({
    compiled <- compileNimble(sirModelFit, myMCMC)
    samples <- runMCMC(compiled$myMCMC, niter = 50000, setSeed = 3)
})
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

```
## Running chain 1 ...
```

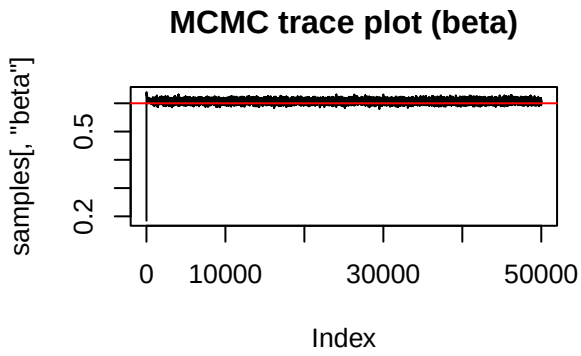
```
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

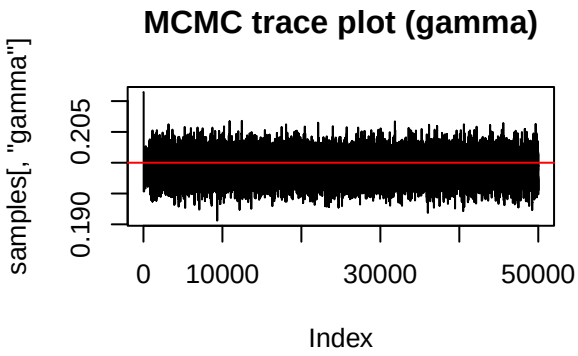
```
##    user  system elapsed
##  21.482   0.844  22.784
```

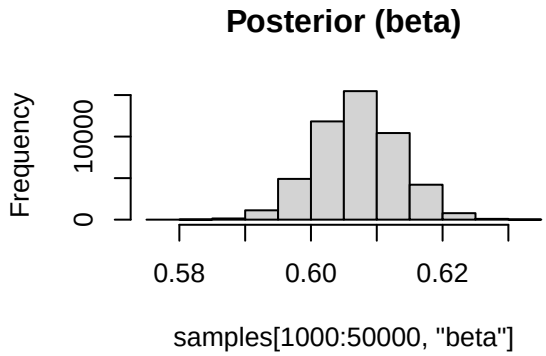
```
head(samples)
```

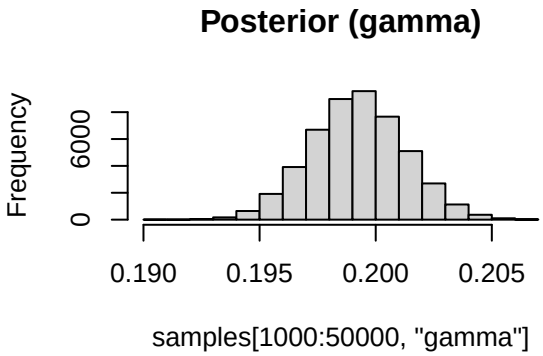
```
##           beta     gamma
## [1,] 0.1848823 0.2114557
## [2,] 0.4436705 0.2114557
## [3,] 0.6394533 0.2114557
## [4,] 0.6394533 0.2114557
## [5,] 0.6394533 0.2114557
## [6,] 0.6394533 0.2114557
```

```
par(mfrow = c(2,2))
plot(samples[,'beta'], type = 'l', main = 'MCMC trace plot (beta)')
abline(h = 0.6, col = 'red')
plot(samples[,'gamma'], type = 'l', main = 'MCMC trace plot (gamma)')
abline(h = 0.2, col = 'red')
hist(samples[1000:50000,'beta'], main = 'Posterior (beta)')
hist(samples[1000:50000,'gamma'], main = 'Posterior (gamma)')
```










```
#
# Posterior Mean and 95% Percentile Interval: Beta
mean(samples[1000:50000,'beta'])
```

```
## [1] 0.6070751
```

```
quantile(samples[1000:50000,'beta'], c(0.025,0.975))
```

```
##      2.5%     97.5%
## 0.5948491 0.6191422
```

```
#
# Posterior Mean and 95% Percentile Interval: Gamma
mean(samples[1000:50000,'gamma'])
```

```
## [1] 0.1992271
```

```
quantile(samples[1000:50000,'gamma'], c(0.025,0.975))
```

```
##      2.5%     97.5%
## 0.1953116 0.2032225
```

```
knitr::knit_exit()
```