Class1StatsTopics

Table of contents

Quarto

```
# install.packages('R2jags', dependencies = TRUE) # installs coda too #
# install.packages('coda', dependencies = TRUE)
# install.packages('lattice',dependencies = TRUE)
# install.packages('MCMCvis', dependencies = TRUE)
```

```
library(R2jags)
```

```
Loading required package: rjags
```

```
Loading required package: coda
```

```
Linked to JAGS 4.3.1
```

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

```
Attaching package: 'R2jags'

The following object is masked from 'package:coda':

    traceplot
```

```
library(MCMCvis)
library(coda)
library(lattice)
```

```
# creating a sample o n size to fit out posterior distribution
# sample(x, size,...)
```

Coin example

```
# model definition
jags.mod.coin <- function() {
    Y ~ dbin(0.5, 10)  # our data model
    P8 <- ifelse(Y > 7, 1, 0)  # the probability of interest, aka 8 or higher out of 10
    ## the ifelse gives the first value if the statement is true and
    ## the second value if the statement is false.  we are not forced
    ## to use the binary 1 and 0, we can use any value
}
## jag has 2 syntaxes the first one is a stochastic dependence ~ the
## second one is a logical dependence <-

## in the jag binom we use the mu parameter and the tau parameter the
## tau parameter is called the precision and is 1/ sigma( ) ^2
```

```
## Generating 100 samples without discarding one with 1 chain

jags.mod.fit.coin <- jags(data = list(), model.file = jags.mod.coin, parameters.to.save =
    "P8"), n.chains = 1, DIC = FALSE, n.burnin = 0, n.iter = 100)  ## here we make DICE=Fa
```

```
module glm loaded

module dic loaded
```

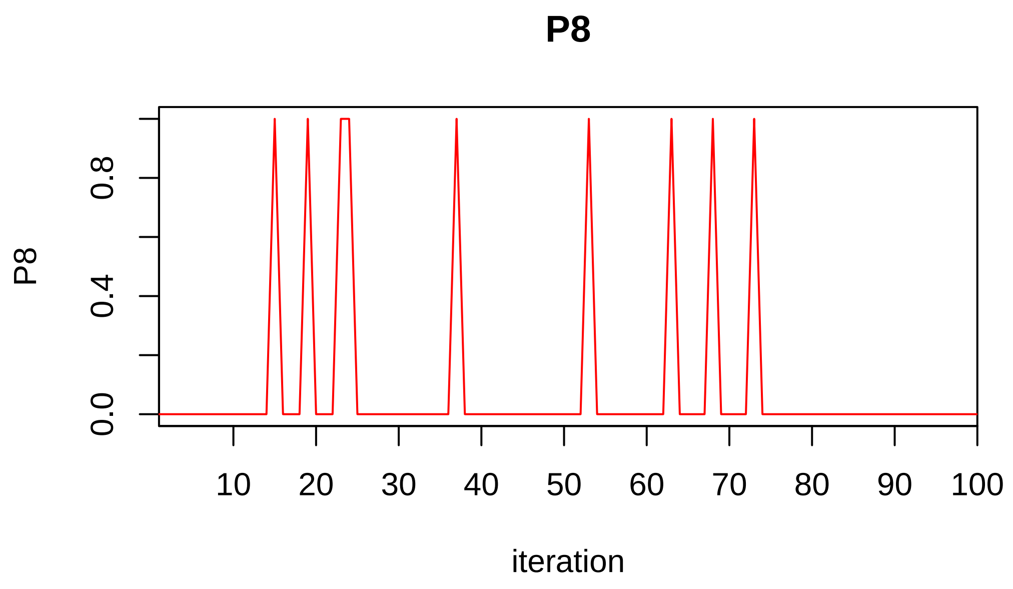
```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 0
   Unobserved stochastic nodes: 1
   Total graph size: 8

Initializing model
```

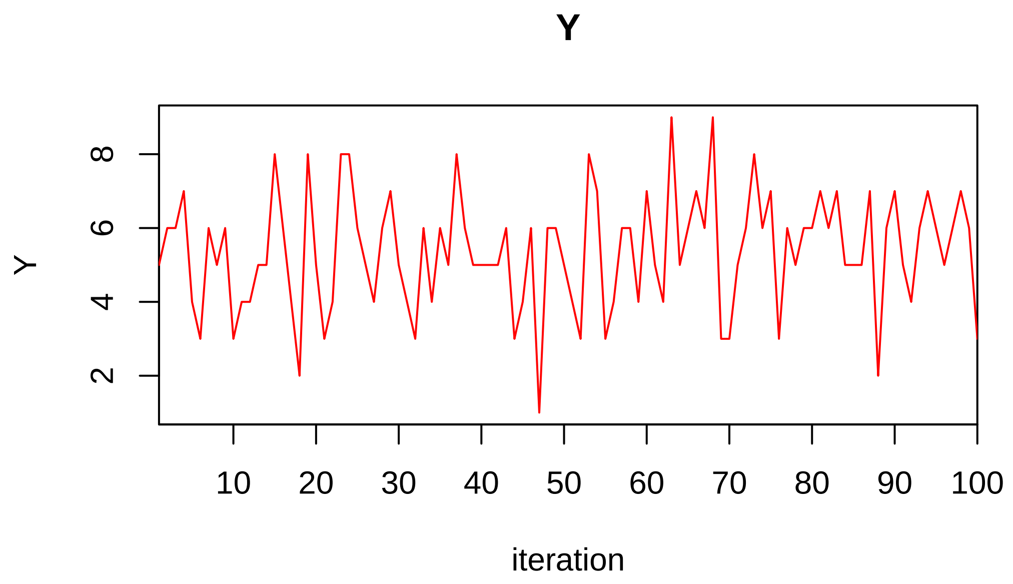
```
# To get the numerical summary of the above model run we use the print
# function

print(jags.mod.fit.coin)
```

```
Inference for Bugs model at "C:/Users/ANDREM~1/AppData/Local/Temp/RtmpSkNrjV/model51dc2219602
 1 chains, each with 100 iterations (first 0 discarded)
 n.sims = 100 iterations saved
   mu.vect sd.vect  2.5% 25% 50% 75% 97.5%
P8    0.09   0.288 0.000   0 0.0   0     1
Y     5.36   1.592 2.475   4 5.5   6     8
```

```
## this gives us the mean, standard deviation and some quantiles of the
## generated samples
```

```
# We can also plot the simulated samples using the traceplot function

traceplot(jags.mod.fit.coin)
```

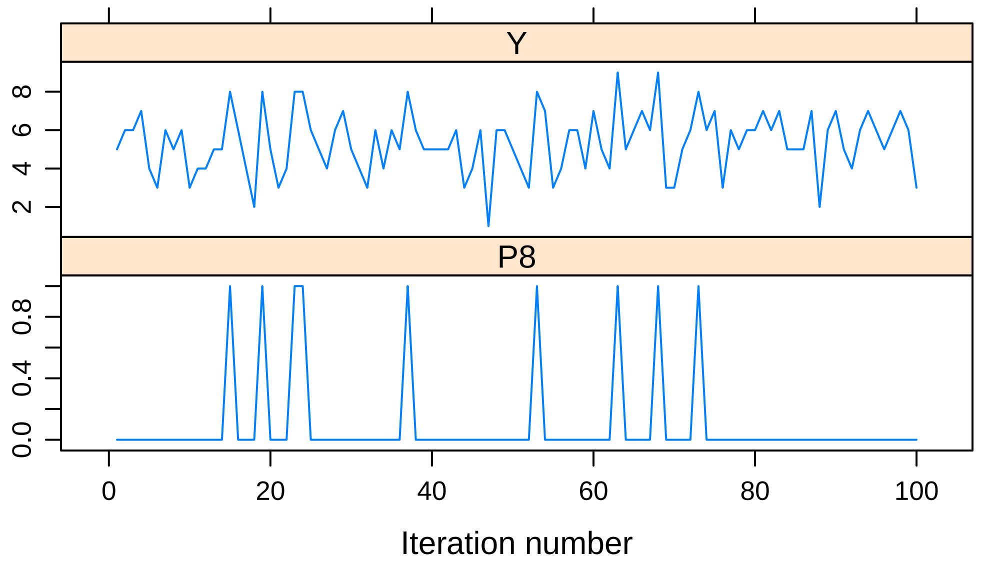

P8
P8
0.0
0.4
0.8
10
20
30
40
50
60
70
80
90
100
iteration


Y
Y
2
4
6
8
10
20
30
40
50
60
70
80
90
100
iteration

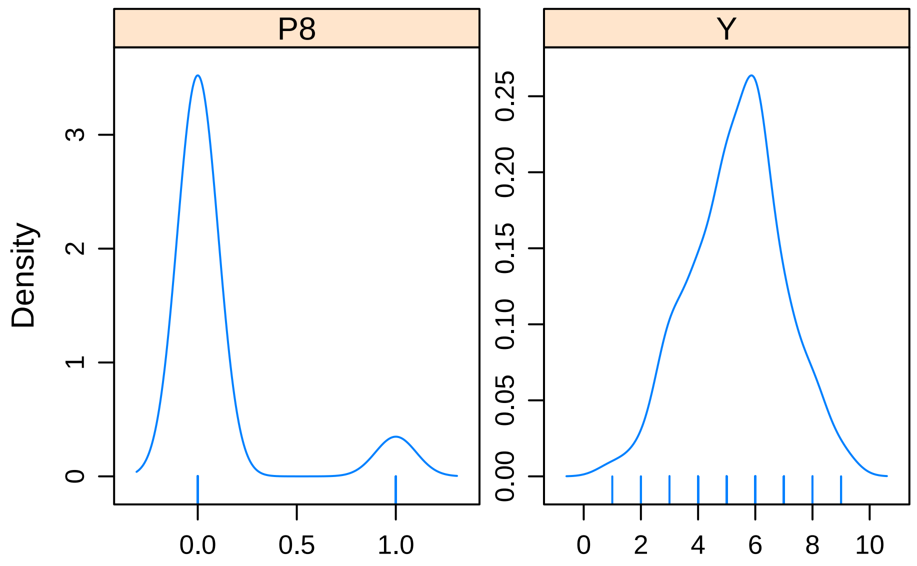
```
## For more diagnostics and visualisation tools we can convert the
## output of the jags function into an MCMC object. Using the MCMC
## object we can look at numerical summaries, traceplots and density
## plots.

# convert into MCMC object
jagsfit.mcmc.coin <- as.mcmc(jags.mod.fit.coin)
# get numerical summary
summary(jagsfit.mcmc.coin)
```

```
Iterations = 1:100
Thinning interval = 1
Number of chains = 1
Sample size per chain = 100

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

   Mean     SD Naive SE Time-series SE
P8 0.09 0.2876  0.02876        0.03649
Y  5.36 1.5924  0.15924        0.12718

2. Quantiles for each variable:

    2.5% 25% 50% 75% 97.5%
P8 0.000   0 0.0   0     1
Y  2.475   4 5.5   6     8
```

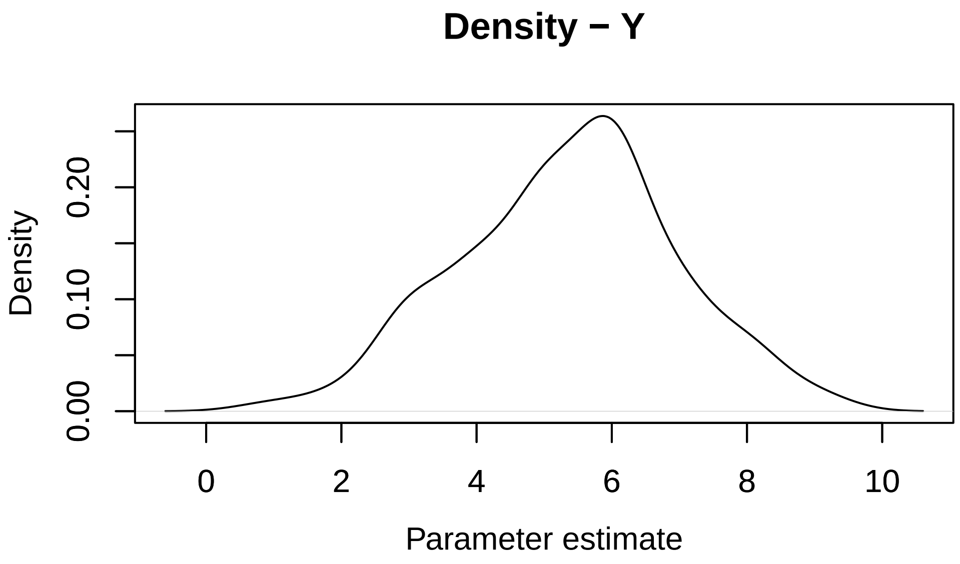
```
## The functions xyplot and densityplot of the lattice package give us
## trace plots and density plots of all the parameters.

# get traceplots
xyplot(jagsfit.mcmc.coin)
```



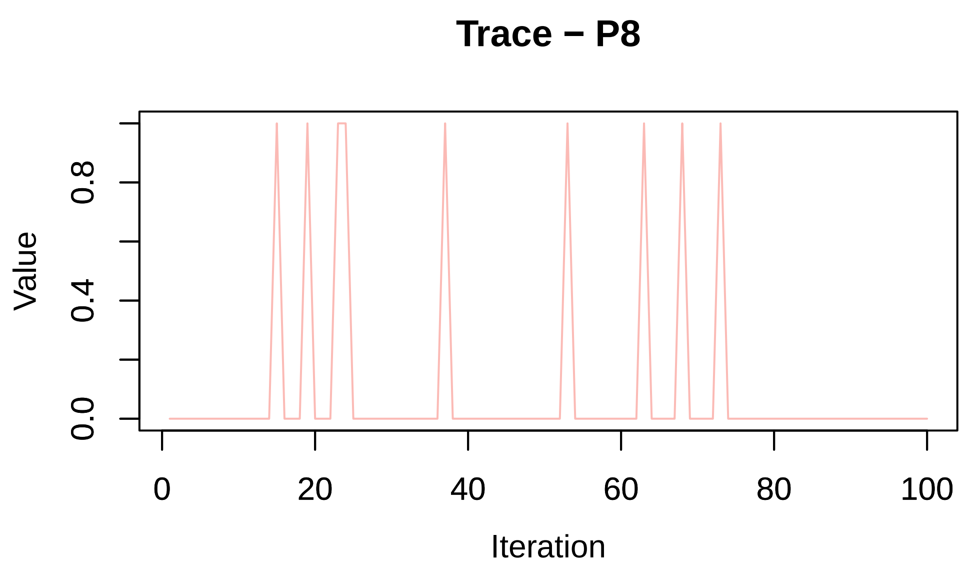
```
# get density estimate
densityplot(jagsfit.mcmc.coin)
```



```
##If we want to concentrate on individual parameters, we can use the MCMCtrace function of

MCMCtrace(jagsfit.mcmc.coin,
params = 'Y', # parameter of interest
type = 'density', # density plot
ind = TRUE, # separate density lines for each chain
pdf = FALSE) # plots are NOT exported into a pdf
```



```
MCMCtrace(jagsfit.mcmc.coin,
params = 'P8',
type = 'trace',
ind = TRUE,
pdf = FALSE)
```



```
## We can also extract summaries of interest. For example, we can get
## the point estimate, and the two endpoints of a 95% credible interval
## using the following

jags.mod.fit.coin$BUGSoutput$summary[, 1]  # mean
```

```
  P8    Y
0.09 5.36
```

```
jags.mod.fit.coin$BUGSoutput$summary[, 3]  # 2.5 percentile
```

```
      P8        Y
0.000000 2.424786
```

```
jags.mod.fit.coin$BUGSoutput$summary[, 7]  # 97.5 percentile
```

```
P8  Y
 1  8
```

Exercise 1

```
## increasing the number of iterations the mean estimate is much closer
## to the truth, more reliable estimates
jags.mod.fit.coin <- jags(data = list(), model.file = jags.mod.coin, parameters.to.save =
    "P8"), n.chains = 1, DIC = FALSE, n.burnin = 0, n.iter = 1e+05)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 0
   Unobserved stochastic nodes: 1
   Total graph size: 8

Initializing model
```

```
print(jags.mod.fit.coin)
```

```
Inference for Bugs model at "C:/Users/ANDREM~1/AppData/Local/Temp/RtmpSkNrjV/model51dc3a851f
 1 chains, each with 1e+05 iterations (first 0 discarded), n.thin = 100
 n.sims = 1000 iterations saved
   mu.vect sd.vect 2.5% 25% 50% 75% 97.5%
P8   0.047   0.212    0   0   0   0     1
Y    4.923   1.609    2   4   5   6     8
```

```
## it becomes more accurate as we can see that the mean is closer to
## the expected real value of 5
```

Exercise 2

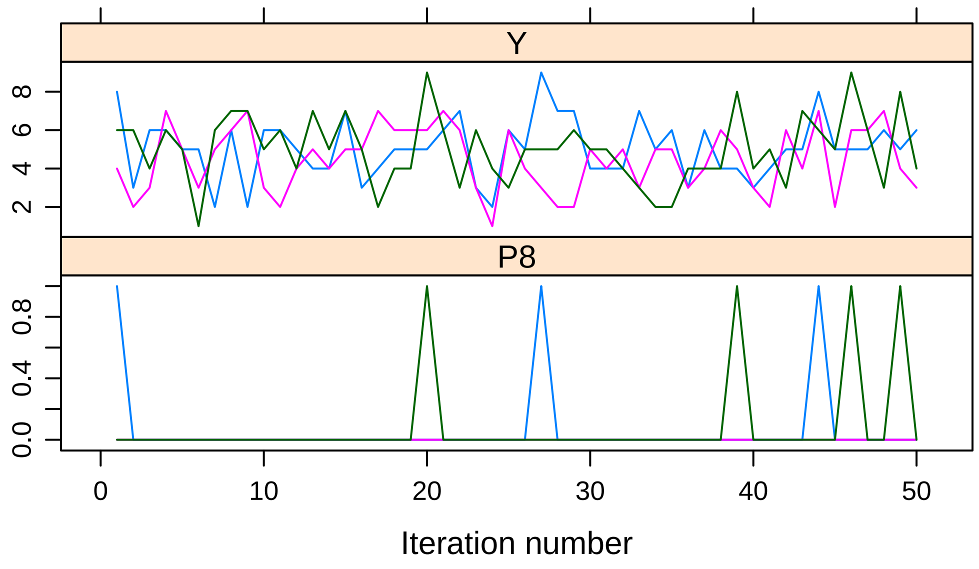
```
## number of chains, how much times does jags runs the simulation
## independently of each other

jags.mod.fit.coin <- jags(data = list(), model.file = jags.mod.coin, parameters.to.save =
    "P8"), n.chains = 3, DIC = FALSE, n.burnin = 0, n.iter = 50)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 0
   Unobserved stochastic nodes: 1
   Total graph size: 8

Initializing model
```

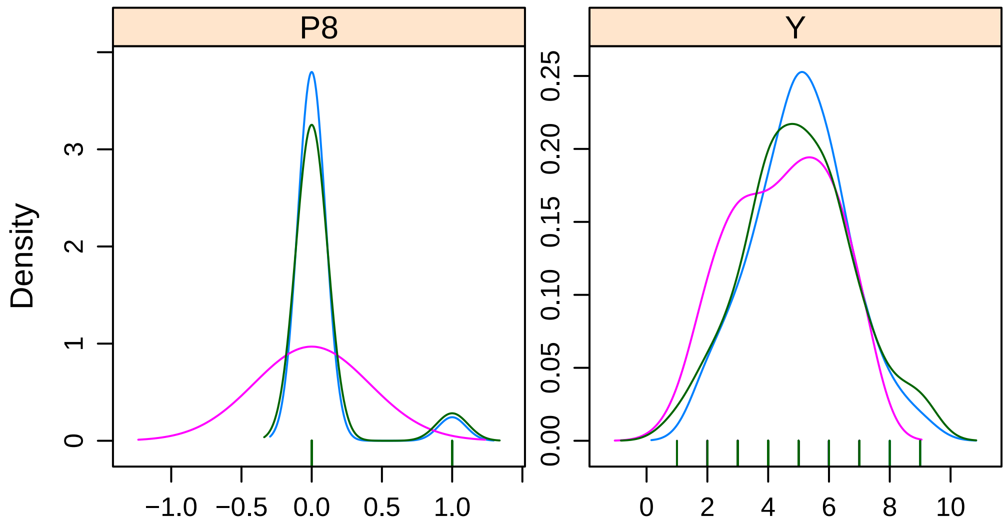
```
# convert into MCMC object
jagsfit.mcmc.coin <- as.mcmc(jags.mod.fit.coin)
# get numerical summary
summary(jagsfit.mcmc.coin)
```

```
Iterations = 1:50
Thinning interval = 1
Number of chains = 3
Sample size per chain = 50

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

      Mean     SD Naive SE Time-series SE
P8 0.04667 0.2116  0.01728        0.01717
Y  4.84667 1.6779  0.13700        0.13549

2. Quantiles for each variable:

   2.5% 25% 50% 75% 97.5%
P8    0   0   0   0     1
Y     2   4   5   6     8
```

```
# get trace plots
xyplot(jagsfit.mcmc.coin)
```



```
# get density estimate
densityplot(jagsfit.mcmc.coin)
```



```
## the outputs are somewhat close for 2 chains but the third one seems
## to not have converged
```

Exercise 3

```
# model definition
jags.mod.clinicalTrial <- function() {
    Y ~ dbin(0.7, 30)  # our data model of prob 0.7 and 30 number of trials
    P15 <- ifelse(Y < 15, 1, 0)  # the probability of interest, aka 15 or less positive re
}

jags.mod.fit.clinicalTrial <- jags(data = list(), model.file = jags.mod.clinicalTrial,
    parameters.to.save = c("Y", "P15"), n.chains = 3, DIC = FALSE, n.burnin = 2000,
    n.iter = 10000)  ## here we make DICE=False because our code does not contain the like
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 0
   Unobserved stochastic nodes: 1
   Total graph size: 8

Initializing model
```

```
print(jags.mod.fit.clinicalTrial)
```

```
Inference for Bugs model at "C:/Users/ANDREM~1/AppData/Local/Temp/RtmpSkNrjV/model51dc59cb253
 3 chains, each with 10000 iterations (first 2000 discarded), n.thin = 8
 n.sims = 3000 iterations saved
    mu.vect sd.vect 2.5% 25% 50% 75% 97.5%  Rhat n.eff
P15   0.007   0.085    0   0   0   0     0 1.042  1500
Y    21.030   2.456   16  19  21  23    26 1.002  1800

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
```

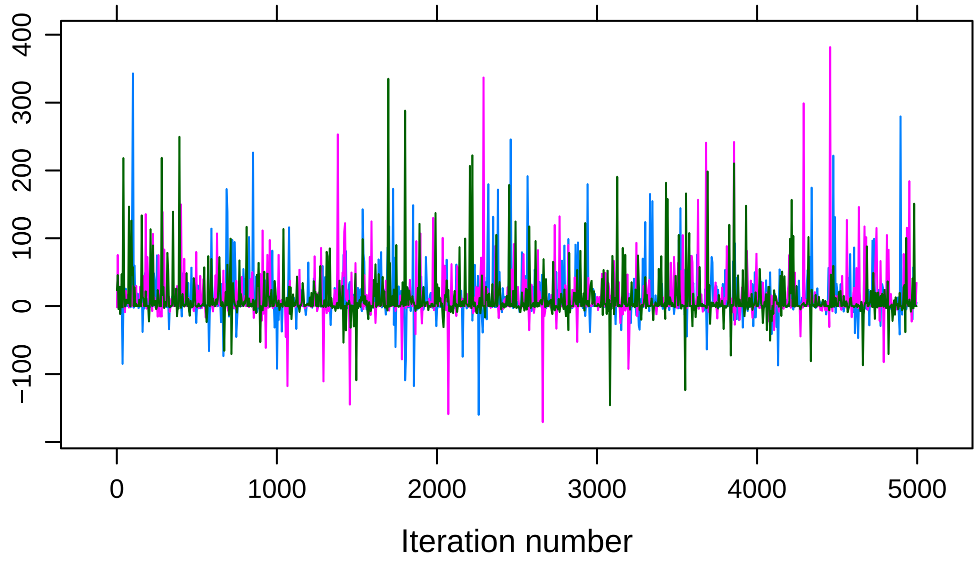
```
# the probability of 15 or less trials is on average 0.006
```

Exercise 4

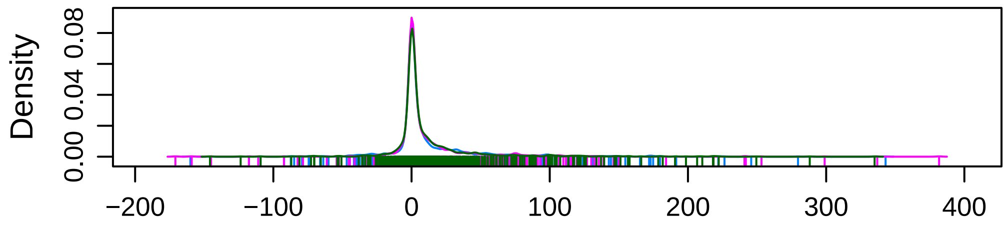
```
# model definition with var 1 and sd 2 attention to how normal
# functions parameters are different in JAGS compared to R
jags.norm3 <- function() {
    Y ~ dnorm(1, 1/4)  # 1/var=1 1/sd^2= 1/4
    X = Y^3
}


jags.mod.fit.norm3 <- jags(data = list(), model.file = jags.norm3, parameters.to.save = c(
    n.chains = 3, DIC = FALSE, n.burnin = 0, n.iter = 5000)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 0
   Unobserved stochastic nodes: 1
   Total graph size: 6

Initializing model
```

```
# jags.mod.fit <- jags(data = list(), model.file = jags.mod.normal3,
# parameters.to.save = c('x'),n.chains=1, DIC=FALSE, n.burnin=0,n.iter
# = 10000) print(jags.mod.fit[drop=F]) # look at mean of x


############ all of these bellow are unecessary but there is not harm
############ in checking the plots convert into MCMC object
jagsfit.mcmc.norm3 <- as.mcmc(jags.mod.fit.norm3)
# get numerical summary summary(jagsfit.mcmc.norm3)

## need to change the entry variable get trace plots
xyplot(jagsfit.mcmc.norm3)
```



```
# get density estimate
densityplot(jagsfit.mcmc.norm3)
```



```r
summary(jagsfit.mcmc.norm3)
```

```
Iterations = 1:4996
Thinning interval = 5
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
       11.8380        36.6952         0.6700         0.6569

2. Quantiles for each variable:

     2.5%       25%       50%       75%     97.5%
-27.44961  -0.03757   0.94261  11.82280 113.34535
```

```r
print(jags.mod.fit.norm3[drop = F])
```

```
$model
JAGS model:

model
{
    Y ~ dnorm(1, 1/4)
    X = Y^3
}

$BUGSoutput
Inference for Bugs model at "C:/Users/ANDREM~1/AppData/Local/Temp/RtmpSkNrjV/model51dc6b2b5c
 3 chains, each with 5000 iterations (first 0 discarded), n.thin = 5
 n.sims = 3000 iterations saved
  mean   sd  2.5% 25% 50%  75% 97.5% Rhat n.eff
X 11.8 36.7 -27.4   0 0.9 11.8 113.3    1  3000

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

$parameters.to.save
```

```
[1] "X"

$model.file
[1] "C:/Users/ANDREM~1/AppData/Local/Temp/RtmpSkNrjV/model51dc6b2b5c1c.txt"

$n.iter
[1] 5000

$DIC
[1] FALSE
```

Trace plot note

We need to check both the trace plots and the coefficients to check for the scale reduction factor, sometimes the factor will be good but we have no conversion.

Drug trial exercise without data

```
### Drug example in JAGS model
jags.mod.drug <- function() {
    theta ~ dbeta(9.2, 13.8)  # prior for the unknown parameter
    y ~ dbin(theta, 20)  # the data model, since it is a positive or negative we use the b
    P.crit <- ifelse(y >= 15, 1, 0)  # quantity of interest, we want to know how the proba
}
```

```
##Even though we don't have any available data, monitoring the parameters y and P.crit wil
##plot the predictive distribution of y, and get a point estimate for the predictive proba
##patients will experience a positive response.


jags.mod.fit.drug <- jags(data = list(), n.iter = 100000, DIC=FALSE, ## large ammount of i
parameters.to.save = c('theta','y','P.crit'),
model.file = jags.mod.drug,n.chains=1)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 0
   Unobserved stochastic nodes: 2
   Total graph size: 10

Initializing model
```

Fying bombs

```
## the gamma distribution is good for when we need a vague prior with
## only positives values on the real line (IR)

## model

jags.mod.bomb <- function() {

    ## likelihood
    for (i in 1:N) {
        y[i] ~ dpois(lambda)  ## the likelihood is only to set up what distribution our da
    }

    ## prior

    lambda ~ dgamma(0.001, 0.001)

}  ## ends the model definition

y <- c(rep(0, 229), rep(1, 211), rep(2, 93), rep(3, 35), rep(4, 7), rep(7,
    1))  ## definite our dataset/occurrences
N <- length(y)
bomb.data <- list("N", "y")

# initial values (for 2 chains)

## since the values we got from the occurrences we can see that the
## values are very low
bomb.inits1 <- list(lambda = 0.8)
bomb.inits2 <- list(lambda = 1)
bomb.inits <- list(bomb.inits1, bomb.inits2)

# parameters to monitor
jags.param <- c("lambda")

# model fit
```

```
jags.mod.fit.bomb <- jags(data = bomb.data, inits = bomb.inits, parameters.to.save = jags.
    n.chains = 2, n.iter = 10000, model.file = jags.mod.bomb)  ## if we don't set our burn
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 576
   Unobserved stochastic nodes: 1
   Total graph size: 579

Initializing model
```

```
# look at numerical summary
print(jags.mod.fit.bomb)
```

```
Inference for Bugs model at "C:/Users/ANDREM~1/AppData/Local/Temp/RtmpSkNrjV/model51dc16b41e
 2 chains, each with 10000 iterations (first 5000 discarded), n.thin = 5
 n.sims = 2000 iterations saved
          mu.vect sd.vect     2.5%      25%      50%      75%    97.5%  Rhat
lambda      0.933   0.040    0.857    0.907    0.933    0.961    1.011 1.002
deviance 1466.161   1.359 1465.190 1465.277 1465.640 1466.531 1469.901 1.002
         n.eff
lambda    1100
deviance  1400

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

DIC info (using the rule, pD = var(deviance)/2)
pD = 0.9 and DIC = 1467.1
DIC is an estimate of expected predictive error (lower deviance is better).
```

```
# point estimate for the parameter is 0.933
```