JAGS example

Paul M

3/27/2020

Introduction to JAGS ("Just Another GIBBS Sampler")

[Based on one of the examples at http://www.johnmyleswhite.com/notebook/2010/08/20/using-jags-in-r-with-the-rjags-package/]

First you must install jags on your computer! rjags won't run without it. e.g., in the terminal using homebrew: brew install jags" Then install the rjags package:

```
#install.packages("rjags")
library(rjags)
```

```
## Loading required package: coda

## Linked to JAGS 4.3.0

## Loaded modules: basemod,bugs
```

```
library(coda)
```

An example using normal rvs, where we are infering the mean and variance:

Let's asusme we have a set of rvs that we know are normally distributed, but wer wish to infer their mean and variance. Generate a test set of data

```
N <- 1000
x <- rnorm(N, 0, 2)

write.table(x,
            file = 'example1.data',
            row.names = FALSE,
            col.names = FALSE)
```

Now we need to write a model specification in JAGS syntax. Put the model specification (below) in a file called example1.bug. The complete model looks like this:

model { for (i in 1:N) {
x[i] ~ dnorm(mu, tau)
}
mu ~ dnorm(0, .0001)
tau <- pow(sigma, -2)
sigma ~ dunif(0, 100)
}

- The first line says that you are specifying a model.
- Then you set up the model for every single data point using a for loop. Here, we say that x[i] is distributed normally with mean mu and precision tau. Note that rjags works with precision rather than variance! (Precision=1/variance) *Then we specify priors for mu and tau.
- mu is assumed to be distributed normally with mean 0 and standard deviation 100. This is an example of a non-informative prior.
- Then we specify tau as a deterministic function (hence the deterministic <- instead of the distributional ~) of sigma, after raising sigma to the -2 power. Then we say that sigma has a uniform prior over the interval [0,100].

Now we invoke the model:

```
# Set up our model object in R
jags <- jags.model('example1.bug',   # specification file
                   data = list('x' = x,   # the data (must use same names as trhe model spec. file)
                               'N' = N),
                   n.chains = 4,   # how many parallel chains to run
                   n.adapt = 100   # we will use adaptive sampling, removing the first 100 iterations
                   )
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1000
##    Unobserved stochastic nodes: 2
##    Total graph size: 1009
##
## Initializing model
```

```
update(jags, 1000)    # run another 1000 iterations. The update function is used to add more iterations

samps <- jags.samples(jags,   # draw 1000 samples from the output for the requested variables
            c('mu', 'tau'),
            1000)

summary(samps)
```

```
##     Length Class   Mode
## mu  4000   mcarray numeric
## tau 4000   mcarray numeric
```

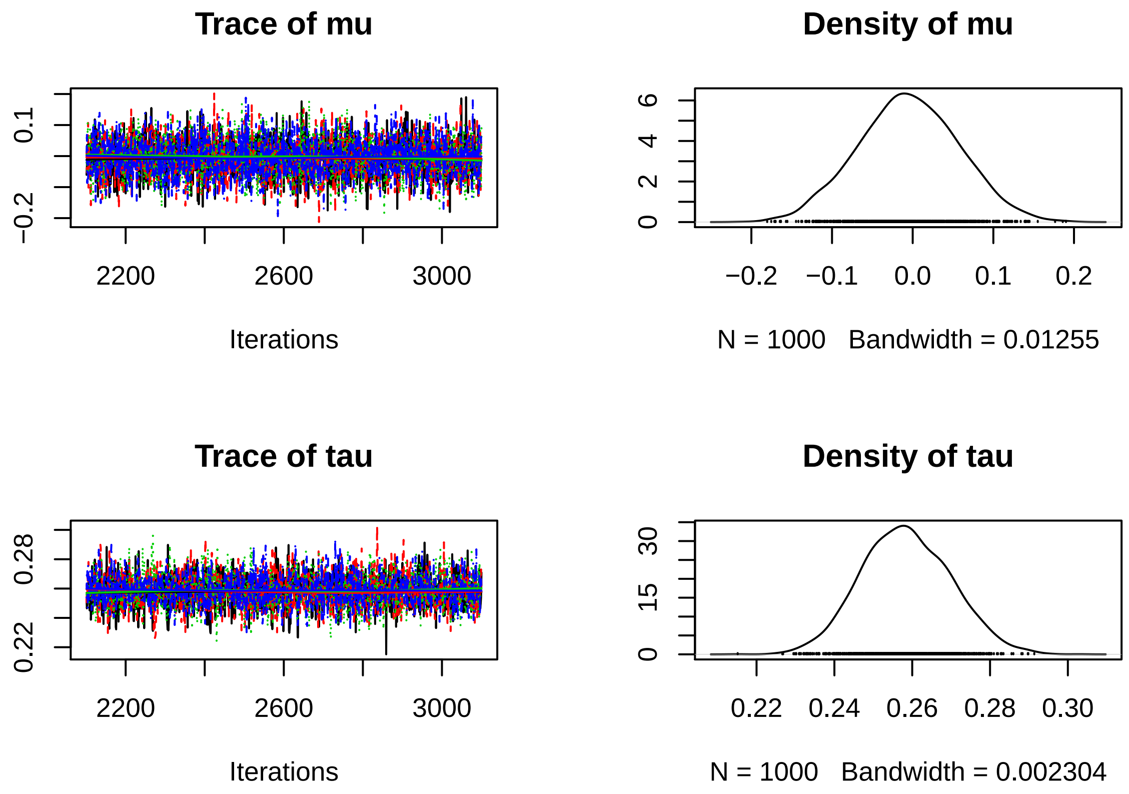
You can also set it for use of coda diagnostics for covergence (which also seems to provide a better summary of the output)

```
samps2 <- coda.samples( jags, c('mu','tau'), 1000 )
summary(samps2)
```

```
##
## Iterations = 2101:3100
## Thinning interval = 1
## Number of chains = 4
## 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
## mu  -0.005113 0.06220 0.0009834      0.0009886
## tau  0.258463 0.01142 0.0001805      0.0001949
##
## 2. Quantiles for each variable:
##
##         2.5%      25%       50%     75%  97.5%
## mu  -0.1248 -0.04706 -0.005097 0.03765 0.1167
## tau  0.2369  0.25051  0.258155 0.26632 0.2811
```

```
plot(samps2)
```



```
show(gelman.diag(samps2))
```

```
## Potential scale reduction factors:
##
##     Point est. Upper C.I.
## mu           1          1
## tau          1          1
##
## Multivariate psrf
##
## 1
```