Using JAGS to run MCMC

Math 315, Fall 2019

For the remainder of the term we will use Just Another Gibbs Sampler (JAGS) to draw MCMC samples from our posterior distributions. In class, we won't be able to discuss every distribution or function you may need to specify you JAGS model, so be sure to keep the URL to the JAGS user's manual handy: https://github.com/aloy/math315-fall2019/blob/master/jags_user_manual.pdf.

Key steps

Every JAGS program has five key components:

1. Specify the model as a string (or save it to a text file)
2. Compile the model using `jags.model()`
3. Draw burn-in samples using `update()`
4. Draw posterior samples using `coda.samples()`
5. Inspect/summarize the results using the `plot()` and `summary()`

Example: Rocket launches

Let's look at a small example first by revisiting our rocket launch example. Let $Y = \#$ successful launches. Then the model is given by:

- Likelihood: We have n iid Bernoulli(θ) trials
- Prior: $\theta \sim \text{Unif}(0.1, 0.9)$

Step 0. Load rjags and data set

```
library(rjags)

# Load data
launches <- read.table("https://www4.stat.ncsu.edu/~wilson/BR/table21.txt", header = TRUE)
y <- launches$Outcome
n <- nrow(launches)
```

Step 1. Specify the model as a string

```
model_string <- textConnection("model{

  # Likelihood
  for(i in 1:n) {
    y[i] ~ dbern(theta)
  }

  # Prior
  theta ~ dunif(a, b)
```

```
}")
```

- Note: ~ indicates that a variable follow a given distribution
- Be sure to use `textConnection()` if you are not saving this model string as a separate text file.

Step 2. Compile the MCMC code

```
inits <- list(theta = 0.5)
data  <- list(y = y, n = n, a = 0.1, b = 0.9)
model <- jags.model(model_string, data = data,
                    inits = inits, n.chains = 1)
```

Note: initsanddata‘ should be **lists**

Step 3. Draw burn-in samples

```
update(model, 1000)
```

Note: Add the argument `progress.bar = "none"` to silence the progress bar in your .Rmd files

Step 4. Draw posterior samples

```
samples <- coda.samples(
  model,
  variable.names = "theta",
  n.iter = 10000,
  progress.bar = "none"
)
```

Note: `variable.names` expects a character vector with the names of the variables to be monitored

Step 5. Inspect/summarize the results

```
summary(samples)
plot(samples)
```

Note: `samples` will be a matrix with one column in this case.

Your turn 1: Two-sample model (redux)

In this first example, we'll return to the two-sample model we explored last week. Recall that the researchers were interested in whether a color distracter slowed student reaction time. The model is given by:

$$
\begin{aligned}
Y_i &\overset{\text{iid}}{\sim} \mathcal{N}(\mu, \sigma^2), \ i = 1, \ldots, 20 \qquad \text{(standard group)} \\
Y_j &\overset{\text{iid}}{\sim} \mathcal{N}(\mu + \delta, \sigma^2), \ j = 21, \ldots, 40 \qquad \text{(color group)} \\
\mu &\sim \mathcal{N}(0, 100^2) \\
\delta &\sim \mathcal{N}(0, 100^2) \\
\sigma^2 &\sim \text{InvGamma}(0.01, 0.01)
\end{aligned}
$$

```
stroop <- read.csv("http://aloy.rbind.io/data/stroop_game.csv")
```

Complete the following code to do the following:

i. Specify the model as a text string. Note that `tau` = $1/\sigma^2$.

ii. Compile the MCMC code so that is runs a single chain.

iii. Update the model to draw 1000 burn-in samples.

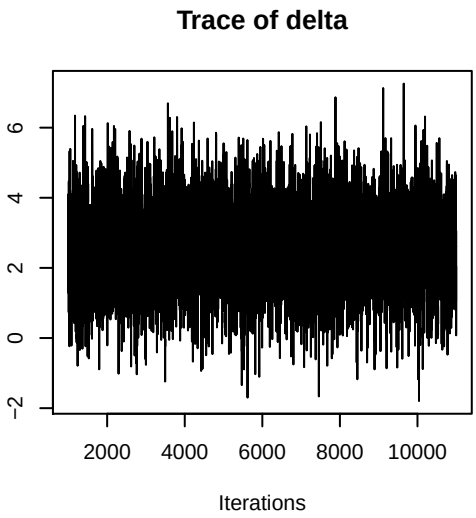
iv. Draw 10,000 MCMC samples after burn-in, and monitor the parameters μ, δ, and σ.

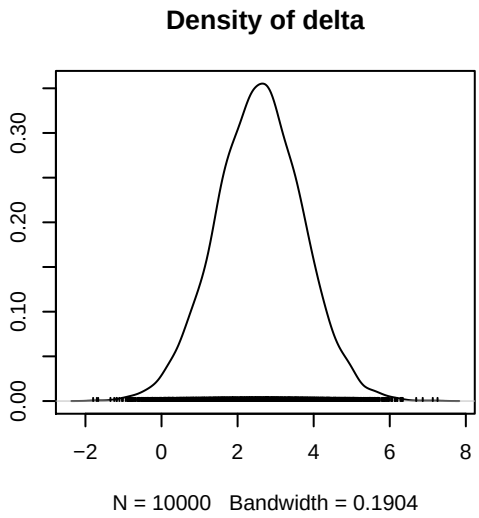
```
# Load rjags
library(rjags)

# Data management
y <- stroop$Time
y.std <- y[stroop$Type == "Standard"]
y.col <- y[stroop$Type == "Color"]
n <- length(y.std)
m <- length(y.col)

# Specify the model
stroop_model_string <- textConnection("model{
  # Specify the likelihood
  for(i in 1:n) {
    y.std[i] ~ dnorm(mu, tau)
  }

  for(i in 1:m) {
    y.col[i] ~ dnorm(mu + delta, tau)
  }

  # Specify the priors
  mu ~ dnorm(0, 0.0001)
  delta ~ dnorm(0, 0.0001)
  tau ~ dgamma(.01, .01)

  # Calculate sigma
  sigma <- 1 / sqrt(tau)
}")

# Compile the MCMC code
inits <- list(mu = mean(y), delta = 0, tau = 1 / sd(y))
```

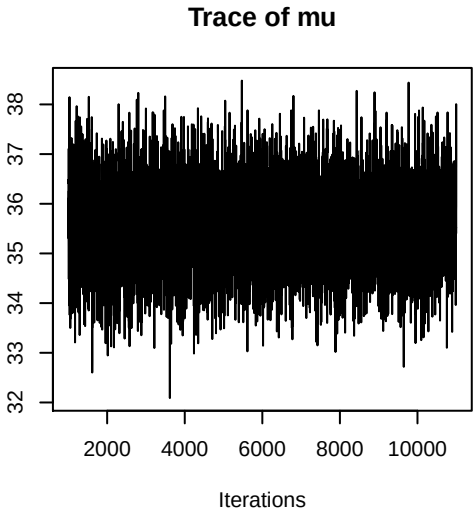
```r
stroop_data  <- list(y.std = y.std, y.col = y.col, n = n, m = m)
stroop_model <- jags.model(stroop_model_string, data = stroop_data,
                           inits = inits, n.chains = 1, quiet = TRUE)


# Draw 1000 burn-in samples
update(stroop_model, 1000)

# Draw 10000 posterior samples
stroop_samples <- coda.samples(stroop_model,
  variable.names = c("mu", "delta", "sigma"),
  n.iter = 10000,
  progress.bar = "none"
)

# Inspect/summarize the results
summary(stroop_samples)
```

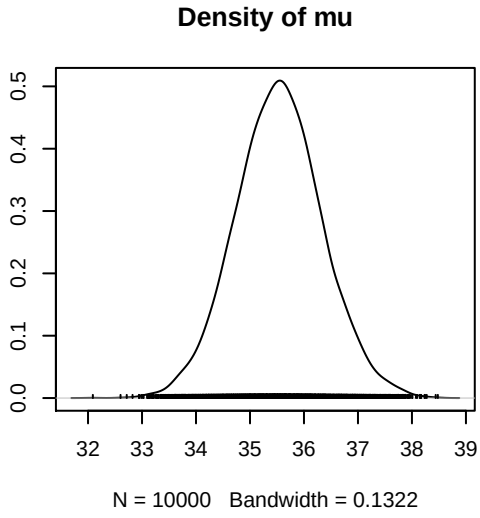
```
##
## Iterations = 1001:11000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##         Mean     SD Naive SE Time-series SE
## delta  2.569 1.1332 0.011332       0.019144
## mu    35.538 0.7998 0.007998       0.013703
## sigma  3.594 0.4294 0.004294       0.004627
##
## 2. Quantiles for each variable:
##
##         2.5%    25%    50%    75%  97.5%
## delta  0.306  1.814  2.576  3.335  4.814
## mu    33.972 35.004 35.533 36.059 37.141
## sigma  2.880  3.285  3.552  3.854  4.541
```

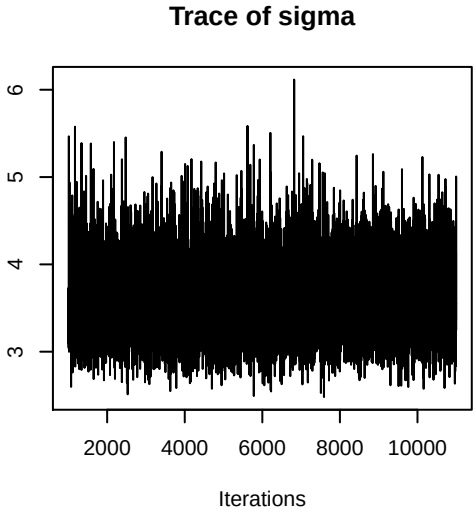
```r
plot(stroop_samples)
```

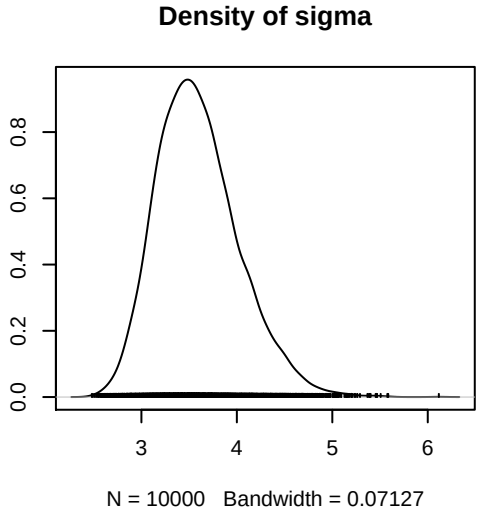












Follow-up questions:

1. Have the chains for μ, δ, and σ converged? How do you know?

   Answer: Yes, the chains have converged. All of the trace plots have a clear center of mass and appear to be "white noise" around this center. (In other words, they look like fuzzy catepillars.)

2. Report the posterior mean and SD for each parameter.

   Answer: The mean and SD are given in the output of `summary()`.

3. Report a 95% credible interval for δ. What does this say about the difference in average completion time between groups?

   Answer: You can report the 95% credible interval given in the `summary()` output. Notice that this interval does not contain 0, and indicates that there is at least a 95% chance (in fact, at least a 97.5% chance) that the color distracter does increase average time to complete the task.

Your turn 2: Survival analysis

Researchers are interested in modeling the survival times, measured in weeks, of patients who were diagnosed with leukemia. The patients were classified according to two characteristics of white bloods cells. In this example, we will only examine those whose blood is AG+. The sample consists of $n = 17$ times in weeks from diagnosis to death.

```
times <- c(65, 156, 100, 134, 16, 108, 121, 4, 39, 143, 56, 26, 22, 1, 1, 5, 65)
```

After discussing the problem with the researchers, you decide to use the Weibull distribution to model the survival time of patient i, Y_i. The Weibull PDF and CDF are given below for your reference, but are available as functions in JAGS.

$$f(y_i|v,\lambda) = \lambda v y_i^{v-1} e^{-\lambda y_i^v}, \; y_i, v, \lambda > 0$$
$$F(y_i|v,\lambda) = 1 - e^{-\lambda y_i^v}$$

In addition, you decide to place independent gamma priors on λ and v, and since you have little prior information about the parameters, so you decide to use Gamma$(0.01, 0.01)$ for both.

Set up question: According to the JAGS user manual, what function should you use for the Weibull density.

Answer: Both `dweib()` and `dweibull()` are Weibull density functions known to JAGS.

Your task: Fit the above model in JAGS and plot the posterior density of the 24-week survival rate, which is defined as $1 - F(24|v,\lambda)$ where F denotes the CDF of a Weibull distribution. (Check the manual for the function that gives the Weibull CDF!) Summarize your findings about the 24-hour survival rate.

```
# Specify the model string
survival_model_string <- textConnection("model{
  # Specify the likelihood
  for(i in 1:n) {
    y[i] ~ dweib(v, lambda)
  }

  # Specify the priors
  v ~ dgamma(0.01, 0.01)
  lambda ~ dgamma(0.01, 0.01)

  # Calculate 24-week survival rate
  s24 <- 1 - pweib(24, v, lambda)
}")
```

```r
# Compile the MCMC code
inits <- list(v = 1, lambda = 1)
survival_data  <- list(y = times, n = length(times))
survival_model <- jags.model(survival_model_string, data = survival_data,
                             inits = inits, n.chains = 3, quiet = TRUE)


# Draw 1000 burn-in samples
update(survival_model, 1000)

# Draw 5000 posterior samples
survival_samples <- coda.samples(survival_model,
  variable.names = "s24",
  n.iter = 5000,
  progress.bar = "none"
)

# Inspect/summarize the results
summary(survival_samples)
```

```
##
## Iterations = 2001:7000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 5000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean             SD       Naive SE Time-series SE
##      0.6407036      0.0957980      0.0007822      0.0035578
##
## 2. Quantiles for each variable:
##
##   2.5%    25%    50%    75%  97.5%
## 0.4408 0.5777 0.6458 0.7099 0.8097
```

```r
plot(survival_samples)
```