A Deeper Dive into JAGS

School of Information Studies
Syracuse University

Learning Goals

Describe the process of conducting Markov Chain Monte Carlo analysis as a way of generating posterior distributions for Bayesian analysis problems

Install and run JAGS and the rjags package; provide an existing model to JAGS for compilation

Run a Bayesian analysis of the difference in means between two independent samples

Interpret the results and diagnostics for a Bayesian analysis using JAGS

School of Information Studies
Syracuse University

Fun With rjags

rjags is the interface between R and JAGS

JAGS – Just another Gibbs sampler – is a variant on the BUGS language that was developed to facilitate MCMC analysis of unknown distributions

John Kruschke uses BUGS in his detailed book on Bayesian analysis

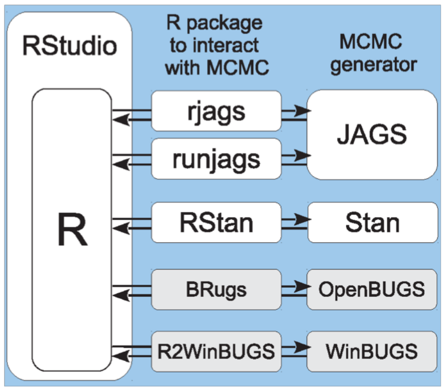


Image credit: J. Kruschke

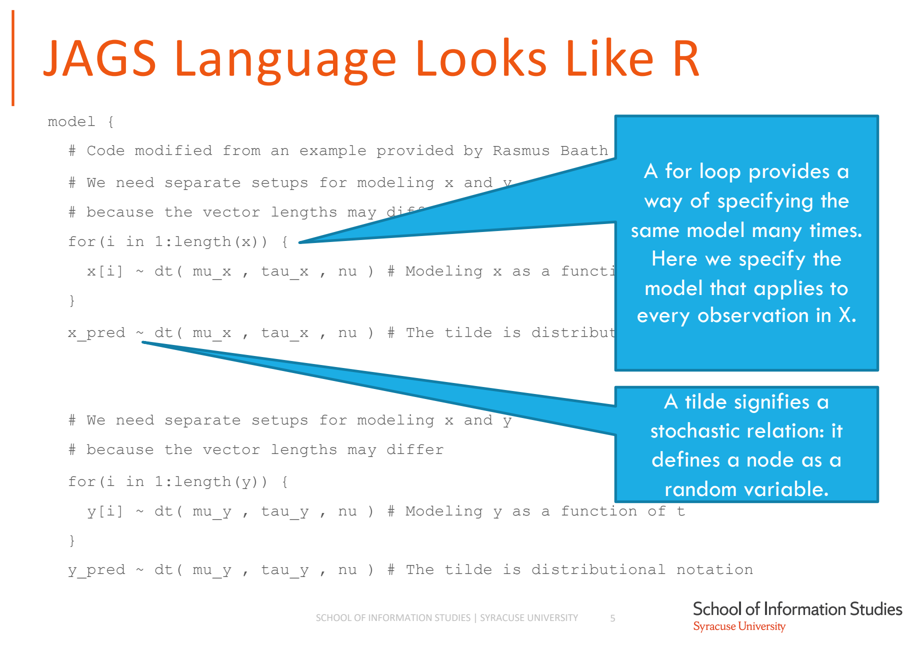
Learning by Example: Posterior Distribution for Two Independent Means

The t-distribution provides a “robust” alternative to the normal distribution for modeling two independent means

As a statistical term, robust means that the statistic performs well even in adverse conditions such as small samples. in the presence of outliers, or with unusual distributions

To develop a posterior distribution for two independent means we need to simultaneously estimate at least five things:

- The mean of group 1 and the mean of group 2 (modeled as t-distributions)
- The standard deviation of group 1 and the standard deviation of group 2
- A parameter to control degrees of freedom (heaviness of the tails of the t-distribution that models the means)



JAGS Language Looks Like R

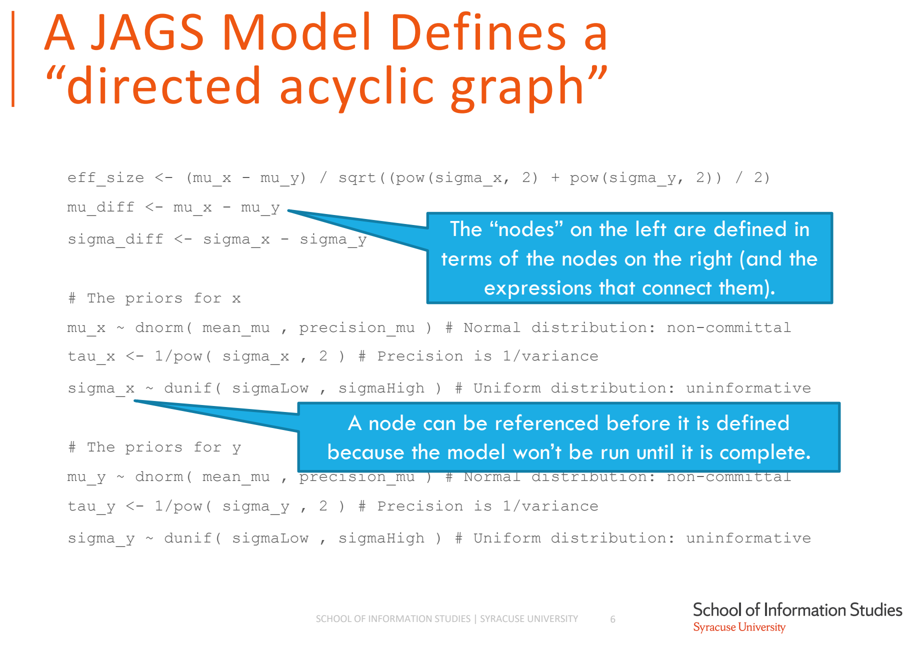
```
model {
  # Code modified from an example provided by Rasmus Baath
  # We need separate setups for modeling x and y
  # because the vector lengths may diff
  for(i in 1:length(x)) {
    x[i] ~ dt( mu_x , tau_x , nu ) # Modeling x as a functi
  }
  x_pred ~ dt( mu_x , tau_x , nu ) # The tilde is distribut


  # We need separate setups for modeling x and y
  # because the vector lengths may differ
  for(i in 1:length(y)) {
    y[i] ~ dt( mu_y , tau_y , nu ) # Modeling y as a function of t
  }
  y_pred ~ dt( mu_y , tau_y , nu ) # The tilde is distributional notation
```

A for loop provides a way of specifying the same model many times. Here we specify the model that applies to every observation in X.

A tilde signifies a stochastic relation: it defines a node as a random variable.

SCHOOL OF INFORMATION STUDIES | SYRACUSE UNIVERSITY 5

School of Information Studies
Syracuse University



A JAGS Model Defines a "directed acyclic graph"

```
eff_size <- (mu_x - mu_y) / sqrt((pow(sigma_x, 2) + pow(sigma_y, 2)) / 2)
mu_diff <- mu_x - mu_y
sigma_diff <- sigma_x - sigma_y

# The priors for x
mu_x ~ dnorm( mean_mu , precision_mu ) # Normal distribution: non-committal
tau_x <- 1/pow( sigma_x , 2 ) # Precision is 1/variance
sigma_x ~ dunif( sigmaLow , sigmaHigh ) # Uniform distribution: uninformative

# The priors for y
mu_y ~ dnorm( mean_mu , precision_mu ) # Normal distribution: non-committal
tau_y <- 1/pow( sigma_y , 2 ) # Precision is 1/variance
sigma_y ~ dunif( sigmaLow , sigmaHigh ) # Uniform distribution: uninformative
```

The "nodes" on the left are defined in terms of the nodes on the right (and the expressions that connect them).

A node can be referenced before it is defined because the model won't be run until it is complete.

SCHOOL OF INFORMATION STUDIES | SYRACUSE UNIVERSITY 6

School of Information Studies
Syracuse University

A JAGS model is compiled and then returned to R for use

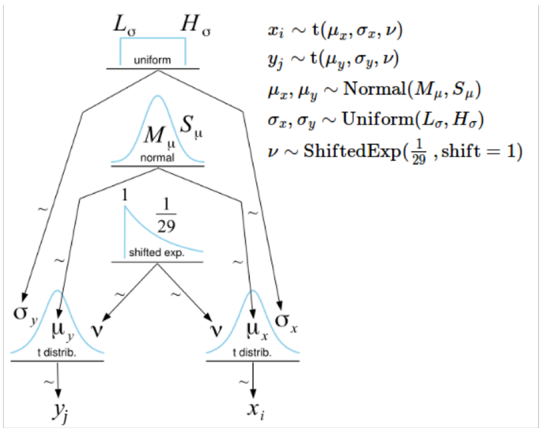
```
  # Set an exponentially distributed prior on nu that starts at 1.
  # 29 is the threshold between small values of df where t has heavy tails
  # and large values of df where t approximates normal distribution
  nu <- nuMinusOne+1
  nuMinusOne ~ dexp(1/29)
}
```

The complete model is enclosed in curly braces.

nu completes the list of five parameters that we must simultaneously explore.

We've used uniform, normal, and exponential dists.

Summary of the Model: Kruschke Notation



$$x_i \sim \mathrm{t}(\mu_x, \sigma_x, \nu)$$
$$y_j \sim \mathrm{t}(\mu_y, \sigma_y, \nu)$$
$$\mu_x, \mu_y \sim \mathrm{Normal}(M_\mu, S_\mu)$$
$$\sigma_x, \sigma_y \sim \mathrm{Uniform}(L_\sigma, H_\sigma)$$
$$\nu \sim \mathrm{ShiftedExp}(\tfrac{1}{29}, \mathrm{shift} = 1)$$

Image Credit: Rasmus Baath

Setup the parameters to compile the JAGS model: Priors

```
# Start by putting one group in x and the other in y
# Here we use mtcars, splitting by am
# Set up the priors
mean_mu = mean(c(x, y)) # Set the means

# Precision is inverse of variance
# mad is mean absolute deviation
precision_mu = 1 / (mad( c(x, y) )^2 * 1000000)
sigmaLow = mad( c(x, y) ) / 1000 # Bottom of the uniform dist
sigmaHigh = mad( c(x, y) ) * 1000 # Top of the uniform dist
```

Setup the parameters to compile the JAGS model: inits

```
inits_list <- list(
  mu_x = mean(x), mu_y = mean(y), # The initial means
  sigma_x = mad(x), sigma_y = mad(y), # The initial SDs
  nuMinusOne = 4) # A low initial guess at nu
```

Setup the parameters to compile the JAGS model: data

```
data_list <- list(
  x = x, y = y, # Data for each of the two groups
  mean_mu = mean_mu, # Means for each group
  precision_mu = precision_mu, # Precision for each group
  sigmaLow = sigmaLow, # Starting points for uniform dist
  sigmaHigh = sigmaHigh)
```

Parameters to Monitor: The list of posteriors we will get

```
# The parameters to monitor.
params <- c("mu_x", "mu_y", "mu_diff", "sigma_x",
          "sigma_y", "sigma_diff",
           "nu", "eff_size", "x_pred", "y_pred")
```

Compiling the JAGS Model

```
model <- jags.model("two_indep_means.jags.txt",
            data = data_list,
            inits = inits_list,
            n.chains = 3,
            n.adapt=1000)
```

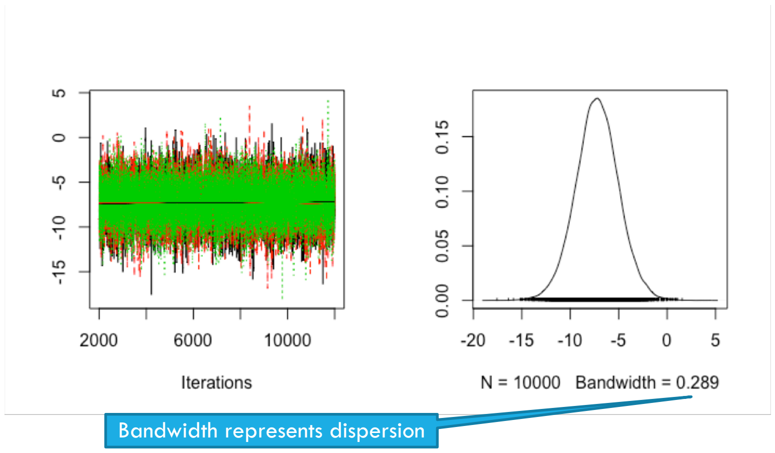
School of Information Studies
Syracuse University

Running and Inspecting the JAGS Model

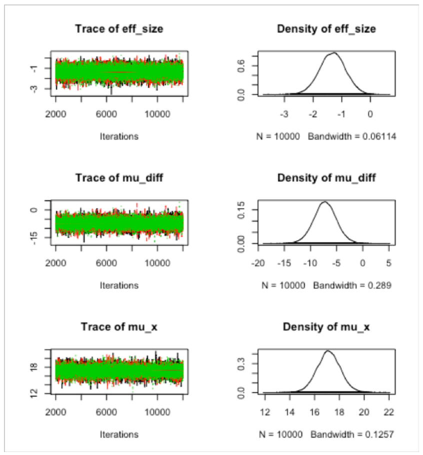
```
update(model, 1000) # Length of the burn in samples
samples <- coda.samples(model, params, n.iter=10000)
```

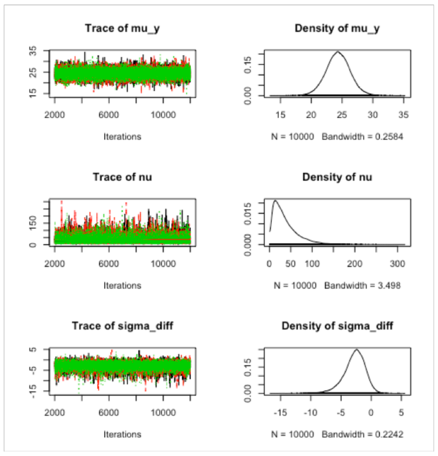
School of Information Studies
Syracuse University

Mean diff by am, mtcars$mpg: plot(samples[,"mu_diff"])

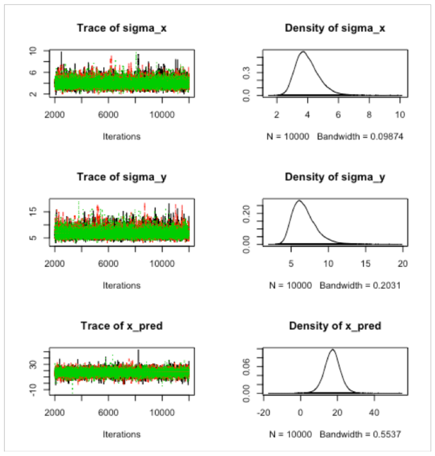


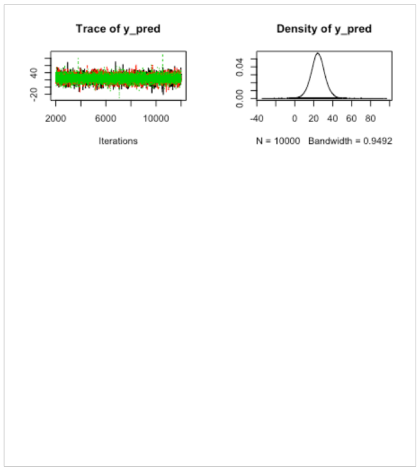
mtcars$mpg: Posterior Plots





mtcars$mpg: Posterior Plots





Conclusion

JAGS Code Expresses the set of relations that connect the quantities whose posterior distributions we want to model

After compilation, the JAGS model returns to R where the real work of estimation will occur

Burn-in samples are necessary at the beginning of the run while the Markov chain gets initialized to a productive starting point for the posterior distribution

Trace plots show the progress of the estimation process over time

HDI plots show the posterior distribution of the quantities we are tracking