Introduction to JAGS and rJAGS

Athena Chen

Tuesday, April 28, 2020

JHU Biostatistics Student Computing Club

Overview

Just Another Gibbs Sampler (JAGS)

- Introduction, installation, and useful resources
- Components of a JAGS program
- The JAGS language
- rJAGS

Example: Normal-Normal Model

Example: Mixture of Normal Distributions

- The label switching problem

Additional Tricks and Tips

- Custom likelihoods: the zero's and one's trick

Just Another Gibbs Sampler (JAGS)

Introduction, installation, and useful resources

Just Another Gibbs Sampler (JAGS) is a program for conducting Bayesian analyses and modeling using MCMC.

- Writen in **C++** and available on all major OS.
- Slice sampler (Neal 2003, *The Annals of Statistics*)

Installation

- **Sourceforge**: https://sourceforge.net/projects/mcmc-jags
- **Homebrew** (MacOS): `brew install jags`

Useful resources

- User manual

The JAGS language

The JAGS language is very friendly to R users.

Relations

deterministic assignment	`mu[i] <- alpha + beta*x[i]`
stochastic assignment	`Y[i]` $\sim$ `dnorm(mu[i], tau)`

Arrays and subsetting[1]

vectors	`mu[1]`, `mu[1:10]`
arrays	`B[r,c]`, `B[1:M,c]`, `B[1:M,1:N]`

Vector-construction and for-loops

Vector construction	`y <- c(x1, x2)`
for-loops	`for(i in 1:N)` $\{\cdots\}$

[1] Indices cannot be stochastic or repeated.

Components of a JAGS Program

1. Model definition
2. Compilation of model into computer memory
3. Initialization of model
4. Adaptation and burn-in
5. Monitoring or saving MCMC-generated values

Example: Normal-Normal Model

Normal-Normal model

Let $N(\mu, \tau^2)$ denote a normal distribution with mean μ and precision τ^2. Suppose

$$
\begin{aligned}
Y_i|\mu, \tau^2 &\overset{iid}{\sim} N(\mu, \tau^2) && \text{likelihood} \\
\mu &\sim N(\mu_0, \tau_0^2) && \text{prior} \\
\tau^2 &\sim \text{Gamma}(a, b)
\end{aligned}
$$

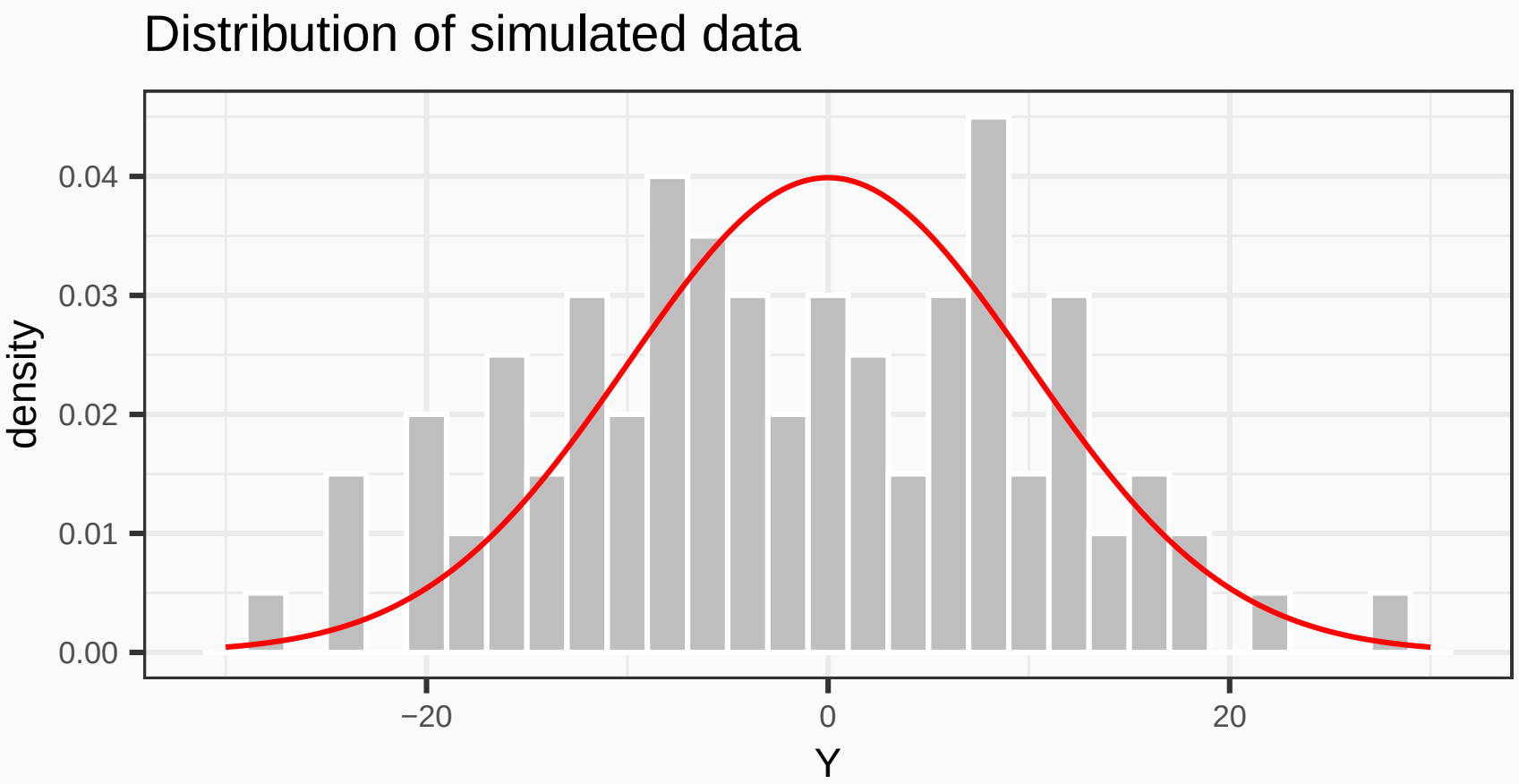
Then we know that,

$$
\mu|\tau^2, \mathbf{Y} \sim N\left(\frac{\tau_0^2\mu_0 + n\tau^2\bar{Y}}{\tau_0^2 + n\tau^2}, \tau_0^2 + n\tau^2\right)
$$

$$
\tau^2|\mu, \mathbf{Y} \sim \text{Gamma}\left(a + \frac{n}{2}, b + \frac{1}{2}\sum_i (Y_i - \mu)^2\right)
$$

Simulate data

```r
mu <- 0; tau2 <- 0.01                # Define simulation parameters

set.seed(2020428)
n <- 100                             # Number of observations in our data
Y <- rnorm(n, mu, sqrt(1/tau2))      # Simulate data
```



Define model

We define the model in the file `nn_model.jags`.

```
model {
  mu ~ dnorm(mu_0, tau2_0)
  tau2 ~ dgamma(a, b)

  for(i in 1:n){
    Y[i] ~ dnorm(mu, tau2)
  }
}
```

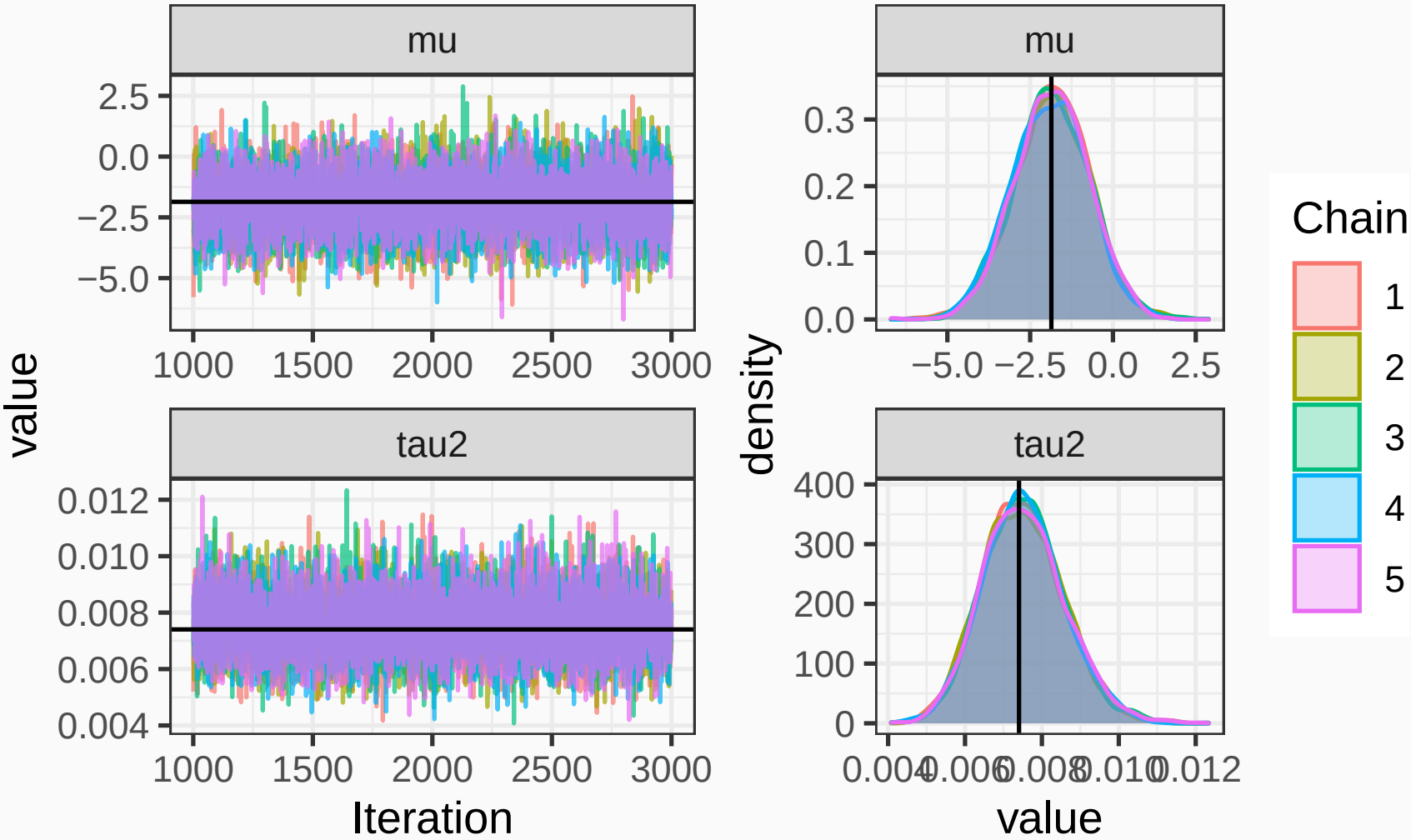
Define priors

```
library(rjags)                          # R interface to JAGS
library(ggmcmc)                         # tidy output objects

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
# Define prior parameters
mu_0 <- 0; tau2_0 <- 0.01               # Prior parameters for mu
a <- 1; b <- 0.1                        # Prior parameters for tau

# Define data
data_list <- list(mu_0 = mu_0,
                  tau2_0 = tau2_0,
                  a = a, b = b,
                  n = n, Y = Y)
```

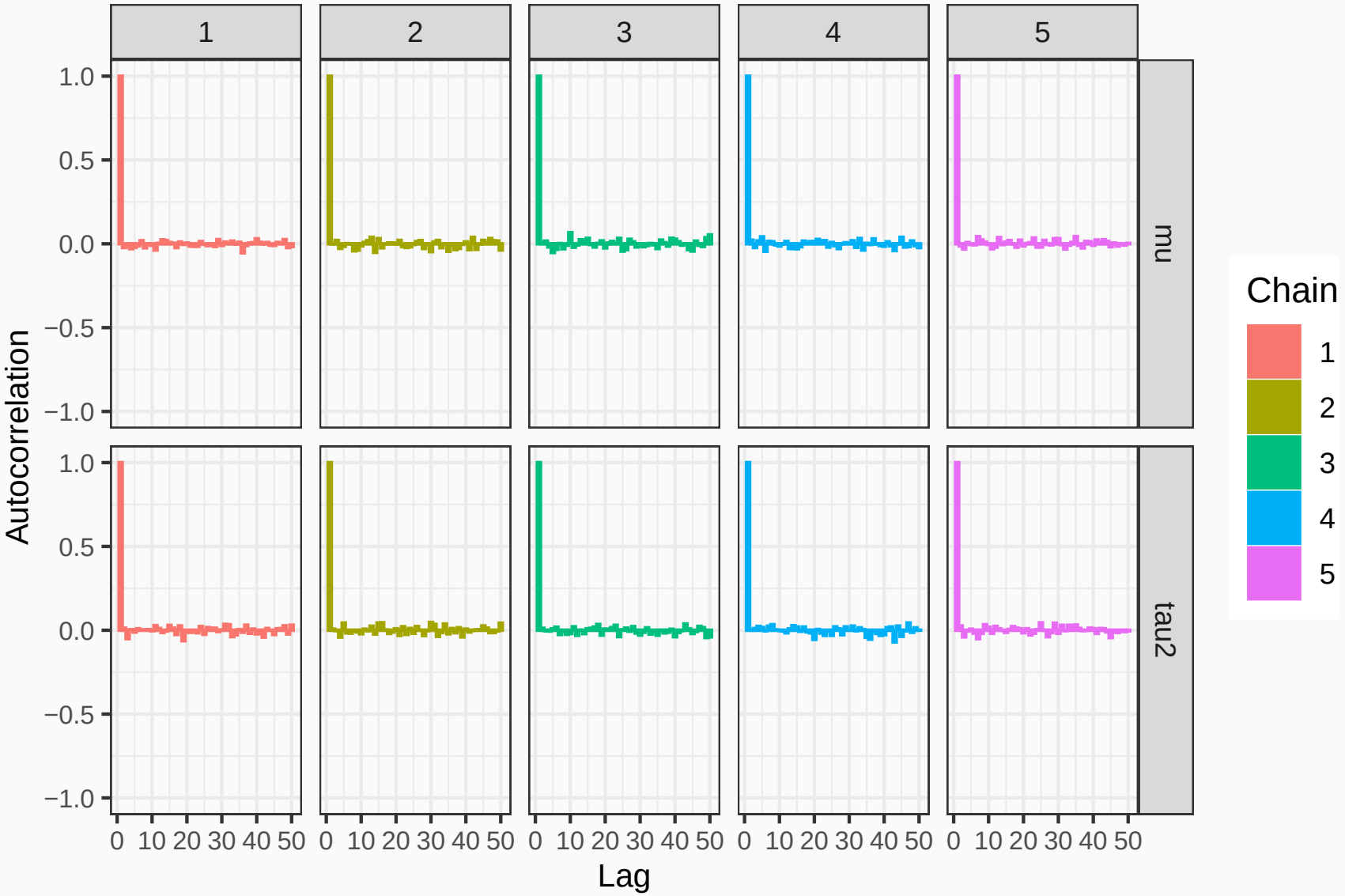
Compile model into memory and run model

```
# Compile model into memory
jags_model <- jags.model("./model/nn_model.bugs",
                         n.chains = 5,
                         data = data_list,
                         quiet = TRUE)

# burn-in of 1000 samples
update(jags_model, 1000)

# draw 2000 and convert samples to tidy format
samples <- coda.samples(jags_model,
                        variable.names = c("mu", "tau2"),
                        n.iter = 2000) %>%
  ggs()
```

Posterior trace and density plots



Chain autocorrelation



Example: Finite Mixture of Normal Distributions

Finite Mixture of Normals

Suppose we have data that comes from a mixture of normal distributions:

$$Z_i|\pi \sim Bernoulli(\pi)$$

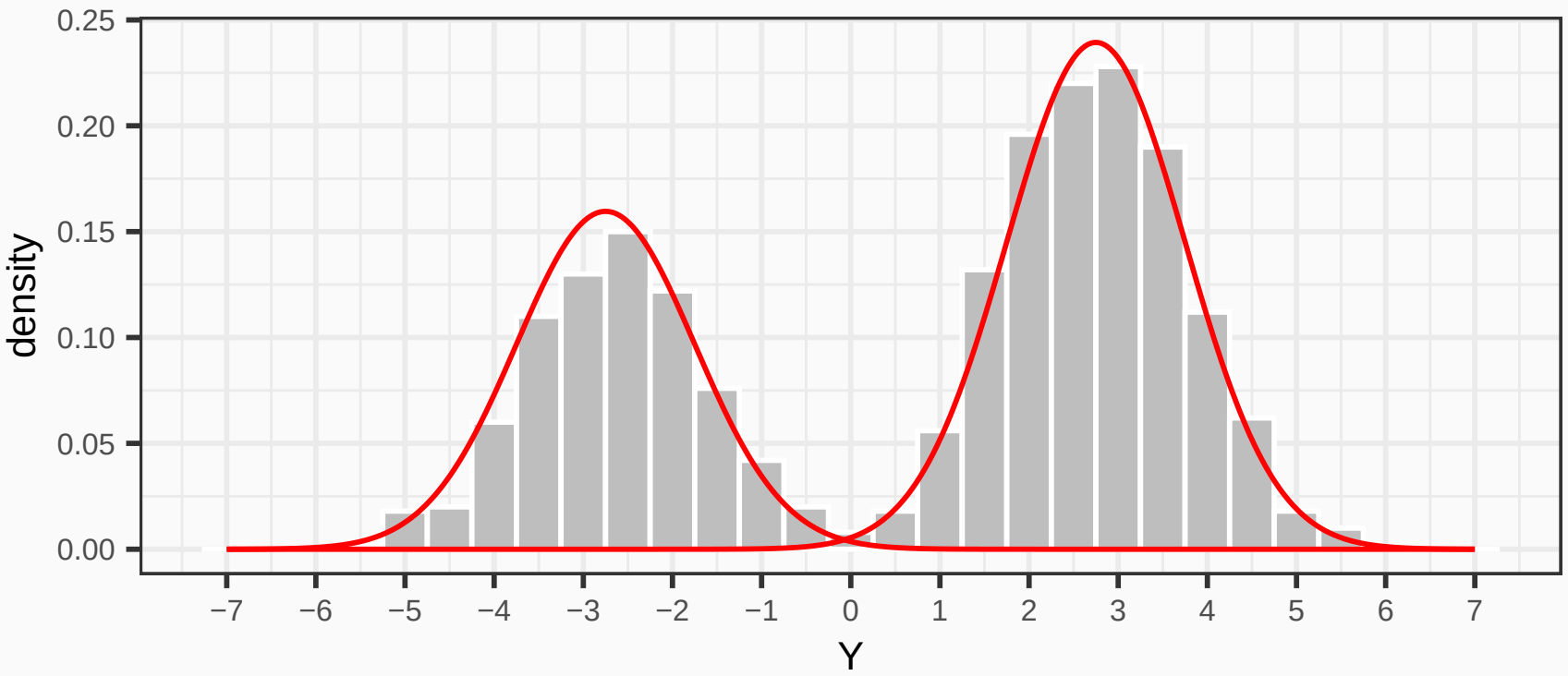
$$Y_i|Z_i, \mu_1, \mu_2, \tau_2^2 \overset{iid}{\sim} \begin{cases} N(\mu_1, \tau^2) & \text{if } Z_i = 1 \\ N(\mu_2, \tau^2) & \text{if } Z_i = 0 \end{cases}$$

Let,

$$\mu_i \sim N(\mu_{0i}, \tau_0^2)$$
$$\tau^2 \sim Gamma(a, b)$$
$$\pi \sim Beta(\alpha, \beta)$$

Simulate data

```r
mu_1 <- -2.75; mu_2 <- 2.75; tau2 <- 1; pi <- 0.4

set.seed(689934)
n <- 1000
Z <- rbinom(n, 1, pi)
Y <- Z*rnorm(1000, mu_1, sqrt(1/tau2)) +
  (1-Z)*rnorm(1000, mu_2, sqrt(1/tau2))
```



Define model

We define the model in the file `mixed_nn_model.jags`.

```
model {
  mu_1 ~ dnorm(mu_01, tau2_0)
  mu_2 ~ dnorm(mu_02, tau2_0)
  tau2 ~ dgamma(a, b)
  pi ~ dbeta(alpha, beta)

  for(i in 1:n){
    Z[i] ~ dbern(pi)
    mu[i] <- Z[i]*mu_1 + (1-Z[i])*mu_2
    Y[i] ~ dnorm(mu[i], tau2)
  }
}
```

Note that in JAGS, we can explicitly model the class label, Z_i.

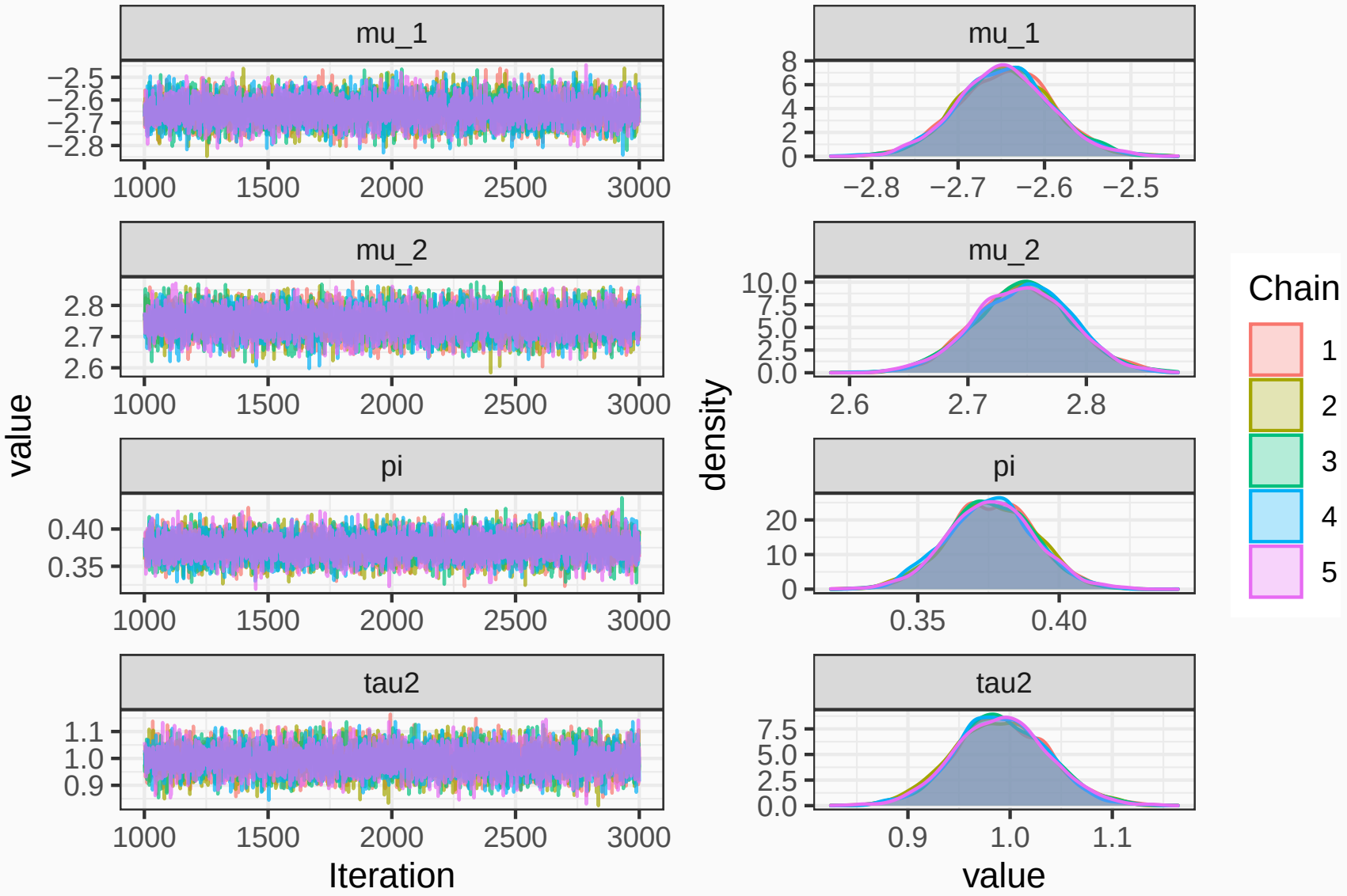
Define priors

```
# Define priors for mu
mu_01 <- -10; mu_02 <- 10;
tau2_0 <- 0.01

# Prior parameters for tau
a <- 1; b <- 0.01

# Prior parameters for pi
alpha <- 4; beta <- 6

# Define data
data_list <- list(mu_01 = mu_01, mu_02 = mu_02,
                  tau2_0 = tau2_0,
                  a = a, b = b,
                  alpha = alpha, beta = beta,
                  n = n, Y = Y)
```

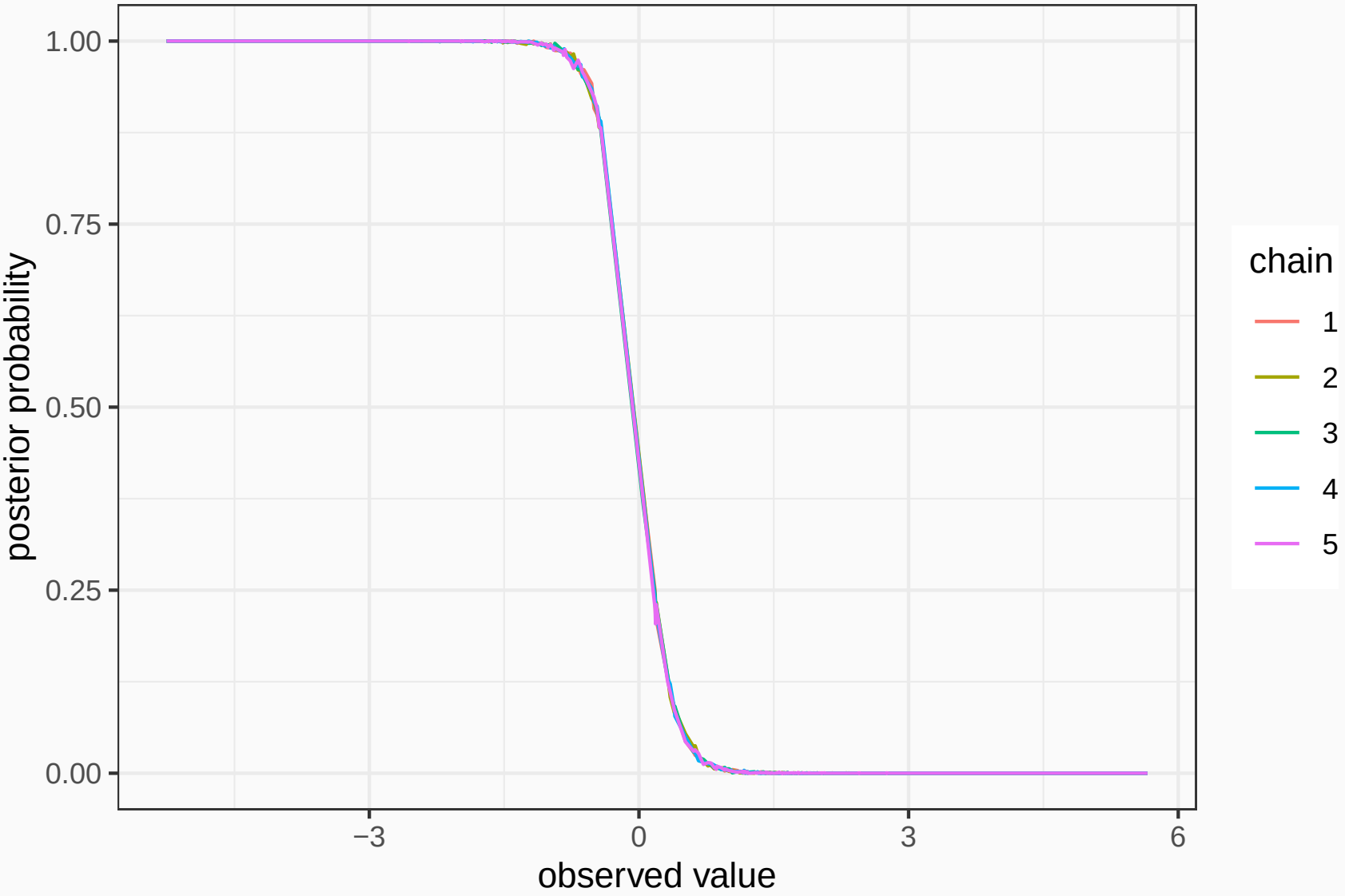
Compile model into memory

```
# Compile model into memory
jags_model <- jags.model("./model/mixed_nn_model.bugs",
                         n.chains = 5,
                         data = data_list,
                         quiet = TRUE)

# burn-in of 1000 samples
update(jags_model, 1000)

# draw 2000 and convert samples in tidy format
samples <- coda.samples(jags_model,
                        variable.names = c("mu_1", "mu_2", "pi",
                                           "tau2", "Z"),
                        n.iter = 2000) %>%
  ggs()
```

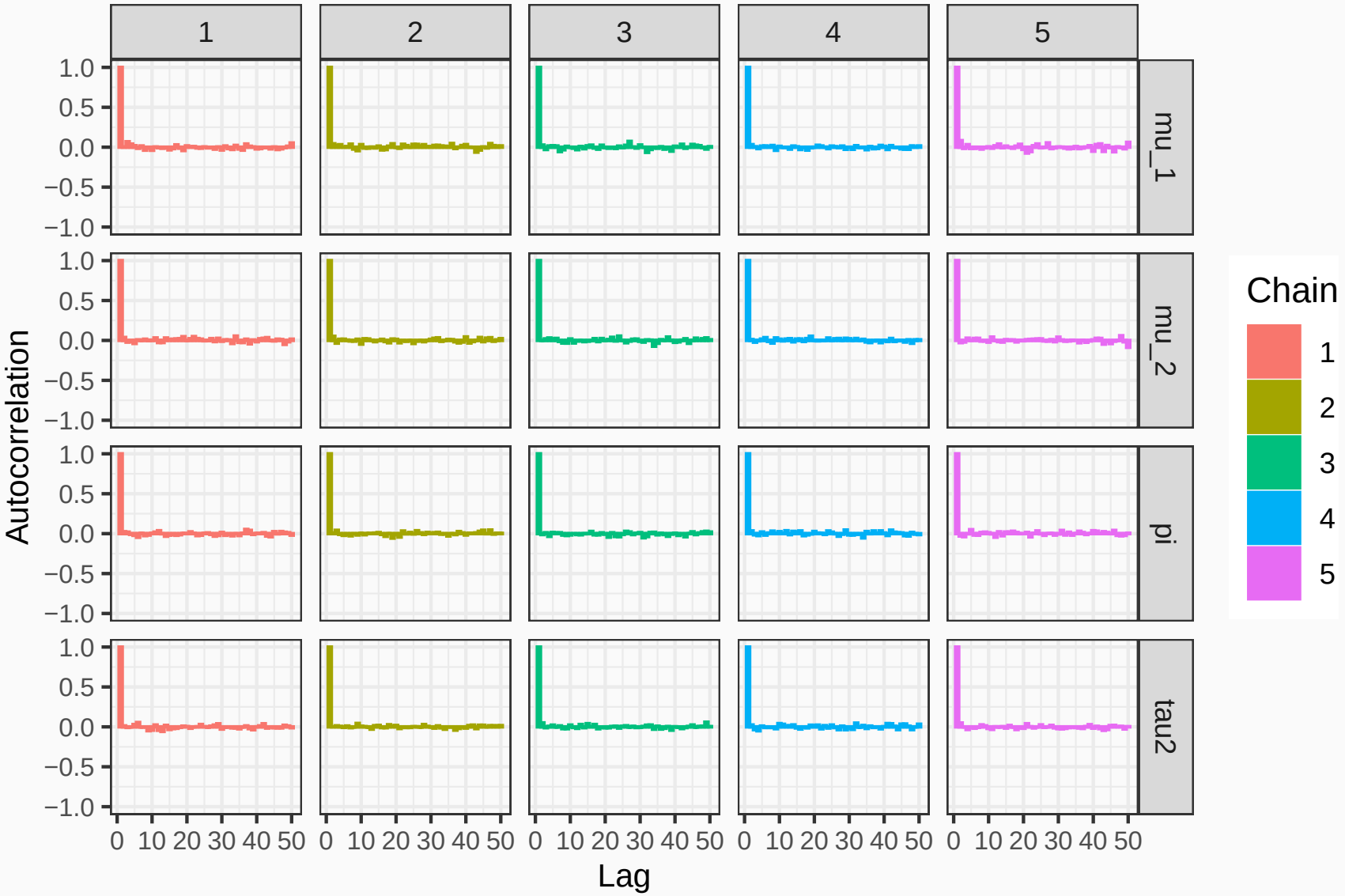
Posterior trace and density plots



Posterior probabilities of group 1 vs group 2



Chain autocorrelation



Non-identifiability and the label-switching problem

One of the main problems with mixture models is an identifiability problem often referred to as the label-switching problem.

Briefly, the label switching problem arises when the prior and likelihood are invariant under permutation of class labels.

Next week, Tuesday, May 5th, Jacob will implement this mixture model in `Stan` and discuss the label-switching problem and how to address it in more depth.

Additional Tips and Tricks

Custom likelihoods: the Zero's Trick

The zero's trick is an approach to specifying custom likelihoods in JAGS.

Suppose L is the custom likelihood we would like to use. The probability of only observing zeros for a Poisson(λ) distribution is given by,

$$f(0|\lambda) = \frac{\lambda^0 e^{-\lambda}}{0!} = e^{-\lambda}$$

Thus, if we define $\lambda = -\log(L)$, $f(0|\lambda) = e^{-(-\log(L))} = L$, and we can obtain the correct likelihood contribution.

```
model{
  # rest of model block
  for(i in 1:N){
    zeros[i] <- 0                    # Observation of all zeros
    zeros[i] ~ dpois(lambda[i])      # Specify poisson dist
    lambda[i] <- -log(L)             # Define custom likelihood
  }
}
```

Custom likelihoods: the One's Trick

An alternative to the zero's trick is the one's trick. Again, if L is the custom likelihood we would like to use, the probability of only observing ones for a Bernoulli(π) distribution is given by,

$$f(1|\pi) = \pi^1(1-\pi)^0 = \pi$$

Thus, if we define $\pi = L$, $f(0|\pi) = L$, and we can obtain the correct likelihood contribution.

```
model{
  # rest of model block
  for(i in 1:N){
    ones[i] <- 1                # Observation of all ones
    ones[i] ~ dbern(ones[i])    # Specify bernoulli dist
    pi[i] <- L                  # Define custom likelihood
  }
}
```