Installing `rjags` in `R`

Caleb Ziolkowski

4/15/2020

Install `rjags`

If you haven't done so already.

```
# Install the program

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

```
## Loading required package: coda
```

```
## Linked to JAGS 4.2.0
```

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

(Install and) load other packages

A lot of these are quite useful (when dealing with JAGS).

```
library(tidyverse)
library(magrittr)
library(pscl)
library(coda)
library(bayesplot)
library(mcmcplots)
```

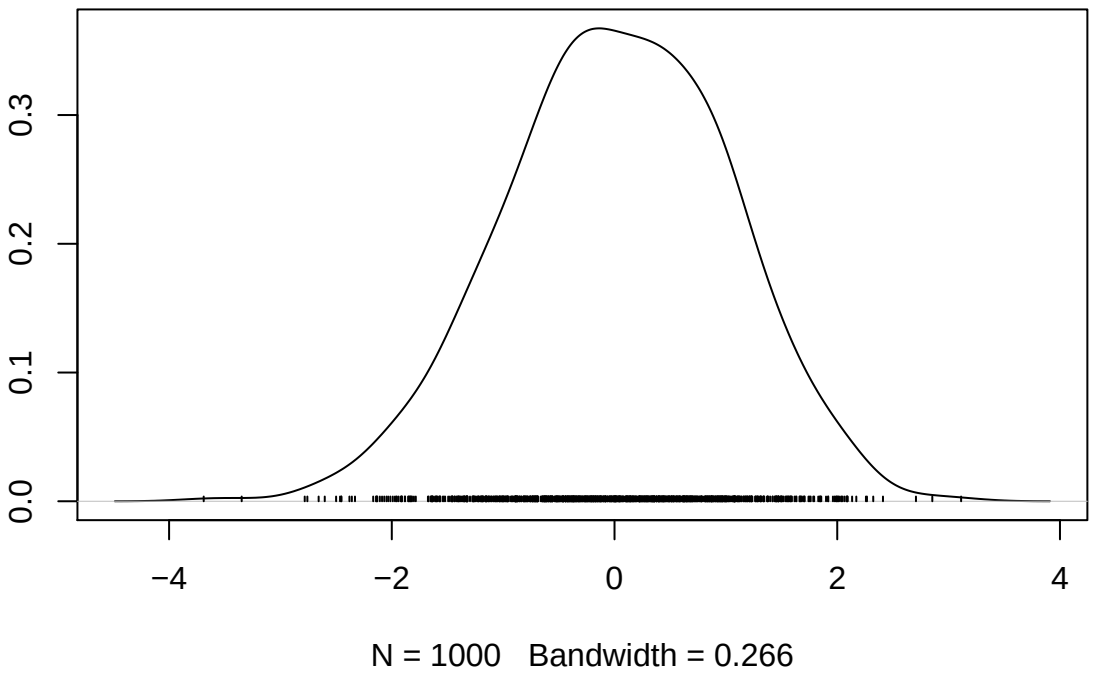
Coding up some simple models

Running through the code (and checking against my results in the `.html` or `.pdf`) will offer a good chance to confirm everythings working.

Normal distribution

Sampling from a normal distribution.

```
code <- '
  model {
    y ~ dnorm(0,1)
  }'

jags <- jags.model(textConnection(code), quiet = T)
samples <- coda.samples(jags, variable.names = c('y'), n.iter = 1000)
densplot(samples)
```



Linear regression

We can build a simple linear regression. First making some data.

```
N <- 100
x1 <- rnorm(N)
x2 <- rnorm(N)
beta <- c(.3, -.3)
y <- .1 + beta[1]*x1 + beta[2]*x2 + rnorm(N, 0)
```

Writing up a model to capture the data generating process.

```
code <- '
  model {

    # specify likelihood for each observation

    for (i in 1:N) {
      y[i] ~ dnorm(mu[i], sigma)
      mu[i] <- a + beta[1]*x1[i] + beta[2]*x2[i]
    }

    # specify priors for theta

    a ~ dnorm(0, .001)

    for (i in 1:2) {
      beta[i] ~ dnorm(0, .001)
    }
```

```
    tau <- 1/(sigma^2)
    sigma ~ dunif(0, 100)
  }
'
```

Conditioning on the data.

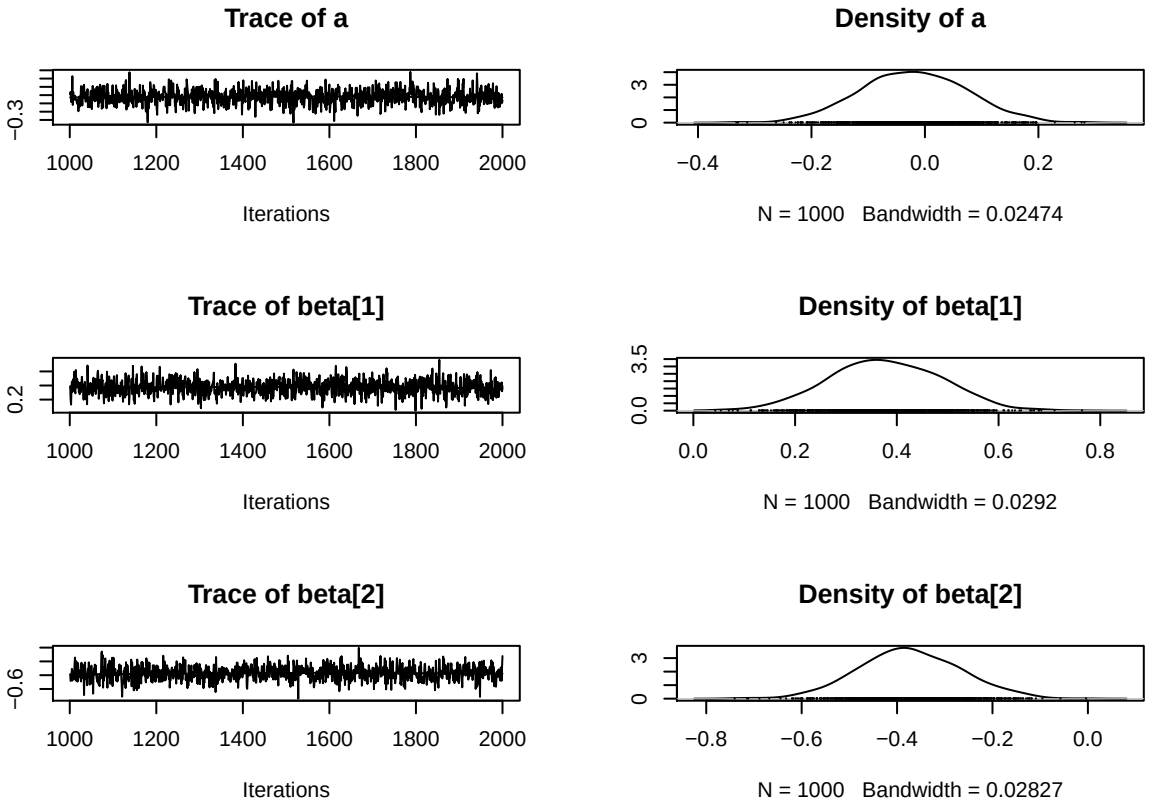
```
## Specification of the model is in the string 'code'
mod_spec <- textConnection(code)

## Complie the code of the model with the user-supplied data
jags <- jags.model(
  mod_spec,
  data = list('N' = N, 'y' = y, 'x1' = x1, 'x2' = x2),
  quiet = T
)
```

Getting samples from the posterior.

```
## Draw samples
samples_coef <- coda.samples(
  jags,
  variable.names = c('a', 'beta'),
  n.iter = 1000, nchain = 4
)
```

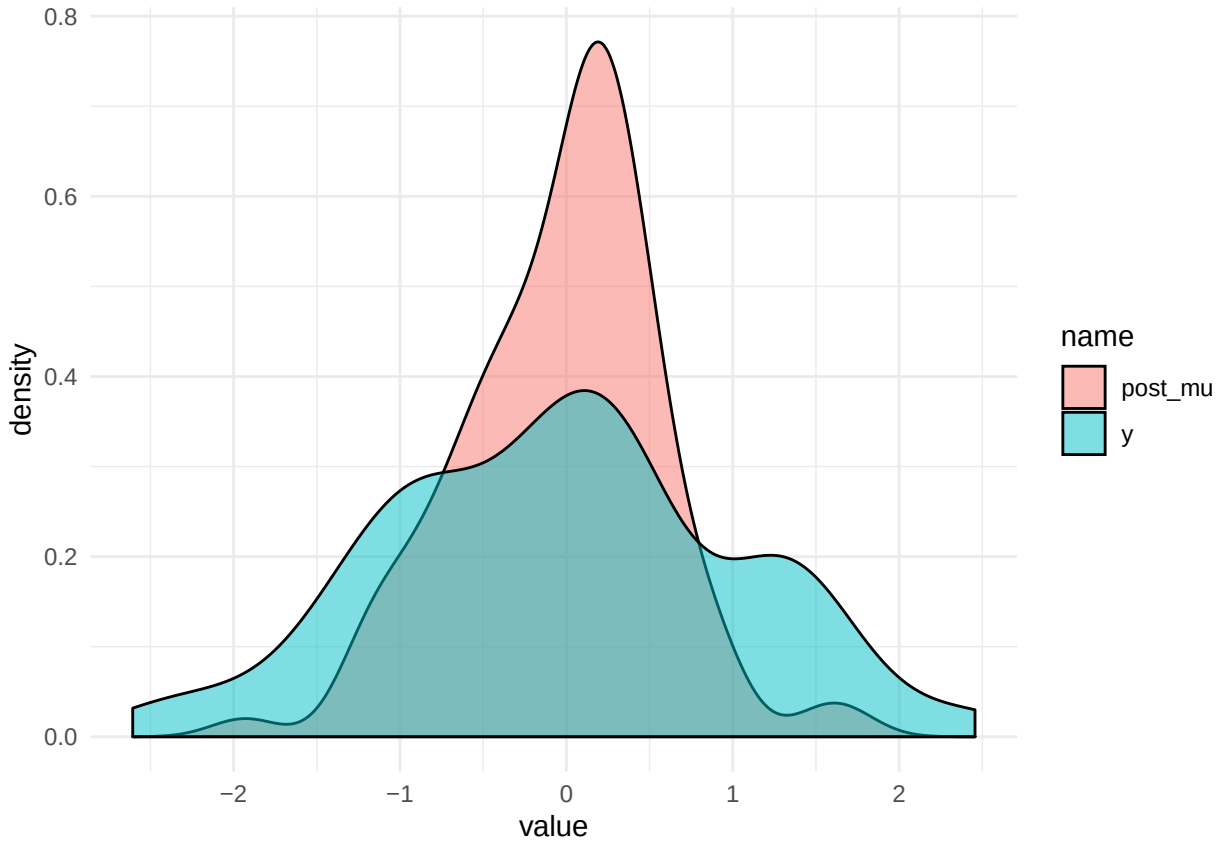
Plotting the posterior distributions of the parameters.

```
plot(samples_coef)
```



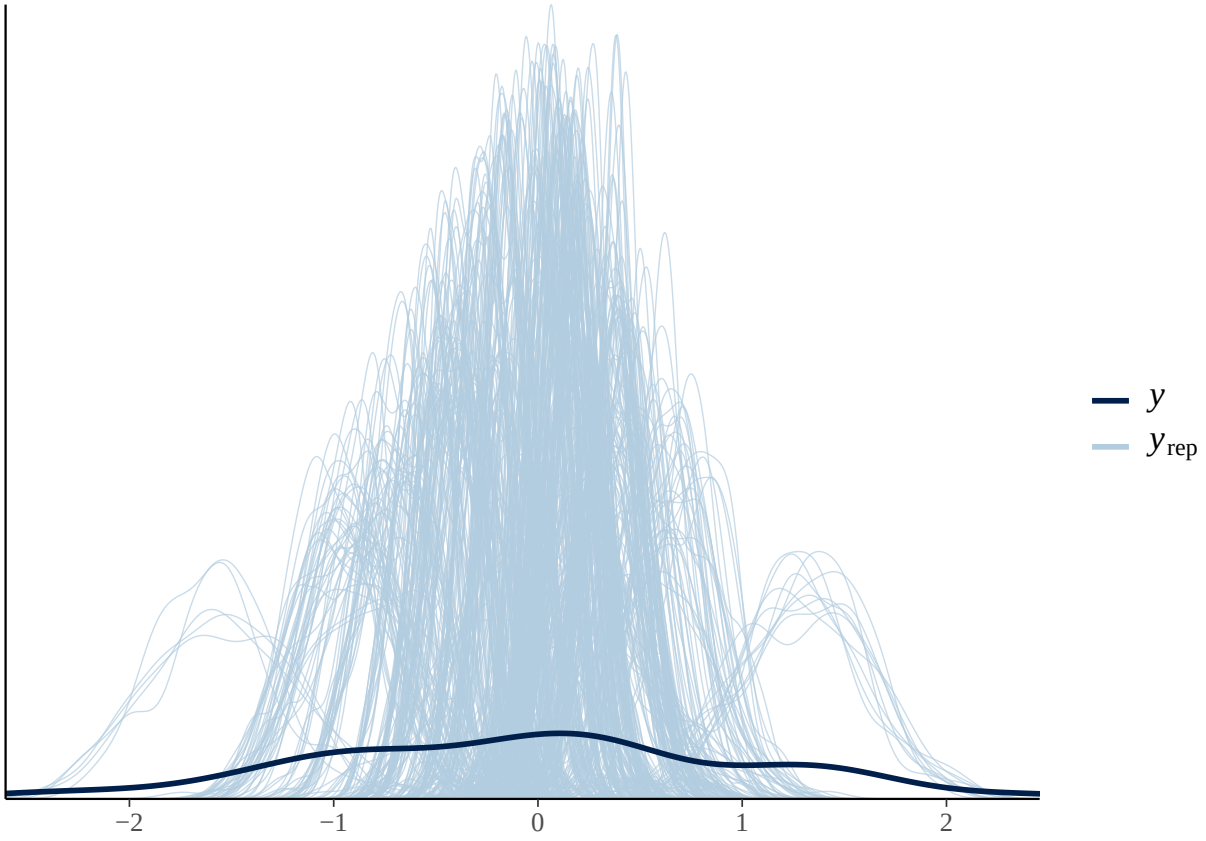
Comparing estimates of μ to the actual ys.

```
samples <- jags.samples(jags, variable.names = c('mu'), n.iter = 1000)

## posterior predictions for mu
dat <- tibble(
  y = y,
  x1 = x1,
  x2 = x2,
  post_mu = samples$mu[1:N]
)

## plot densities of y and mu
dat %>%
  select(y, post_mu) %>%
  pivot_longer(y: post_mu) %>%
  ggplot(aes(value, fill = name)) +
  geom_density(alpha = .5) +
  theme_minimal()
```



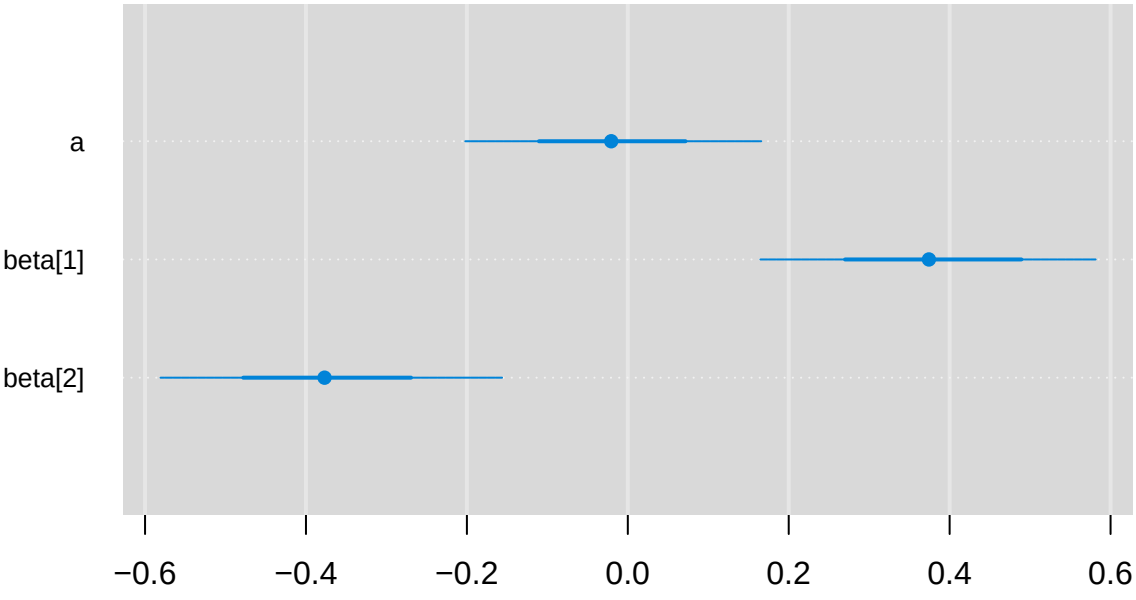
We can also compare the ys to posterior draws of y.

```
## plot a set of posterior draws
yrep <- matrix(samples$mu, ncol = N)
ppc_dens_overlay(y, yrep[1:500,])
```



Here's a function to easily create a plot displaoying estimates of the coefficients.

```
caterplot(samples_coef, reorder = F)
```



An example of something going wrong

What if we write up a model that isn't identifiable? Like if it has two intercepts?

```
## non-identifiable model
code <- '
  model {
    # Likelihood with 2 intercepts

  for (i in 1:N) {
      y[i] ~ dnorm(mu[i], sigma)
      mu[i] <- a1 + a2 + beta[1]*x1[i] + beta[2]*x2[i]
    }

    # specify priors for theta

    a1 ~ dnorm(0, .001)
    a2 ~ dnorm(0, .001)

    for (i in 1:2) {
      beta[i] ~ dnorm(0, .001)
    }

    tau <- 1/(sigma^2)
    sigma ~ dunif(0, 100)
  }
'
```

And we condition on the data.

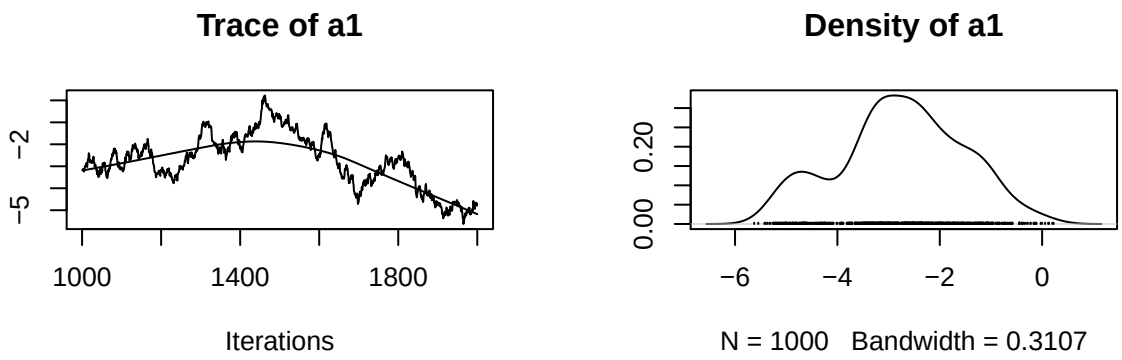
```
mod_spec <- textConnection(code)

jags <- jags.model(
  mod_spec,
  data = list('N' = N, 'y' = y, 'x1' = x1, 'x2' = x2),
  quiet = T
)
```

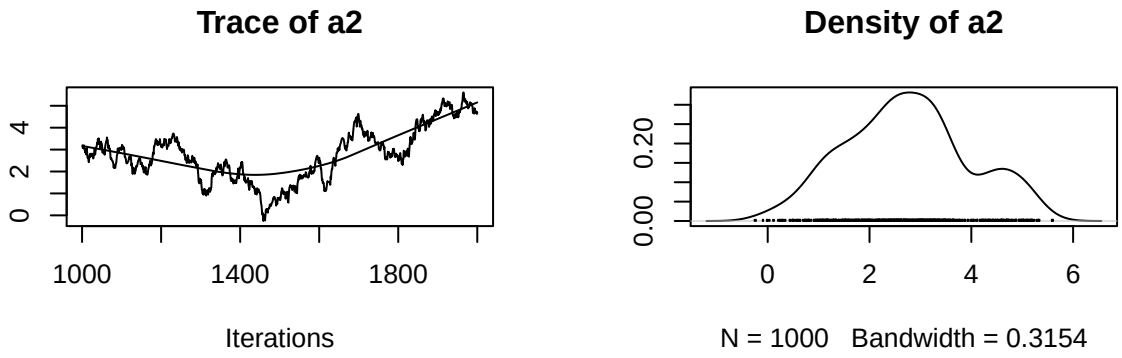
Make draws from the posterior distributions of the intercepts.

```
## Draw samples
samples_ints <- coda.samples(
  jags,
  variable.names = c('a1', 'a2'),
  n.iter = 1000, nchain = 4
)
```

Examining trace plots shows that we've got problems (we want hairy caterpillars, but that is not what we got). You can actually see how the draws of `a1` and `a2` "mirror" one another, a result of the non-identifiability we wrote into the model.

```
plot(samples_ints)
```





Binary response

Last example, we'll build a probit model. First, the data.

```
beta <- c(.4, -.2)
y_star <- -.2 + .4*x1 - .2*x2 + rnorm(N)
y <- rbinom(N, 1, pnorm(y_star))
```

Writing the model.

```
code <- '
  model {
    for (i in 1:N) {
      y[i] ~ dbern(p[i])
      probit(p[i]) <- a + beta[1]*x1[i] + beta[2]*x2[i]
    }

    a ~ dnorm(0, .001)

    for (i in 1:2) {
      beta[i] ~ dnorm(0, .001)
    }
  }
'
```

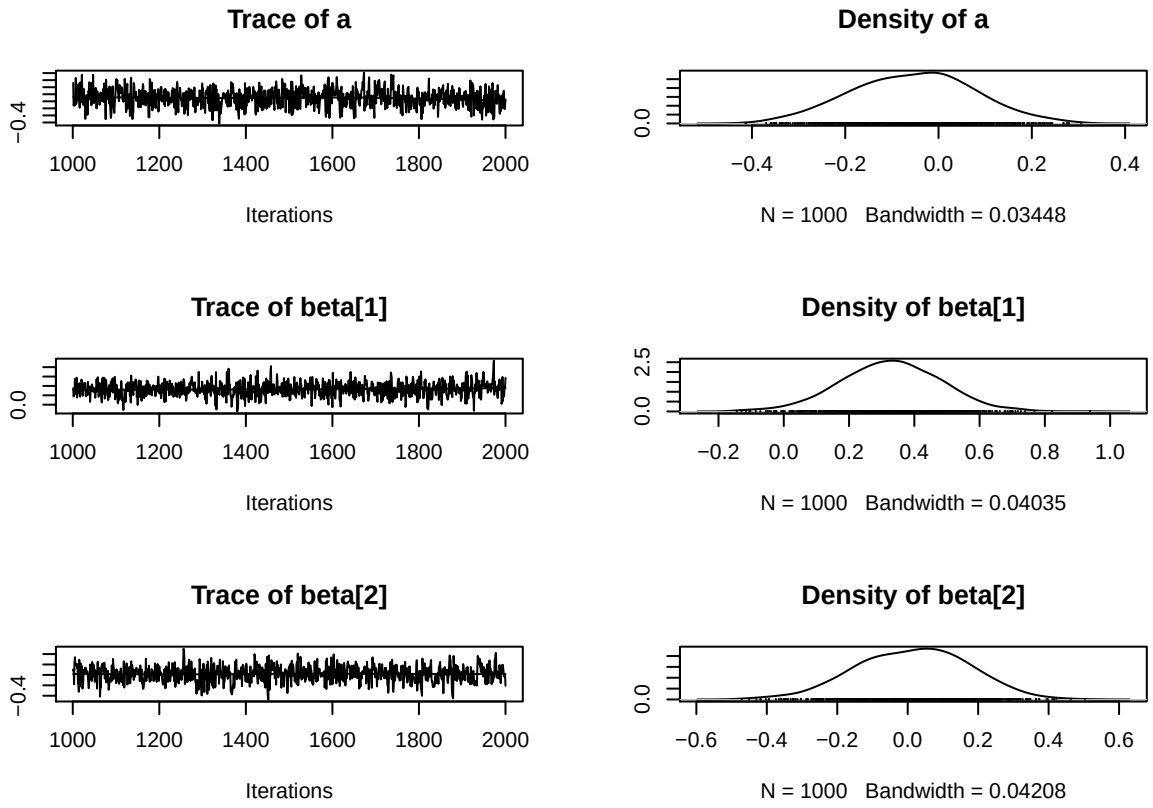
Conditioning the model on the data.

```
mod_spec <- textConnection(code)

jags <- jags.model(
```

```
  mod_spec,
  data = list('N' = N, 'y' = y, 'x1' = x1, 'x2' = x2),
  quiet = T
)
```

Drawing samples.

```
## Draw samples
samples <- coda.samples(
  jags,
  variable.names = c('a', 'beta'),
  n.iter = 1000, nchain = 4
)
plot(samples)
```



We can build quantities of interest (QOIs) into our code for the model. Here we'll add some predicted probabilities.

```
code <- '
  model {
    for (i in 1:N) {
      y[i] ~ dbern(p[i])
      probit(p[i]) <- a + beta[1]*x1[i] + beta[2]*x2[i]
    }

    a ~ dnorm(0, .001)

    for (i in 1:2) {
      beta[i] ~ dnorm(0, .001)
```

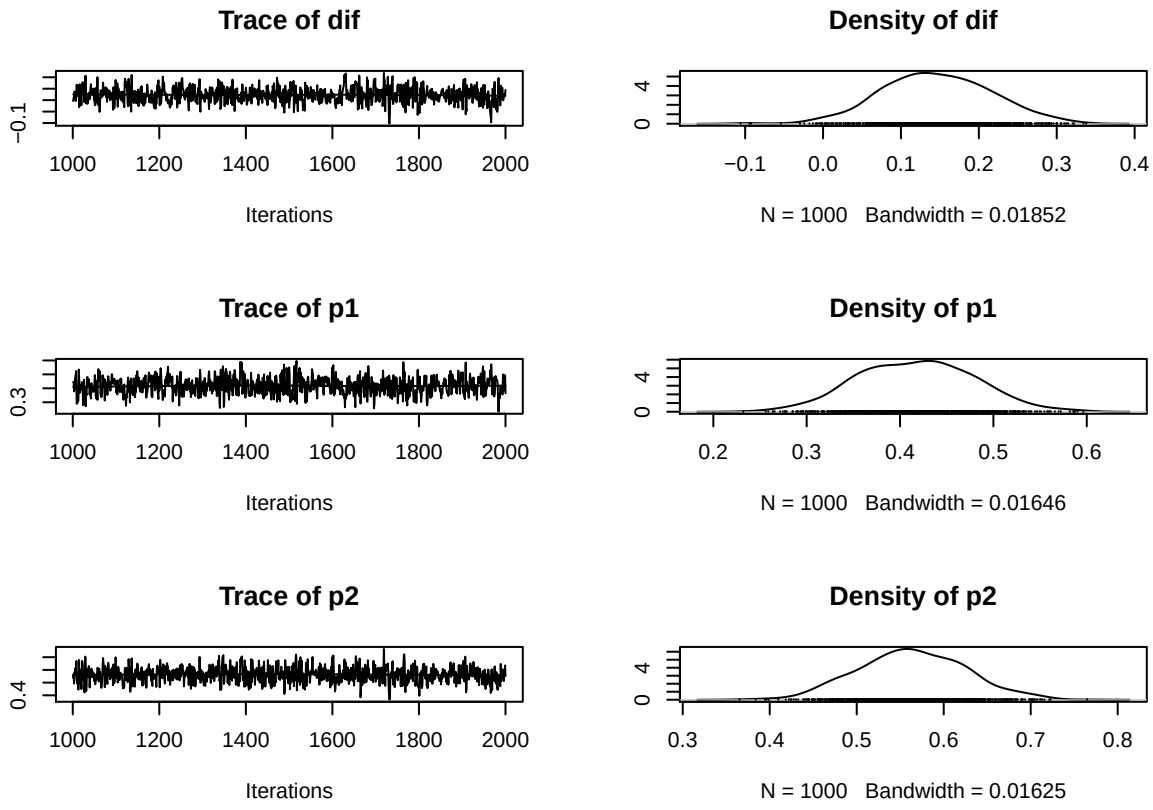
```
    }

    p1 <- phi(a + beta[1]*lox1 + beta[2]*medx2)
    p2 <- phi(a + beta[1]*hix1 + beta[2]*medx2)
    dif <- p2 - p1
  }
'
```

We need to provide some data to `JAGS` on the counterfactuals of interest.

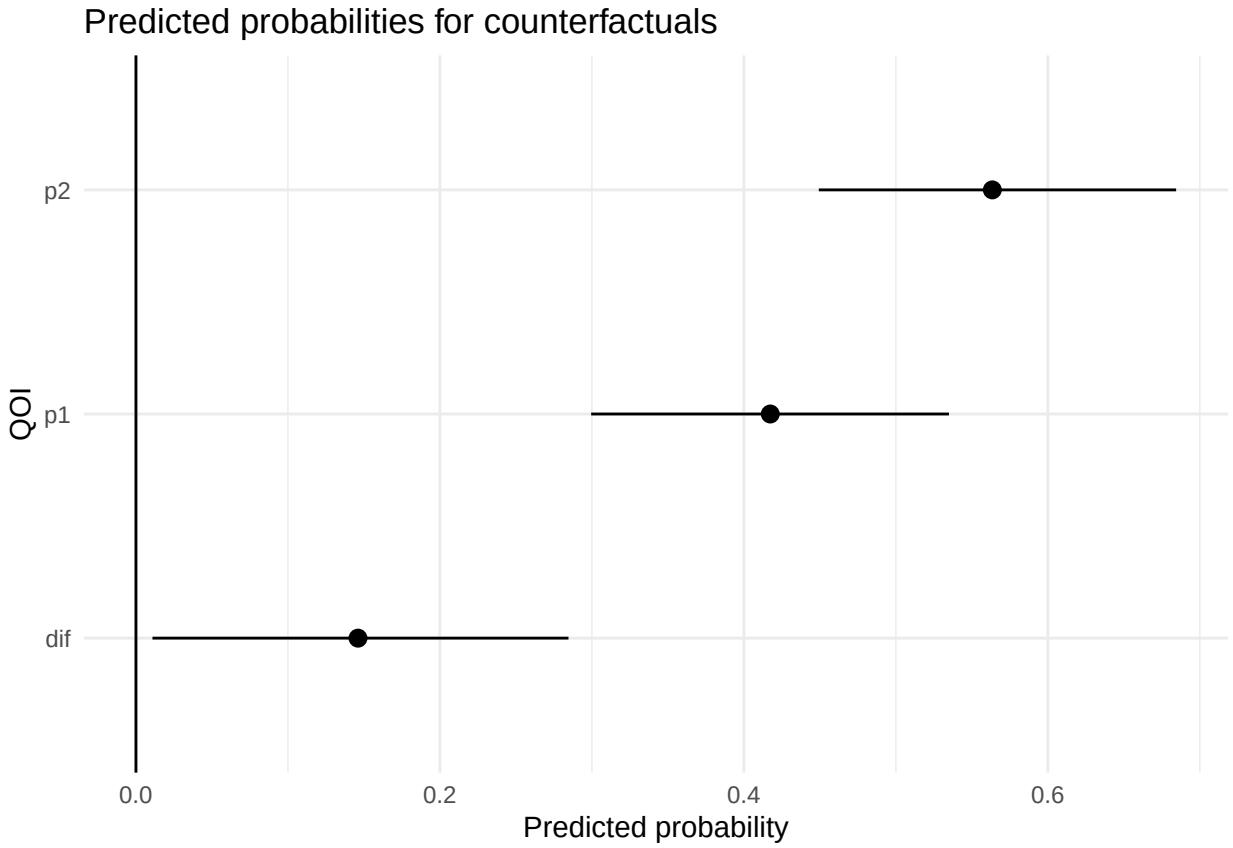
```
lox1 <- quantile(x1, .25)
hix1 <- quantile(x1, .75)
medx2 <- median(x2)

mod_spec <- textConnection(code)

jags <- jags.model(
  mod_spec,
  data = list('N' = N, 'y' = y, 'x1' = x1,
              'x2' = x2, 'lox1' = lox1,
              'hix1' = hix1, 'medx2' = medx2),
  quiet = T
)
```

We can then analyze those QOIs.

```
samples <- coda.samples(
  jags,
  variable.names = c('p1', 'p2', 'dif'),
  n.iter = 1000, nchain = 4
)

plot(samples)
```




```
samples[[1]][1:1000,] %>%
  as_tibble() %>%
  summarize_all(list(~mean(.), lo = ~quantile(., .025), hi = ~quantile(., .975))) %>%
  pivot_longer(dif_mean:p2_hi) %>%
  separate(name, c('pred', 'quantity')) %>%
  pivot_wider(names_from = quantity, values_from = value) %>%
  mutate(pred = factor(pred, levels = c('dif', 'p1', 'p2'))) %>%
  ggplot(aes(pred, mean, ymin = lo, ymax = hi)) +
  geom_pointrange() +
  geom_hline(yintercept = 0) +
  coord_flip() +
  theme_minimal() +
  xlab('QOI') +
  ylab('Predicted probability') +
  ggtitle('Predicted probabilities for counterfactuals')
```


Predicted probabilities for counterfactuals
QOI
p2
p1
dif
0.0
0.2
0.4
0.6
Predicted probability