Bayesian Modeling with RJAGS

DataCamp

13/09/2021

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(rjags)
```

```
## Warning: package 'rjags' was built under R version 4.0.5
```

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

```
## Warning: package 'coda' was built under R version 4.0.5
```

```
## Linked to JAGS 4.3.0
```

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

```
library(openintro)
```

```
## Warning: package 'openintro' was built under R version 4.0.5
```

```
## Loading required package: airports
```

```
## Warning: package 'airports' was built under R version 4.0.5
```

```
## Loading required package: cherryblossom

## Warning: package 'cherryblossom' was built under R version 4.0.5

## Loading required package: usdata

## Warning: package 'usdata' was built under R version 4.0.5
```

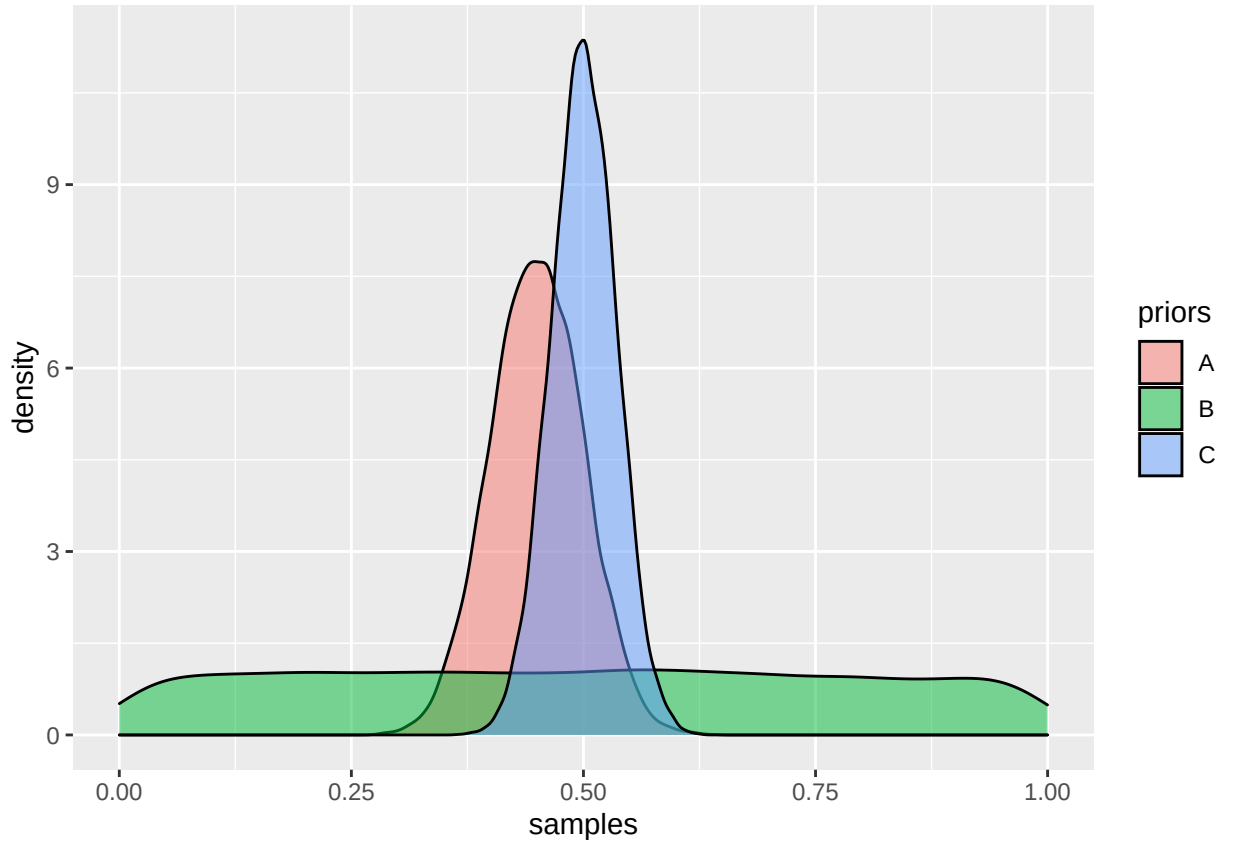
```
library(mosaic)
```

```
## Warning: package 'mosaic' was built under R version 4.0.4

## Registered S3 method overwritten by 'mosaic':
##   method                           from
##   fortify.SpatialPolygonsDataFrame ggplot2

##
## The 'mosaic' package masks several functions from core packages in order to add
## additional features.  The original behavior of these functions should not be affected by this.

##
## Attaching package: 'mosaic'

## The following object is masked from 'package:Matrix':
##
##     mean

## The following object is masked from 'package:openintro':
##
##     dotPlot

## The following objects are masked from 'package:dplyr':
##
##     count, do, tally

## The following object is masked from 'package:ggplot2':
##
##     stat

## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cor.test, cov, fivenum, IQR, median, prop.test,
##     quantile, sd, t.test, var

## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum
```

Introduction to Bayesian Modelling

Beta Priors

```r
# Sample 10000 draws from the Beta(45,55) prior
prior_A <- rbeta(n=10000, shape1 = 45,shape2 = 55)

# Sample 10000 draws from the Beta(1,1) prior
prior_B <- rbeta(n = 10000, shape1 = 1, shape2 = 1)

# Sample 10000 draws from the Beta(100,100) prior
prior_C <- rbeta(n = 10000, shape1 = 100, shape2 = 100)

# Combine the results in a single data frame
prior_sim <- data.frame(samples = c(prior_A, prior_B, prior_C),
        priors = rep(c("A","B","C"), each = 10000))

# Plot the 3 priors
ggplot(prior_sim, aes(x = samples, fill = priors)) +
    geom_density(alpha = 0.5)
```



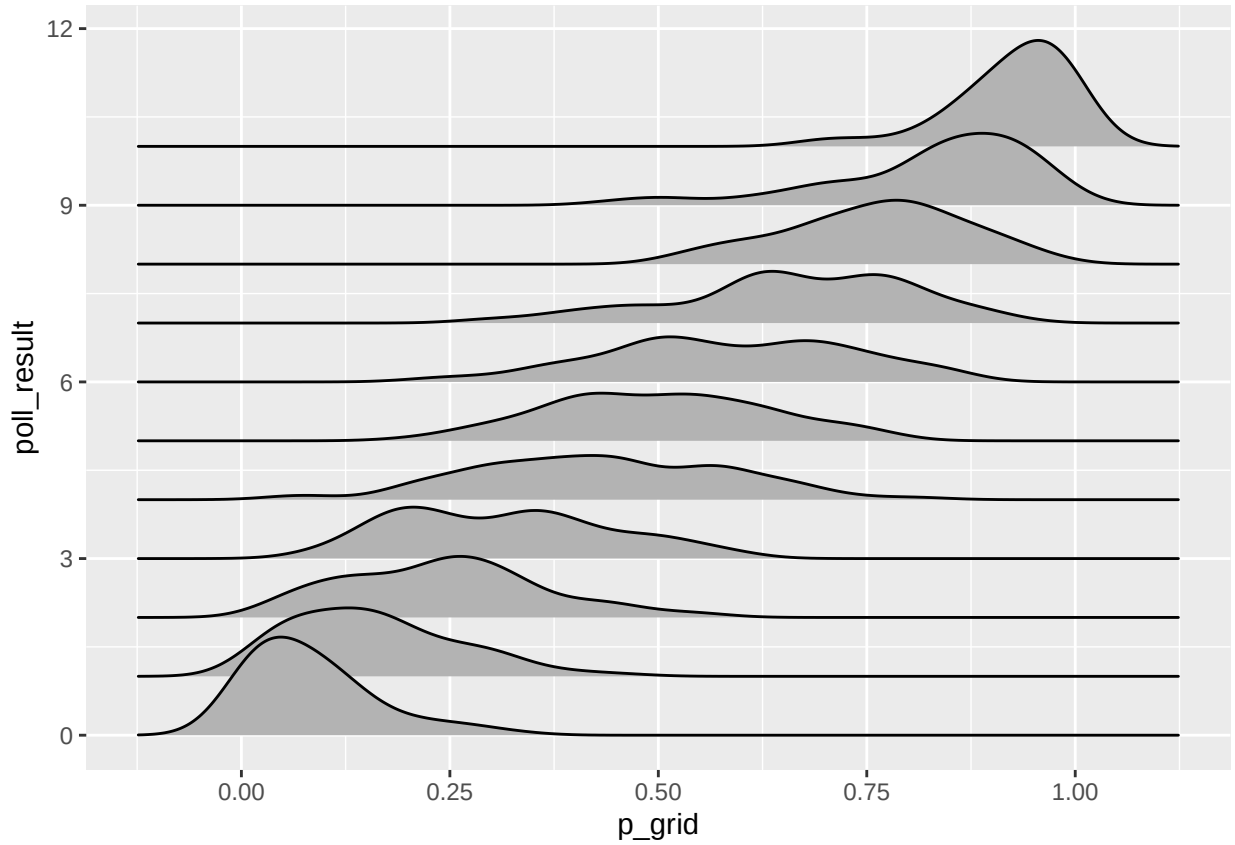
Prior B reflects 'vague' prior information about p - it gives equal prior weight to all values of p between 0 and 1. Prior C reflects more prior certainty about p - it has less spread and is centered around a mean that's greater than that for Prior A.

The data and the likelihood

Likelihood is a function of p that depends on the observed data X. How likily the unknown parameter p is given by the data X we observed. $L(p) = Prob(X|p)$.

```r
library(ggridges)
# simulate a bonomial model

# Define a vector of 1000 p values
p_grid <- seq(from = 0, to = 1, length.out = 1000)

# Simulate 1 poll result for each p in p_grid
poll_result <- rbinom(1000,10,p_grid) # 1 poll is 10 votes, so size=10

# Create likelihood_sim data frame
likelihood_sim <- data.frame(p_grid, poll_result)

# Density plots of p_grid grouped by poll_result
ggplot(likelihood_sim, aes(x = p_grid, y = poll_result, group = poll_result)) +
    geom_density_ridges()
```

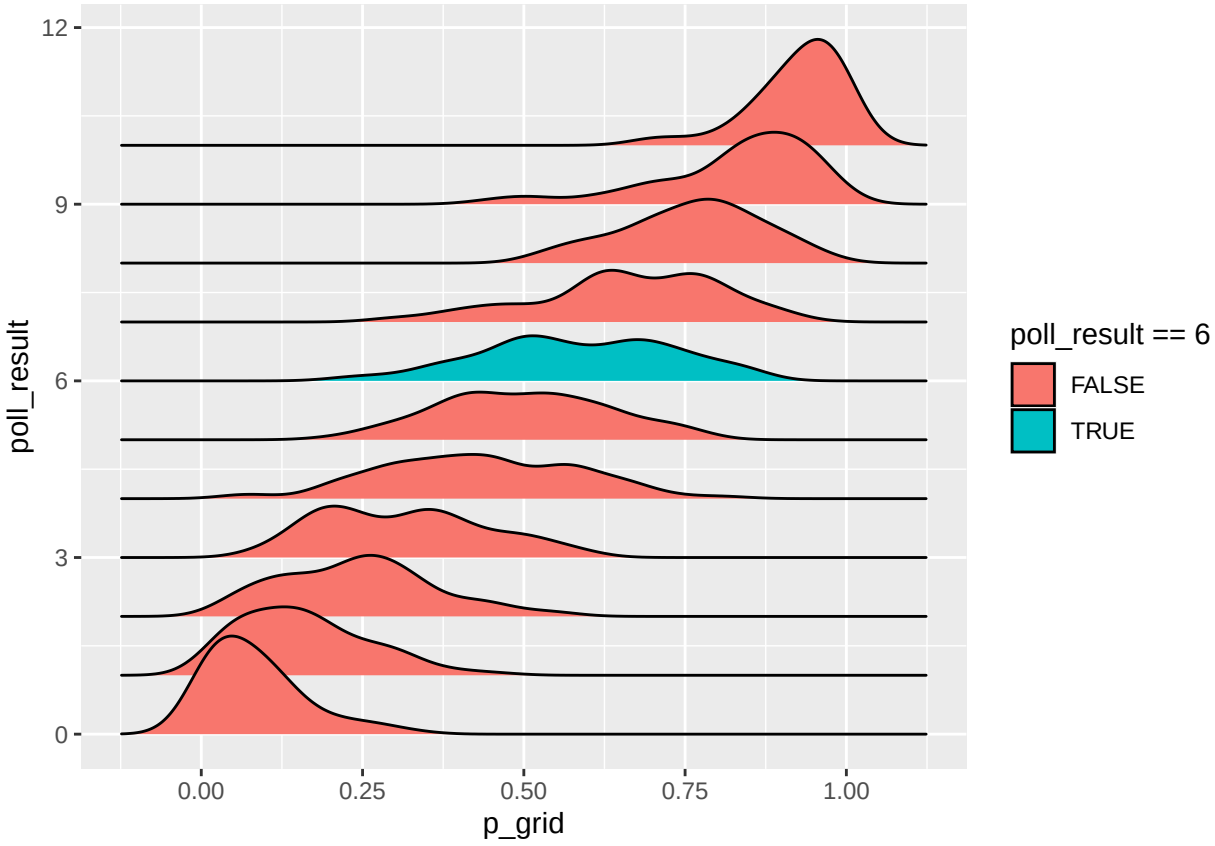
```
## Picking joint bandwidth of 0.0412
```



Polls in which 0 people supported you (`poll_result = 0`) correspond to smaller values of underlying support p (`p_grid`). The opposite is true for polls in which all 10 people supported you.

```r
# highlight the most likely probability p given by the X=6 we observed
ggplot(likelihood_sim, aes(x = p_grid, y = poll_result, group = poll_result, fill = poll_result==6)) +
    geom_density_ridges()
```

```
## Picking joint bandwidth of 0.0412
```



It indicates that the simulated surveys in which 6 of 10 voters supported you corresponded to underlying support p that ranged from approximately 0.25 to 1, with p around 0.6 being the most common.

The probability of having 6 people voted you is 0.6.

The posterior model

Prior: $p \sim Beta(45, 55)$

Likelihood: $X \sim Bin(10, p)$

Bayes's Rule: $Posterior \propto prior * likelihood$

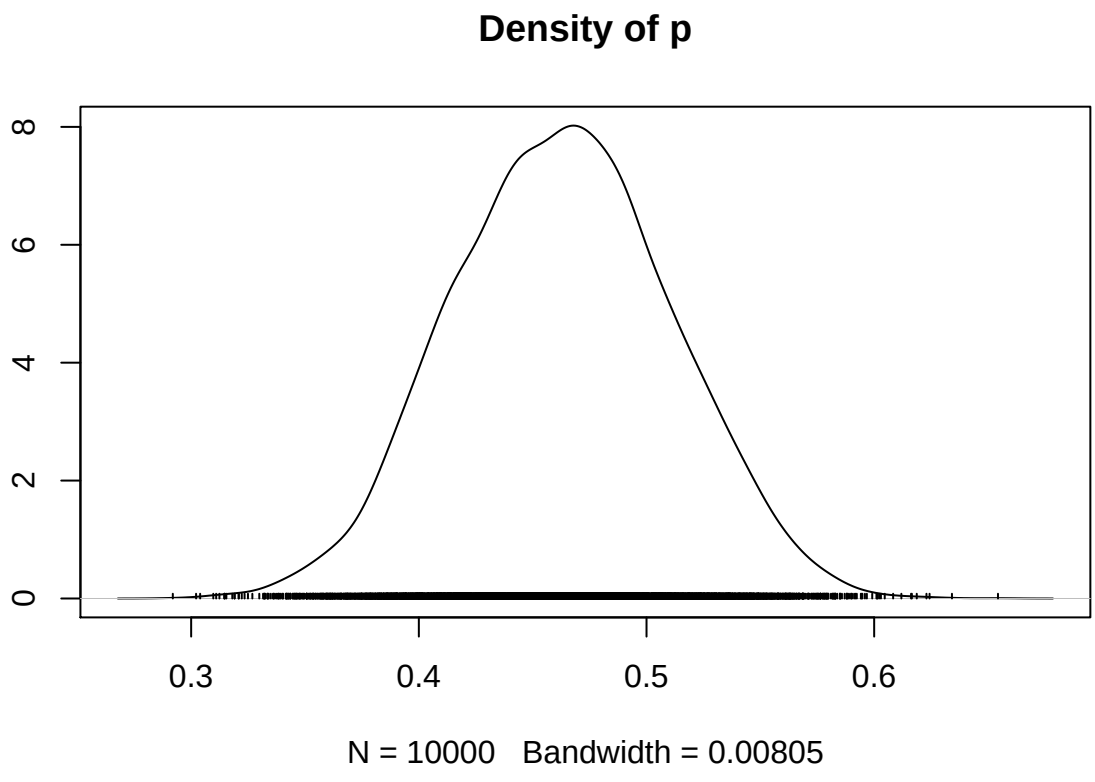
The `rjags` function `dbin()` is different than `dbinom()` in the base r. `rjags` works different than the base r.

```
library(rjags)

# define the model
vote_model <- "model{
     # Likelihood model for X
     X ~ dbin(p,n)

     # prior model for p
     p ~ dbeta(a,b)
}"

# comile the model using the JAGS
vote_jags <- jags.model(textConnection(vote_model), # textConnection() to defined 'vote_model' string
```

```
       data=list(a=45,b=55,X=6,n=10), #supply the parameters value
       inits = list(.RNG.name="base::Wichmann-Hill", .RNG.seed=100))
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1
##    Unobserved stochastic nodes: 1
##    Total graph size: 5
##
## Initializing model
```

```
# it's using the jags out of r to design an algorithm to sample from the posterior

# simulate the posterior
# in this step, we draw 10000 samples from the posterior
vote_sim <- coda.samples(model=vote_jags, # model we compiled
                         variable.names = c("p"), # parameter we interested
                         n.iter = 10000) # desired sample size of 10000 iterations

# plot the simulated posterior distribution of p from 10000 coda samples
plot(vote_sim,trace=FALSE)
```



After observing a poll in which 6 of 10 (60%) of voters supported you, your updated posterior optimism and certainty about your underlying support, p, are slightly higher than they were prior to the poll.

Bayesian Models & Markov Chains

The normal-normal model

Modelling change in reaction time

Y_i = change in reaction time (ms) after 3 days of sleep deprivation $Y_i \sim N(m, s^2)$

Assumption: Y_i is normally distributed with mean reaction time m and standard deviation s.

Prior information for parameter m:

1. Average reaction time is ~250ms with normal sleep.
2. Expect average increase by 50ms after 3 days of sleep deprivation.
3. Average reaction time is unlikely to decrease after 3 days of sleep deprivation, and also unlikely to increase by more than 150ms.

With above prior information, the average change in reaction time $m \sim N(50, 25^2)$.

Prior information for parameter s:

1. $s > 0$
2. with normal sleep, s.d in reaction time is 30ms.
3. s is equally likely to be anywhere from 0 to 200ms.

$s \sim Uniform(0, 200)$.

Likelihood:

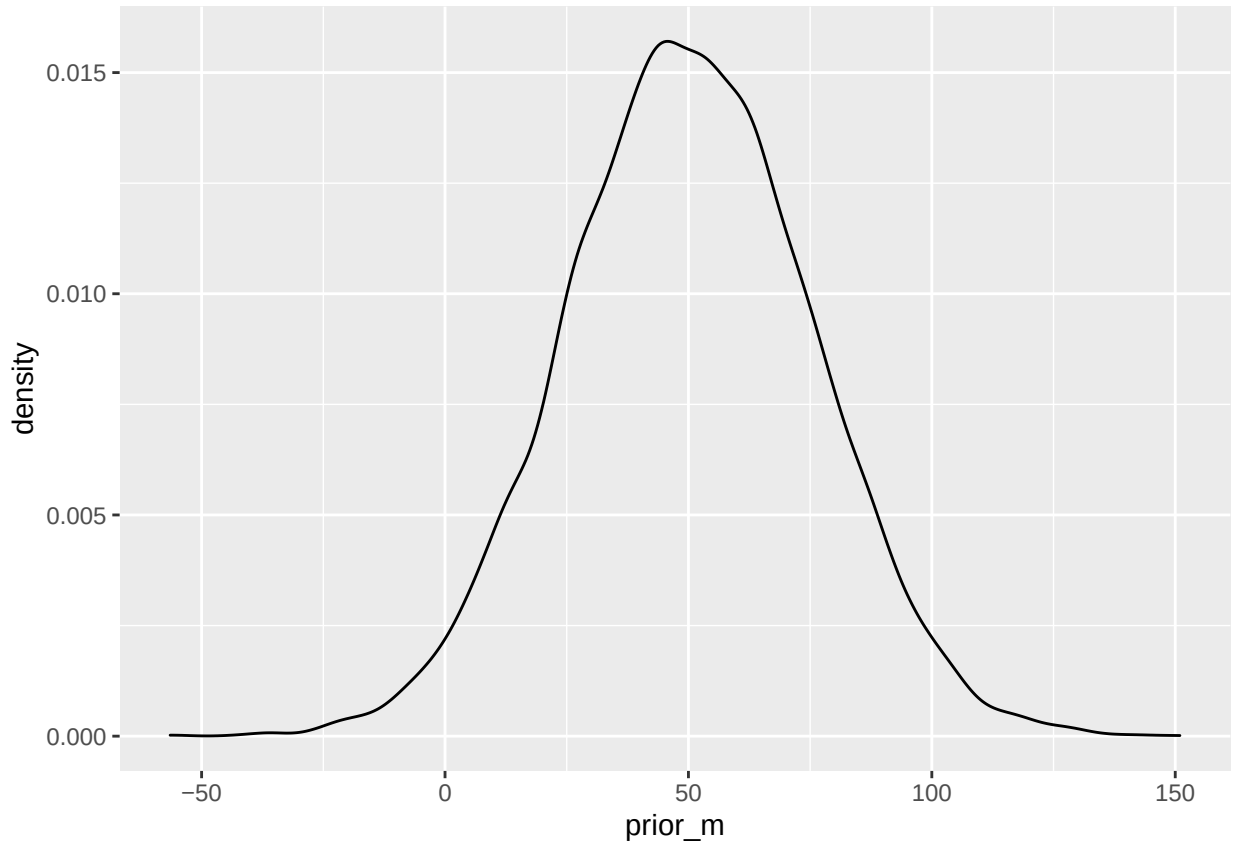
$Y_i \sim N(m, s^2)$

Prior:

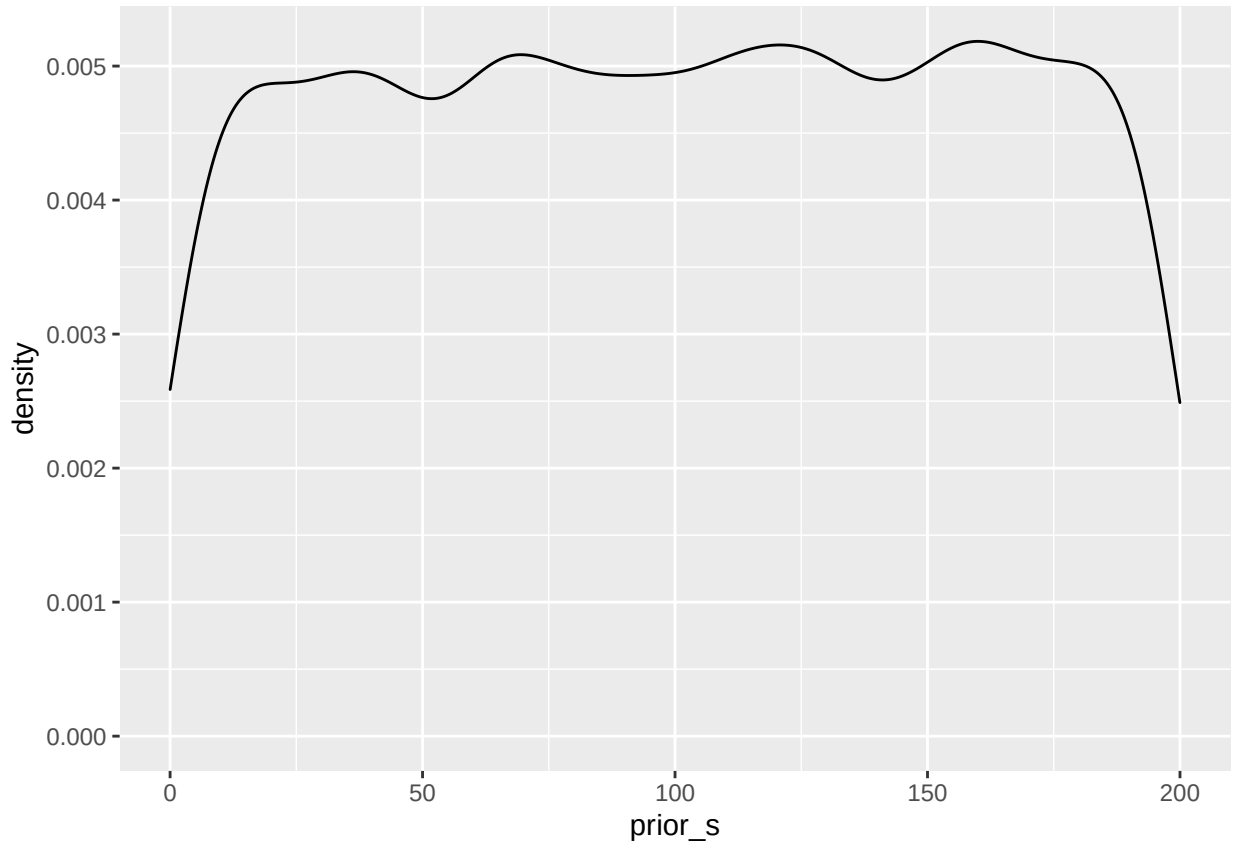
$m \sim N(50, 25^2)$

$s \sim Uniform(0, 200)$

```
# generate prior distribution of m and s

# Take 10000 samples from the m prior
prior_m <- rnorm(10000,50,25)

# Take 10000 samples from the s prior
prior_s <- runif(10000,0,200)

# Store samples in a data frame
samples <- data.frame(prior_m, prior_s)

# Density plots of the prior_m & prior_s samples
ggplot(samples, aes(x = prior_m)) +
    geom_density()
```




```
ggplot(samples, aes(x = prior_s)) +
    geom_density()
```



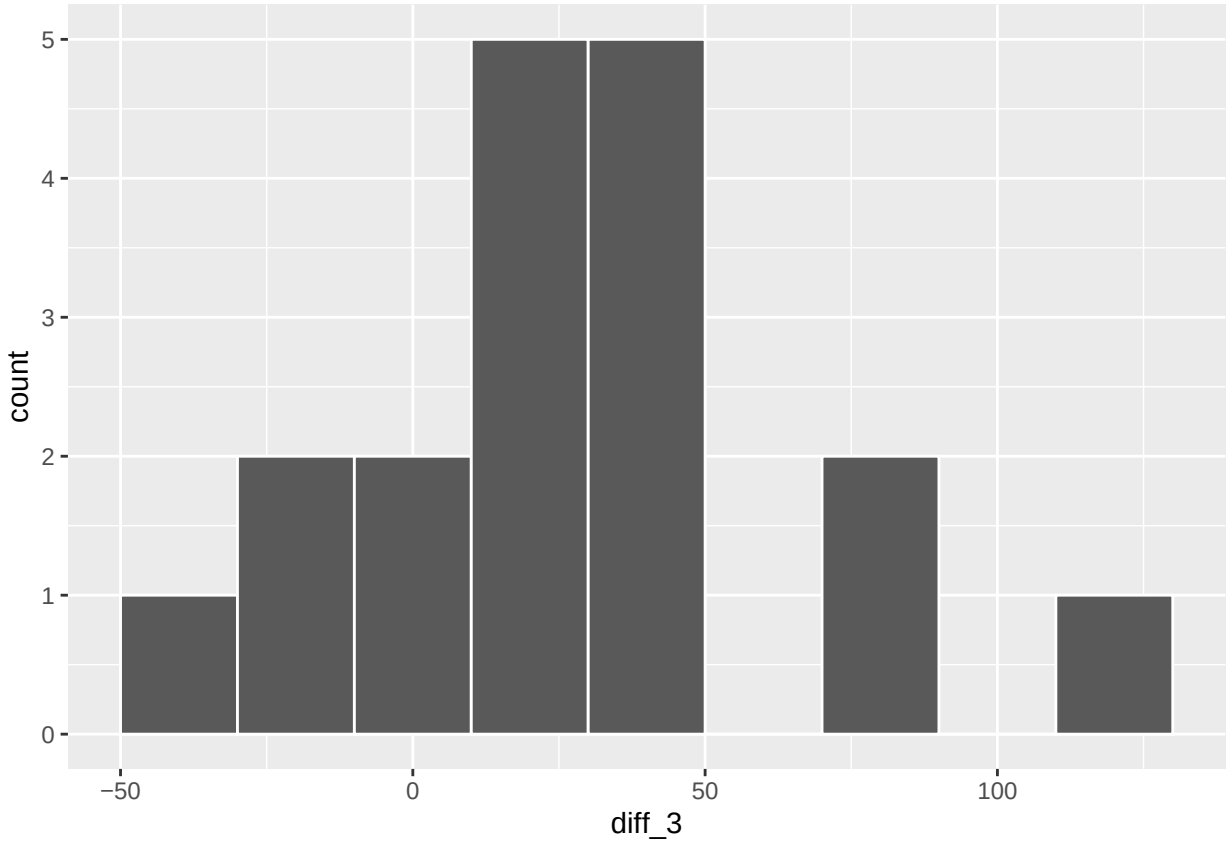
Researchers enrolled 18 subjects in a sleep deprivation study. Their observed `sleep_study` data are loaded in the workspace. These data contain the `day_0` reaction times and `day_3` reaction times after 3 sleep deprived nights for each `subject`.

```
# load data
sleep_study <- read.table(file="sleep_study.txt", header = TRUE)
head(sleep_study)
```

```
##   subject    day_0    day_3
## 1     308 249.5600 321.4398
## 2     309 222.7339 204.7070
## 3     310 199.0539 232.8416
## 4     330 321.5426 285.1330
## 5     331 287.6079 320.1153
## 6     332 234.8606 309.7688
```

```
library(dplyr)
```

```
# define the observed difference in reaction times for each subject
sleep_study <- sleep_study %>%
    mutate(diff_3 = day_3-day_0)

# Histogram of diff_3
ggplot(sleep_study, aes(x = diff_3)) +
    geom_histogram(binwidth = 20, color = "white")
```




```
# Mean and standard deviation of diff_3, the likelihood
sleep_study %>%
    summarize(mean(diff_3), sd(diff_3))
```

```
##   mean(diff_3) sd(diff_3)
## 1     26.34021   37.20764
```

Reaction times increased by an average of 26 ms with a standard deviation of 37 ms. Further, only 4 of the 18 test subjects had faster reaction times on day 3 than on day 0.

Likelihood:

$Y_i \sim N(26, 37)$

Prior:

$m \sim N(50, 25^2)$

$s \sim Uniform(0, 200)$

Then we can simulate the posterior distribution of change of reaction time after 3 days of sleep deprivation.

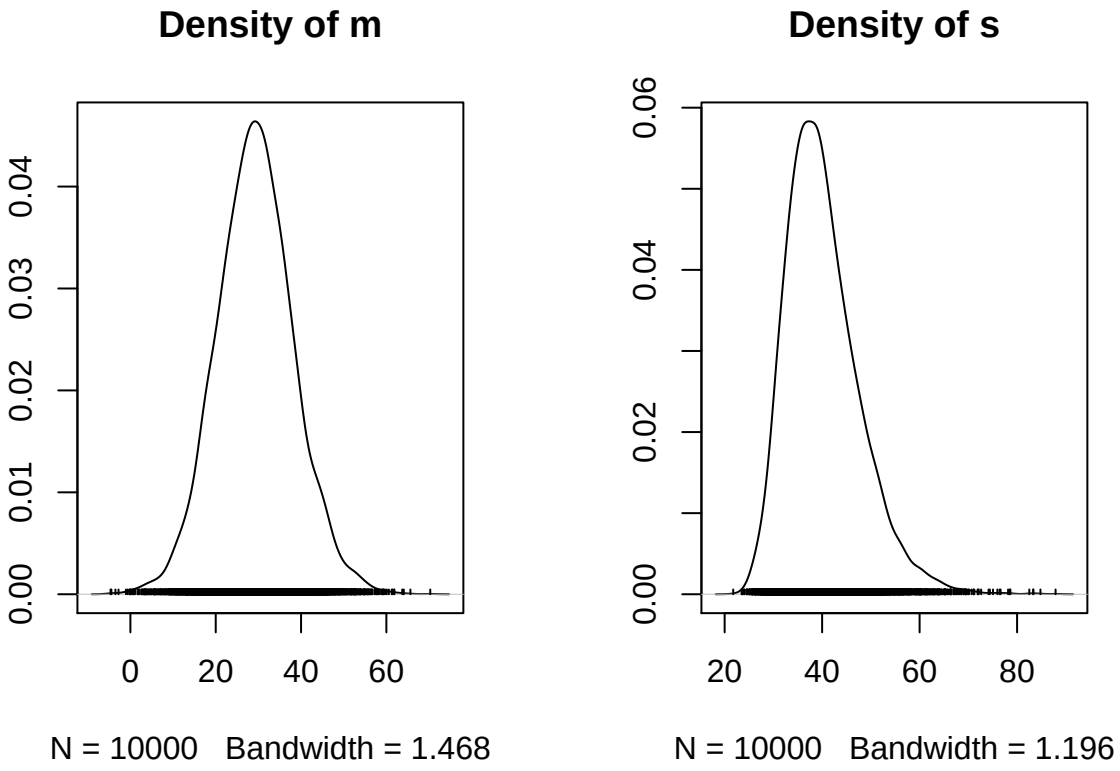
Note: In `RJAGS`, the `dnorm` is defined by `mean` and `precision` or the inverse variance. $precision = variance^{-1} = s.d^{-2}$

```
# DEFINE the model
sleep_model <- "model{
    # Likelihood model for Y[i]
```

```
    for(i in 1:length(Y)) {
        Y[i] ~ dnorm(m,s^(-2))

    }

    # Prior models for m and s
    m ~ dnorm(50,25^(-2))
    s ~ dunif(0,200)
}"

# COMPILE the model
sleep_jags <- jags.model(
  textConnection(sleep_model),
  data = list(Y = sleep_study$diff_3),
  inits = list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 1989)
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 18
##    Unobserved stochastic nodes: 2
##    Total graph size: 28
##
## Initializing model
```

```
# SIMULATE the posterior
sleep_sim <- coda.samples(model = sleep_jags, variable.names = c("m","s"), n.iter = 10000)

# PLOT the posterior
plot(sleep_sim, trace = FALSE)
```



Your posterior model is more narrow and lies almost entirely above 0, thus you're more confident that the average reaction time increases under sleep deprivation. Further, the location of the posterior is below that of the prior. This reflects the strong insight from the observed sleep study data in which the increase in average reaction time was only 26 ms.

Markov Chain

The sample of m values in `sleep_sim` is a dependent **Markov chain**, the distribution of which converges to the posterior.

```
# Check out the head of sleep_sim
head(sleep_sim)
```

```
## [[1]]
## Markov Chain Monte Carlo (MCMC) output:
## Start = 1001
## End = 1007
## Thinning interval = 1
##             m        s
## [1,] 17.25796 31.46256
## [2,] 34.58469 37.88655
## [3,] 36.45480 39.58056
## [4,] 25.00971 39.69494
## [5,] 29.95475 35.90001
## [6,] 28.43894 37.46466
```

```
## [7,] 38.32427 35.44081
##
## attr(,"class")
## [1] "mcmc.list"
```

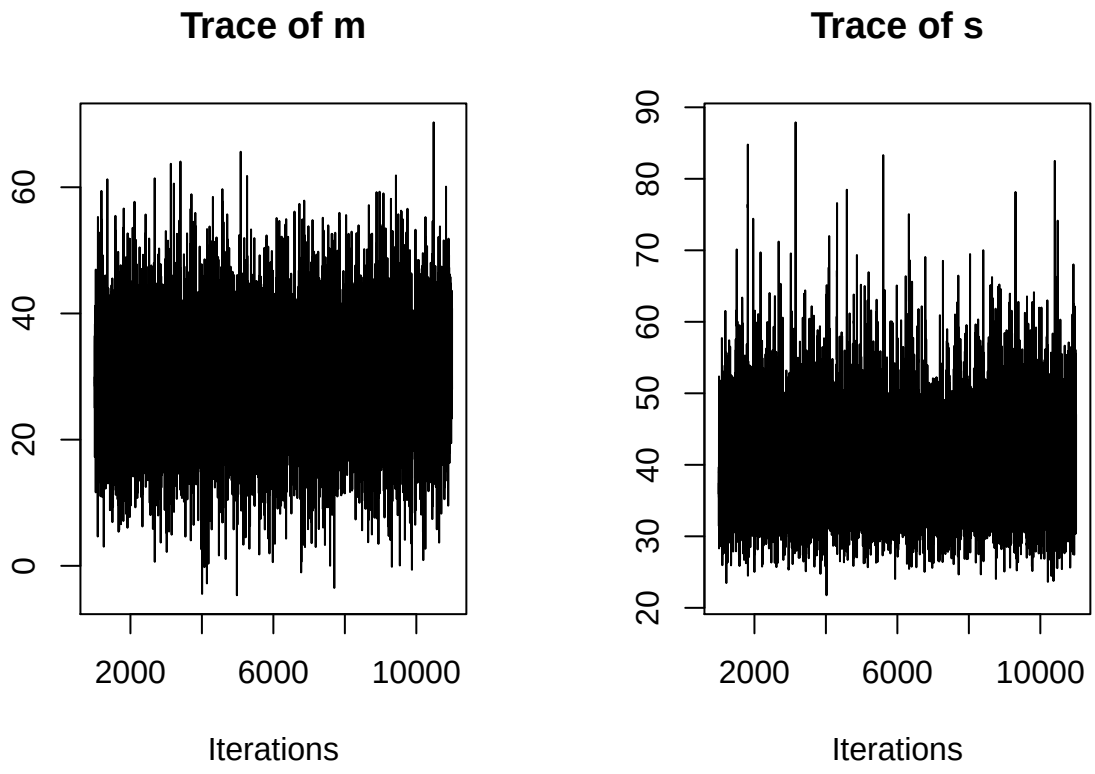
```
# Store the chains in a data frame,obtained the first list item
sleep_chains <- data.frame(sleep_sim[[1]], iter = 1:10000)

# Check out the head of sleep_chains
head(sleep_chains)
```

```
##          m        s iter
## 1 17.25796 31.46256    1
## 2 34.58469 37.88655    2
## 3 36.45480 39.58056    3
## 4 25.00971 39.69494    4
## 5 29.95475 35.90001    5
## 6 28.43894 37.46466    6
```

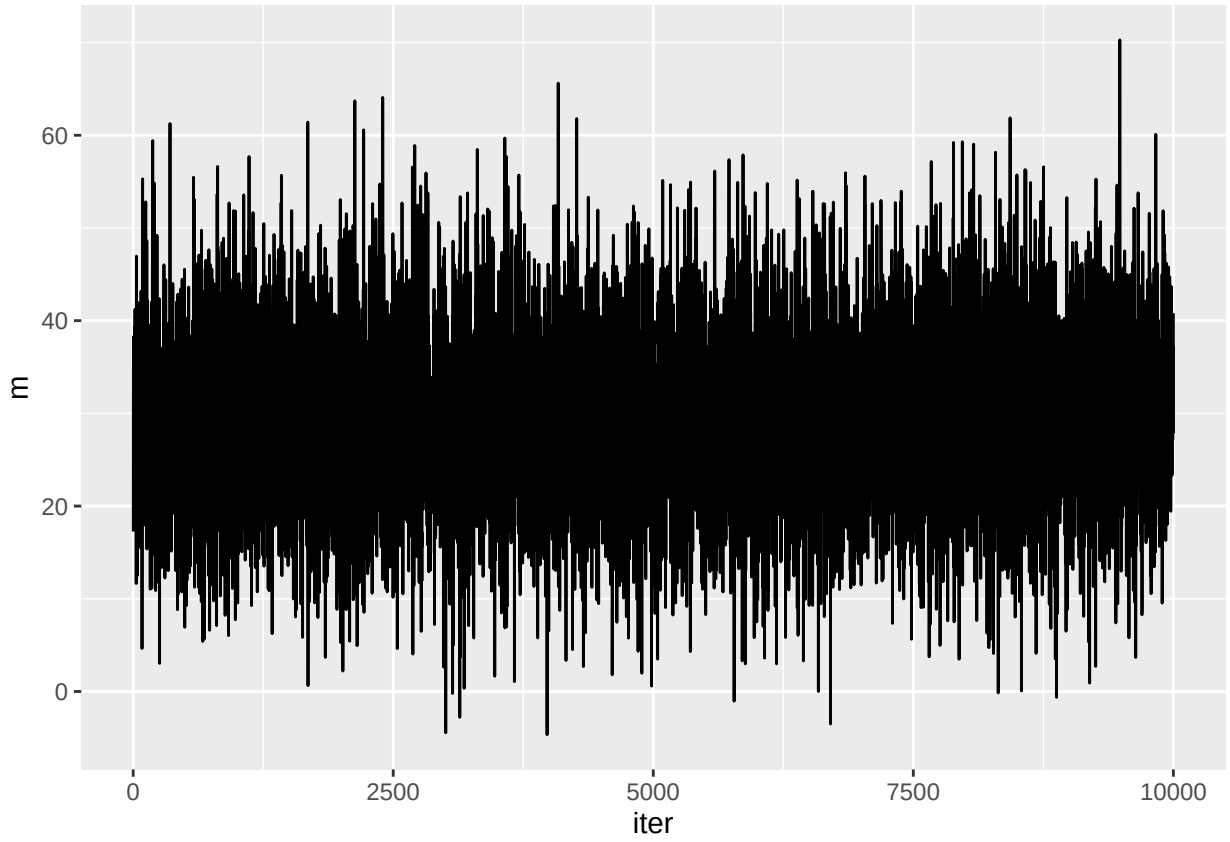
MC Trace Plot

A trace plot provides a visualization of a Markov chain's longitudinal behavior. Specifically, a trace plot for the m chain plots the observed chain value (y-axis) against the corresponding iteration number (x-axis).

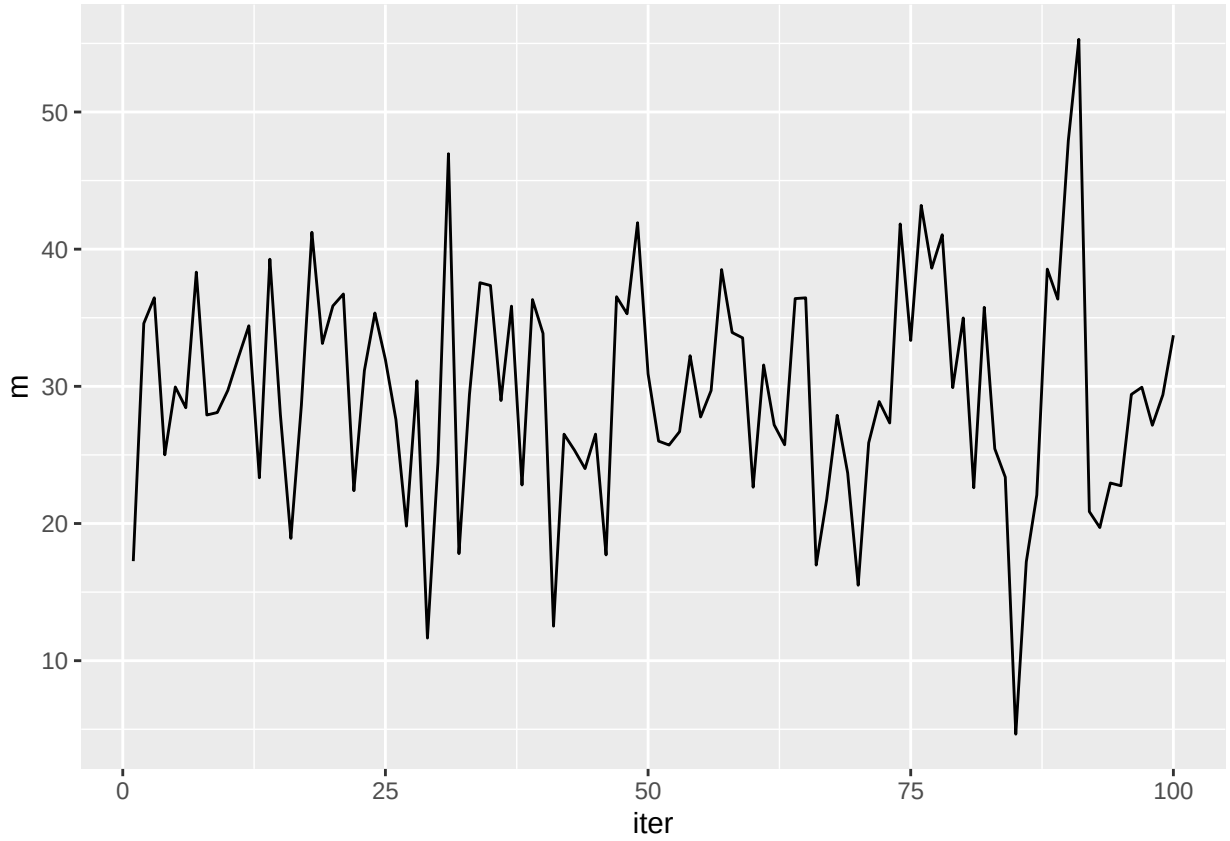
```
# Use plot() to construct trace plots of the m and s chains
plot(sleep_sim,density=FALSE)
```



```
# Use ggplot() to construct a trace plot of the m chain
ggplot(sleep_chains, aes(x = iter, y = m)) +
    geom_line()
```




```
# Trace plot the first 100 iterations of the m chain
ggplot(sleep_chains[1:100,], aes(x =iter , y = m)) +
    geom_line()
```

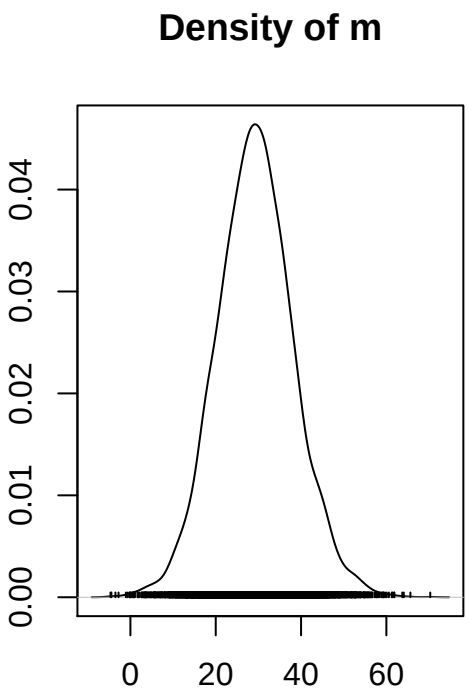


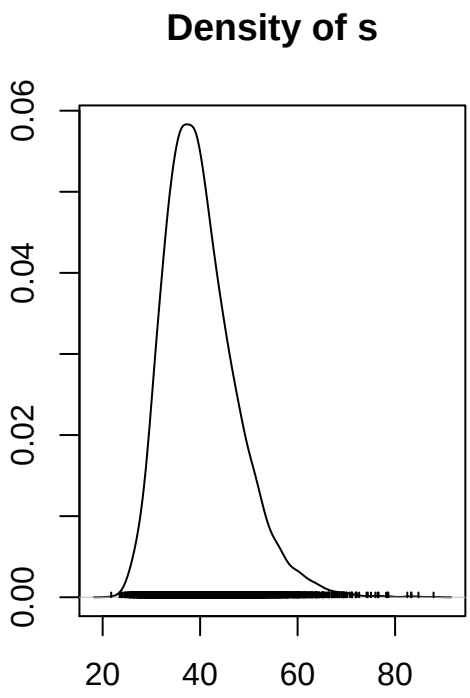
Note that the longitudinal behavior of the chain appears quite random and that the trend remains relatively constant. This is a good thing - it indicates that the Markov chain (likely) converges quickly to the posterior distribution of m.

MC Density Plot

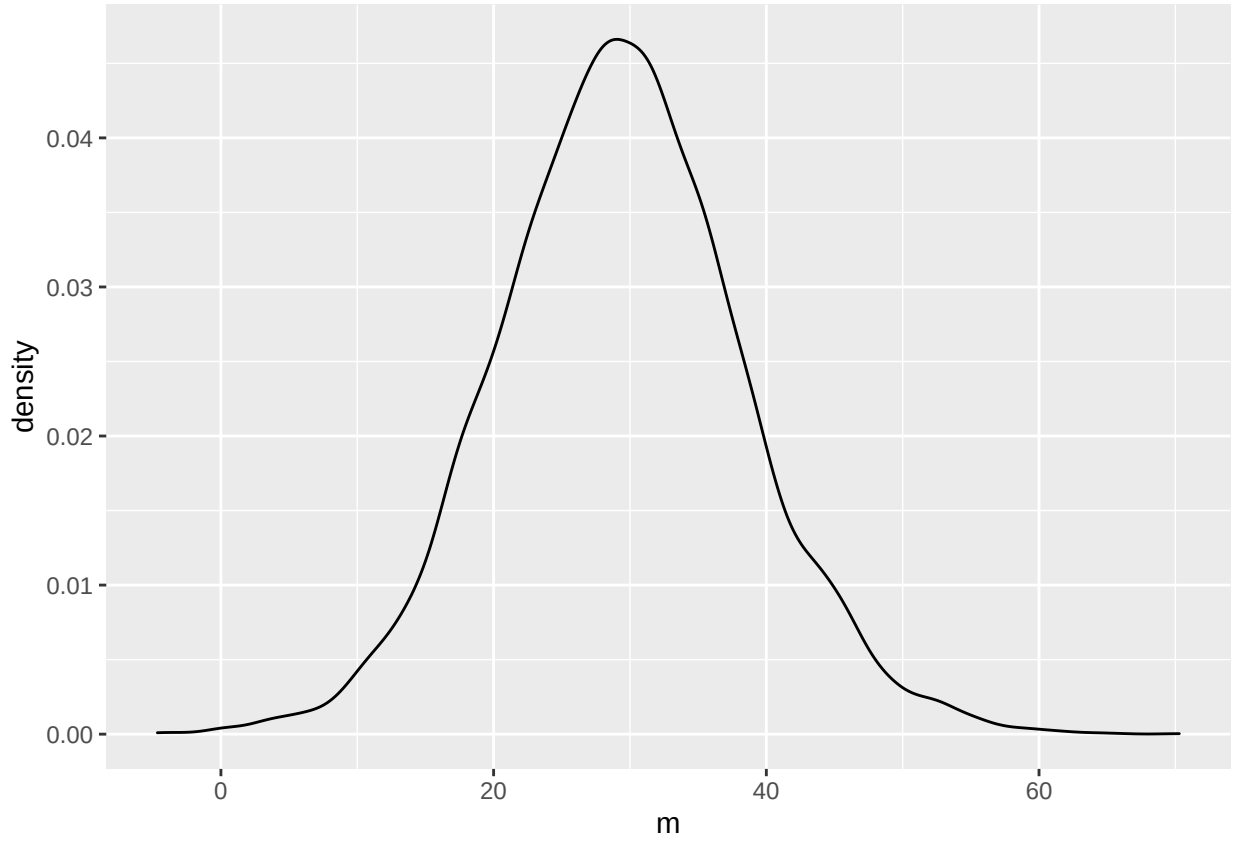
Whereas a trace plot captures a Markov chain's longitudinal behavior, a **density plot** illustrates the final distribution of the chain values. In turn, the density plot provides an approximation of the posterior model.

```
# Use plot() to construct density plots of the m and s chains
plot(sleep_sim,trace=FALSE)
```






```
# Use ggplot() to construct a density plot of the m chain
ggplot(sleep_chains, aes(x = m)) +
    geom_density()
```



These density plots approximate the posterior models of m and s.

Markov Chain Work Flow

1. Define, compile, simulate the model
2. Examine the following diagnosis
    - Trace Plot
    - Multiple chain output
    - Standard errors
3. Finalize the simulation

```
# multiple chain check

# COMPILE the model, run 4 parnell chains
sleep_jags_multi <- jags.model(textConnection(sleep_model),
                               data = list(Y = sleep_study$diff_3),                n.cha
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 18
##    Unobserved stochastic nodes: 2
##    Total graph size: 28
```

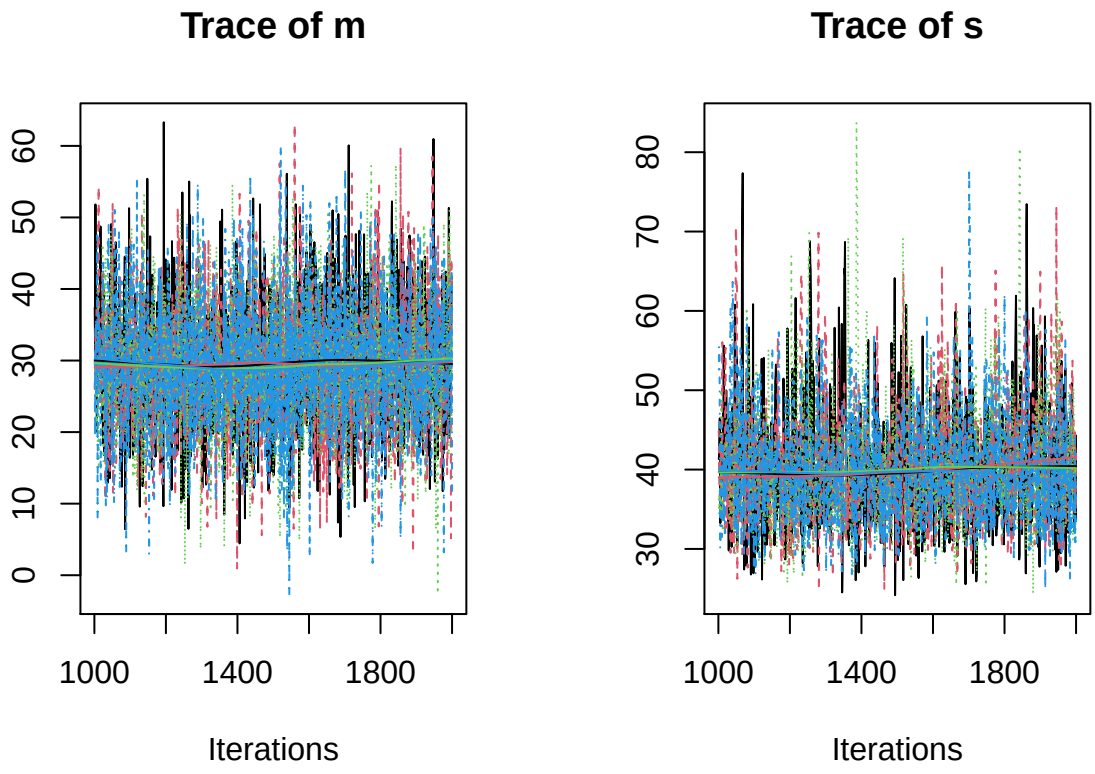
```
##
## Initializing model
```

```
# SIMULATE the posterior
sleep_sim_multi <- coda.samples(model = sleep_jags_multi, variable.names = c("m", "s"), n.iter = 1000)

# Check out the head of sleep_sim_multi
head(sleep_sim_multi)
```

```
## [[1]]
## Markov Chain Monte Carlo (MCMC) output:
## Start = 1001
## End = 1007
## Thinning interval = 1
##             m        s
## [1,] 37.37058 41.33889
## [2,] 31.41652 41.65825
## [3,] 51.81564 44.25068
## [4,] 26.24024 46.46334
## [5,] 34.62180 33.25510
## [6,] 29.65425 33.32577
## [7,] 20.06391 34.13543
##
## [[2]]
## Markov Chain Monte Carlo (MCMC) output:
## Start = 1001
## End = 1007
## Thinning interval = 1
##             m        s
## [1,] 26.00325 44.90733
## [2,] 28.44372 39.90834
## [3,] 34.72244 37.21488
## [4,] 26.44007 36.31541
## [5,] 33.45940 34.65404
## [6,] 37.47727 40.31669
## [7,] 37.46959 42.41181
##
## [[3]]
## Markov Chain Monte Carlo (MCMC) output:
## Start = 1001
## End = 1007
## Thinning interval = 1
##             m        s
## [1,] 21.74712 36.26315
## [2,] 30.34083 34.32884
## [3,] 29.65315 37.41460
## [4,] 21.26290 35.41328
## [5,] 33.54624 34.57563
## [6,] 32.38083 38.85590
## [7,] 20.02291 33.68017
##
## [[4]]
## Markov Chain Monte Carlo (MCMC) output:
## Start = 1001
```

```
## End = 1007
## Thinning interval = 1
##              m        s
## [1,] 31.33337 52.60297
## [2,] 19.35499 55.44027
## [3,] 32.81669 54.81575
## [4,] 35.80936 46.29193
## [5,] 33.96355 30.14370
## [6,] 25.39097 31.12658
## [7,] 28.71736 37.58679
##
## attr(,"class")
## [1] "mcmc.list"
```

```
# Construct trace plots of the m and s chains
plot(sleep_sim_multi, density = FALSE)
```



The most important thing to notice here is the similarity and stability among the 4 parallel chains. This provides some reassurance about the quality and consistency of our Markov chain simulation.

The mean of the m Markov chain provides an estimate of the posterior mean of m. The **naive standard error** provides a measure of the potential error in this estimate. In turn, we can use this measure to determine an appropriate chain length. For example, suppose your goal is to estimate the posterior mean of within a standard error of `0.1` ms. If your observed naive standard error exceeds this target, no problem! Simply run a longer chain - the error in using a Markov chain to approximate a posterior tends to decrease as chain length increases.

```r
# Naive standard error check

# SIMULATE the posterior
sleep_sim_1 <- coda.samples(model = sleep_jags, variable.names = c("m", "s"), n.iter = 1000)

# Summarize the m and s chains of sleep_sim_1
summary(sleep_sim_1)
```

```
##
## Iterations = 11001:12000
## Thinning interval = 1
## Number of chains = 1
## 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
## m 29.55 9.032   0.2856         0.2856
## s 40.81 7.654   0.2420         0.3414
##
## 2. Quantiles for each variable:
##
##    2.5%   25%   50%   75% 97.5%
## m 12.00 23.62 29.31 35.58 47.48
## s 28.34 35.25 39.97 45.41 58.60
```

The naive standard error of m and s exceed `0.1` ms, so will run a longer chain.

```r
# RE-SIMULATE the posterior
sleep_sim_2 <- coda.samples(model = sleep_jags, variable.names = c("m", "s"), n.iter = 10000)

# Summarize the m and s chains of sleep_sim_2
summary(sleep_sim_2)
```

```
##
## Iterations = 12001:22000
## 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
## m 29.33 9.080  0.09080         0.0942
## s 40.12 7.483  0.07483         0.1115
##
## 2. Quantiles for each variable:
##
##    2.5%   25%   50%   75% 97.5%
## m 11.69 23.25 29.25 35.24 47.55
## s 28.61 34.89 39.07 44.25 57.82
```

If the standard errors associated with your Markov chain are too big, simply increase the number of iterations. In general, naive standard error will decrease as the chain length increases.

Bayesian Regression

Regression Priors

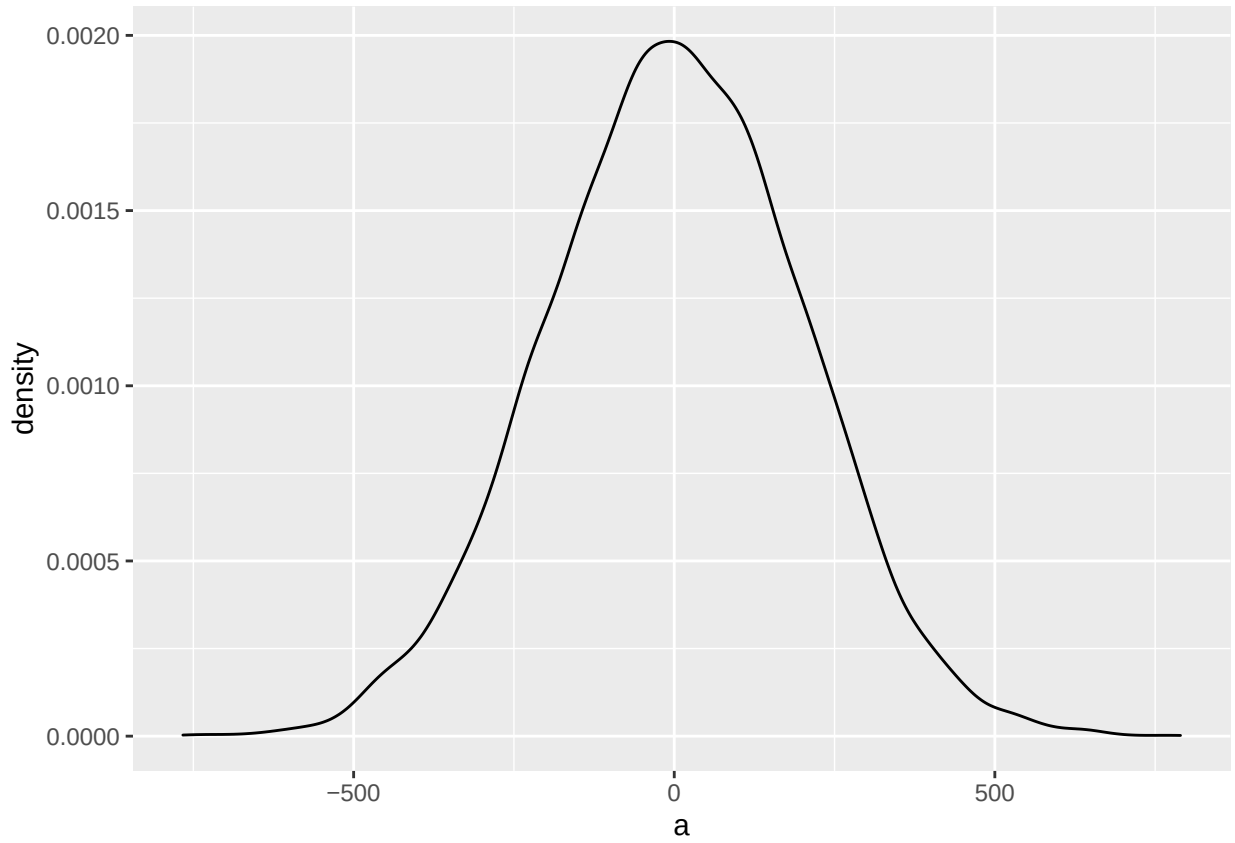
Let Y_i be the weight (in kg) of subject i. Past studies have shown that weight is linearly related to height X_i(in cm). The average weight m_i among adults of any shared height X_i can be written as $m_i = a + bX_i$. But height isn't a perfect predictor of weight - individuals vary from the trend. To this end, it's reasonable to assume that Y_i are Normally distributed around m_i with residual standard deviation s:$Y_i \sim N(m_i, s^2)$.

Note the 3 parameters in the model of weight by height: intercept a, slope b, & standard deviation s. In the first step of your Bayesian analysis, you will simulate the following prior models for these parameters:

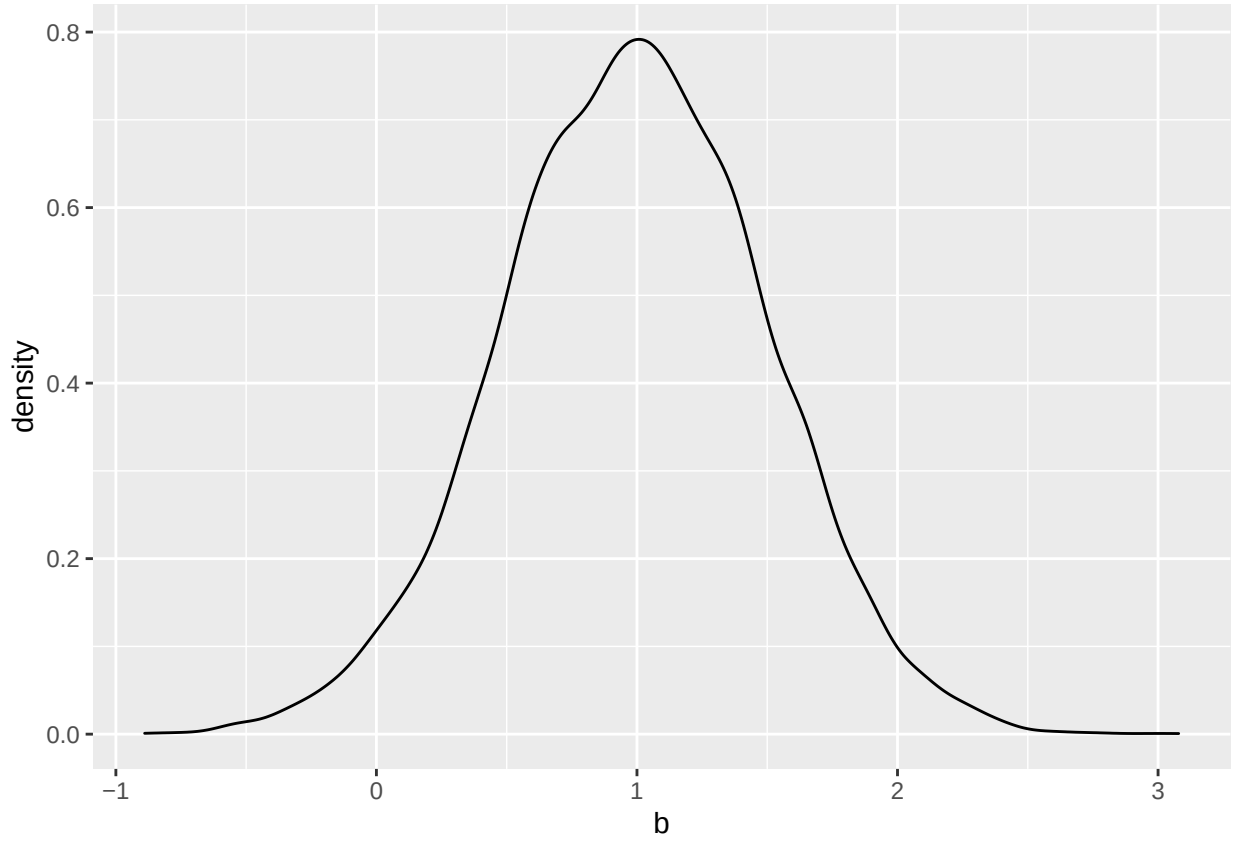
$a_{N(0,\,200^2)}$ $_b N(1, 0.5^2)$ $s \sim Unif(0\ 20)$.

Sample 10,000 draws from each of the a, b, and s priors. Assign the output to `a`, `b`, and `s`. These are subsequently combined in the `samples` data frame along with `set = 1:10000`, an indicator of the draw numbers.

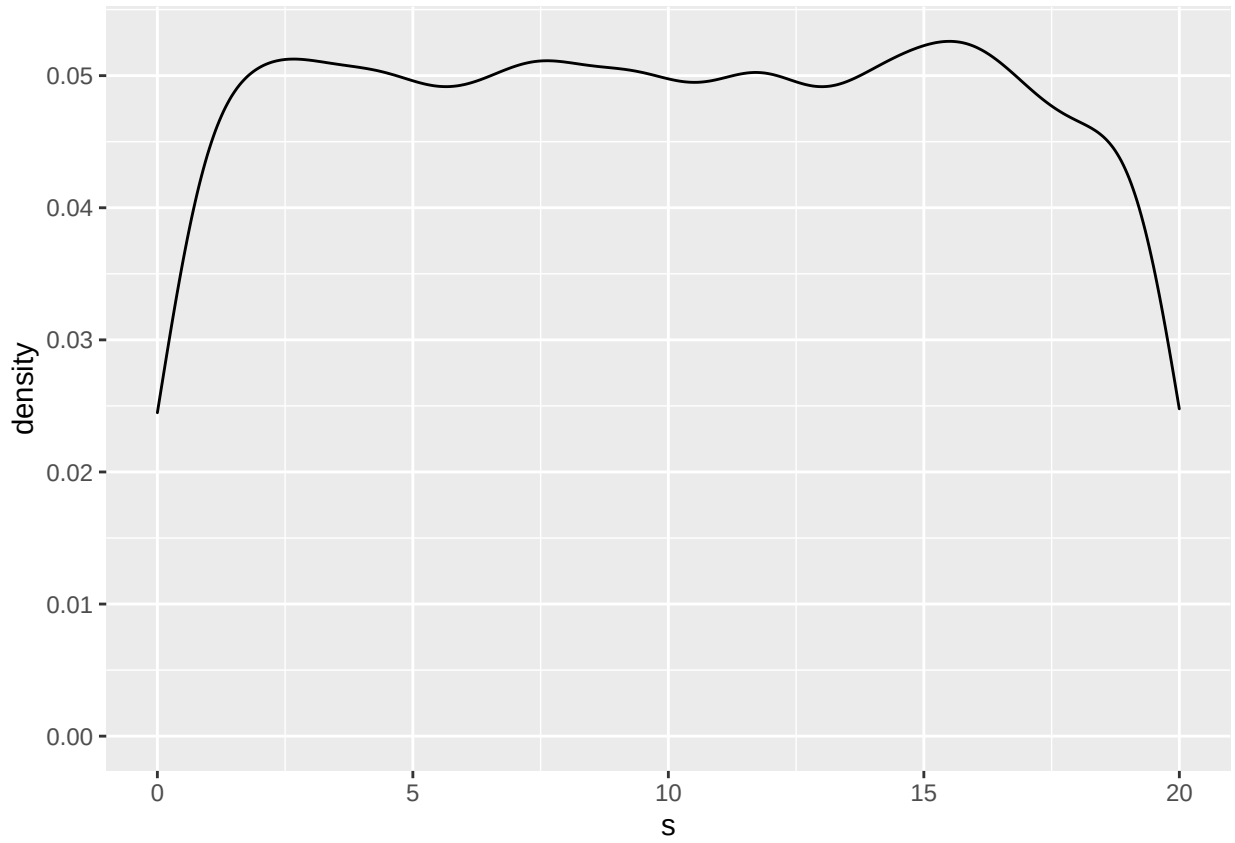
```
library(ggplot2)
# Take 10000 samples from the a, b, & s priors
a <- rnorm(10000, mean=0,sd=200)
b <- rnorm(10000, mean=1,sd=0.5)
s <- runif(10000, 0,20)

# Store samples in a data frame
samples <- data.frame(set = 1:10000, a, b, s)

# Construct density plots of the prior samples
ggplot(samples, aes(x = a)) +
    geom_density()
```




```
ggplot(samples, aes(x = b)) +
    geom_density()
```




```
ggplot(samples, aes(x = s)) +
    geom_density()
```



Visualizing the regression priors

In the previous exercise, you simulated 10,000 `samples` for each parameter (a, b, s) in the Bayesian regression model of weight Y by height X: $Y \sim N(m, s^2)$ with mean $m = a + bX$. The set of a, b, and s values in each row of `samples` represents a prior plausible regression scenario. To explore the scope of these prior scenarios, you will simulate 50 pairs of height and weight values from *each of the first 12 sets of prior parameters* a, b, and s.

Create a data frame `prior_simulation` which includes `n = 50` replicates of the first 12 sets of prior parameters in `samples` (600 rows in total!).

```
library(dplyr)
# Replicate the first 12 parameter sets 50 times each
prior_scenarios_rep <- bind_rows(replicate(n = 50, expr = samples[1:12, ], simplify = FALSE))
```

For each of the 600 `prior_simulation` rows:

Simulate a `height` value from a $N(170, 10^2)$ model.

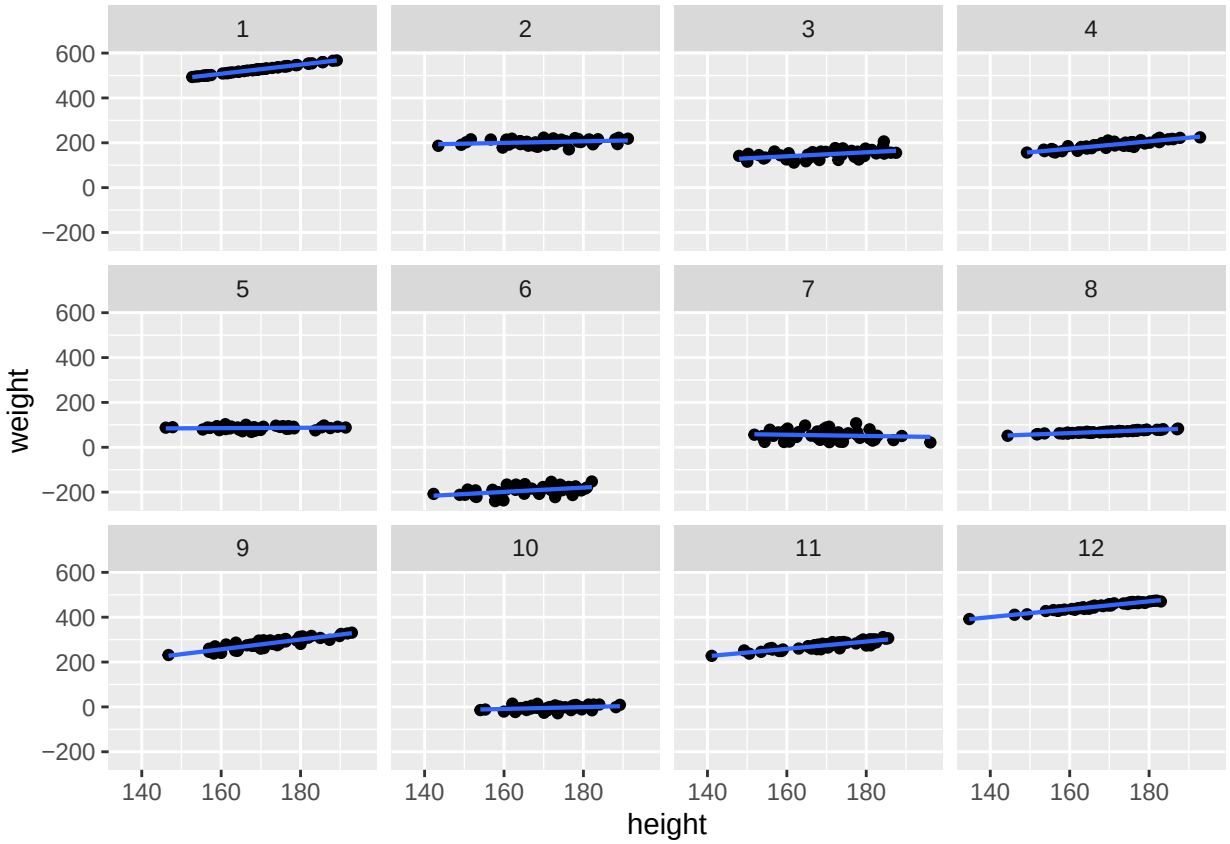
Simulate a `weight` value from $N(a + bX, s^2)$ where X is height and (a, b, s) are the prior parameter set.

```
# Simulate 50 height & weight data points for each parameter set
prior_simulation <- prior_scenarios_rep %>%
    mutate(height = rnorm(n = 600, mean = 170, sd = 10)) %>%
    mutate(weight = rnorm(n = 600, mean = (a+b*height), sd = s))
```

You now have 50 simulated `height` and `weight` pairs for each of the 12 parameter sets. Use `ggplot()` to construct a scatterplot of these 50 pairs for each `set` of parameter values. Be sure to put `weight` on the y-axis!

```
# Plot the simulated data & regression model for each parameter set
ggplot(prior_simulation, aes(x = height, y = weight)) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE, size = 0.75) +
    facet_wrap(~ set)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



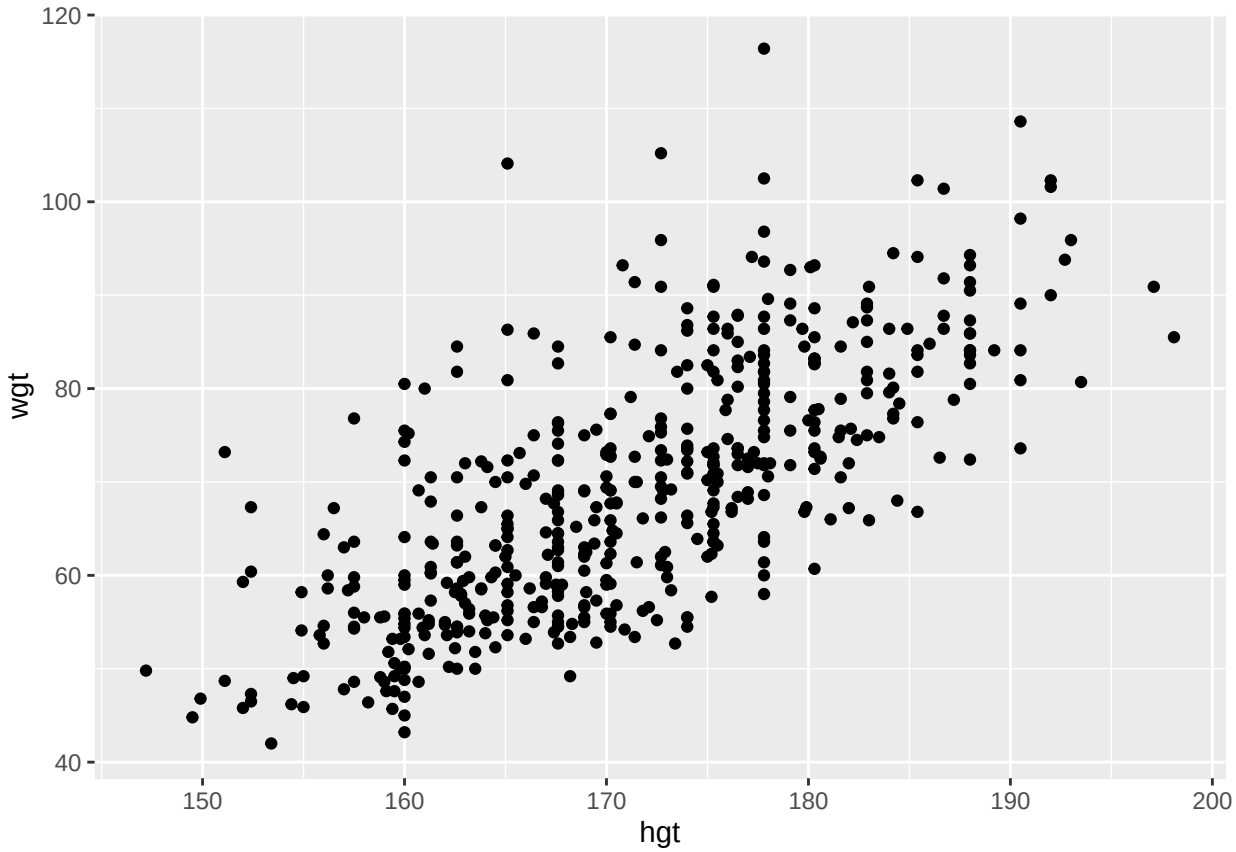
These 12 plots demonstrate the range of prior plausible models. These models have different intercepts, slopes, and residual standard deviations. Almost all of the models have positive slopes, demonstrating the prior information that there is likely a positive association between weight & height. Given your vague prior for `a`, some of these models are even biologically impossible!

Weight and Height Data

The `bdims` data set from the `openintro` package is loaded in your workspace. `bdims` contains physical measurements on a sample of 507 individuals, including their weight in kg (`wgt`) and height in cm (`hgt`). You will use these data to build insights into the relationship between weight and height.

Construct a scatterplot of `wgt` (y-axis) vs `hgt` (x-axis) using `ggplot()` with a `geom_point()` layer.

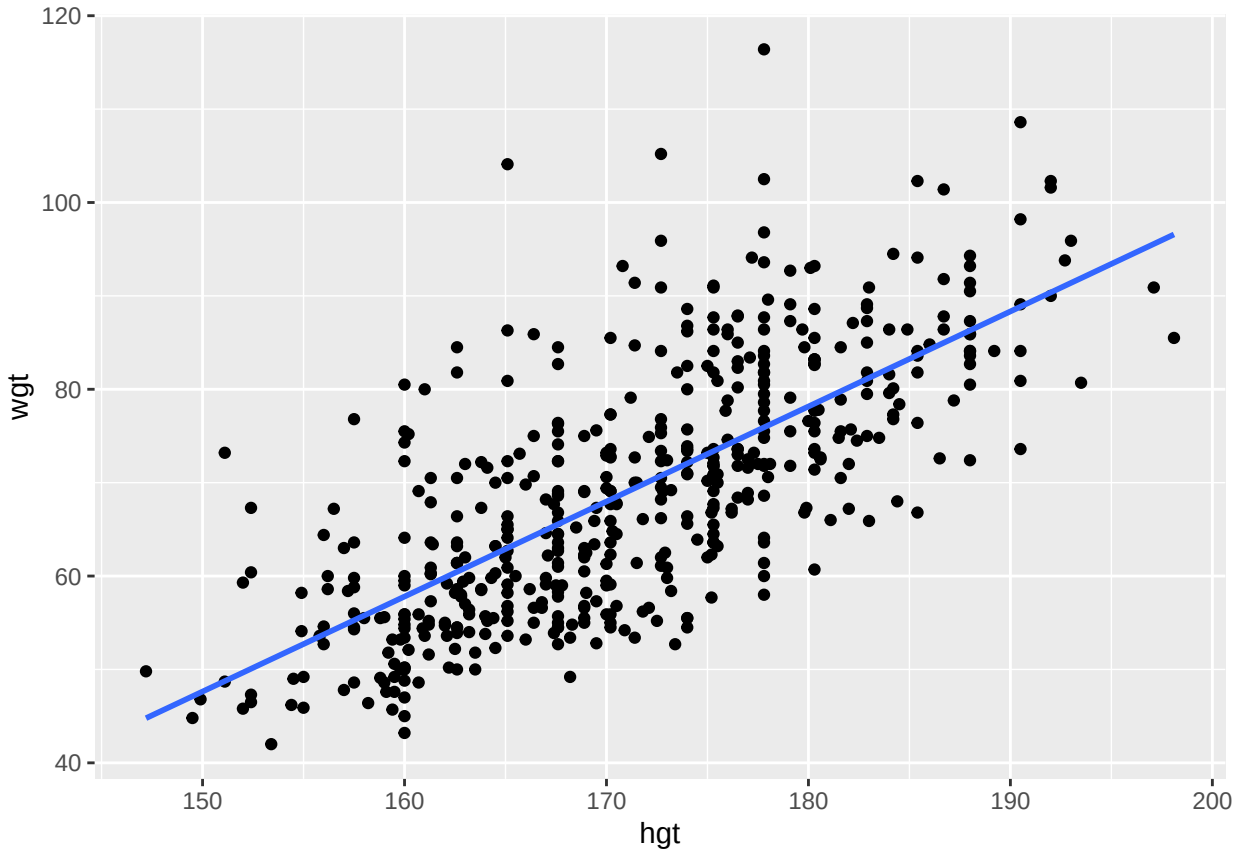
```
library(openintro)

# Construct a scatterplot of wgt vs hgt
ggplot(bdims, aes(x = hgt, y = wgt)) +
    geom_point()
```



Construct a scatterplot of `wgt` vs `hgt` which includes a `geom_smooth()` of the linear relationship between these 2 variables.

```
# Add a model smooth
ggplot(bdims, aes(x = hgt, y = wgt)) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



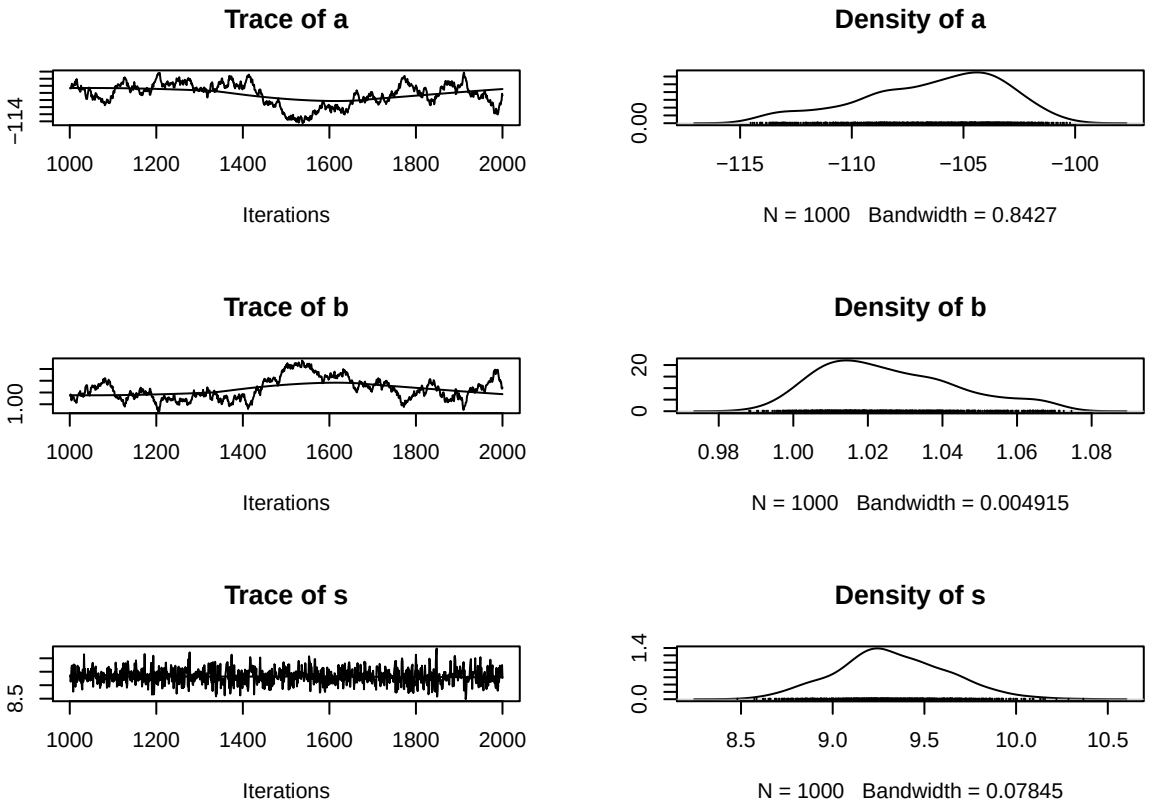
These data support your prior information about a positive association between weight and height. With insights from the priors and data in place, you're ready to simulate the posterior regression model in RJAGS!

```
# Define the Bayesian model
weight_model <- "model{
                # Likelihood model for Y[i]
                for (i in 1:length(Y)) {
                   Y[i] ~ dnorm(m[i],s^(-2))
                   m[i] <- a+b*X[i]
                   }
                # Priors models for a, b and s
                a ~ dnorm(0,200^(-2))
                b ~ dnorm(1, 0.5^(-2))
                s ~ dunif(0,20)
}"

# COMPILE the model
weight_jags <- jags.model(
  textConnection(weight_model),
  data = list(Y = bdims$wgt, X = bdims$hgt),
  inits = list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 1989)
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
```

```
## Graph information:
##    Observed stochastic nodes: 507
##    Unobserved stochastic nodes: 3
##    Total graph size: 1321
##
## Initializing model
```

```
# SIMULATE the posterior
weight_sim <- coda.samples(model =weight_jags, variable.names = c('a','b','s'), n.iter = 1000)

# PLOT the posterior
plot(weight_sim)
```



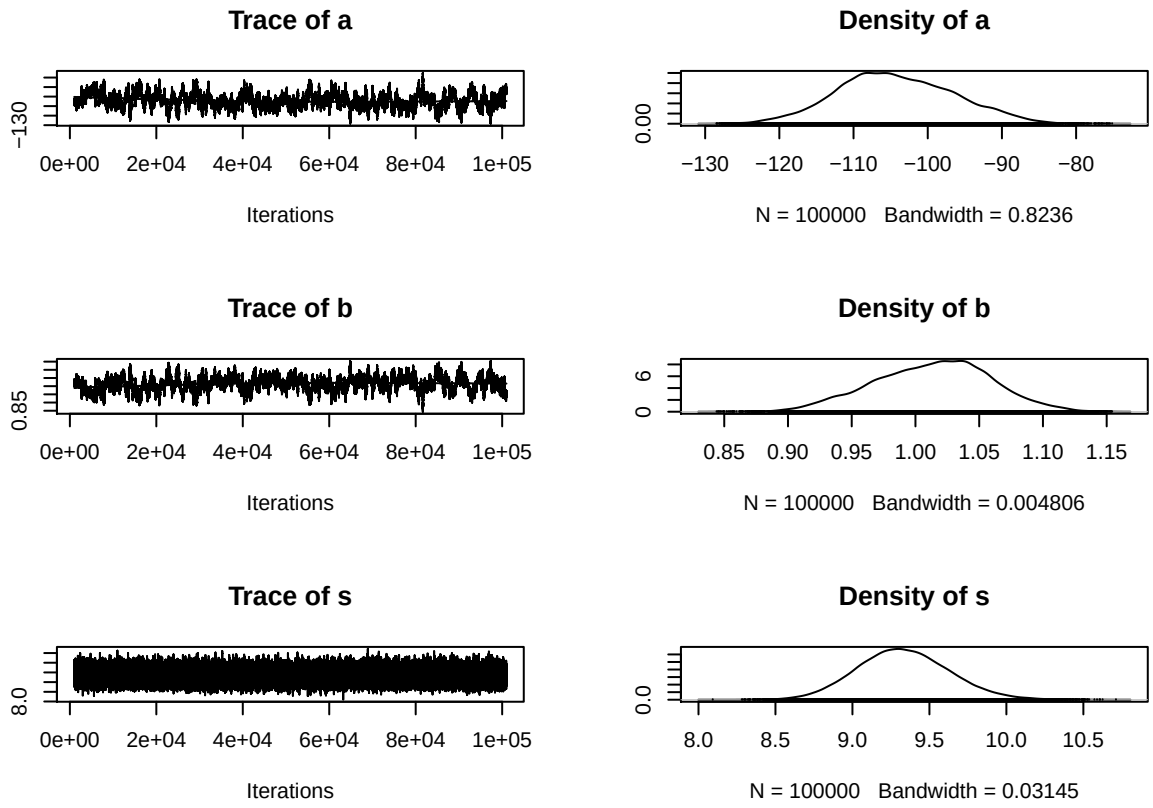
The Markov Chian mixing is not good. The length of chain 1000 is too short. We might also increase the stability of our simulation by *standardizing the height data*, but this is beyond the scope of our current discussion.

```
# Construct a 100000 length chain

# Define the Bayesian model
weight_model <- "model{
                # Likelihood model for Y[i]
                for (i in 1:length(Y)) {
                   Y[i] ~ dnorm(m[i],s^(-2))
                   m[i] <- a+b*X[i]
                   }
```

```
                # Priors models for a, b and s
                a ~ dnorm(0,200^(-2))
                b ~ dnorm(1, 0.5^(-2))
                s ~ dunif(0,20)
}"

# COMPILE the model
weight_jags <- jags.model(
  textConnection(weight_model),
  data = list(Y = bdims$wgt, X = bdims$hgt),
  inits = list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 1989)
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 507
##    Unobserved stochastic nodes: 3
##    Total graph size: 1321
##
## Initializing model
```

```
# SIMULATE the posterior
weight_sim_big <- coda.samples(model =weight_jags, variable.names = c('a','b','s'), n.iter = 100000)

# PLOT the posterior
plot(weight_sim_big)
```




```
# Store the chains in a data frame,obtained the first list item
weight_chains <- data.frame(weight_sim_big[[1]], iter = 1:100000)
```

Posterior point estimates

Recall the likelihood of the Bayesian regression model of weight Y by height X: $Y \sim N(m, s^2)$ where $m = a + bX$. A 100,000 iteration RJAGS simulation of the posterior, `weight_sim_big`, is in your workspace along with a data frame of the Markov chain output:

```
head(weight_chains, 2)
```

```
##           a        b        s iter
## 1 -104.7251 1.014141 9.369227    1
## 2 -104.5230 1.016743 9.131354    2
```

The posterior means of the intercept & slope parameters, a & b, reflect the posterior mean trend in the relationship between weight & height. In contrast, the full posteriors of a & b reflect the range of plausible parameters, thus posterior uncertainty in the trend. You will examine the trend and uncertainty in this trend below. The `bdims` data are in your workspace.

Obtain `summary()` statistics of the `weight_sim_big` chains.

```
# Summarize the posterior Markov chains
summary(weight_sim_big)
```

```
##
## Iterations = 1001:101000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 1e+05
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##       Mean      SD  Naive SE Time-series SE
## a -104.252 7.76988 0.0245705       0.646283
## b    1.013 0.04534 0.0001434       0.003782
## s    9.331 0.29674 0.0009384       0.001212
##
## 2. Quantiles for each variable:
##
##        2.5%       25%      50%     75%   97.5%
## a -118.8879 -109.7149 -104.717 -98.855 -88.712
## b    0.9224    0.9817    1.016   1.045   1.098
## s    8.7742    9.1271    9.322   9.526   9.939
```

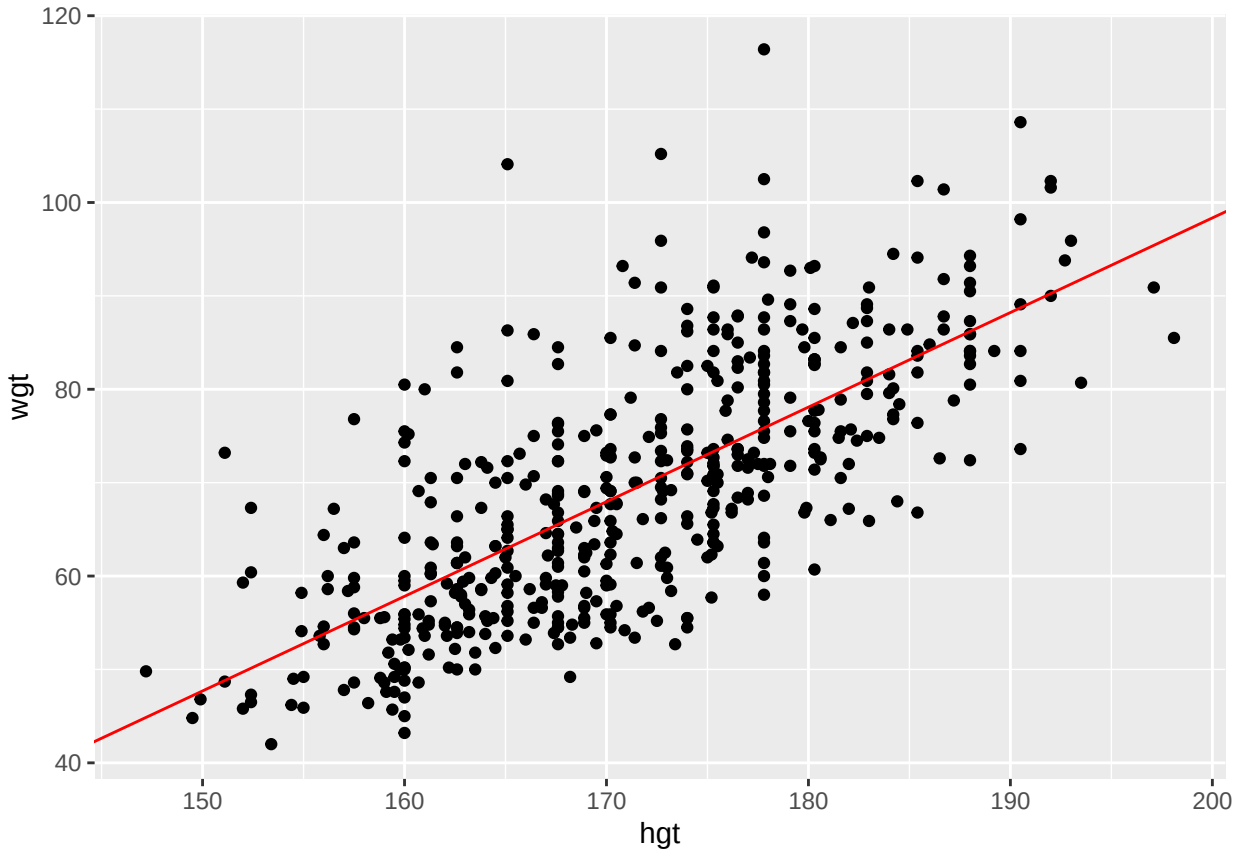
The posterior mean of b is reported in Table 1 of the `summary()`. Use the raw `weight_chains` to verify this calculation.

```
mean(weight_chains$b)
```

```
## [1] 1.013191
```

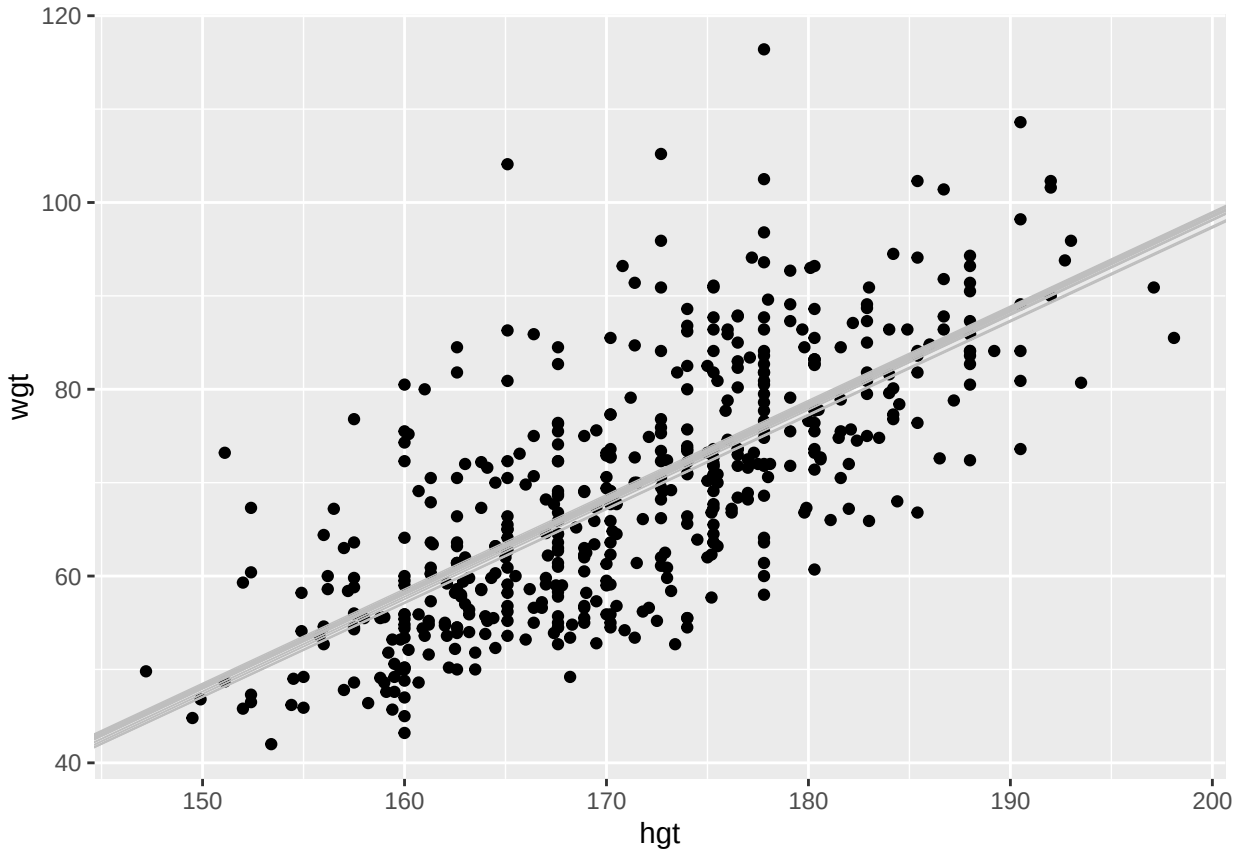
Construct a scatterplot of the `wgt` vs `hgt` data in `bdims`. Use `geom_abline()` to superimpose the posterior mean trend.

```
# Plot the posterior mean regression model
ggplot(bdims, aes(x = hgt, y = wgt)) +
    geom_point() +
    geom_abline(intercept = -104.252, slope = 1.013, color = "red")
```



Construct another scatterplot of `wgt` vs `hgt`. Superimpose the 20 regression lines defined by the first 20 sets of a & b parameter values in `weight_chains`.

```
# Visualize the range of 20 posterior regression models
ggplot(bdims, aes(x = hgt, y = wgt)) +
    geom_point() +
    geom_abline(intercept = weight_chains$a[1:20], slope = weight_chains$b[1:20], color = "gray", size =
```



You now have a sense of the posterior mean trend in the linear relationship between weight and height as well as the posterior uncertainty in this trend. Given the size of the data and selection of priors, the posterior uncertainty is noticeably small as evidenced by the tight distribution of the gray posterior plausible lines around the trend.

Posterior credible intervals

Let's focus on slope parameter b, the rate of change in weight over height. The *posterior mean* of b reflects the trend in the posterior model of the slope. In contrast, a posterior *credible interval* provides a range of posterior plausible slope values, thus reflects posterior uncertainty about b. For example, the 95% credible interval for b ranges from the 2.5th to the 97.5th quantile of the posterior. Thus there's a 95% (posterior) chance that b is in this range.

You will use RJAGS simulation output to approximate credible intervals for b. The 100,000 iteration RJAGS simulation of the posterior, `weight_sim_big`, is in your workspace along with a data frame of the Markov chain output, `weight_chains`.

Obtain `summary()` statistics of the `weight_sim_big` chains.

```
# Summarize the posterior Markov chains
summary(weight_sim_big)
```

```
##
## Iterations = 1001:101000
## Thinning interval = 1
## Number of chains = 1
```

```
## Sample size per chain = 1e+05
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##       Mean      SD  Naive SE Time-series SE
## a -104.252 7.76988 0.0245705       0.646283
## b    1.013 0.04534 0.0001434       0.003782
## s    9.331 0.29674 0.0009384       0.001212
##
## 2. Quantiles for each variable:
##
##        2.5%       25%      50%     75%   97.5%
## a -118.8879 -109.7149 -104.717 -98.855 -88.712
## b    0.9224    0.9817    1.016   1.045   1.098
## s    8.7742    9.1271    9.322   9.526   9.939
```

The `2.5%` and `97.5%` posterior quantiles for b are reported in Table 2 of the `summary()`. Apply `quantile()` to the raw `weight_chains` to verify these calculations. Save this as `ci_95` and print it.

```
# Calculate the 95% posterior credible interval for b
ci_95 <- quantile(weight_chains$b, probs = c(0.025, 0.975))
ci_95
```

```
##      2.5%     97.5%
## 0.9224311 1.0983272
```

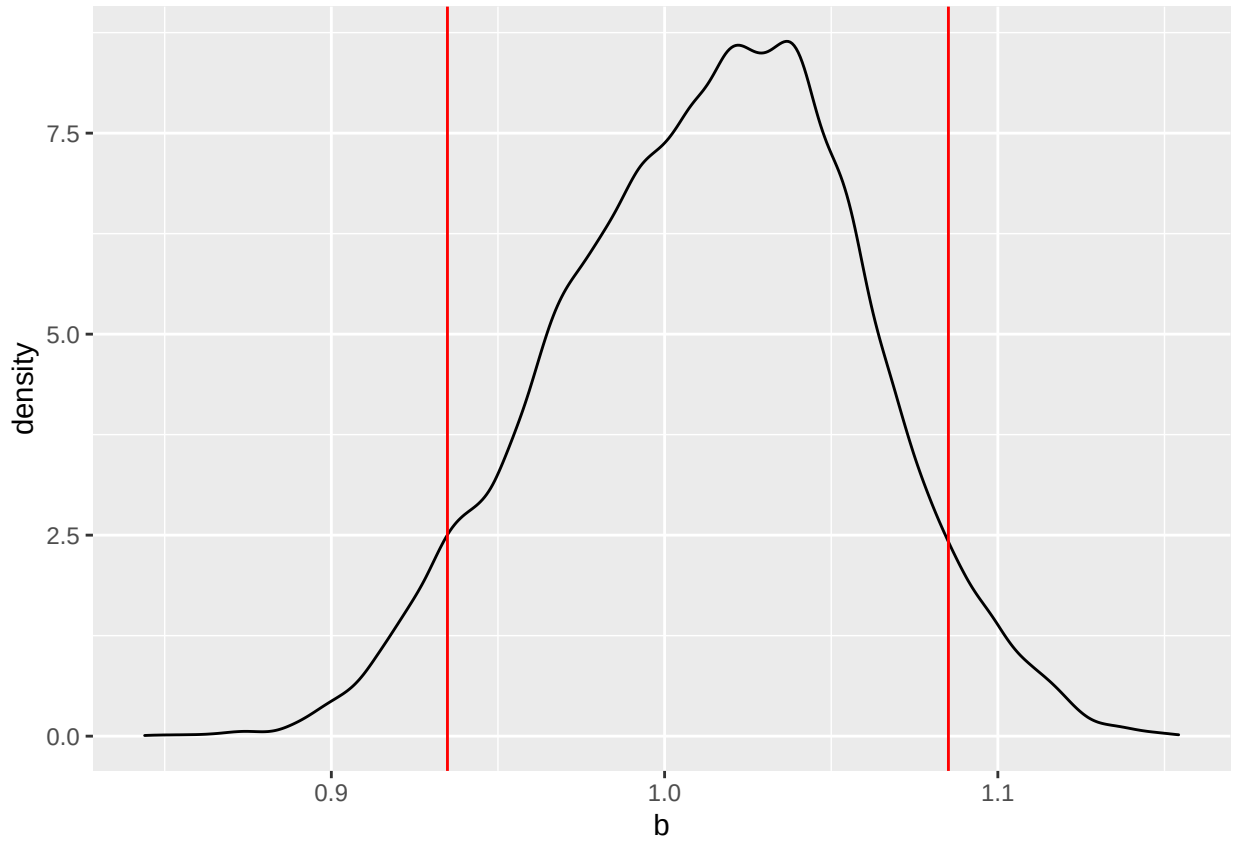
Similarly, use the `weight_chains` data to construct a 90% credible interval for b. Save this as `ci_90` and print it.

```
# Calculate the 90% posterior credible interval for b
ci_90 <- quantile(weight_chains$b, probs=c(0.05,0.95))
ci_90
```

```
##        5%       95%
## 0.9348413 1.0851638
```

Construct a density plot of the b Markov chain values. Superimpose vertical lines representing the 90% credible interval for b using `geom_vline()` with `xintercept = ci_90`.

```
# Mark the 90% credible interval
ggplot(weight_chains, aes(x = b)) +
    geom_density() +
    geom_vline(xintercept = ci_90, color = "red")
```



Based on your calculations we can say that there's a 90% (posterior) probability that, on average, the increase in weight per 1 cm increase in height is between 0.93 and 1.08 kg.

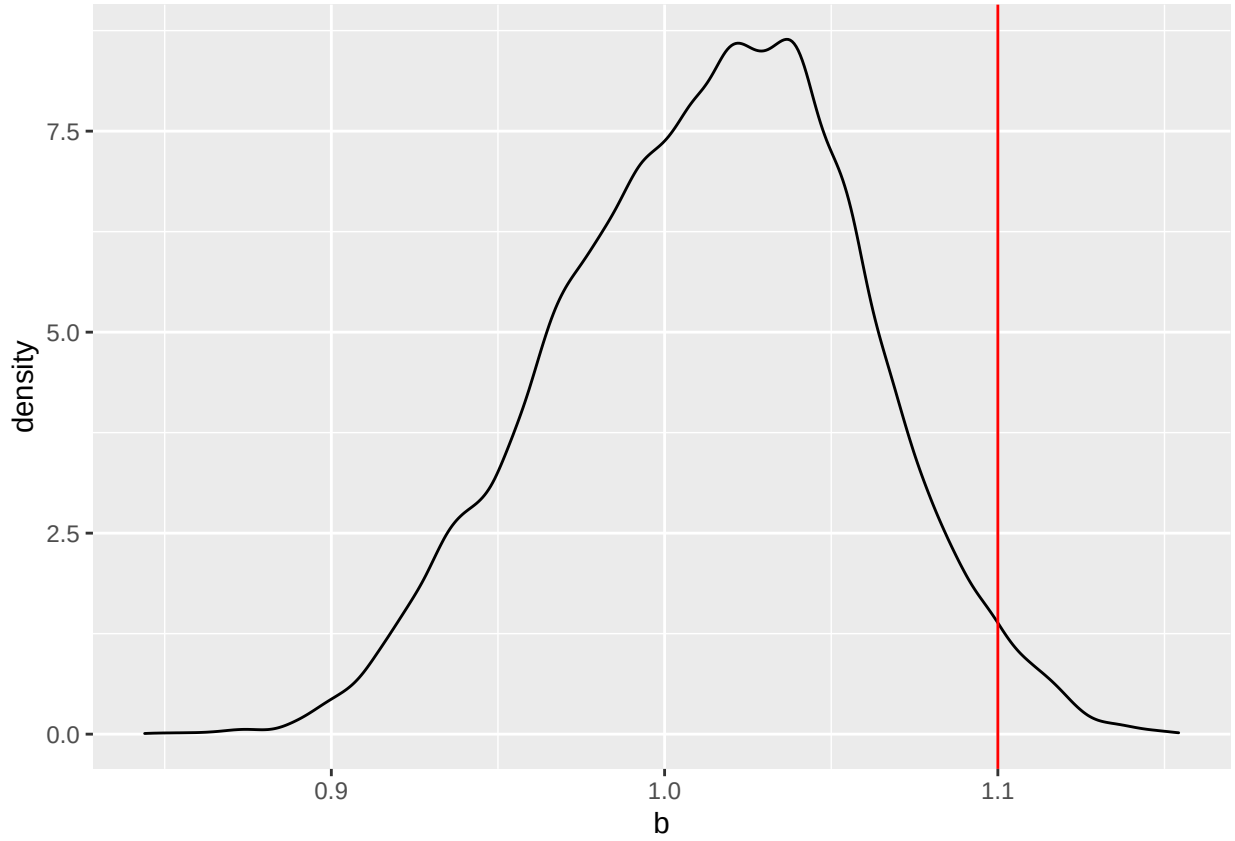
Posterior probabilities

You've used RJAGS output to explore and quantify the posterior trend & uncertainty b. You can also use RJAGS output to assess specific hypotheses. For example: What's the posterior probability that, on average, weight increases by more than 1.1 kg for every 1 cm increase in height? That is, what's the posterior probability that $b > 1.1$?

You will approximate this probability by the proportion of b Markov chain values that exceed 1.1. The `weight_chains` data frame with the 100,000 iteration Markov chain output is in your workspace.

Construct a density plot of the b Markov chain values and use `geom_vline()` to superimpose a vertical line at 1.1.

```
# Mark 1.1 on a posterior density plot for b
ggplot(weight_chains, aes(x = b)) +
    geom_density() +
    geom_vline(xintercept = 1.1, color = "red")
```



Use `table()` to summarize the number of b Markov chain values that exceed 1.1.

```
# Summarize the number of b chain values that exceed 1.1
table(weight_chains$b>1.1)
```

```
##
## FALSE  TRUE
## 97751  2249
```

Use `mean()` to calculate the proportion of b Markov chain values that exceed 1.1.

```
# Calculate the proportion of b chain values that exceed 1.1
mean(weight_chains$b>1.1)
```

```
## [1] 0.02249
```

Based on your calculations we can say that there's only a ~2% (posterior) chance that, on average, the increase in weight per 1 cm increase in height exceeds 1.1 kg.

Inference for the posterior trend

You will use RJAGS simulation output to approximate the posterior trend in weight among 180 cm tall adults as well as the posterior uncertainty in this trend. The 100,000 iteration RJAGS simulation of the

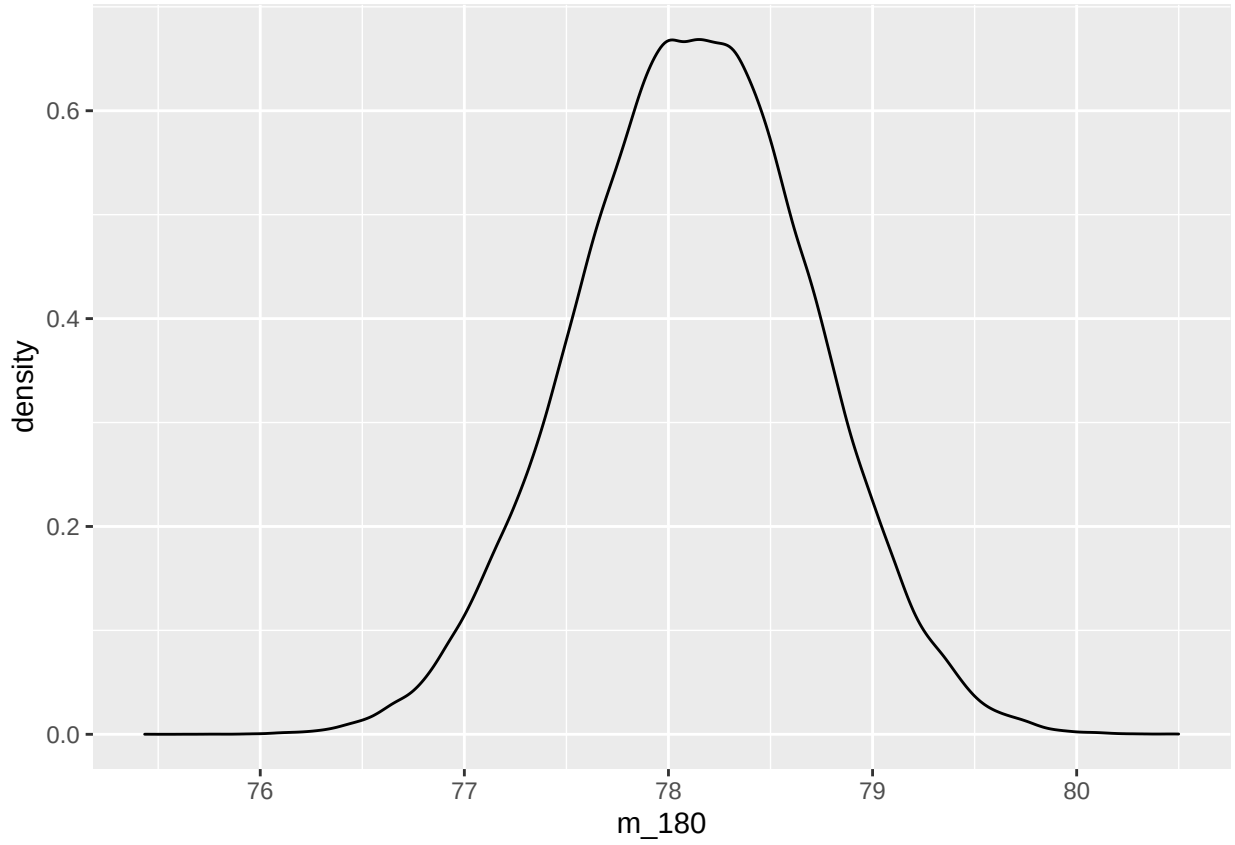
posterior, `weight_sim_big`, is in your workspace along with a data frame of the Markov chain output, `weight_chains`.

`weight_chains` contains 100,000 sets of posterior plausible parameter values of a and b. From each, calculate the mean (typical) weight among 180 cm tall adults, $a + b * 180$. Store these trends as a new variable `m_180` in `weight_chains`.

```
# Calculate the trend under each Markov chain parameter set
weight_chains <- weight_chains  %>%
    mutate(m_180 = a+b*180)
```

Construct a posterior density plot of 100,000 `m_180` values.

```
# Construct a posterior density plot of the trend
ggplot(weight_chains, aes(x = m_180)) +
    geom_density()
```



Use the 100,000 `m_180` values to calculate a 95% posterior credible interval for the mean weight among 180 cm tall adults.

```
# Construct a posterior credible interval for the trend
quantile(weight_chains$m_180, probs = c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 76.97899 79.23839
```

The posterior trend of your regression model indicates that the typical weight among 180 cm tall adults is roughly 78 kg. However, posterior uncertainty in the regression model trickles down to uncertainty about this trend. This uncertainty is communicated through your credible interval: there's a 95% (posterior) chance that the typical weight at a height of 180 cm is between 76.98 and 79.24 kg.

Calculating posterior predictions

You just explored the posterior trend in weight Y among adults with height $X = 180 : m_{180} = a + b * 180 : 180$. The `weight_chains` data frame contains 100,000 posterior plausible values of m_{180} that you calculated from the corresponding values of a and b:

```
head(weight_chains, 2)
```

```
##           a        b        s iter    m_180
## 1 -104.7251 1.014141 9.369227    1 77.82020
## 2 -104.5230 1.016743 9.131354    2 78.49081
```

Forget the trend - what if you wanted to predict the weight of a specific 180 cm tall adult? You can! To do so, you must account for individual variability from the trend, modeled by

$Y_{180} \sim N(m_{180}, s^2)$

Using this model, you will simulate predictions of weight under each set of posterior plausible parameters in `weight_chains`.

Use `rnorm()` to simulate a single prediction of weight under the parameter settings in the first row of `weight_chains`.

```
# Simulate 1 prediction under the first parameter set
rnorm(n = 1, mean = weight_chains$m_180[1], sd = weight_chains$s[1])
```

```
## [1] 67.40882
```

Repeat the above using the parameter settings in the second row of `weight_chains`.

```
# Simulate 1 prediction under the second parameter set
rnorm(n = 1, mean = weight_chains$m_180[2], sd = weight_chains$s[2])
```

```
## [1] 65.6943
```

Simulate a single prediction of weight under each of the 100,000 parameter settings in `weight_chains`. Store these as a new variable `Y_180` in `weight_chains`.

```
# Simulate & store 1 prediction under each parameter set
weight_chains <- weight_chains  %>%
    mutate(Y_180 = rnorm(n = 100000, mean = weight_chains$m_180, sd = weight_chains$s))
```

Print the first 6 rows of parameter values & predictions in `weight_chains`.

```
# Print the first 6 parameter sets & predictions
head(weight_chains,6)
```

```
##           a        b        s iter    m_180     Y_180
## 1 -104.7251 1.014141 9.369227    1 77.82020  76.06890
## 2 -104.5230 1.016743 9.131354    2 78.49081  62.71583
## 3 -104.7267 1.013953 9.219877    3 77.78481  82.16713
## 4 -103.8473 1.006570 9.654584    4 77.33538 102.14154
## 5 -103.4073 1.011546 9.725802    5 78.67105  75.33764
## 6 -103.9034 1.012012 9.523006    6 78.25880  81.88009
```

You now have 100,000 predictions for the weight of a 180 cm tall adult that reflect the range of posterior plausible parameter settings.

Posterior predictive distribution

The `weight_chains` data frame (in your workspace) contains your 100,000 posterior predictions, `Y_180`, for the weight of a 180 cm tall adult:

```
head(weight_chains, 2)
```

```
##           a        b        s iter    m_180    Y_180
## 1 -104.7251 1.014141 9.369227    1 77.82020 76.06890
## 2 -104.5230 1.016743 9.131354    2 78.49081 62.71583
```

You will use these 100,000 predictions to approximate the *posterior predictive distribution* for the weight of a 180 cm tall adult. The `bdims` data are in your workspace.

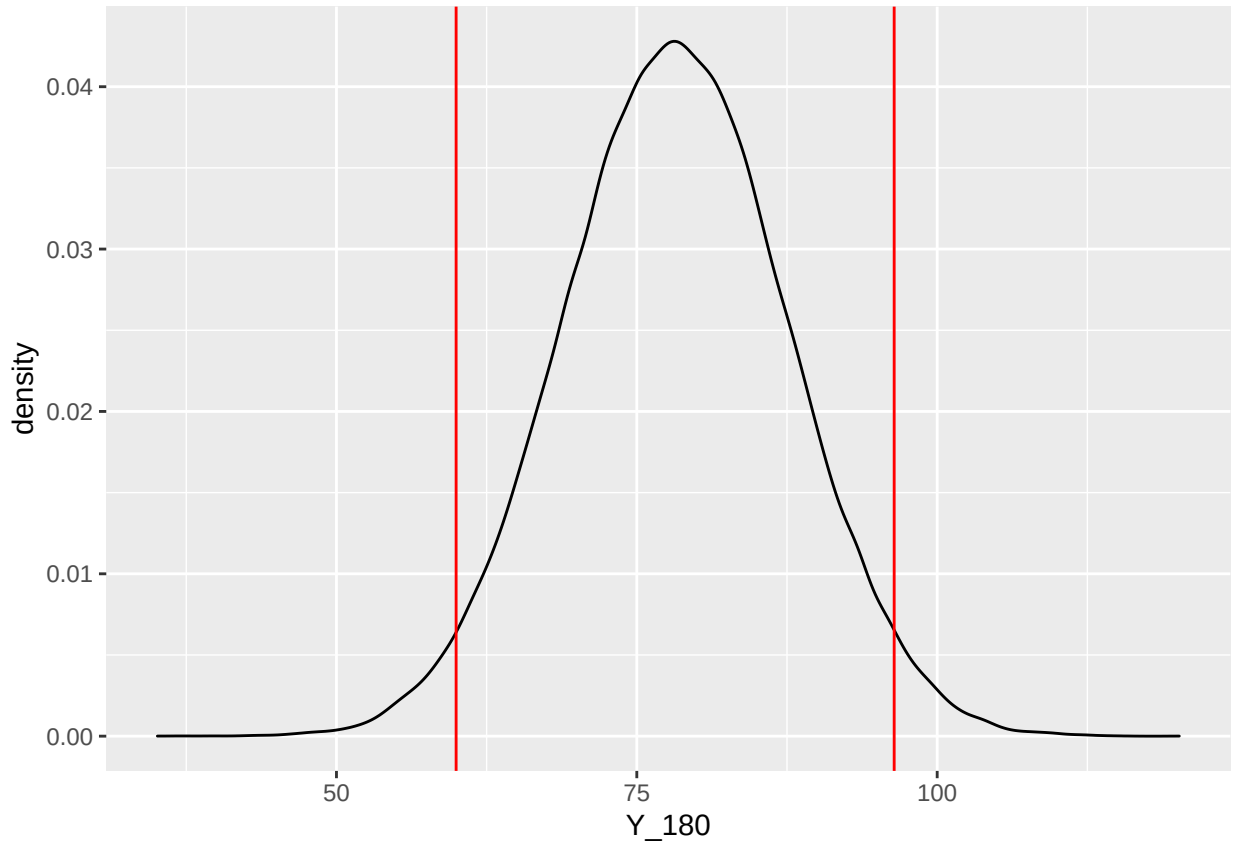
Use the 10,000 `Y_180` values to construct a 95% posterior credible interval for the weight of a 180 cm tall adult.

```
# Construct a posterior credible interval for the prediction
ci_180 <- quantile(weight_chains$Y_180, probs = c(0.025, 0.975))
ci_180
```

```
##     2.5%    97.5%
## 59.95932 96.41951
```

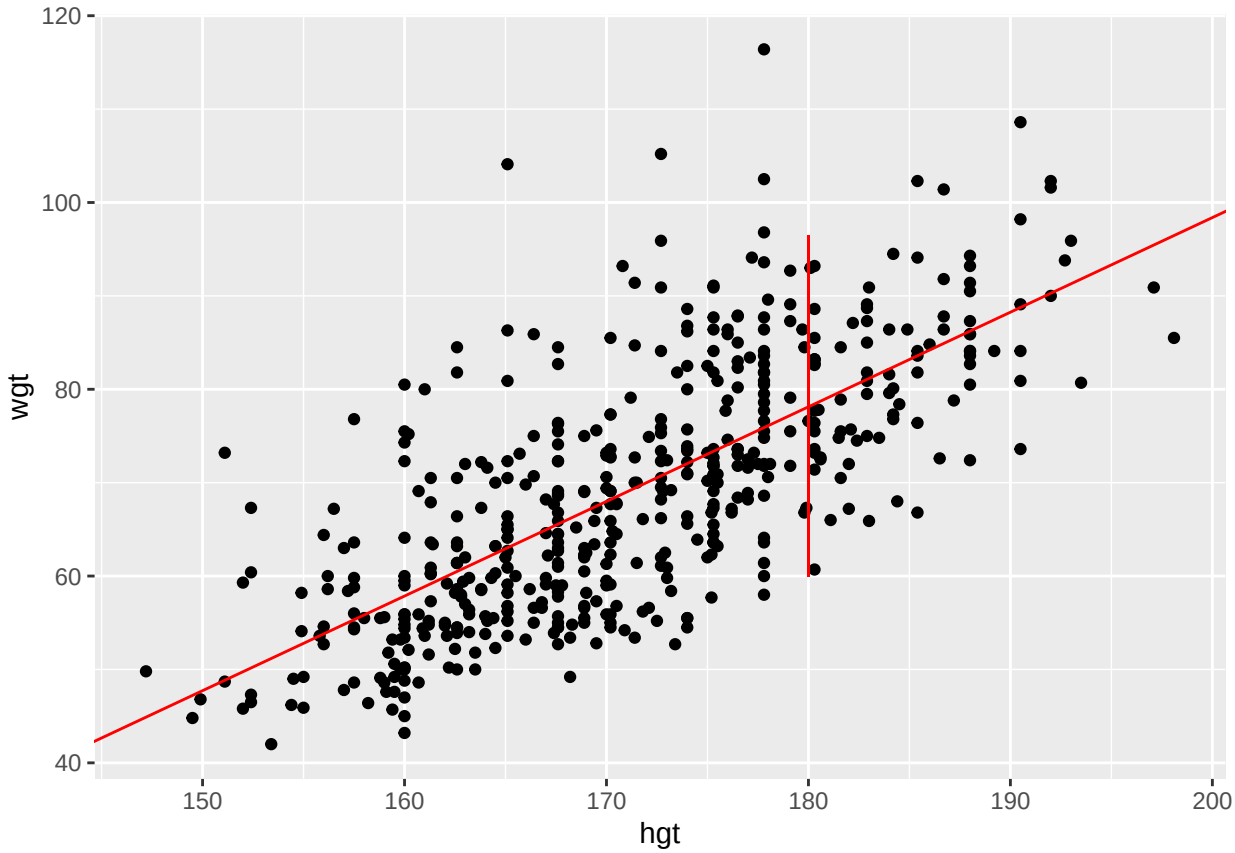
Construct a density plot of your 100,000 posterior plausible predictions.

```
# Construct a density plot of the posterior predictions
ggplot(weight_chains, aes(x = Y_180)) +
    geom_density() +
    geom_vline(xintercept = ci_180, color = "red")
```



Construct a scatterplot of the `wgt` vs `hgt` data in `bdims`. Use `geom_abline()` to superimpose the posterior regression trend. Use `geom_segment()` to superimpose a vertical line at a `hgt` of 180 that represents the lower & upper limits (`y` and `yend`) of `ci_180`.

```
# Visualize the credible interval on a scatterplot of the data
ggplot(bdims, aes(x = hgt, y = wgt)) +
    geom_point() +
    geom_abline(intercept = mean(weight_chains$a), slope = mean(weight_chains$b), color = "red") +
    geom_segment(x = 180, xend = 180, y = ci_180[1], yend = ci_180[2] , color = "red")
```



You've simulated your first posterior predictive distribution. Your 100,000 posterior plausible weights for a given 180 cm tall adult ranged from roughly 36 to 117 kg. Eliminating the most extreme 5% of these predictions, you observed that there's a 95% (posterior) chance that the weight is between 59.9 and 96.4 kg.

Multivariate & Generalized Linear Models

RailTrail sample data

The `RailTrail` data frame from the `mosaic` package is loaded in your workspace. `RailTrail` contains data collected by the Pioneer Valley Planning Commission on the usage of a local rail-trail. For each of 90 days, they recorded the rail-trail `volume` (number of users) and whether it was a `weekday` (`TRUE` if yes and `FALSE` otherwise). You will explore the trends in weekday vs weekend volume below.

```
head(RailTrail)
```

```
##   hightemp lowtemp avgtemp spring summer fall cloudcover precip volume weekday
## 1       83      50    66.5      0      1    0        7.6   0.00    501    TRUE
## 2       73      49    61.0      0      1    0        6.3   0.29    419    TRUE
## 3       74      52    63.0      1      0    0        7.5   0.32    397    TRUE
## 4       95      61    78.0      0      1    0        2.6   0.00    385   FALSE
## 5       44      52    48.0      1      0    0       10.0   0.14    200    TRUE
## 6       69      54    61.5      1      0    0        6.6   0.02    375    TRUE
##   dayType
## 1 weekday
```

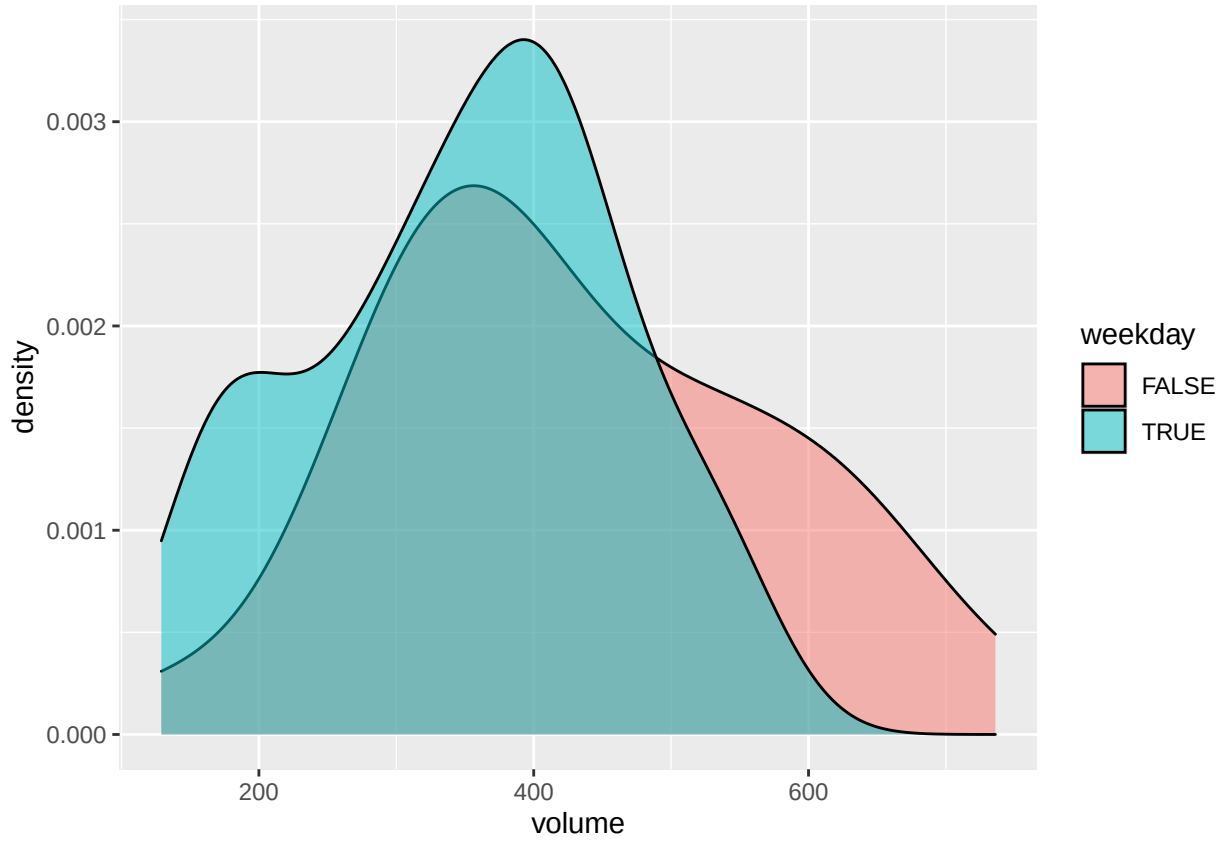
```
## 2 weekday
## 3 weekday
## 4 weekend
## 5 weekday
## 6 weekday
```

```
# Confirm that weekday is a factor variable
class(RailTrail$weekday)
```

```
## [1] "logical"
```

Construct density plots of weekday `volume` and weekend `volume` on the same frame.

```
# Construct a density plot of volume by weekday
ggplot(RailTrail, aes(x = volume, fill = weekday)) +
    geom_density(alpha = 0.5)
```



Notice that, as might be intuitive, rail-trail volume tends to be slightly higher on weekends (~430 users per day) than on weekdays (~350 users per day).

RJAGS simulation with categorical variables

Consider the Normal regression model of volume Y_i by weekday status X_i:

Likelihood: $Y_i \sim N(m_i, s^2)$ where $m_i = a + bX_i$.

Priors: $a \sim N(400, 100^2)$, $b \sim N(0, 200^2)$, $s \sim Unif(0, 200)$.

You explored the relationship between Y_i and X_i for the 90 days recorded in `RailTrail` (in your workspace). In light of these data and the priors above, you will update your posterior model of this relationship. This differs from previous analyses in that X_i is categorical. In `rjags` syntax, its coefficient b is defined by two elements, `b[1]` and `b[2]`, which correspond to the weekend and weekday levels, respectively. For reference, `b[1]` is set to 0. In contrast, `b[2]` is modeled by the prior for b.

DEFINE your Bayesian model.

Define the likelihood model of $Y[i]$ given $m[i]$ and s where $m[i] < -a + b[X[i]]$. Note the new notation $b[X[i]]$ here!

Specify the priors for a, b (via $b[1]$ and $b[2]$), and s.

Store the model string as $rail_model_1$.

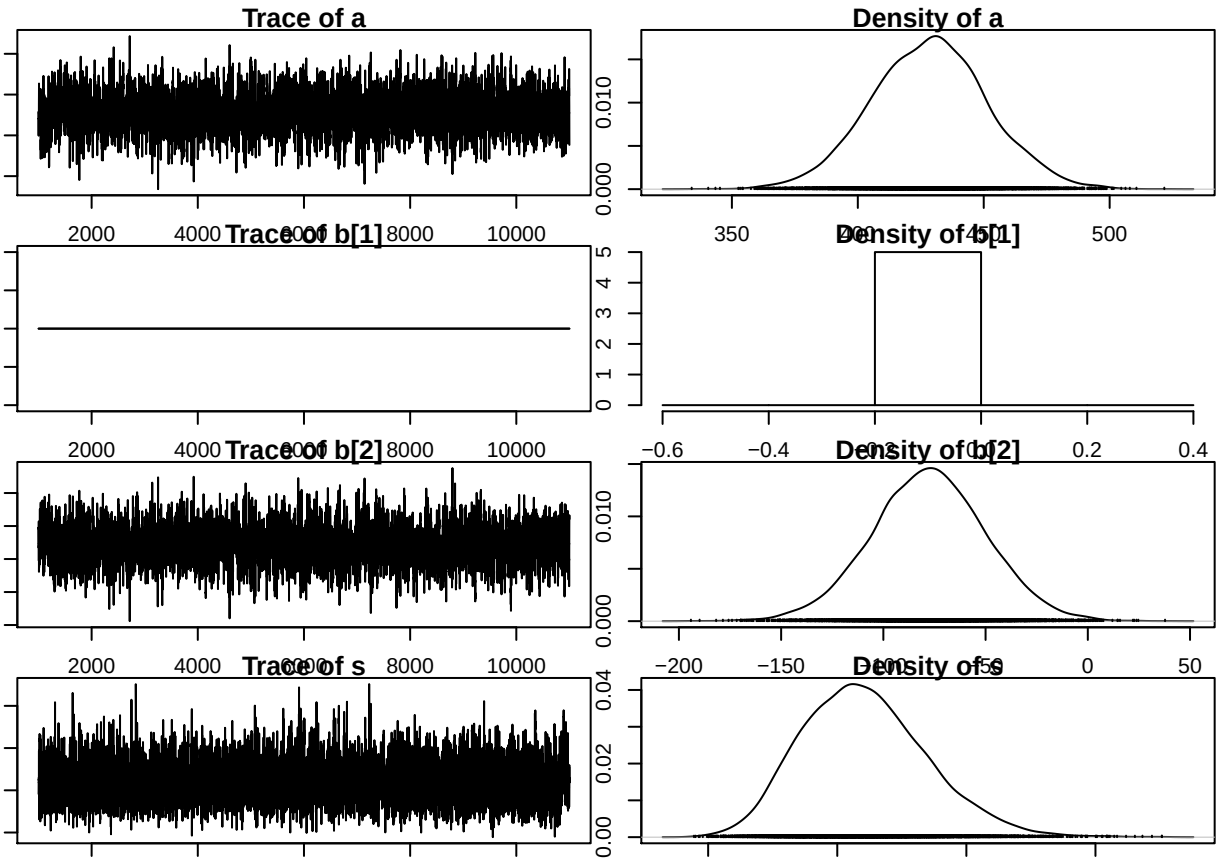
```
RailTrail$weekday <- factor(RailTrail$weekday)


class(RailTrail$weekday)
```

```
## [1] "factor"
```

```
# DEFINE the model
rail_model_1 <- "model{
    # Likelihood model for Y[i]
    for(i in 1:length(Y)){
      Y[i] ~ dnorm(m[i],s^(-2))
      m[i] <- a+b[X[i]]
    }

    # Prior models for a, b, s
    a ~ dnorm(400,100^(-2))
    b[1] <- 0
    b[2] ~ dnorm(0,200^(-2))
    s ~ dunif(0,200)
}"

# COMPILE the model
rail_jags_1 <- jags.model(
  textConnection(rail_model_1),
  data = list(Y = RailTrail$volume, X = RailTrail$weekday),
  inits = list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 10)
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 90
##    Unobserved stochastic nodes: 3
##    Total graph size: 194
##
## Initializing model
```

```
# SIMULATE the posterior
rail_sim_1 <- coda.samples(model = rail_jags_1, variable.names = c("a", "b", "s"), n.iter = 10000)

# Store the chains in a data frame
rail_chains_1 <- data.frame(rail_sim_1[[1]])

# PLOT the posterior
# windows() ## create window to plot your file
par(mar=c(1, 1, 1, 1))
plot(rail_sim_1)
```




```
# dev.off()
```

Interpreting categorical coefficients

In your Bayesian model $m_i = a + bX_i$ specified the dependence of typical trail volume on weekday status X_i (1 for weekdays and 0 for weekends). A `summary()` of your RJAGS model simulation provides posterior mean estimates of parameters a and b, the latter corresponding to `b.2.` here.

```
summary(rail_sim_1)
```

```
##
## 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
## a    428.47 23.052  0.23052         0.5321
## b[1]   0.00  0.000  0.00000         0.0000
## b[2] -77.78 27.900  0.27900         0.6422
## s    124.25  9.662  0.09662         0.1335
##
## 2. Quantiles for each variable:
##
##        2.5%   25%    50%    75%  97.5%
## a     383.1 412.9 428.78 443.62 474.20
## b[1]    0.0   0.0   0.00   0.00   0.00
## b[2] -133.2 -96.3 -77.65 -59.43 -23.07
## s     107.5 117.3 123.59 130.32 144.90
```

Typically, there are 428.47 trail users on a weekend day and 77.78 fewer users (~350.69) on a weekday.

Parameter a describes the typical weekend volume whereas b describes the contrast between weekday and weekend volume.

Inference for volume by weekday

The 10,000 iteration RJAGS simulation output, `rail_sim_1`, is in your workspace along with a data frame of the Markov chain output:

```
head(rail_chains_1, 2)
```

```
##          a b.1.      b.2.        s
## 1 420.6966    0 -54.30783 118.2328
## 2 399.5823    0 -52.02570 119.9499
```

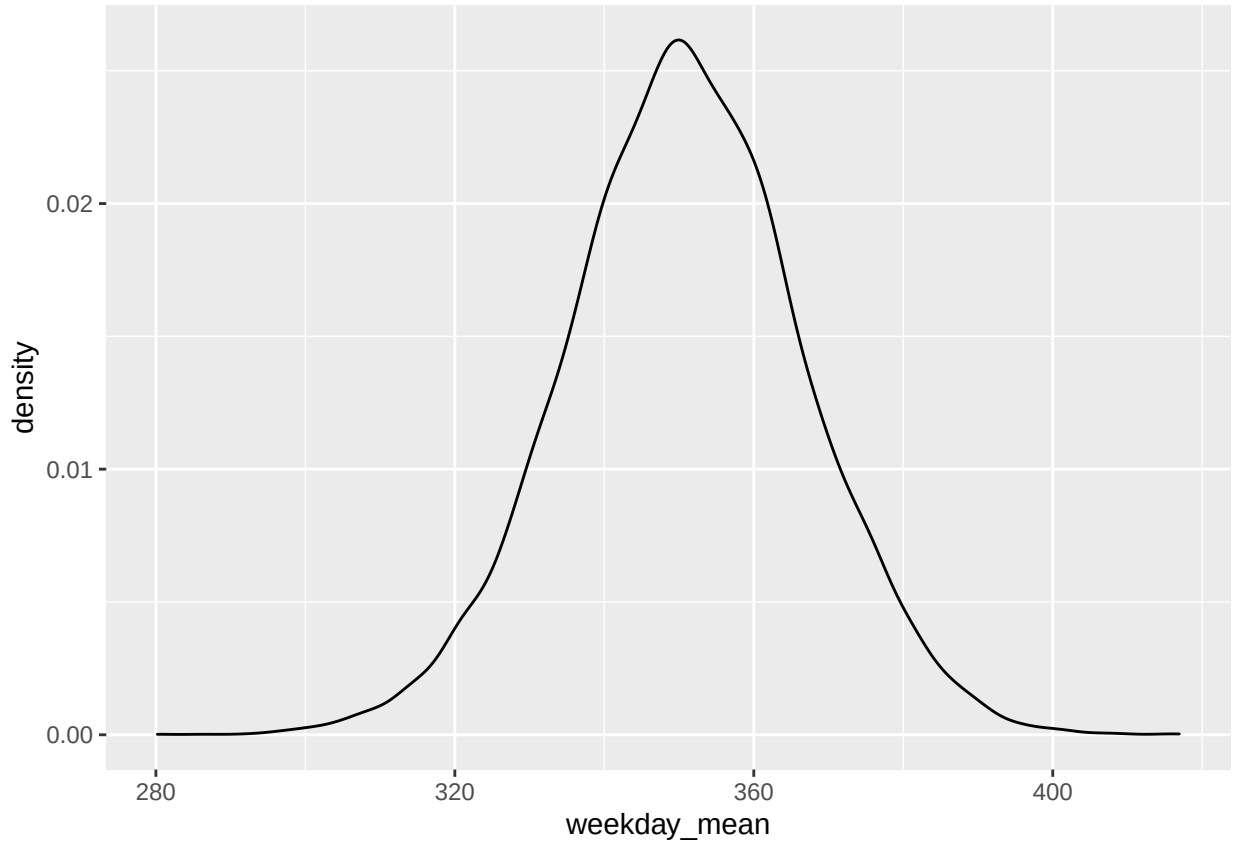
These chains provide 10,000 unique sets of values for `a`, the typical trail volume on weekend days, and `b.2.`, the *contrast* between typical weekday volume vs weekend volume. For example, the first set of parameters indicate that there are typically `420.6966` riders on weekend days and `54.30783` fewer riders on weekdays. Thus there are typically `420.6966 - 54.30783 = 366.3888` riders on weekdays. You will utilize these simulation data to make inferences about weekday trail volume.

Combine the `a` and `b.2.` chain values to construct a chain of 10,000 values for the typical weekday trail volume. Store this as `weekday_mean` in `rail_chains_1`.

```
# Construct a chain of values for the typical weekday volume
rail_chains_1 <- rail_chains_1 %>%
    mutate(weekday_mean = rail_chains_1$a+rail_chains_1$b.2.)
```

Use `ggplot()` to construct a density plot of the `weekday_mean` chain values.

```
# Construct a density plot of the weekday chain
ggplot(rail_chains_1, aes(x = weekday_mean)) +
    geom_density()
```



Construct a 95% credible interval for the typical weekday trail volume.

```
# 95% credible interval for typical weekday volume
quantile(rail_chains_1$weekday_mean,c(0.025,0.975))
```

```
##     2.5%    97.5%
## 318.9520 381.7573
```

You've shown that there's a 95% posterior chance that the typical weekday volume is between 319 and 382 trail users.

Multivariate Bayesian regression

Re-examining the RailTrail data

In your previous work, you observed that rail-trail `volume` tends to be lower on a weekday than a `weekend`. Some of the variability in `volume` might also be explained by outside temperature. For example, we might expect trail volume to increase on warm, pleasant days.

The `RailTrail` data set in your workspace includes `hightemp`, the observed high temperature (F) for each of the 90 days in the study period. You will use these data to explore the associations between trail `volume`, `weekday` status, and `hightemp`.

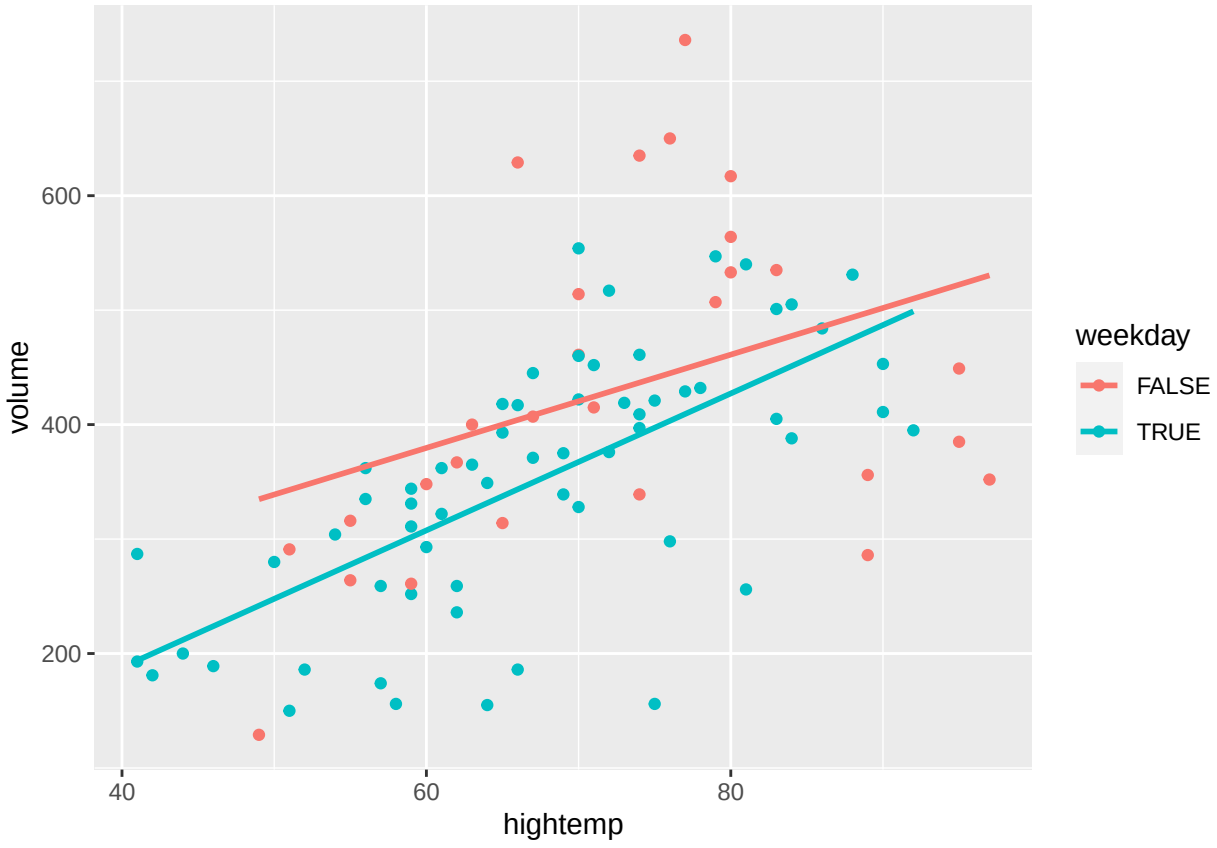
Construct a scatterplot of volume by `hightemp`:

Use `color` to distinguish between weekdays & weekends.

Use `geom_smooth()` to highlight the linear relationship between the observed `volume` & `hightemp` values.

```
# Construct a plot of volume by hightemp & weekday
ggplot(RailTrail, aes(y = volume, x = hightemp, color = weekday)) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Notice that for the 90 days in the study period, volume tends to increase with temperature. Further, volume tends to be higher on weekends than on weekdays of the same temperature.

RJAGS simulation for multivariate regression

Consider the following Bayesian model of volume Y_i by weekday status X_i and temperature Z_i:

Likelihood: $Y_i \sim N(m_i, s^2)$ where $m_i = a + bX_i + cZ_i$ (b is 1 for weekdays and b is 0 for weekends)

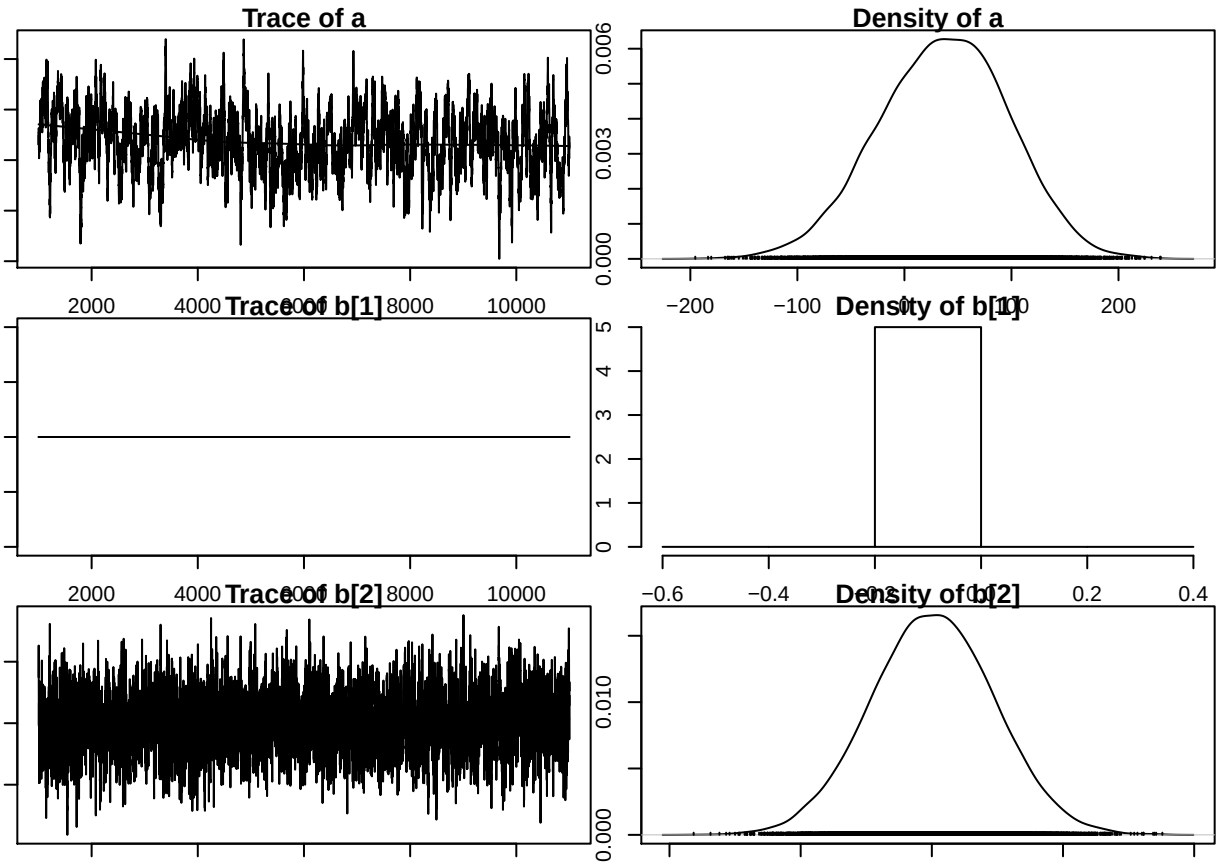
Priors: $a \sim N(0, 200^2)$, $b \sim N(0, 200^2)$, $c \sim N(0, 20^2)$, $s \sim Unif(0, 200)$

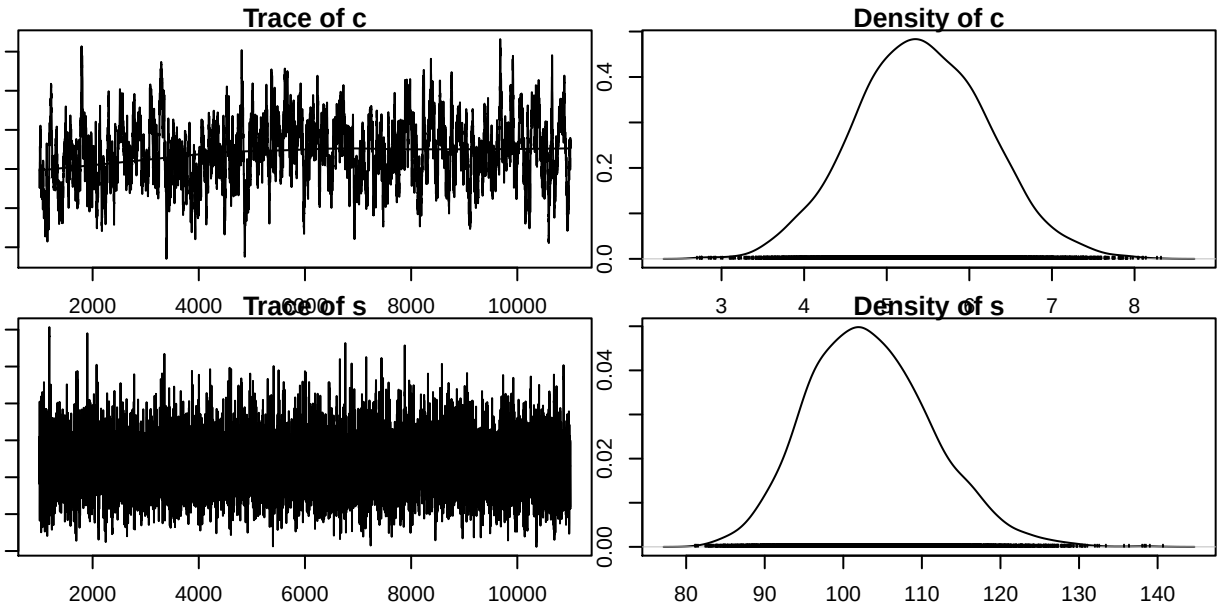
Your previous exploration of the relationship between `volume`, `weekday`, and `hightemp` in the `RailTrail` data provided some insight into this relationship. You will combine this with insight from the priors to develop a posterior model of this relationship using RJAGS. The `RailTrail` data are in your work space.

```
# DEFINE the model
rail_model_2 <- "model{
    # Likelihood model for Y[i]
    for (i in 1:length(Y)) {
        Y[i] ~ dnorm(m[i],s^(-2))
        m[i] <- a+b[X[i]]+c*Z[i]
    }
    # Prior model for a, b, c, s
    a ~ dnorm(0,200^(-2))
    b[1] <- 0
    b[2] ~ dnorm(0,200^(-2))
    c ~ dnorm(0,20^(-2))
    s ~ dunif(0,200)
}"


# COMPILE the model
rail_jags_2 <- jags.model(textConnection(rail_model_2),
    data = list(Y = RailTrail$volume, X = RailTrail$weekday, Z = RailTrail$hightemp),
    inits = list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 10))
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 90
##    Unobserved stochastic nodes: 4
##    Total graph size: 385
##
## Initializing model
```

```
# SIMULATE the posterior
rail_sim_2 <- coda.samples(model = rail_jags_2, variable.names = c("a","b","c","s"), n.iter = 10000)

# Store the chains in a data frame
rail_chains_2 <- data.frame(rail_sim_2[[1]])

# PLOT the posterior
par(mar=c(1, 1, 1, 1))
plot(rail_sim_2)
```


Trace of a
Density of a
Trace of b[1]
Density of b[1]
Trace of b[2]
Density of b[2]



Interpreting multivariate regression parameters

Your Bayesian model explored the dependence of typical trail volume on weekday status X_i and temperature Z_i: $m_i = a + bX_i + cZ_i$. A `summary()` of your RJAGS model simulation provides posterior mean estimates of parameters a, b, and c:

```
summary(rail_sim_2)
```

```
##
## 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
## a      36.592 60.6238 0.606238        4.19442
## b[1]    0.000  0.0000 0.000000        0.00000
## b[2] -49.610 23.4930 0.234930        0.55520
## c      5.417  0.8029 0.008029        0.05849
## s    103.434  7.9418 0.079418        0.11032
##
```

```
## 2. Quantiles for each variable:
##
##         2.5%     25%     50%     75%   97.5%
## a    -83.443  -4.631  38.160  78.413 150.306
## b[1]   0.000   0.000   0.000   0.000   0.000
## b[2] -96.181 -65.468 -49.547 -33.633  -3.644
## c      3.865   4.865   5.402   5.965   7.007
## s     89.466  97.766 102.875 108.503 120.205
```

For example, the posterior mean of `c` indicates that for both weekends and weekdays, typical rail volume increases by ~5.4 users for every 1 degree increase in temperature. The posterior mean of `b[2]` indicates volume is ~50 less on weekdays than on weekends of the same temperature.

Posterior inference for multivariate regression

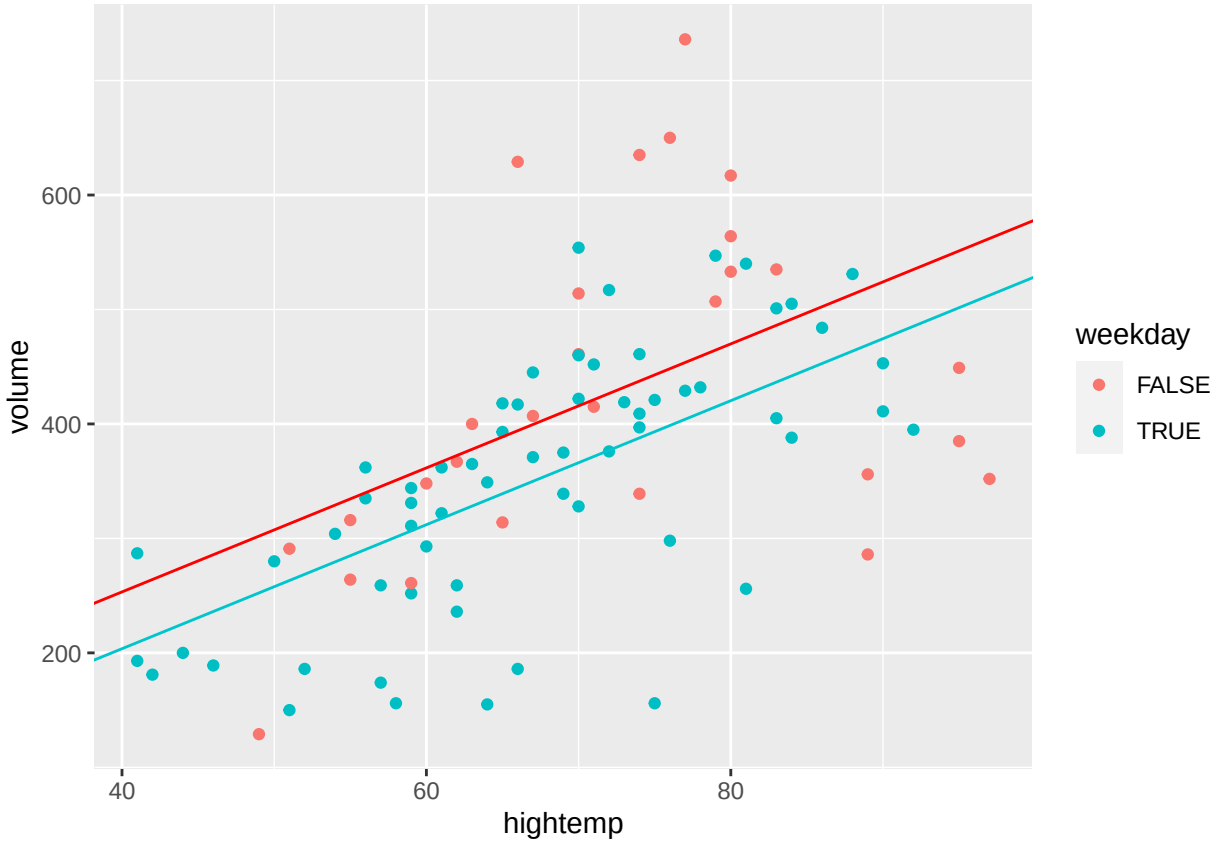
The 10,000 iteration RJAGS simulation output, `rail_sim_2`, is in your workspace along with a data frame of the Markov chain output:

```
head(rail_chains_2, 2)
```

```
##          a b.1.      b.2.        c         s
## 1 49.76954    0 -12.62112 4.999202 111.02247
## 2 30.22211    0  -3.16221 4.853491  98.11892
```

You will use these 10,000 unique sets of parameter values to summarize the posterior mean trend in the relationships between trail `volume`, `weekday` status, and `hightemp`.

```
# Plot the posterior mean regression models
ggplot(RailTrail, aes(x = hightemp, y = volume, color = weekday)) +
    geom_point() +
    geom_abline(intercept = mean(rail_chains_2$a), slope = mean(rail_chains_2$c), color = "red") +
    geom_abline(intercept = mean(rail_chains_2$a) + mean(rail_chains_2$b.2.), slope = mean(rail_chains_
```



Red line that represents the posterior mean trend of the linear relationship between `volume` and `hightemp` for *weekends*: `m = a + c Z`

Blue line represents the posterior mean trend of the linear relationship between `volume` and `hightemp` for *weekdays*: `m = (a + b.2.) + c Z`

Your posterior analysis suggests that there's a positive association between volume and temperature. Further, the typical weekday volume is less than that on weekends of the same temperature.

Bayesian Poisson regression

The Poisson Model

Y = volume (number of users) on a given day.

$Y \sim Pois(\lambda)$

Y is the number of independent events that occur in a fixed interval (0,1,2,…).

Rate parameter λ represent the typical number of events per time interval ($\lambda > 0$).

Poisson Regression

$Y_i \sim Pois(\lambda_i)$ where $\lambda_i > 0$, $\lambda_i = a + bX_i + cZ_i$.

However linking λ_i directly to the linear model assumes λ_i can be negative and it's not possible in real world. Alternatively, we can use a *log link function* to link λ_i to the linear model. In turn: $\lambda_i = e^{a+bX_i+cZ_i}$. It guaranteed to be positive, thus preserves the Poisson properties.

RJAGS simulation for Poisson regression

In the previous video we engineered a Poisson regression model of volume Y_i by weekday status X_i and temperature Z_i:

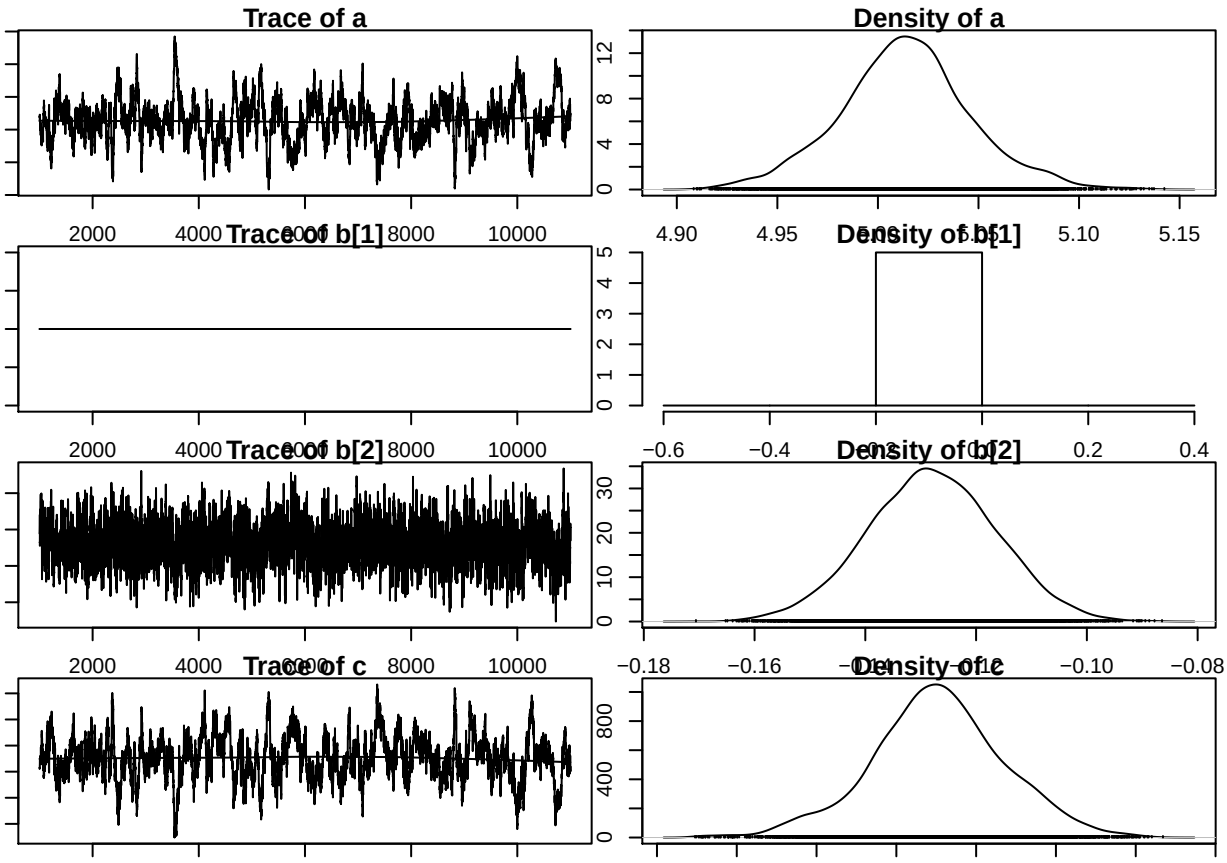
Likelihood: $Y_i \sim Pois(\lambda_i)$ where $log(\lambda_i) = a + bX_i + cZ_i$

Priors: $a \sim N(0, 200^2)$, $b \sim N(0, 2^2)$, and $c \sim N(0, 2^2)$

```
# DEFINE the model
poisson_model <- "model{
    # Likelihood model for Y[i]
    for (i in 1:length(Y)) {
        Y[i] ~ dpois(l[i])
        log(l[i]) <- a+b[X[i]]+c*Z[i]
        }
    # Priors model for a, b and c
    a ~ dnorm(0,200^(-2))
    b[1] <- 0
    b[2] ~ dnorm(0,2^(-2))
    c ~ dnorm(0,2^(-2))
}"

# COMPILE the model
poisson_jags <- jags.model(textConnection(poisson_model),
                data=list(Y = RailTrail$volume, X = RailTrail$weekday, Z = RailTrail$hightemp),
                inits=list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 10))
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 90
##    Unobserved stochastic nodes: 3
##    Total graph size: 441
##
## Initializing model
```

```
# SIMULATE the posterior
poisson_sim <- coda.samples(model=poisson_jags,variable.names=c("a","b","c"), n.iter=10000)

# Store the chains in a data frame
poisson_chains <- data.frame(poisson_sim[[1]])

# PLOT the posterior
par(mar=c(1, 1, 1, 1))
plot(poisson_sim)
```



Plotting the Poisson regression model

Recall the likelihood structure for your Bayesian Poisson regression model of volume Y_i by weekday status X_i and temperature Z_i: $Y_i \sim Pois(\lambda_i)$ where $log(\lambda_i) = a + bX_i + cZ_i$; thus $\lambda_i = e^{a+bX_i+cZ_i}$

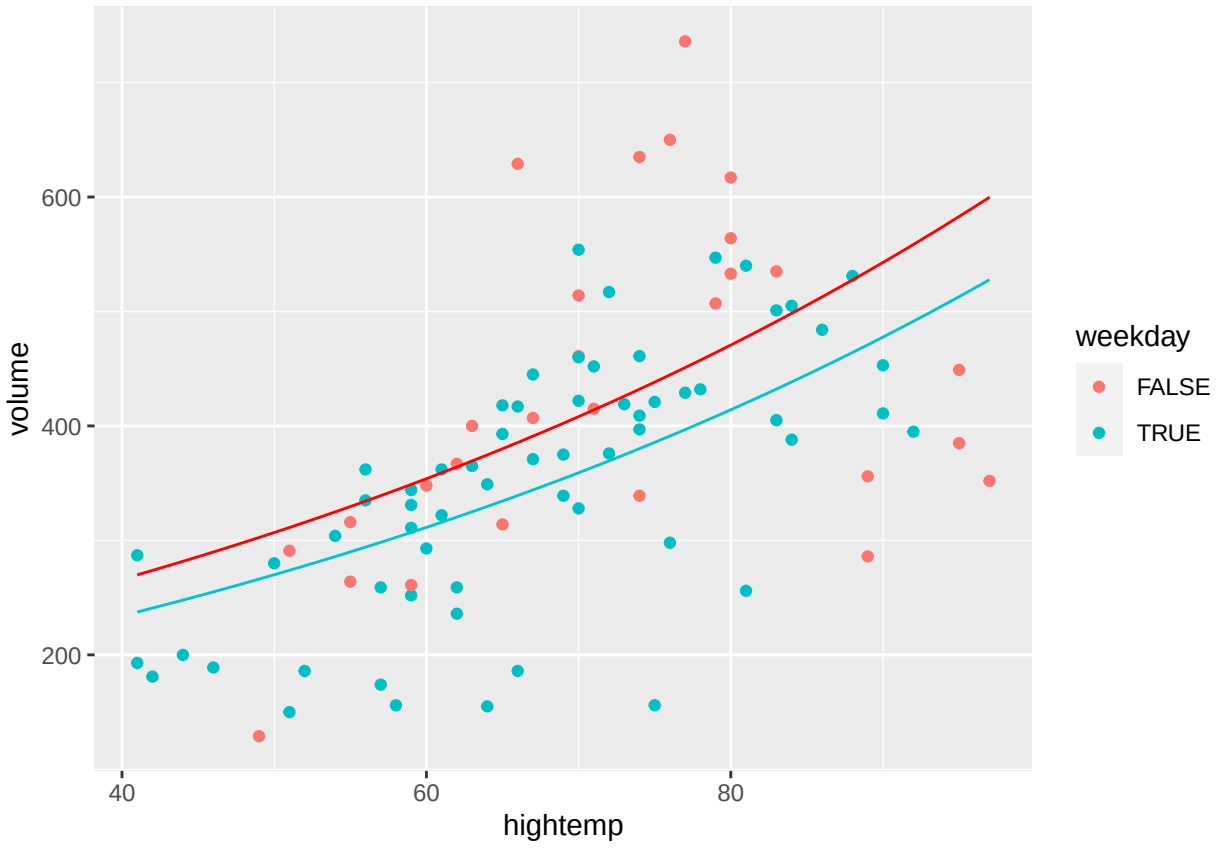
Your 10,000 iteration RJAGS simulation of the model posterior, `poisson_sim`, is in your workspace along with a data frame of the Markov chain output:

```
head(poisson_chains, 2)
```

```
##          a b.1.       b.2.          c
## 1 5.019807    0 -0.1222143 0.01405269
## 2 5.018642    0 -0.1217608 0.01407691
```

You will use these results to plot the posterior Poisson regression trends. These nonlinear trends can be added to a `ggplot()` using `stat_function()`. For example, specifying `fun = function(x){x^2}`would return a quadratic trend line.

```
# Plot the posterior mean regression models
ggplot(RailTrail, aes(x = hightemp, y = volume, color = weekday)) +
    geom_point() +
    stat_function(fun = function(x){exp(mean(poisson_chains$a) + mean(poisson_chains$c) * x)}, color =
    stat_function(fun = function(x){exp(mean(poisson_chains$a) + mean(poisson_chains$b.2.) + mean(poiss
```



Inference for the Poisson rate parameter

Again, recall the likelihood structure for your Bayesian Poisson regression model of volume Y_i by weekday status X_i and temperature Z_i: $Y_i \sim Pois(\lambda_i)$ where $\lambda_i = exp(a + bX_i + cZ_i)$.

Your 10,000 iteration RJAGS simulation of the model posterior, `poisson_sim`, is in your workspace along with a data frame of the Markov chain output:

```
head(poisson_chains, 2)
```

```
##          a b.1.       b.2.          c
## 1 5.019807    0 -0.1222143 0.01405269
## 2 5.018642    0 -0.1217608 0.01407691
```

Using these 10,000 unique sets of posterior plausible values for parameters a, b, and c you will make inferences about the typical trail volume on *80 degree days*.

From each set of `poisson_chains` parameter values, calculate the typical trail volumes on an 80 degree weekend day. Store these trends as a new variable, `l_weekend`, in `poisson_chains`. Similarly, calculate the typical trail volumes on an 80 degree weekday. Store these as a new variable, `l_weekday`.

```
# Calculate the typical volume on 80 degree weekends & 80 degree weekdays
poisson_chains <- poisson_chains %>%
    mutate(l_weekend = exp(a + c * 80)) %>%
    mutate(l_weekday = exp(a + b.2. + c * 80))
```

Calculate 95% posterior credible intervals for the typical volume on an 80 degree weekend day and the typical volume on an 80 degree weekday.

```
# Construct a 95% CI for typical volume on 80 degree weekend
quantile(poisson_chains$l_weekend,c(0.025,0.975))
```

```
##     2.5%    97.5%
## 462.1431 479.3101
```

```
# Construct a 95% CI for typical volume on 80 degree weekday
quantile(poisson_chains$l_weekday,c(0.025,0.975))
```

```
##     2.5%    97.5%
## 407.4644 420.7102
```

Poisson posterior prediction

Your `l_weekday` variable reflects the trend in volume on 80 degree weekdays:

```
head(poisson_chains, 2)
```

```
##          a b.1.       b.2.          c l_weekend l_weekday
## 1 5.019807    0 -0.1222143 0.01405269  465.9240  412.3235
## 2 5.018642    0 -0.1217608 0.01407691  466.2839  412.8292
```

Now that you understand the trend, let's make some predictions! Specifically, let's predict trail volumes on the next 80 degree weekday. To do so, you must take into account individual variability from the trend, modeled by the likelihood $Y_i \sim Pois(\lambda_i)$.

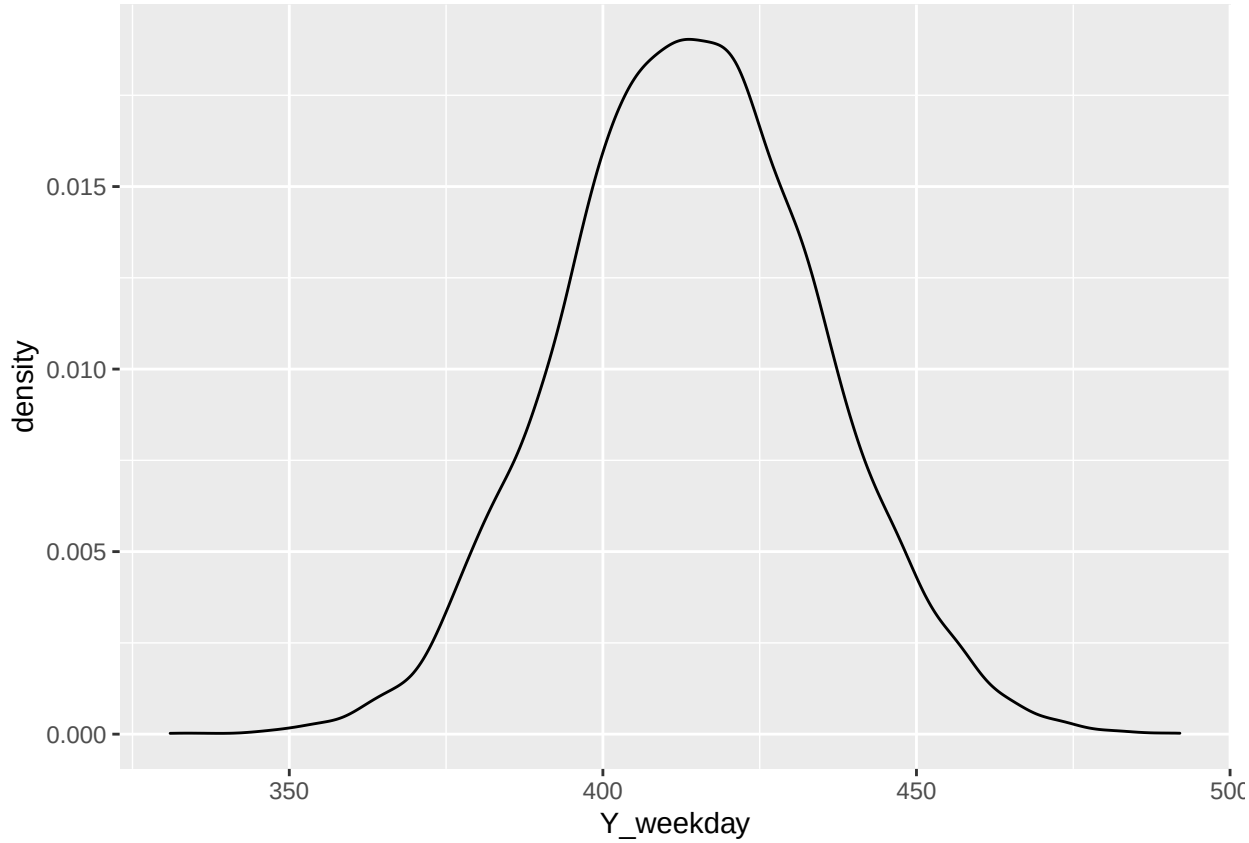
Using `rpois(n, lambda)` for sample size `n` and rate parameter `lambda`, you will simulate Poisson predictions of volume under each value of the posterior plausible trend in `poisson_chains`.

From each of the 10,000 `l_weekday` values in `poisson_chains`, use `rpois()` to predict volume on an 80 degree weekday. Store these as `Y_weekday`in `poisson_chains`.

```
# Simulate weekday predictions under each parameter set
poisson_chains <- poisson_chains %>%
    mutate(Y_weekday = rpois(n = 10000, lambda = l_weekday))
```

Use `ggplot()` to construct a density plot of your `Y_weekday` predictions.

```
# Construct a density plot of the posterior weekday predictions
ggplot(poisson_chains, aes(x = Y_weekday)) +
    geom_density()
```



Approximate the posterior probability that the volume on an 80 degree weekday is less than 400 users.

```
# Posterior probability that weekday volume is less 400
mean(poisson_chains$Y_weekday<400)
```

```
## [1] 0.2373
```

Recall that one of our motivations in applying the Poisson model was to accommodate the count nature of the volume data. This trickled down to your volume predictions `Y_weekday` - notice that these predictions, like the raw volume data, are discrete counts.