Week 3 assignment: Bayesian inference

Example solutions

January 17, 2015

Problem 1

Maure et al. (2015) conducted experiments to see how ladybird beetle diet affected interactions with parasitoid wasps. Individual beetles were randomly assigned to one of five diet treatments:

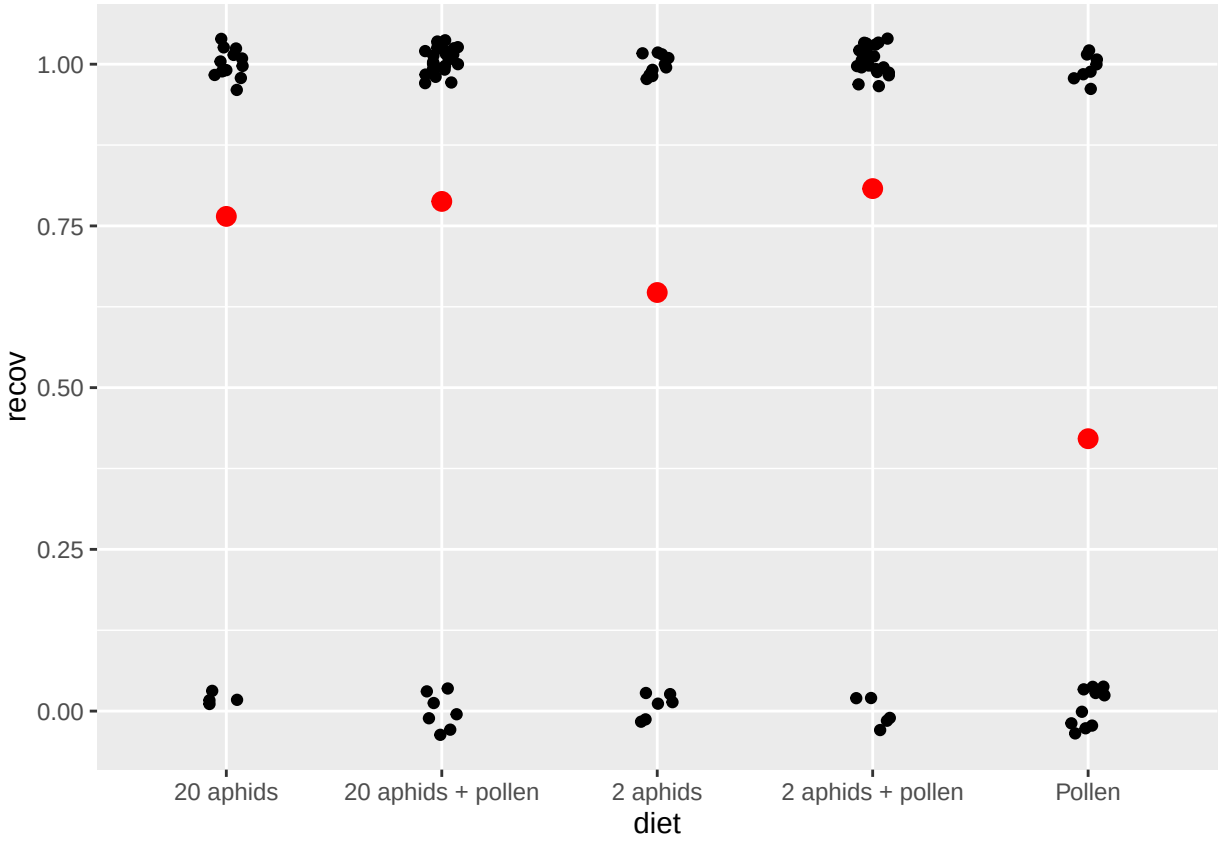
1. 20 aphids per day
2. 2 aphids per day
3. 20 aphids per day + pollen
4. 2 aphids per day + pollen
5. pollen only

Each beetle was exposed to a female parisitoid wasp for 3 hours, which deposited eggs in the beetle, and the beetles were then fed each day and the outcome of the interaction monitored. The authors wanted to know whether diet affected the probability of recovering from parasitoid attack. Your task is to build a Bayesian model with Stan for beetle survival as a function of experimental treatment, estimating the probability of survival for each treatment. To keep things simple, consider the treatments to be categorical and unordered. The data are archived in Dryad here.

Important: do not manipulate the original raw data file! This is is error prone, leaves no written record, encourages spreadsheet farming, and is not reproducible. And, in this case, the data are already well organized. Read in the .xls file using the `readxl` package, then use R to clean the data as necessary.

```
## Loading the raw data
library(readxl) # use the read_excel() function to load the data
library(dplyr)
library(ggplot2)
d <- read_excel('data/diet_data.xlsx')

## Cleaning the data
# remove NA columns
d <- d[, !is.na(names(d))]

# replace spaces in column names with underscores and convert to lowercase
names(d) <- gsub(' ', '_', names(d)) %>%
  tolower()

# create binary recovery variable
d <- d %>%
  mutate(recov = ifelse(ladybird_recovery == 'YES', 1, 0))

# calculate the proportion recovered for each diet
p_sample <- d %>%
  group_by(diet) %>%
  summarize(p_recov = mean(recov))

## Visualizing the data
```

```r
ggplot(d, aes(x=diet, y=recov)) +
  geom_jitter(position = position_jitter(width=.2, height=.1)) +
  geom_point(aes(y = p_recov), data = p_sample, col='red', size=3)
```



Write out your model using LaTeX:

$y_i \sim Bern(p_j[i])$

$p_j \sim Beta(1, 1)$

Paste your Stan model statement in the code block below, and ensure that your written model matches the notation in your Stan file:

```
data {
  // define the types and names of the data
  int n; // n is an integer
  int<lower=0, upper=1> y[n]; // y is an integer vector with n elements
  int n_trt;
  int<lower=1, upper=n_trt> treatment[n];
}
parameters {
  vector<lower=0, upper=1>[n_trt] p; // p is a real number between 0 and 1
}
model {
  // define priors
  p ~ beta(1, 1);

  // likelihood
```

```
  for (i in 1:n)
    y[i] ~ bernoulli(p[treatment[i]]);
}
```

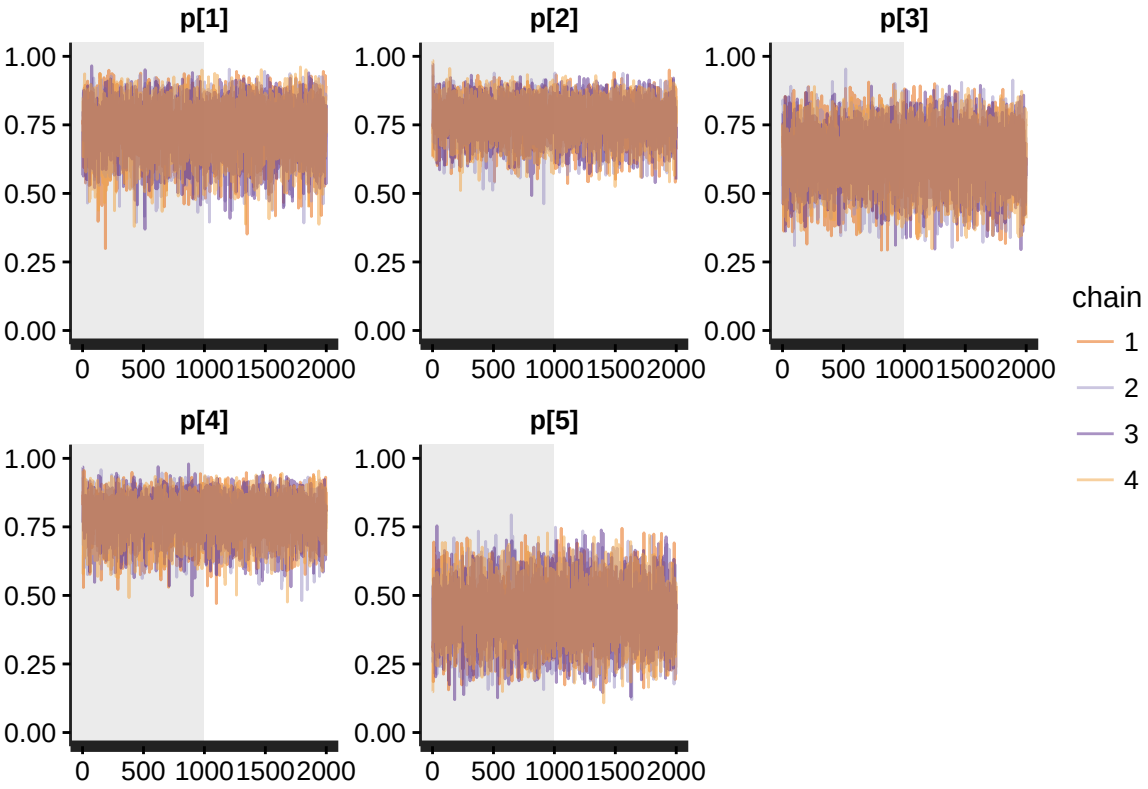
Now, use `rstan` to fit your model. Evaluate convergence by inspecting the $\hat{R}$ statistic and the traceplots.

```
library(rstan)

# bundle data into a list
stan_d <- list(n = nrow(d),
               y = d$recov,
               n_trt = length(unique(d$diet)),
               treatment = as.numeric(factor(d$diet)))

# fit the model
m_fit <- stan('bern_mod.stan', data = stan_d)
```

```
# print summary
m_fit
```

```
## Inference for Stan model: bern_mod.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##        mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## p[1]   0.74    0.00 0.10   0.52   0.67   0.74   0.81   0.90  2856    1
## p[2]   0.77    0.00 0.07   0.62   0.72   0.77   0.82   0.89  3351    1
## p[3]   0.63    0.00 0.11   0.41   0.56   0.64   0.71   0.83  3323    1
## p[4]   0.79    0.00 0.08   0.62   0.74   0.79   0.84   0.91  2851    1
## p[5]   0.43    0.00 0.11   0.23   0.35   0.42   0.50   0.64  2749    1
## lp__ -73.74    0.04 1.57 -77.54 -74.59 -73.44 -72.57 -71.56  1563    1
##
## Samples were drawn using NUTS(diag_e) at Fri Feb  5 19:10:10 2016.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
# plot the chains
traceplot(m_fit, inc_warmup=TRUE, alpha=.5) + ylim(0, 1)
```



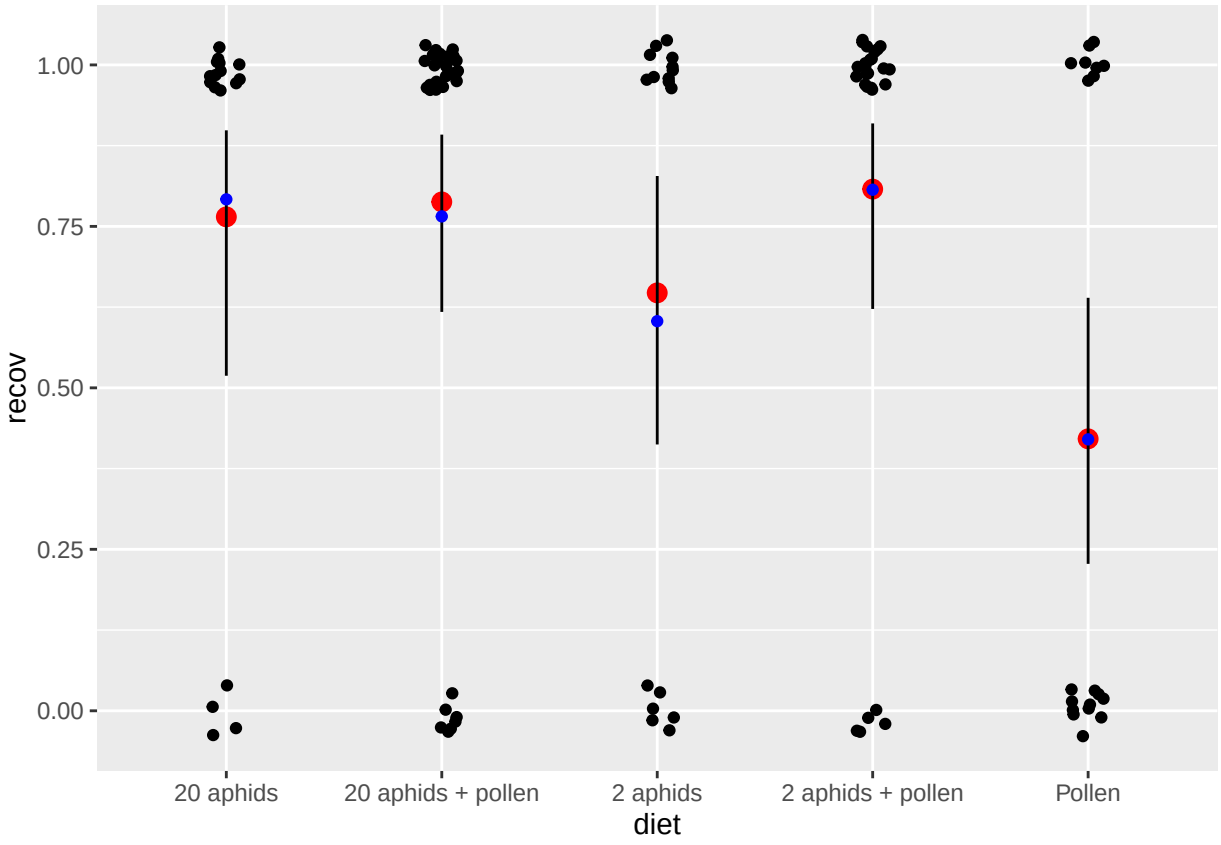
Calculate posterior credible intervals, medians, means, and modes for the survival probabilities for each treatment. Hint: posterior draws can be extracted with the `rstan::extract` function, which returns a list of arrays.

```
library(reshape2)
library(modeest)

# extract the posterior draws
post <- rstan::extract(m_fit)

# make a data frame that contains all of the posterior draws
post_df <- post$p %>%
  melt(varnames = c('iter', 'trt'))

# match numeric treatment indicators to diet names
post_df$diet <- levels(factor(d$diet))[post_df$trt]

# summarize each treatment's posterior in terms of quantiles and central tendency
post_sum <- post_df %>%
  group_by(diet) %>%
  summarize(lo = quantile(value, .025),
            hi = quantile(value, .975),
            mode = mlv(value, method='mfv')$M,
            median = median(value),
            mean = mean(value))

# plot the data frame
post_sum
```

```
## Source: local data frame [5 x 6]
##
##                  diet        lo        hi      mode    median      mean
##                 (chr)     (dbl)     (dbl)     (dbl)     (dbl)     (dbl)
## 1           20 aphids 0.5187319 0.8987272 0.7918305 0.7423156 0.7354246
## 2 20 aphids + pollen 0.6175643 0.8920791 0.7654963 0.7730917 0.7686639
## 3            2 aphids 0.4123219 0.8280028 0.6032315 0.6367875 0.6321267
## 4  2 aphids + pollen 0.6221582 0.9094115 0.8067297 0.7923127 0.7861481
## 5              Pollen 0.2274927 0.6394435 0.4202510 0.4249765 0.4287385
```

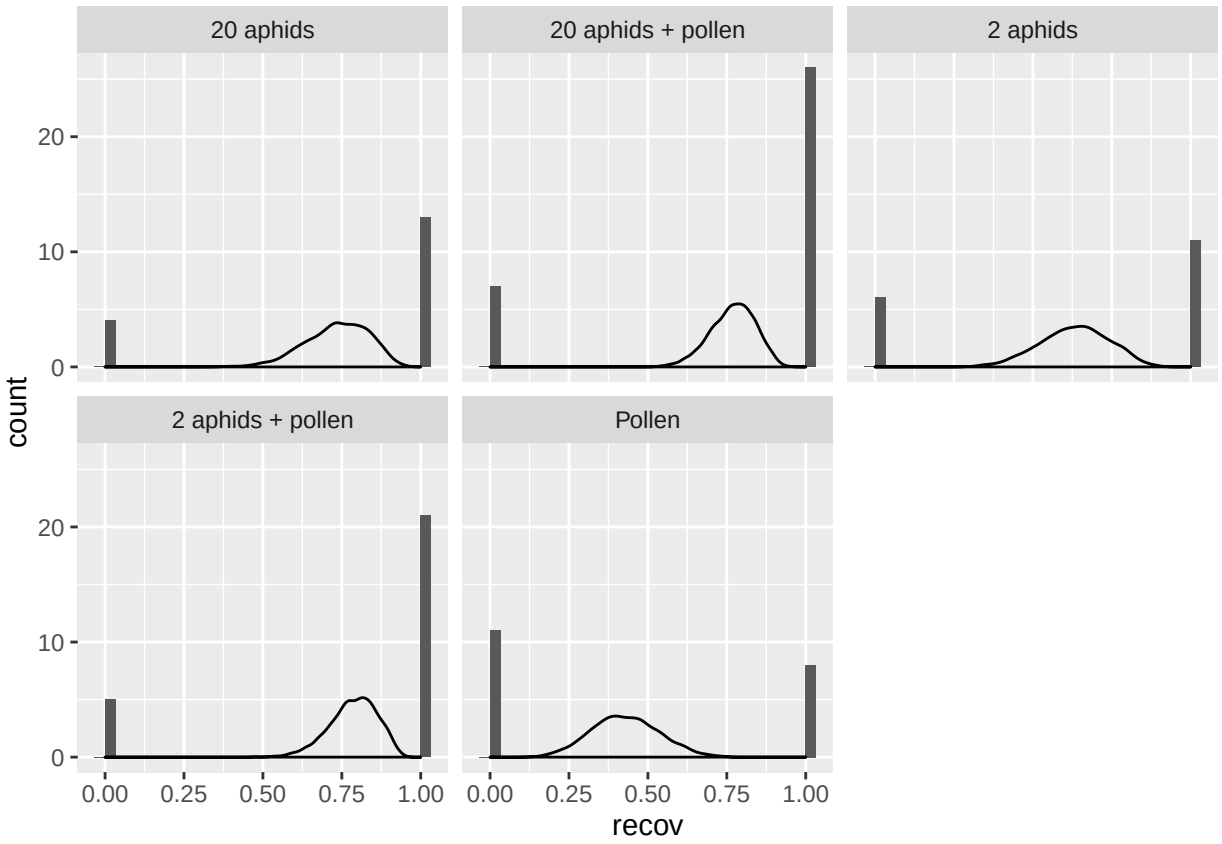
Generate a plot that shows all of the raw data along with the posterior probability distributions of recovery for each treatment:

```
# plot the raw data with interval estimates
ggplot(d, aes(x=diet, y=recov)) +
  geom_jitter(position = position_jitter(width=.2, height=.1)) +
  geom_point(aes(y = p_recov), data = p_sample, col='red', size=3) +
  geom_segment(aes(xend = diet, y = lo, yend = hi), data = post_sum) +
  geom_point(aes(y=mode), data = post_sum, color='blue')
```




```
ggplot(d, aes(x=recov)) +
  geom_histogram() +
  geom_density(aes(x=value), data = post_df) +
  facet_wrap(~ diet)
```

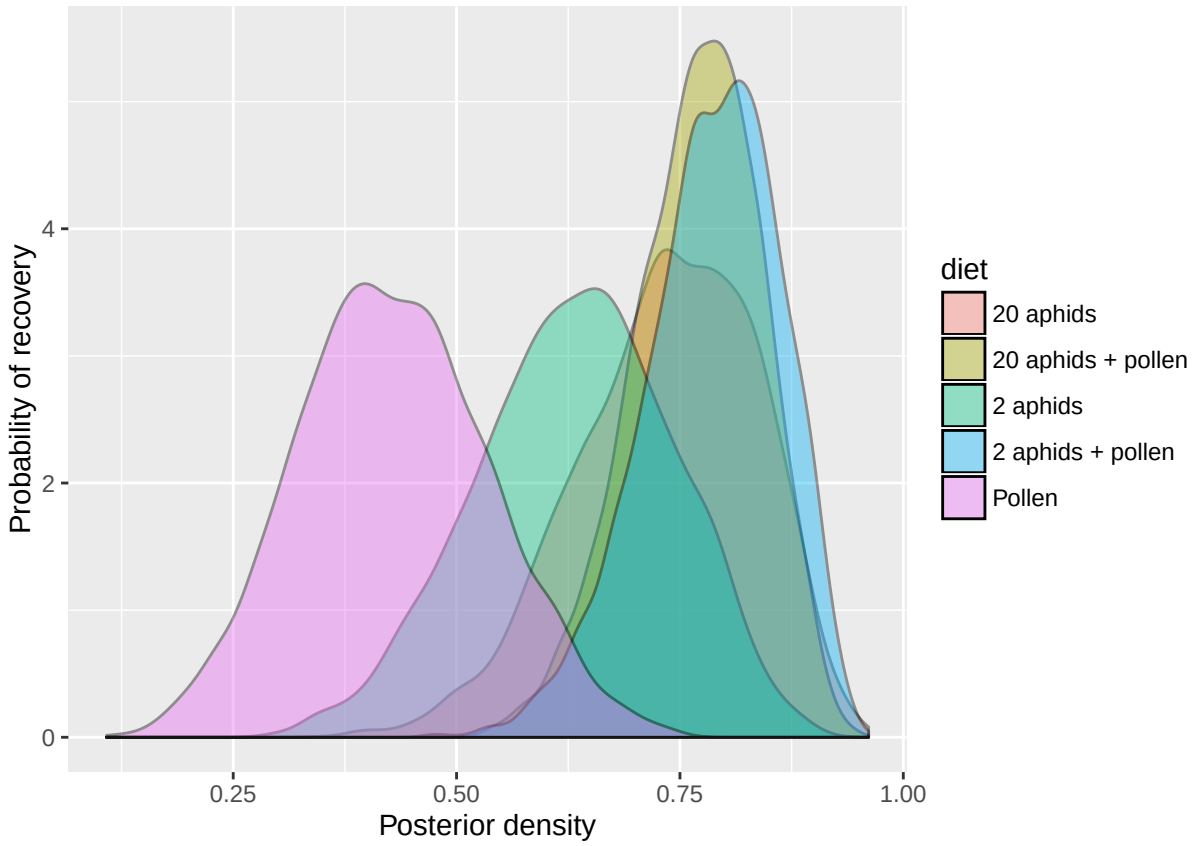
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The authors reported statistically significant differences in ladybird beetle recovery between the diet treatments. What is your conclusion for the effect of diet on ladybird beetle recovery?

It doesn't seem like the recovery probabilities were very different between the five treatments. Beetles without aphids had the lowest probabilities of recovery, but there is considerable overlap with the other diets. See the posterior comparisons below.

```
ggplot(post_df, aes(x=value, fill=diet)) +
  geom_density(alpha=.4) +
  xlab('Posterior density') +
  ylab('Probability of recovery')
```



Problem 2

One of the biggest advantages of Bayesian approaches is the ease with which you can make inference on **derived parameters**. For example, we might want to know which diet treatment gives the highest survival probability. In one draw from the posterior distribution, we should have five estimated probabilities. The highest probability can be recorded and stored in an object (say `best`). We can do this for each posterior draw to produce a vector of the "best" treatments (from the beetle's perspective). To find the posterior probability that each particular treatment is best, count the frequency of each treatment in the `best` vector, and divide by the total number of posterior draws. Do this below using the results from problem 1, and report the posterior probabilities.

```
best <- apply(post$p, 1, which.max)
p_best <- rep(NA, stan_d$n_trt)
for (i in 1:stan_d$n_trt){
  p_best[i] <- mean(best == i)
}
data.frame(diet = levels(factor(d$diet)), p_best)
```

```
##                 diet  p_best
## 1          20 aphids 0.23275
## 2 20 aphids + pollen 0.29650
## 3           2 aphids 0.04200
## 4  2 aphids + pollen 0.42850
## 5             Pollen 0.00025
```

Which treatment was best? What is the probability of that treatment being the best, conditional on the data?

The 2 aphids + pollen treatment was the best with probability ≈ 0.4.

Problem 3

Simulate data from a normal distribution for three groups, each with a different mean. You can assume that the standard deviations for each group are equal. In generating data, use a design matrix to acquire the expected value for your data (somewhere in your code there should be `X %*% beta`).

```
# your code here
n <- 50
ngroup <- 3
group_vector <- sample(ngroup, n, replace = TRUE) %>%
  sort()

is_in_group <- function(x, group) {
  # tests whether elements in x are in a group
  return(as.numeric(x == group))
}

# make design matrix
X <- matrix(nrow = n, ncol = ngroup)
for (i in 1:ngroup){
  X[, i] <- is_in_group(group_vector, i)
}
X
```

```
##       [,1] [,2] [,3]
##  [1,]    1    0    0
##  [2,]    1    0    0
##  [3,]    1    0    0
##  [4,]    1    0    0
##  [5,]    1    0    0
##  [6,]    1    0    0
##  [7,]    1    0    0
##  [8,]    1    0    0
##  [9,]    1    0    0
## [10,]    1    0    0
## [11,]    1    0    0
## [12,]    1    0    0
## [13,]    1    0    0
## [14,]    1    0    0
## [15,]    1    0    0
## [16,]    1    0    0
## [17,]    0    1    0
## [18,]    0    1    0
## [19,]    0    1    0
## [20,]    0    1    0
## [21,]    0    1    0
## [22,]    0    1    0
## [23,]    0    1    0
```

```
## [24,]    0    1    0
## [25,]    0    1    0
## [26,]    0    1    0
## [27,]    0    1    0
## [28,]    0    1    0
## [29,]    0    1    0
## [30,]    0    1    0
## [31,]    0    1    0
## [32,]    0    1    0
## [33,]    0    1    0
## [34,]    0    1    0
## [35,]    0    0    1
## [36,]    0    0    1
## [37,]    0    0    1
## [38,]    0    0    1
## [39,]    0    0    1
## [40,]    0    0    1
## [41,]    0    0    1
## [42,]    0    0    1
## [43,]    0    0    1
## [44,]    0    0    1
## [45,]    0    0    1
## [46,]    0    0    1
## [47,]    0    0    1
## [48,]    0    0    1
## [49,]    0    0    1
## [50,]    0    0    1
```

```r
# choose group means and residual sd
beta <- c(-2.3, .1, 1.5)
sigma <- .5

# simulate data
y <- rnorm(n, X %*% beta, sigma)
```

Write a Stan model statement for a linear model that you can use to estimate the parameters.

```
data {
  int n; //
  int p; //
  matrix[n, p] X;
  vector[n] y;
}
parameters {
  vector[p] beta;
  real<lower=0> sigma;
}
model {
  beta ~ normal(0, 5);
  sigma ~ normal(0, 5);
  y ~ normal(X * beta, sigma);
}
```

Use Stan to estimate the parameters of your model.

```r
library(rstan)
stan_d <- list(X = X, p = ngroup, n = n, y = y)
m_fit <- stan('lm.stan', data = stan_d)
```

```
##
## SAMPLING FOR MODEL 'lm' NOW (CHAIN 1).
##
## Chain 1, Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1, Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1, Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1, Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1, Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1, Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1, Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1, Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1, Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1, Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1, Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1, Iteration: 2000 / 2000 [100%]  (Sampling)#
## #  Elapsed Time: 0.040581 seconds (Warm-up)
## #                0.039924 seconds (Sampling)
## #                0.080505 seconds (Total)
## #
##
## SAMPLING FOR MODEL 'lm' NOW (CHAIN 2).
##
## Chain 2, Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2, Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2, Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2, Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2, Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2, Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2, Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2, Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2, Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2, Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2, Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2, Iteration: 2000 / 2000 [100%]  (Sampling)#
## #  Elapsed Time: 0.040754 seconds (Warm-up)
## #                0.041449 seconds (Sampling)
## #                0.082203 seconds (Total)
## #

## The following numerical problems occured the indicated number of times after warmup on chain 2

##                                                                          count
## Exception thrown at line 16: normal_log: Scale parameter is 0, but must be > 0!     1

## When a numerical problem occurs, the Metropolis proposal gets rejected.

## However, by design Metropolis proposals sometimes get rejected even when there are no numerical probl

## Thus, if the number in the 'count' column is small, do not ask about this message on stan-users.
```

```
##
## SAMPLING FOR MODEL 'lm' NOW (CHAIN 3).
##
## Chain 3, Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3, Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3, Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3, Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3, Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3, Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3, Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3, Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3, Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3, Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3, Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3, Iteration: 2000 / 2000 [100%]  (Sampling)#
## #  Elapsed Time: 0.037938 seconds (Warm-up)
## #                0.045314 seconds (Sampling)
## #                0.083252 seconds (Total)
## #
##
## SAMPLING FOR MODEL 'lm' NOW (CHAIN 4).
##
## Chain 4, Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4, Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4, Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4, Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4, Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4, Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4, Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4, Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4, Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4, Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4, Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4, Iteration: 2000 / 2000 [100%]  (Sampling)#
## #  Elapsed Time: 0.038501 seconds (Warm-up)
## #                0.042629 seconds (Sampling)
## #                0.08113 seconds (Total)
## #
```

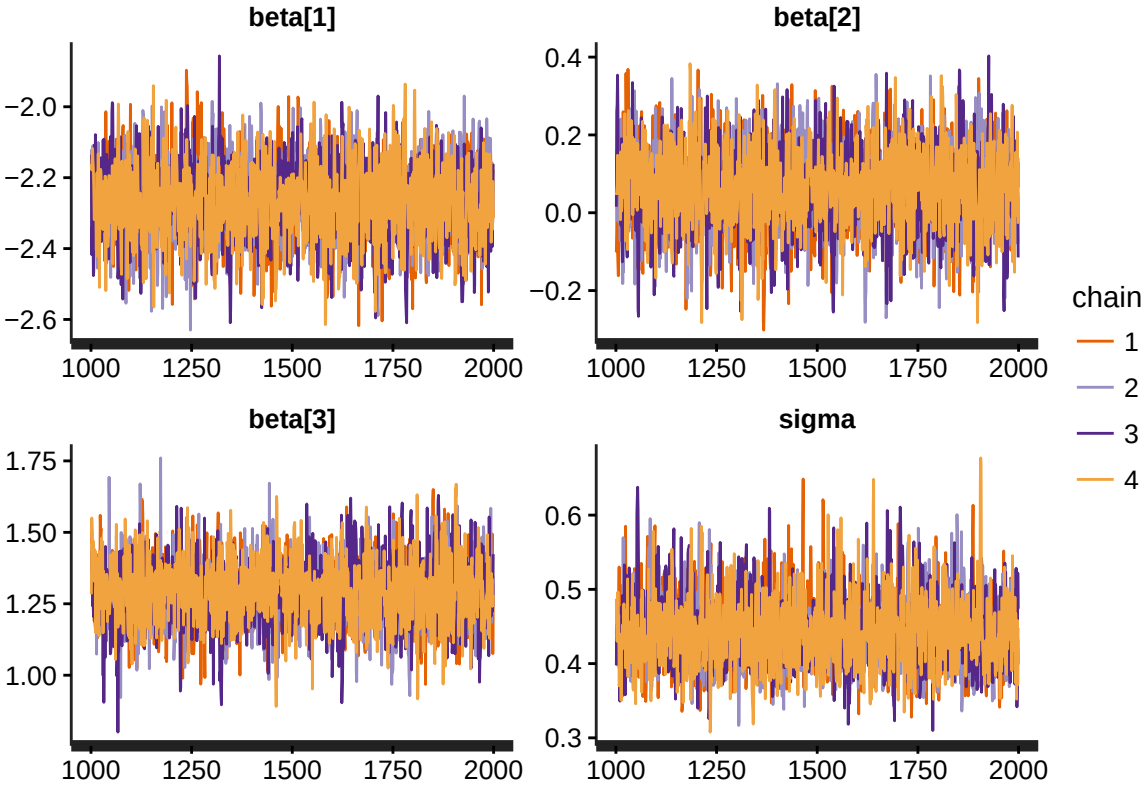
Assess convergence of the MCMC algorithm graphically and with the Rhat statistic.

```
m_fit
```

```
## Inference for Stan model: lm.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## beta[1] -2.27    0.00 0.11 -2.49 -2.34 -2.27 -2.19 -2.06  2318    1
## beta[2]  0.06    0.00 0.11 -0.15 -0.01  0.06  0.13  0.26  2541    1
## beta[3]  1.29    0.00 0.11  1.06  1.22  1.29  1.36  1.52  2072    1
## sigma    0.44    0.00 0.05  0.36  0.41  0.44  0.47  0.55  2273    1
## lp__    14.87    0.04 1.52 11.04 14.12 15.22 15.99 16.73  1447    1
```

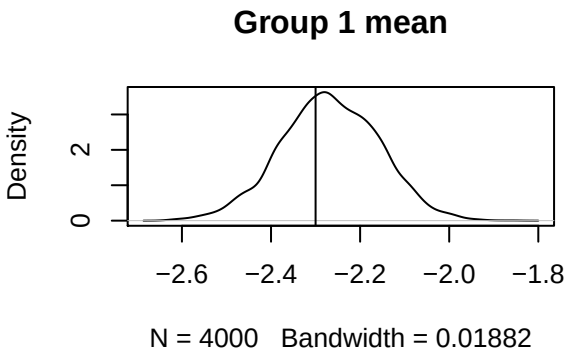
```
##
## Samples were drawn using NUTS(diag_e) at Fri Feb  5 19:10:13 2016.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

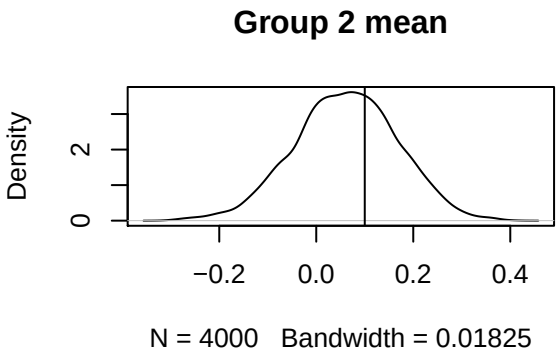
```
traceplot(m_fit)
```

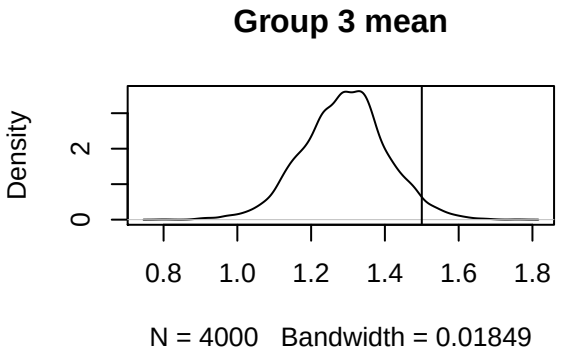


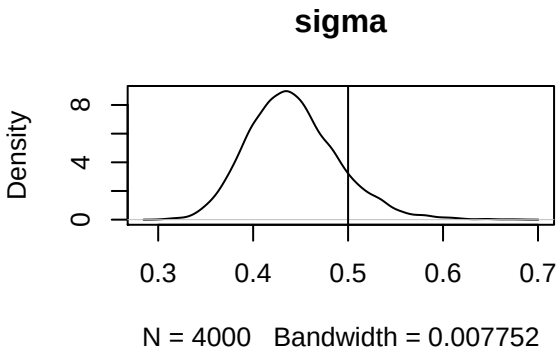
Plot the marginal posterior distributions for each parameter with a vertical line at the true value.

```
post <- rstan::extract(m_fit)
par(mfrow=c(2, 2))
for (i in 1:ngroup){
  plot(density(post$beta[, i]), main = paste('Group', i, 'mean', sep = ' '))
  abline(v = beta[i])
}
plot(density(post$sigma), main = 'sigma')
abline(v = sigma)
```


Group 1 mean
Density
0
2
−2.6
−2.4
−2.2
−2.0
−1.8
N = 4000 Bandwidth = 0.01882


Group 2 mean
Density
0
2
−0.2
0.0
0.2
0.4
N = 4000 Bandwidth = 0.01825


Group 3 mean
Density
0
2
0.8
1.0
1.2
1.4
1.6
1.8
N = 4000 Bandwidth = 0.01849


sigma
Density
0
4
8
0.3
0.4
0.5
0.6
0.7
N = 4000 Bandwidth = 0.007752