An introduction to Bayesian inference (solutions)

Nate Pope

28 October 2016

```
library(rstan)
library(shinystan)
library(ggplot2)
library(dplyr)
```

Exercise 1:

The prior on μ can be interpreted as follows: the mean of the prior is our best guess as to the value of μ, while standard deviation of the prior represents the amount of uncertainty we have regarding this guess. In this case our prior guess about μ is -1, far from the true value of 2, and the disagreement between the prior and likelihood will result in a posterior that is a compromise between the two. Copy the functions above and modify them to answer the following questions:

1. What happens to the posterior if you decrease the standard deviation of the prior on μ to 0.1? Why?
2. What happens to the posterior if you change the mean of the prior on μ to 2? Why?
3. What happens to the posterior if you simulate new data, but increase the number of data points to 10? Why?

Redfine functions that that I can supply values for the prior parameters:

```
# simulate some data
set.seed(101)
fake_data <- rnorm(n = 3, mean = 2, sd = 2)
fake_data # look at it
```

```
## [1] 1.3479270 3.1049237 0.6501123
```

```
# a function that returns the likelihood for given values of mu and sigma
likelihood <- function(mu, sigma, data_values){
  prod( dnorm(data_values, mean = mu, sd = sigma) )
}

# a function that returns the prior for given values of mu and sigma
prior <- function(mu, sigma,
                  mu.prior.mean = -1,
                  mu.prior.sd = 1,
                  sigma.prior.sd = 1){
  # normal prior on mu, half-normal prior on sigma
  dnorm(mu, mean = mu.prior.mean, sd = mu.prior.sd) *
    2*dnorm(abs(sigma), mean = 0, sd = sigma.prior.sd)
}
```

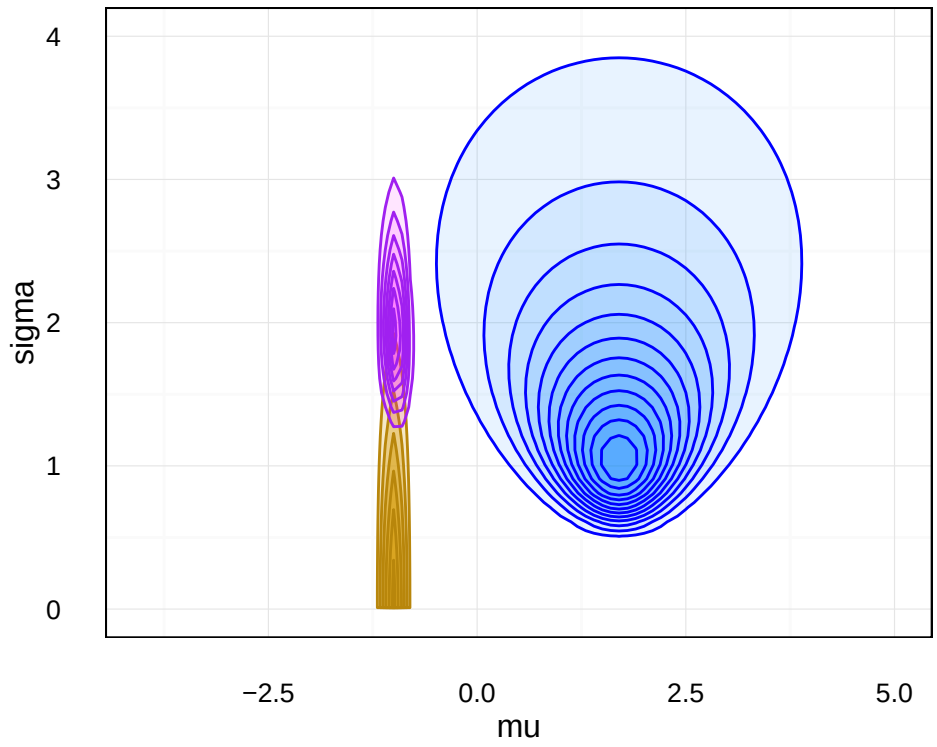
```
# a function that computes the unnormalized posterior
# given the functions likelihood(), prior() defined above
posterior <- function(mu, sigma, data_values,
                  mu.prior.mean = -1,
                  mu.prior.sd = 1,
                  sigma.prior.sd = 1){
  likelihood(mu, sigma, data_values) *
    prior(mu, sigma, mu.prior.mean, mu.prior.sd, sigma.prior.sd)
}
```

Might as well automate the plot, too:

```
bayes_plot <- function(data_values,
                  mu.prior.mean = -1,
                  mu.prior.sd = 1,
                  sigma.prior.sd = 1){
  expand.grid(mu = seq(-4, 5, 0.1), sigma = seq(0.01, 4, 0.1)) %>%
    rowwise() %>%
    mutate(likelihood = likelihood(mu, sigma, data_values),
      prior = prior(mu, sigma, mu.prior.mean, mu.prior.sd, sigma.prior.sd),
      posterior = posterior(mu, sigma, data_values, mu.prior.mean, mu.prior.sd, sigma.prior.sd)) %>%
    ggplot(aes(x=mu, y=sigma)) +
    stat_contour(geom="polygon", aes(z = likelihood, alpha = ..level..),
                 color="blue", fill="dodgerblue") +
    stat_contour(geom="polygon", aes(z = prior, alpha = ..level..),
             color="darkgoldenrod", fill="goldenrod") +
    stat_contour(geom="polygon", aes(z = posterior, alpha = ..level..),
             color="purple", fill="magenta") +
    xlim(-4,5) + ylim(0, 4) + theme_minimal() +
    theme(legend.position = "none", panel.border=element_rect(fill=NA))
}
```

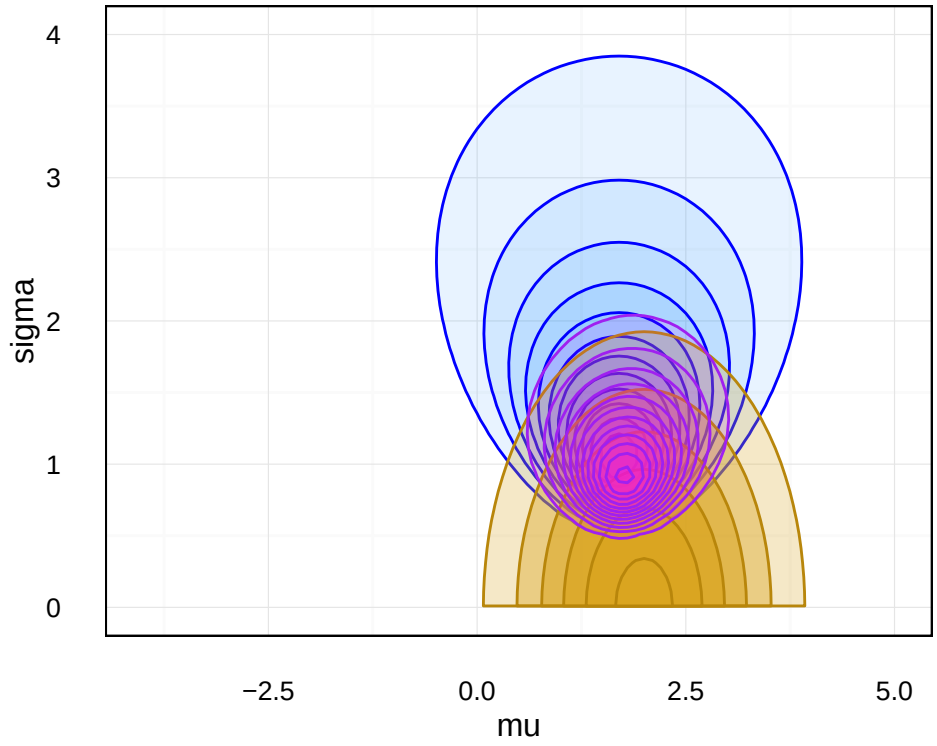
Now with prior standard deviation of μ as 0.1, the prior has a stronger influence on the posterior. Essentially, this is because we've 'increased' our belief in our prior guess at μ.

```
bayes_plot(fake_data, mu.prior.sd = 0.1)
```



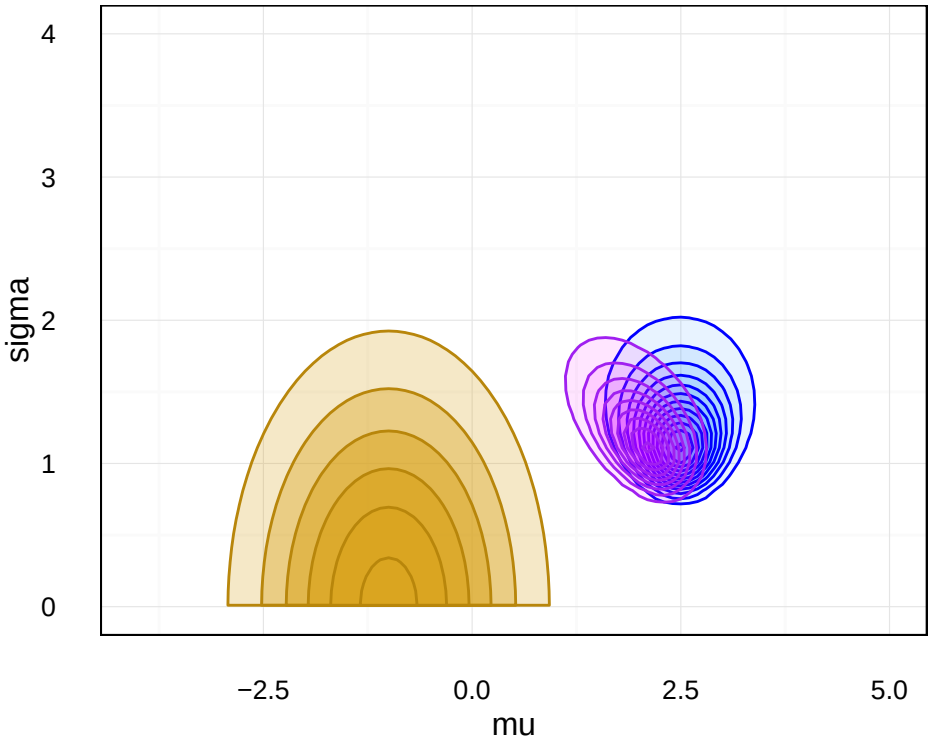
Now with mean of prior on μ set to 2, the prior and the likelihood are more in agreement. Thus, the posterior reflects the likelihood better than the previous example.

```
bayes_plot(fake_data, mu.prior.mean = 2)
```



Now with a new dataset of 10 observations, the likelihood is "stronger": the probability mass is more concentrated around particular values of μ and σ. Thus, the posterior more resembles the likelihood: essentially, the likelihood has dominated the prior.

```
set.seed(101)
fake_data2 <- rnorm(n = 10, mean = 2, sd = 2)
bayes_plot(fake_data2)
```



Exercise 2.

Modify the regression model to include the covariate `drat` in addition to `wt`. You'll need to change all three blocks to include: a new covariate, a new parameter, a new prior and likelihood. How might you write a model that could include an arbitrary number of covariates?

```
exercise2 <- "
data {
  int N; // the number of data points
  vector[N] Y; // the response variable is an N-dimensional vector
  vector[N] X; // the covariate is also an N-dimensional vector
  vector[N] X2; // the covariate is also an N-dimensional vector
}

parameters {
  real alpha; // the intercept
  real beta; // the first slope
  real beta2; // the second slope
```

```
  real<lower=0> sigma; // the standard deviation
}

model {
  // priors
  target += cauchy_lpdf(sigma | 0, 5); // Cauchy prior distribution on std. deviation
  target += normal_lpdf(alpha | 0, 10); // Normal(0,10) prior on intercept
  target += normal_lpdf(beta | 0, 10); // Normal(0,10) prior on first slope
  target += normal_lpdf(beta2 | 0, 10); // Normal(0,10) prior on second slope

  // likelihood
  target += normal_lpdf(Y | alpha + X*beta + X2*beta2, sigma);
}
"

library(rstan)
# fit model with stan
ex2data <- list(N=nrow(mtcars), Y=mtcars$mpg, X=mtcars$wt, X2=mtcars$drat)
ex2_fit <- stan(model_name = "exercise2",
                      model_code = exercise2,
                      data = ex2data,
                      chains = 3, iter = 1000,
                      thin = 1, warmup = 300)
```

```
##
## SAMPLING FOR MODEL 'exercise2' NOW (CHAIN 1).
##
## Chain 1, Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1, Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1, Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1, Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1, Iteration: 301 / 1000 [ 30%]  (Sampling)
## Chain 1, Iteration: 400 / 1000 [ 40%]  (Sampling)
## Chain 1, Iteration: 500 / 1000 [ 50%]  (Sampling)
## Chain 1, Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1, Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1, Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1, Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1, Iteration: 1000 / 1000 [100%]  (Sampling)
##  Elapsed Time: 0.036007 seconds (Warm-up)
##                0.084338 seconds (Sampling)
##                0.120345 seconds (Total)
##
##
## SAMPLING FOR MODEL 'exercise2' NOW (CHAIN 2).
##
## Chain 2, Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2, Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2, Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2, Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2, Iteration: 301 / 1000 [ 30%]  (Sampling)
## Chain 2, Iteration: 400 / 1000 [ 40%]  (Sampling)
## Chain 2, Iteration: 500 / 1000 [ 50%]  (Sampling)
```

```
## Chain 2, Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2, Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2, Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2, Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2, Iteration: 1000 / 1000 [100%]  (Sampling)
##  Elapsed Time: 0.046532 seconds (Warm-up)
##                0.117651 seconds (Sampling)
##                0.164183 seconds (Total)
##
##
## SAMPLING FOR MODEL 'exercise2' NOW (CHAIN 3).
##
## Chain 3, Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3, Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3, Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3, Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3, Iteration: 301 / 1000 [ 30%]  (Sampling)
## Chain 3, Iteration: 400 / 1000 [ 40%]  (Sampling)
## Chain 3, Iteration: 500 / 1000 [ 50%]  (Sampling)
## Chain 3, Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3, Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 3, Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3, Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3, Iteration: 1000 / 1000 [100%]  (Sampling)
##  Elapsed Time: 0.035942 seconds (Warm-up)
##                0.077163 seconds (Sampling)
##                0.113105 seconds (Total)
```

Exercise 3.

Modify the regression model to use the non-negative, integer-valued variable `hp` as a response variable instead of `mpg`. Use a Poisson distribution for the likelihood. The Poisson has a single parameter: the mean, which is constrained to be positive. To respect this constraint, we exponentiate the regression equation $\alpha + x_i\beta$ (this type of transformation – to respect contraints – is often called a "link function" in the statistical literature). The model is

$$y_i \sim \text{Poisson}(\exp(\alpha + x_i\beta))$$

Note that the function to implement the Poisson distribution in Stan is `poisson_lpmf( variable | mean )`.

```
# your code goes here
exercise3 <- "
data {
  int N; // the number of data points
  int Y[N]; // the response variable is an N-dimensional integer
  vector[N] X; // the covariate is also an N-dimensional vector
}

parameters {
  real alpha; // the intercept
```

```
  real beta; // the slope
}

model {
  // priors
  target += normal_lpdf(alpha | 0, 10); // Normal(0,10) prior on intercept
  target += normal_lpdf(beta | 0, 10); // Normal(0,10) prior on first slope

  // likelihood
  target += poisson_lpmf(Y | exp(alpha + X*beta) );
}
"

library(rstan)
# fit model with stan
ex3data <- list(N=nrow(mtcars), Y=mtcars$hp, X=mtcars$wt)
ex3_fit <- stan(model_name = "exercise3",
                      model_code = exercise3,
                      data = ex3data,
                      chains = 3, iter = 1000,
                      thin = 1, warmup = 300)
```

```
##
## SAMPLING FOR MODEL 'exercise3' NOW (CHAIN 1).
##
## Chain 1, Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1, Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1, Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1, Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1, Iteration: 301 / 1000 [ 30%]  (Sampling)
## Chain 1, Iteration: 400 / 1000 [ 40%]  (Sampling)
## Chain 1, Iteration: 500 / 1000 [ 50%]  (Sampling)
## Chain 1, Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1, Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1, Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1, Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1, Iteration: 1000 / 1000 [100%]  (Sampling)
##  Elapsed Time: 0.017381 seconds (Warm-up)
##                0.044838 seconds (Sampling)
##                0.062219 seconds (Total)
##
##
## SAMPLING FOR MODEL 'exercise3' NOW (CHAIN 2).
##
## Chain 2, Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2, Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2, Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2, Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2, Iteration: 301 / 1000 [ 30%]  (Sampling)
## Chain 2, Iteration: 400 / 1000 [ 40%]  (Sampling)
## Chain 2, Iteration: 500 / 1000 [ 50%]  (Sampling)
## Chain 2, Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2, Iteration: 700 / 1000 [ 70%]  (Sampling)
```

```
## Chain 2, Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2, Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2, Iteration: 1000 / 1000 [100%]  (Sampling)
##  Elapsed Time: 0.021542 seconds (Warm-up)
##                0.045686 seconds (Sampling)
##                0.067228 seconds (Total)
##
##
## SAMPLING FOR MODEL 'exercise3' NOW (CHAIN 3).
##
## Chain 3, Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3, Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3, Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3, Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3, Iteration: 301 / 1000 [ 30%]  (Sampling)
## Chain 3, Iteration: 400 / 1000 [ 40%]  (Sampling)
## Chain 3, Iteration: 500 / 1000 [ 50%]  (Sampling)
## Chain 3, Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3, Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 3, Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3, Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3, Iteration: 1000 / 1000 [100%]  (Sampling)
##  Elapsed Time: 0.023082 seconds (Warm-up)
##                0.042379 seconds (Sampling)
##                0.065461 seconds (Total)
```

Exercise 4.

An optionial block is called `generated quantities {}`. If you wanted to calculate functions of the parameters (and save the results across MCMC samples), you would do so inside this block. An example of such a function is the fitted values given by the equation `alpha + beta*X`. Add a `generated quantities` block to the regression model, and store the fitted values in a vector called `fitted_values`.

```
exercise4 <- "
data {
  int N; // the number of data points
  vector[N] Y; // the response variable is an N-dimensional vector
  vector[N] X; // the covariate is also an N-dimensional vector
}

parameters {
  real alpha; // the intercept
  real beta; // the slope
  real<lower=0> sigma; // the standard deviation
}

model {
  // priors
  target += cauchy_lpdf(sigma | 0, 5); // Cauchy prior distribution on std. deviation
  target += normal_lpdf(alpha | 0, 10); // Normal(0,10) prior on intercept
  target += normal_lpdf(beta | 0, 10); // Normal(0,10) prior on first slope
```

```
  // likelihood
  target += normal_lpdf(Y | alpha + X*beta, sigma);
}

generated quantities {
  // calculate the fitted values and save as vector
  vector[N] fitted;
  fitted = alpha + X*beta;

}
"

library(rstan)
# fit model with stan
ex4data <- list(N=nrow(mtcars), Y=mtcars$mpg, X=mtcars$wt)
ex4_fit <- stan(model_name = "exercise4",
                        model_code = exercise4,
                        data = ex4data,
                        chains = 3, iter = 1000,
                        thin = 1, warmup = 300)
```

```
##
## SAMPLING FOR MODEL 'exercise4' NOW (CHAIN 1).
##
## Chain 1, Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1, Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1, Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1, Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1, Iteration: 301 / 1000 [ 30%]  (Sampling)
## Chain 1, Iteration: 400 / 1000 [ 40%]  (Sampling)
## Chain 1, Iteration: 500 / 1000 [ 50%]  (Sampling)
## Chain 1, Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1, Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1, Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1, Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1, Iteration: 1000 / 1000 [100%]  (Sampling)
##  Elapsed Time: 0.010917 seconds (Warm-up)
##                0.021327 seconds (Sampling)
##                0.032244 seconds (Total)
##
##
## SAMPLING FOR MODEL 'exercise4' NOW (CHAIN 2).
##
## Chain 2, Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2, Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2, Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2, Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2, Iteration: 301 / 1000 [ 30%]  (Sampling)
## Chain 2, Iteration: 400 / 1000 [ 40%]  (Sampling)
## Chain 2, Iteration: 500 / 1000 [ 50%]  (Sampling)
## Chain 2, Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2, Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2, Iteration: 800 / 1000 [ 80%]  (Sampling)
```

```
## Chain 2, Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2, Iteration: 1000 / 1000 [100%]  (Sampling)
##  Elapsed Time: 0.011447 seconds (Warm-up)
##                0.023344 seconds (Sampling)
##                0.034791 seconds (Total)
##
##
## SAMPLING FOR MODEL 'exercise4' NOW (CHAIN 3).
##
## Chain 3, Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3, Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3, Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3, Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3, Iteration: 301 / 1000 [ 30%]  (Sampling)
## Chain 3, Iteration: 400 / 1000 [ 40%]  (Sampling)
## Chain 3, Iteration: 500 / 1000 [ 50%]  (Sampling)
## Chain 3, Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3, Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 3, Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3, Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3, Iteration: 1000 / 1000 [100%]  (Sampling)
##  Elapsed Time: 0.010956 seconds (Warm-up)
##                0.022449 seconds (Sampling)
##                0.033405 seconds (Total)
```

Exercise 5.

Modify the model above to include a categorical covariate, the number of gears `gear`. To do so, you'll have to allow `alpha` to vary across data points in the likelihood. `alpha` will have to be defined as a vector of length 3.

Hint: there are several ways to approach this problem, but one involves the following indexing 'trick': if `alpha` is a vector of three numbers, and `indices` is a set of `N` integers that only include 1,2,3, then `alpha[indices]` is a vector with `N` elements (the elements of `alpha` 'repeated' corresponding to the pattern given in `indices`. This is similar to how R works, and can be demonstrated (in R) as follows:

```
alpha <- c(-0.5, 3, -18)
indices <- c(1,1,3,2,2,1,2,3)
alpha[indices]
```

```
## [1]  -0.5  -0.5 -18.0   3.0   3.0  -0.5   3.0 -18.0
```

The solution:

```
exercise5 <- "
data {
  int N; // the number of data points
  vector[N] Y; // the response variable is an N-dimensional vector
  vector[N] X; // the covariate is also an N-dimensional vector
  int Trans[N]; // an indicator for number of gears
}
```

```
parameters {
  vector[3] alpha; // intercepts, one for each type of gear
  real beta; // the slope
  real<lower=0> sigma; // the standard deviation
}

model {
  // priors
  target += cauchy_lpdf(sigma | 0, 5); // Cauchy prior distribution on std. deviation
  target += normal_lpdf(alpha | 0, 10); // Normal(0,10) prior on intercept
  target += normal_lpdf(beta | 0, 10); // Normal(0,10) prior on first slope

  // likelihood
  target += normal_lpdf(Y | alpha[Trans] + X*beta, sigma);
}
"

library(rstan)
# fit model with stan
ex5data <- list(N=nrow(mtcars), Y=mtcars$mpg, X=mtcars$wt,
                Trans=as.integer(as.factor(mtcars$gear)))
ex5_fit <- stan(model_name = "exercise5",
                      model_code = exercise5,
                      data = ex5data,
                      chains = 3, iter = 1000,
                      thin = 1, warmup = 300)
```

```
##
## SAMPLING FOR MODEL 'exercise5' NOW (CHAIN 1).
##
## Chain 1, Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1, Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1, Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1, Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1, Iteration: 301 / 1000 [ 30%]  (Sampling)
## Chain 1, Iteration: 400 / 1000 [ 40%]  (Sampling)
## Chain 1, Iteration: 500 / 1000 [ 50%]  (Sampling)
## Chain 1, Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1, Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1, Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1, Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1, Iteration: 1000 / 1000 [100%]  (Sampling)
##  Elapsed Time: 0.024364 seconds (Warm-up)
##                0.035979 seconds (Sampling)
##                0.060343 seconds (Total)
##
##
## SAMPLING FOR MODEL 'exercise5' NOW (CHAIN 2).
##
## Chain 2, Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2, Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2, Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2, Iteration: 300 / 1000 [ 30%]  (Warmup)
```

```
## Chain 2, Iteration: 301 / 1000 [ 30%]  (Sampling)
## Chain 2, Iteration: 400 / 1000 [ 40%]  (Sampling)
## Chain 2, Iteration: 500 / 1000 [ 50%]  (Sampling)
## Chain 2, Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2, Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2, Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2, Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2, Iteration: 1000 / 1000 [100%]  (Sampling)
##  Elapsed Time: 0.024323 seconds (Warm-up)
##                0.043401 seconds (Sampling)
##                0.067724 seconds (Total)
##
##
## SAMPLING FOR MODEL 'exercise5' NOW (CHAIN 3).
##
## Chain 3, Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3, Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3, Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3, Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3, Iteration: 301 / 1000 [ 30%]  (Sampling)
## Chain 3, Iteration: 400 / 1000 [ 40%]  (Sampling)
## Chain 3, Iteration: 500 / 1000 [ 50%]  (Sampling)
## Chain 3, Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3, Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 3, Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3, Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3, Iteration: 1000 / 1000 [100%]  (Sampling)
##  Elapsed Time: 0.029625 seconds (Warm-up)
##                0.038652 seconds (Sampling)
##                0.068277 seconds (Total)
```