Problem Set 2: R Part

Your Name Goes Here

Details

How to Write Up

The written part of this assignment can be either typeset using latex or hand written.

Grading

5% of your grade on this assignment is for turning in something legible. This means it should be organized, and any Rmd files should knit to pdf without issue.

An additional 15% of your grade is for completion. A quick pass will be made to ensure that you've made a reasonable attempt at all problems.

Across both the written part and the R part, in the range of 1 to 3 problems will be graded more carefully for correctness. In grading these problems, an emphasis will be placed on full explanations of your thought process. You don't need to write more than a few sentences for any given problem, but you should write complete sentences! Understanding and explaining the reasons behind what you are doing is at least as important as solving the problems correctly.

Solutions to all problems will be provided.

Collaboration

You are allowed to work with others on this assignment, but you must complete and submit your own write up. You should not copy large blocks of code or written text from another student.

Sources

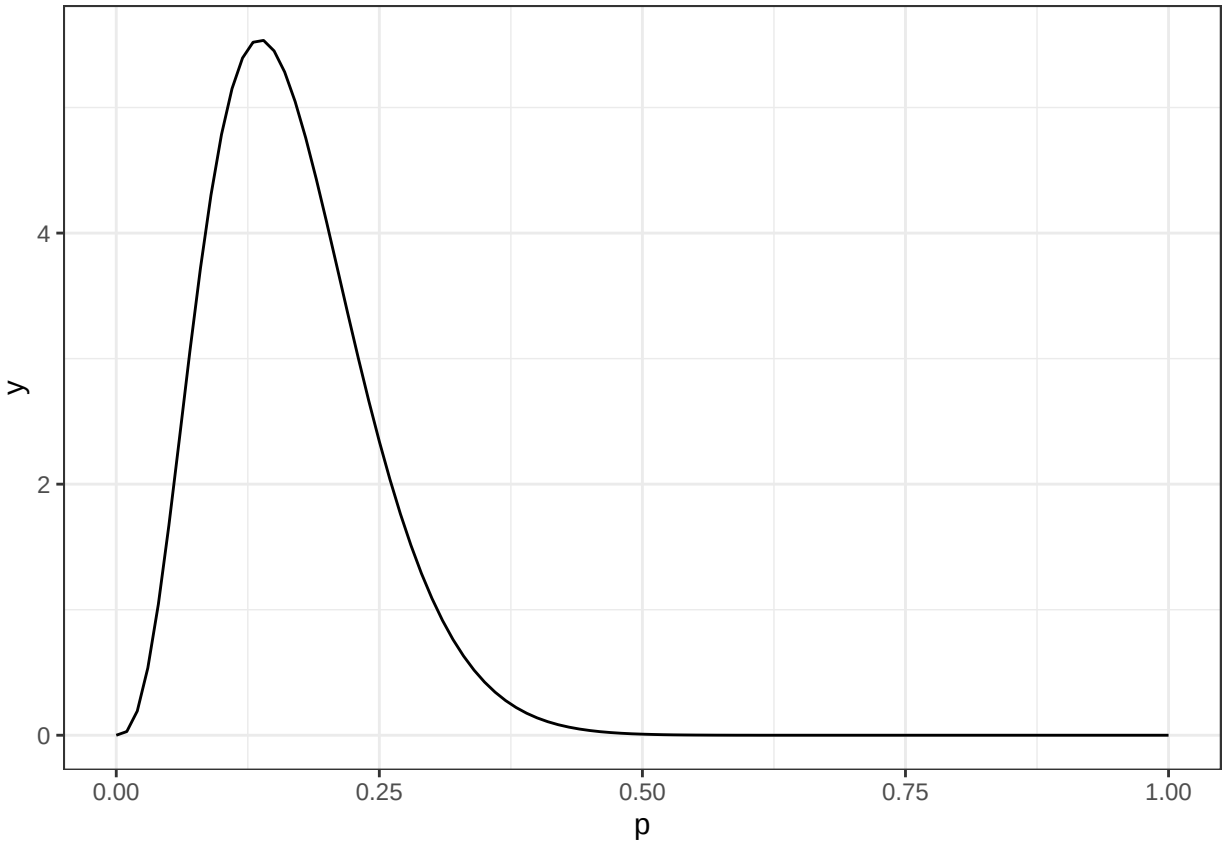
You may refer to our text, Wikipedia, and other online sources. All sources you refer to must be cited in the space I have provided at the end of this problem set.

Problem I: M&M's

The code below does four things for specified values of *alpha* and *beta*: 1. Makes a plot of the pdf of the distribution 2. Calculates the mean of the distribution 3. Calculates the 25th and 75th percentiles of the distribution 4. Calculates the 5th and 95th percentiles of the distribution

```
alpha <- 4
beta <- 20

ggplot(data = data.frame(p = c(0, 1)), mapping = aes(x = p)) +
  stat_function(fun = dbeta, args = list(shape1 = alpha, shape2 = beta)) +
  theme_bw()
```




```
paste0("The mean is: ", alpha/(alpha + beta))
```

```
## [1] "The mean is: 0.166666666666667"
```

```
paste0("The 25th and 75th percentiles are: ",
  paste0(qbeta(c(0.25, 0.75), shape1 = alpha, shape2 = beta), collapse = ", "))
```

```
## [1] "The 25th and 75th percentiles are: 0.111448829155192, 0.212034001937022"
```

```
paste0("The 5th and 95th percentiles are: ",
  paste0(qbeta(c(0.05, 0.95), shape1 = alpha, shape2 = beta), collapse = ", "))
```

```
## [1] "The 5th and 95th percentiles are: 0.0616755174210953, 0.303637701557462"
```

Modify the values of the parameters `alpha` and `beta` in the code above until you get a beta distribution that fairly well matches your guesses about the proportion of M&M's that are blue from the written part of the assignment. That is, you should approximately have:

- A mean of the beta distribution that is similar to your best guess of the proportion of M&M's that are blue from your answer to problem I (a) on the written part.
- 25th and 75th percentiles that are similar to your 50% credible interval $[a, b]$ from problem I (b) on the written part.
- 5th and 95th percentiles that are similar to your 90% credible interval $[c, d]$ from problem I (c) on the written part.

You do not need to get a distribution that matches your answers from the written part exactly! Approximate is good enough.

To make this easier, it is helpful to know that the expected value of a Beta(α, β) random variable is $\frac{\alpha}{\alpha+\beta}$. This gives you a constraint on the kinds of values you can try for α and β while keeping the mean about equal to your best guess of the proportion of M&M's that are blue.

Problem II: Wind Speeds

The R code below reads in the data:

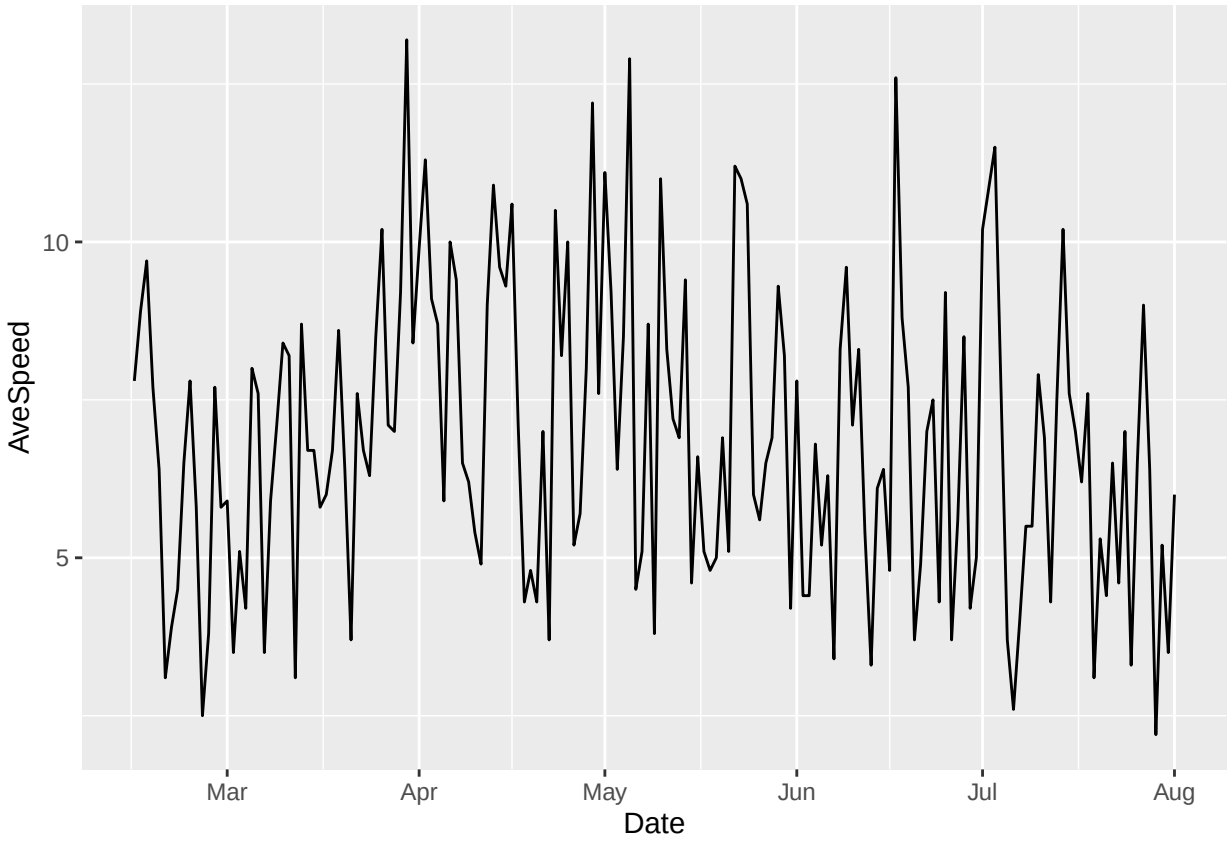
```
wind_speeds <- read_csv("http://www.evanlray.com/data/chihara_hesterberg/Turbine.csv") %>%
  mutate(Date = mdy(paste0(Date2010, " 2010")))
```

```
## Parsed with column specification:
## cols(
##   Date2010 = col_character(),
##   AveKW = col_double(),
##   AveSpeed = col_double(),
##   Production = col_double()
## )
```

```
head(wind_speeds)
```

```
## # A tibble: 6 x 5
##   Date2010 AveKW AveSpeed Production Date
##   <chr>    <dbl>    <dbl>      <dbl> <date>
## 1 Feb 14   548.       7.8      13146 2010-02-14
## 2 Feb 15   776        8.9      18626 2010-02-15
## 3 Feb 16   944.       9.7      22667 2010-02-16
## 4 Feb 17   506.       7.7      12148 2010-02-17
## 5 Feb 18   323.       6.4       7742 2010-02-18
## 6 Feb 19    67.9      3.1       1585 2010-02-19
```

```
ggplot(data = wind_speeds, mapping = aes(x = Date, y = AveSpeed)) +
  geom_line()
```



Background about for loops in R (feel free to skip if you are comfortable with this)

For loops let you repeat a block of code many times. Here is an example:

```
for(i in seq_len(10)) {
  print(paste0("i = ", i))
  print(i^2)
```

```
}
```

```
## [1] "i = 1"
## [1] 1
## [1] "i = 2"
## [1] 4
## [1] "i = 3"
## [1] 9
## [1] "i = 4"
## [1] 16
## [1] "i = 5"
## [1] 25
## [1] "i = 6"
## [1] 36
## [1] "i = 7"
## [1] 49
## [1] "i = 8"
## [1] 64
## [1] "i = 9"
## [1] 81
## [1] "i = 10"
## [1] 100
```

The first line of the code above starts the for loop. `seq_len(10)` creates a `seq`uence of integers of `len`gth 10. The code inside the following curly braces runs once for each number in that sequence. On each iteration of the loop (each time the code runs), the variable `i` will have a different value from the sequence of integers from 1 to 10 we specified. When you run the code, the on the first iteraton of the loop, `i` is equal to 1, so when we print `i^2` we print 1. On the second iteration of the loop, `i = 2`, so the print statement prints 4, and so on.

It is possible to `break` out of a loop early if we don't need every iteration of the loop to run. Here's an example of that:

```
for(i in seq_len(10)) {
  print(paste0("i = ", i))
  if(i >= 8) {
    break
  }
  print(i^2)
}
```

```
## [1] "i = 1"
## [1] 1
## [1] "i = 2"
## [1] 4
## [1] "i = 3"
## [1] 9
## [1] "i = 4"
## [1] 16
## [1] "i = 5"
## [1] 25
## [1] "i = 6"
## [1] 36
## [1] "i = 7"
## [1] 49
## [1] "i = 8"
```

Here, we added a check on the value of `i` to the code in the loop. On every iteration of the loop, we check whether the current value of `i` is at least 8. If it is, we `break` the loop, so that the current iteration stops and any remaining iterations do not run. In the output, you can see that the value of `i^2` was only calculated for values from 1 to 7. The 8th iteration started to run, but the `break` statement ran before we got to calculating `i^2` for `i = 8`. The 9th and 10th iterations of the loop never started.

(1) Find the maximum likelihood estimate of k by implementing the Newton's method optimization algorithm you wrote down in problem III (1) in the written part.

I have given you some starter code to do this. The code below defines functions to evaluate the first and second derivatives of the function $\tilde{\ell}(k|x_1, \ldots, x_n)$ from the introduction to the problem in the written part.

```
d_l_tilde_dk <- function(k) {
  n <- nrow(wind_speeds)
  x_power_k <- wind_speeds$AveSpeed^k
  log_x <- log(wind_speeds$AveSpeed)

  return(n/k - n * sum(x_power_k * log_x) / sum(x_power_k) + sum(log_x))
}

d2_l_tilde_dk2 <- function(k) {
  n <- nrow(wind_speeds)
  x_power_k <- wind_speeds$AveSpeed^k
  log_x <- log(wind_speeds$AveSpeed)

  return(-n/(k^2) - n * sum(x_power_k * log_x^2) / sum(x_power_k) - n * sum(x_power_k * log_x)^2 / sum(x_powe
}
```

For example, if $k = 1$ then the first and second derivatives of $\tilde{\ell}$ can be calculated with the following code:

```
d_l_tilde_dk(1)
```

```
## [1] 147.444
```

```
d2_l_tilde_dk2(1)
```

```
## [1] -1508.801
```

The code below contains an outline for how you can set up the algorithm. You just need to fill in a few steps and run the code.

```
# Starting value for k
k_current <- 2

# Maximum number of iterations and tolerance for stopping
max_iter <- 1000
tolerance <- 1e-8

for(i in seq_len(max_iter)) {
  # You don't need to change the next line.  It sets up for the next iteration of the loop
  k_previous <- k_current

  # replace NA on the next line with a calculation of the update to k
  k_current <- k_previous - d_l_tilde_dk(k_previous) / d2_l_tilde_dk2(k_previous)

  # replace NA on the next lne with a calculation of a quantity you will need to determine whether to stop th
  # this will involve k_current and k_previous
  diff <- abs(k_current - k_previous)

  # replace the TRUE in the if statement below with a check of whether we should stop the algorithm
  if(diff < tolerance) {
    break
  }
}
```

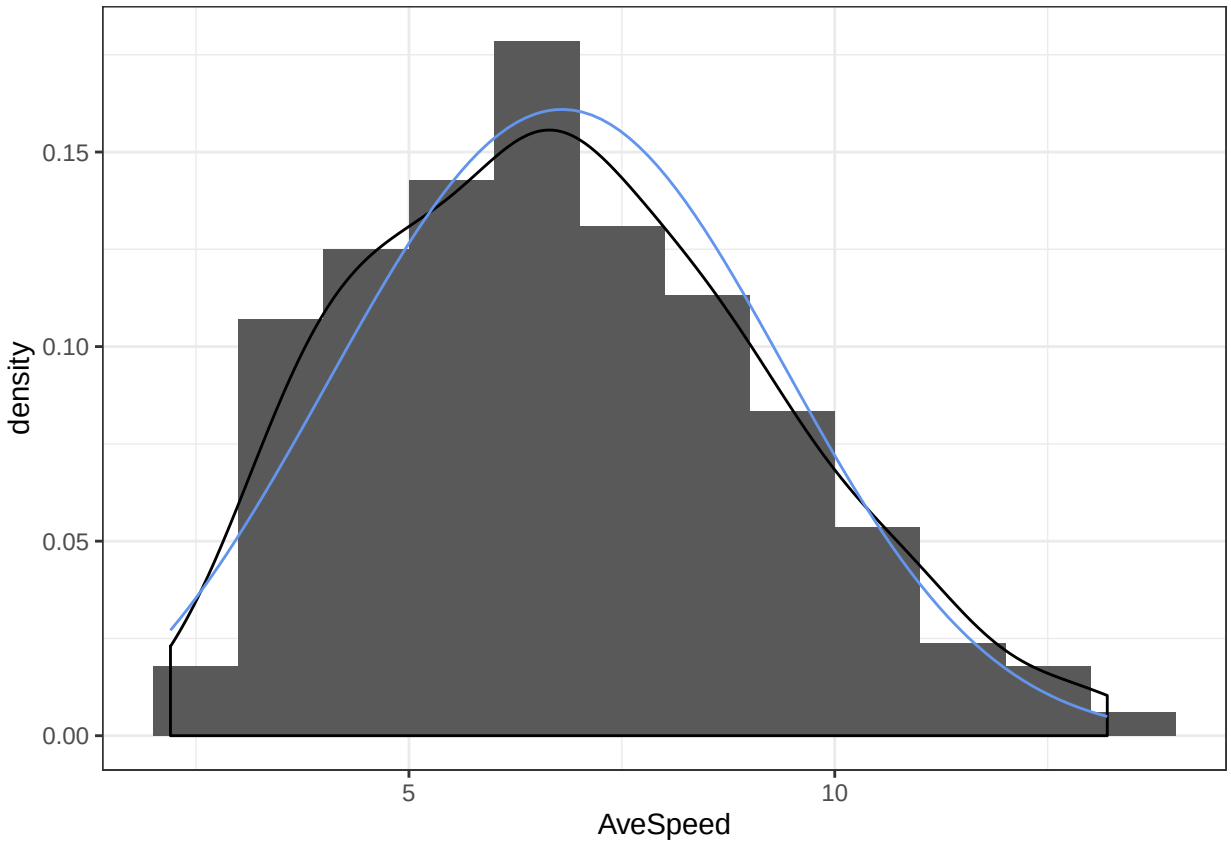
(2) Find the maximum likelihood estimate of λ using the formula you wrote down in problem III (2) in the written part. The daily average wind speeds are in the variable `AveSpeed` in the `wind_speeds` data frame.

```
lambda <- mean(wind_speeds$AveSpeed^k_current)^(1/k_current)
```

(3) Add a representation of the estimated Weibull distribution to the histogram of the data below, using your favorite color. Comment briefly (1 sentence) on how the model fit looks.

You can use the `dweibull` function, which has arguments `shape` (k) and `scale` (lambda).

```
ggplot(data = wind_speeds, mapping = aes(x = AveSpeed)) +
  geom_histogram(binwidth = 1, center = 0.5, mapping = aes(y = ..density..)) +
  geom_density() +
  stat_function(fun = dweibull, args = list(shape = k_current, scale = lambda), color = "cornflowerblue") +
  theme_bw()
```



Problem III: Movie Ratings

The following R cell reads in a data frame with information on a sample of 1794 movies. Among other variables, the data set includes a variable called `imdb_rating`, which has the average rating for the movie from users of the website www.imdb.com on a scale of 0 to 10, and a variable called `imdb_rating_0_1`, which has the movie ratings scaled to lie between 0 and 1.

```
library(tidyverse)
library(rstan)
movies <- read_csv("http://www.evanlray.com/data/misc/imdb/imdb.csv")
head(movies)
```

```
## # A tibble: 6 x 13
##    year title imdb_rating imdb_rating_0_1 num_imdb_ratings mpaa_rating
##   <dbl> <chr>       <dbl>           <dbl>            <dbl> <chr>
## 1  2013 21 &...         5.9            0.59            64520 R
## 2  2012 Dred...         7.1            0.71           217487 R
## 3  2013 12 Y...         8.1            0.81           501013 R
```

```
## 4  2013 2 Gu...         6.7            0.67           168308 R
## 5  2013 42            7.5            0.75           70755 PG-13
## 6  2013 47 R...         6.3            0.63           125401 PG-13
## # ... with 7 more variables: run_time_min <dbl>, budget <dbl>,
## #   domgross <dbl>, intgross <dbl>, budget_2013 <dbl>,
## #   domgross_2013 <dbl>, intgross_2013 <dbl>
```

```
dim(movies)
```

```
## [1] 1794   13
```

If we let X_i be the average rating for movie number i on the scale between 0 and 1, we might use the model $X_i \overset{\text{i.i.d.}}{\sim} \text{Beta}(\alpha, \beta)$.

Recall from the first problem set that a Beta distribution is a distribution for continuous random variables with support on the interval $[0, 1]$. The parameters α and β are required to be positive.

(1) Find the maximum likelihood estimates of the parameters α and β.

Note that the maximum likelihood parameter estimates for the Beta distribution cannot be found in closed form, so you will need to find the maximum likelihood estimates via numerical optimization.

Please use Stan for this. I have included a starter .stan file in this repository with definitions of the data, parameters, and model blocks to fill in. The Stan function for the beta distribution is `beta`; documentation for this function is here: https://mc-stan.org/docs/2_22/functions-reference/beta-distribution.html

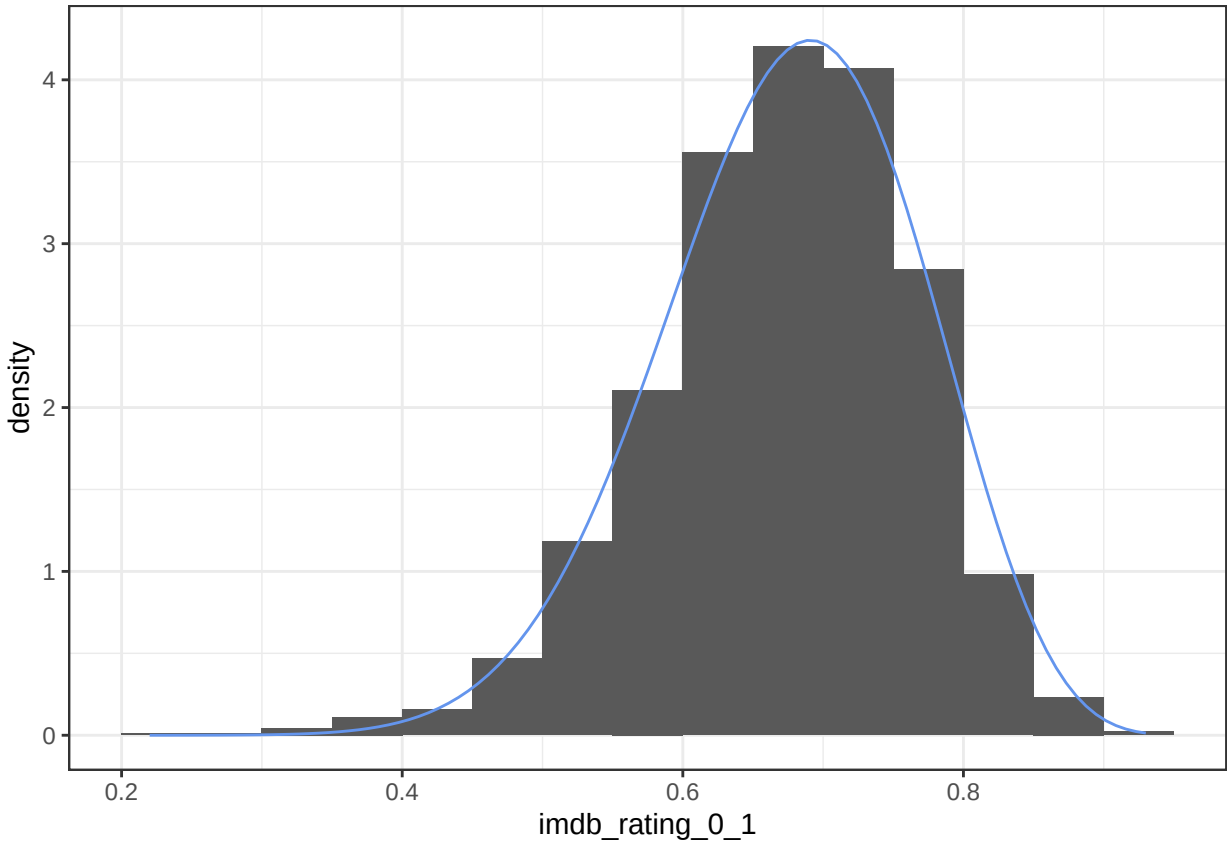
```
# Set up list with data Stan will need to know about
stan_data <- list(
  n = nrow(movies),
  x = movies$imdb_rating_0_1
)

# Compile the Stan model definition to an executable.  Takes a few seconds to run.
movies_model_compiled <- stan_model(file = "movies_model.stan")

# Call Stan to do optimization
movies_fit <- optimizing(movies_model_compiled,
  data = stan_data,
  seed = 8742,
  init = "random"
)
```

(2) Make a histogram or density plot showing the values of the `imdb_rating_0_1` variable in this data set, overlaid with the probability density function for the Beta distribution with the maximum likelihood parameter estimates.

```
ggplot(data = movies, mapping = aes(x = imdb_rating_0_1)) +
  geom_histogram(binwidth = 0.05, boundary = 0, mapping = aes(y = ..density..)) +
  stat_function(fun = dbeta,
    args = list(shape1 = movies_fit$par["alpha"], shape2 = movies_fit$par["beta"]),
    color = "cornflowerblue") +
  theme_bw()
```


density
4
3
2
1
0
0.2
0.4
0.6
0.8
imdb_rating_0_1