CS 1675 Spring 2022 Homework: 04

Assigned February 2, 2022; Due: February 9, 2022

Sameera Boppana

Submission time: February 9, 2022 at 11:00PM EST

Collaborators

Include the names of your collaborators here.

Overview

This assignment focuses on the mathematics of likelihoods, priors, and posterior distributions. You will work with the Binomial likelihood and a Beta prior throughout this assignment.

IMPORTANT: code chunks are created for you. Each code chunk has `eval=FALSE` set in the chunk options. You **MUST** change it to be `eval=TRUE` in order for the code chunks to be evaluated when rendering the document.

Load packages

You will use the `tidyverse` in this assignment, as you have done in the previous assignments.

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──
```

```
## ✓ ggplot2 3.3.5     ✓ purrr   0.3.4
## ✓ tibble  3.1.6     ✓ dplyr   1.0.7
## ✓ tidyr   1.1.4     ✓ stringr 1.4.0
## ✓ readr   2.1.1     ✓ forcats 0.5.1
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Problem 01

Baseball has a rich history of quantitative analysis, even before the rise in the popularity of advanced analytics techniques. Batting averages, slugging percentages, and other metrics have been used to evaluate a players offensive performance for over one hundred years. The batting average requires the number of at bats (or trials) and the number of successful hits (or events). It measures the fraction of at bats a player successfully gets a hit.

You will practice working with the Binomial distribution in order to study the probability that a player gets a hit.

1a)

A certain player is considered to have had a good offensive season. He had 189 hits out of 602 at bats. The number of hits and at bats are provided as variables for you in the code chunk below.

```
player_hits <- 189
player_atbats <- 602
```

We will assume that the number of hits, H, is Binomially distributed, conditioned on the number of at bats, AB, and the probability of a successful hit, μ.

Write out the formula for the Maximum Likelihood Estimate (MLE) for the probability of a successful hit, μ_{ML}, based on the number of at bats, AB, and number of hits, H. Calculate the MLE for the player based on the data provided in the above code chunk. Save the MLE to the variable `player_mle` and print the value to the screen.

SOLUTION

The MLE for the probability of a successful hit is:

$$\mu_{MLE} = H/AB$$

For this particular example:

```
player_mle <- player_hits / player_atbats
player_mle
```

```
## [1] 0.3139535
```

1b)

Let's check your answer in Problem 1a) by visualizing the log-likelihood with respect to the unknown probability, μ. You will work with the un-normalized log-likelihood in this problem. Un-normalized corresponds to dropping the constants or the terms that do not involve the unknown variable, which in this case is the probability μ.

Write out the expression for the un-normalized Binomial log-likelihood for the number of hits H, given the number of at bats, AB, and the probability of a hit, μ. The equation block is started for you, showing that the log-likelihood is just proportional to the expression on the right hand side.

SOLUTION

$$\log(p\left(H \mid AB, \mu\right)) \propto \log(\ \mu^{H}(1-\mu)^{AB-H}))$$

1c)

Let's now generate the visualization. The code chunk below is started for you. It consists of a `tibble` containing a variable `mu`. You will complete the code chunk and generate a visualization for the un-normalized log-likelihood with respect to `mu`, over the interval $\mu = 0.1$ to $\mu = .6$.

Set the variable `mu` equal to a vector consisting of 101 evenly spaced elements from 0.1 to 0.6. Pipe the `tibble` into the `mutate()` function and create a new variable `log_lik`. Evaluate the un-normalized log-likelihood using the number of hits and at bats for the player. Pipe the result into `ggplot()` and map the `x` aesthetic `mu` and the `y` aesthetic to `log_lik`. Use a `geom_line()` to display those aesthetics. As a reference point, include your calculated MLE on the probability with a `geom_vline()`. The `geom_vline()` displays a vertical line at a specified `xintercept` value. You do not need to place `xintercept` within the `aes()` function.

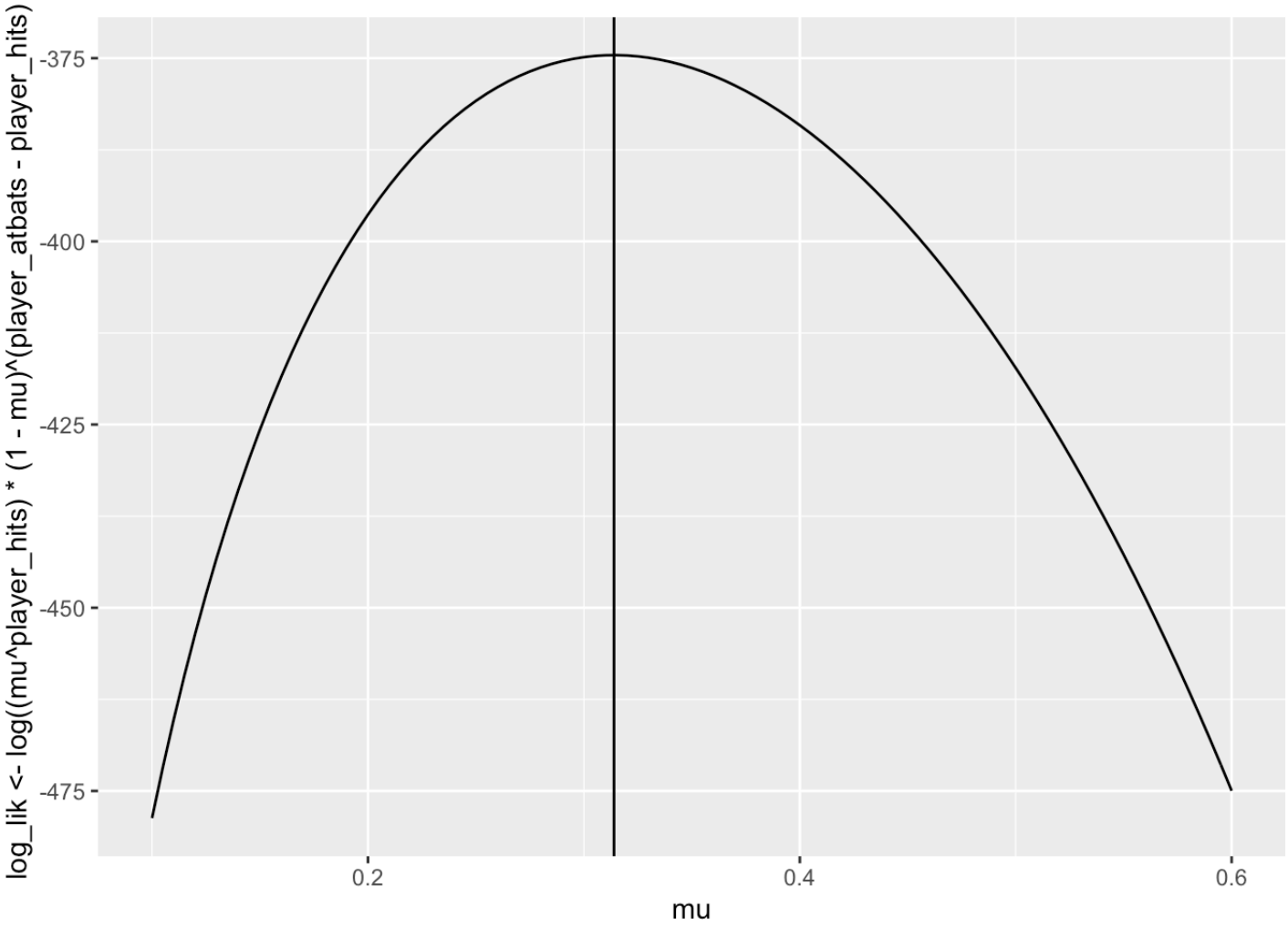
The MLE is corresponds to what on the plotted curve?

HINT: The `seq()` function allows you to create a vector of evenly spaced points from a starting value to an upper value. The `length.out` argument allows specifying the number of points to use.

HINT: Remember that when you pipe data.frames/tibbles you have full access to the variables contained within them.

SOLUTION

```
tibble::tibble(
  mu = seq(0.1,0.6,length.out = 101)
) %>%
  mutate(
    log_lik <- log( (mu ^ player_hits) * (1-mu)^(player_atbats - player_hits))
  )  %>% ggplot(mapping = aes(x = mu, y =  log_lik <- log( (mu ^ player_hits) * (1-mu
)^(player_atbats - player_hits)))) + geom_line() + geom_vline(xintercept = player_mle
)
```



What do you think?
The MLE corresponds to the mode of the curve.

1d)

If we were interested in evaluating the log-likelihood over and over again, it might be tedious to have to rewrite the expression many times. Instead of doing that, we can define a function to streamline the calculation of the un-normalized log-likelihood.

A function in `R` has the basic syntax shown in the code chunk below. The `function()` call is used to assign the newly created function to a variable. Specifically in the example below, the newly created function is named `my_example_function()`. The function receives two input arguments, `input_1` and `input_2`. You can define functions that require zero inputs or many inputs. The actions you want the function to perform are contained within curly braces, `{}`. The last comment states you can use the `return()` function to return a data object from a function. *Alternatively*, the last line evaluated within the function will be returned by default.

```
my_example_function <- function(input_1, input_2)
{
  ### PERFORM ACTIONS

  ### return objects with return()
}
```

To call our newly created function we could use either syntax shown below. If you do not name the input arguments, as in the second example below, by default `R` assumes the defined order for the inputs. Thus, the first call to the function below assumes that `input_1` equals 1 and `input_2` equals 2. It can be good practice to name the inputs when you're starting out to help you get familiar with the syntax.

```
my_example_function(1, 2)
```

```
## NULL
```

```
my_example_function(input_1 = 1, input_2 = 2)
```

```
## NULL
```

Let's now create a function to calculate the un-normalized Binomial log-likelihood. The function `log_binom_pmf_unnorm()` is started for you in the code chunk below. You will use more general names than hits and at bats for the function. The number of events will be denoted as `events` and the number of trials will be referred to as `trials`. The probability of the event will be denoted as `prob`.

Complete the code chunk below by specifying the inputs to `log_binom_pmf_unnorm()` in the following order, `events`, `trials`, and `prob`. Within the function calculate the un-normalized log-likelihood and assign the result to the variable `log_lik`. Return `log_lik` as the result of the function call.

SOLUTION

```
### set the input arguments !!!
log_binom_pmf_unnorm <- function(events, trials, prob)
{
  # calculate log_lik
  log_lik <- log( (prob ^ events) * (1-prob)^(trials - events))

  # return log_lik
  return (log_lik)
}
```

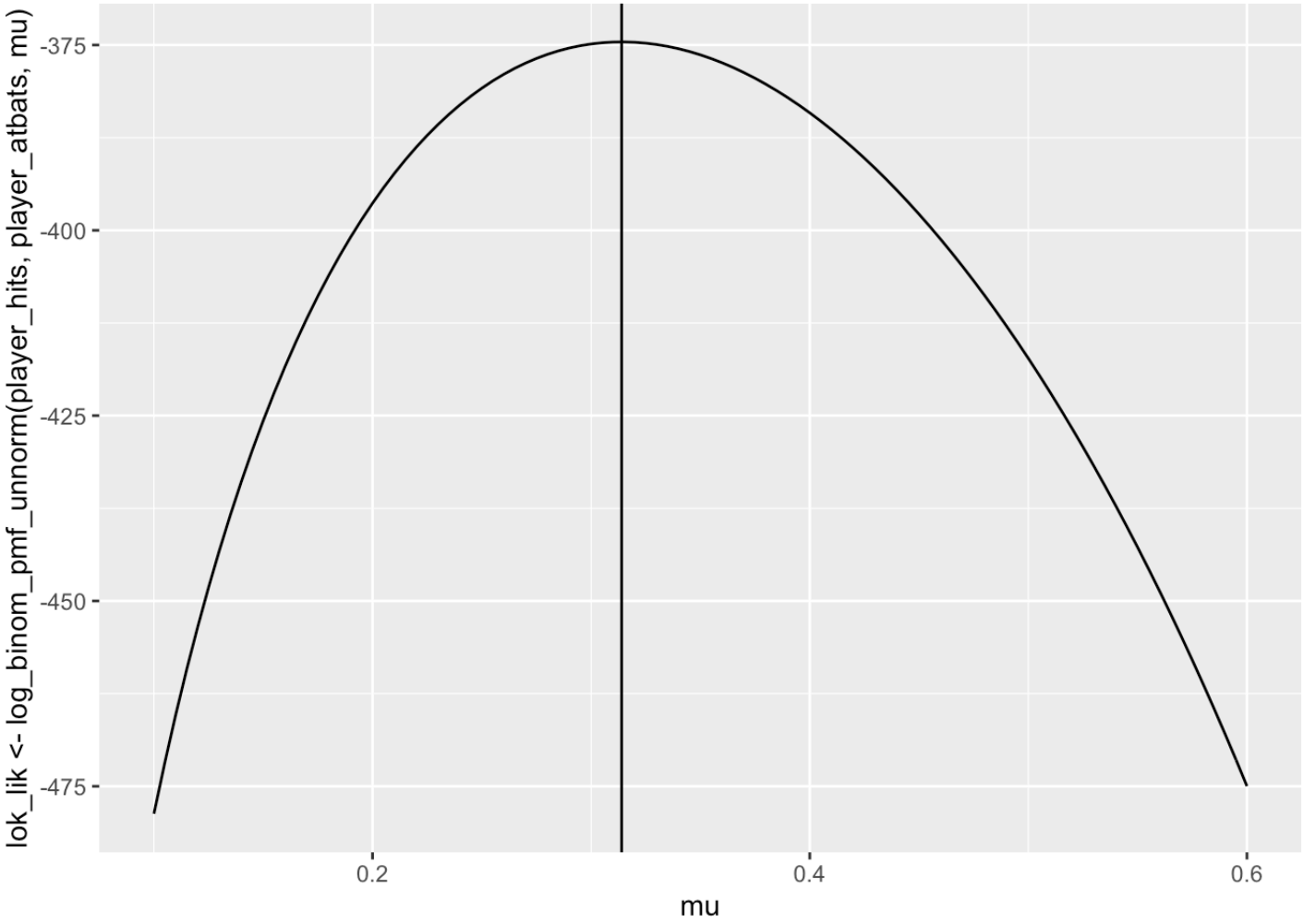
1e)

Let's now use the `log_binom_pmf_unnorm()` function to recreate the figure from Problem 4c).

Recreate the figure from Problem 1c), but this time call the `log_binom_pmf_unnorm()` function rather than typing out the expression in order to calculate the `log_lik` variable. Define the variable `mu` as you did before, as a vector of evenly spaced points between 0.1 and 0.6.

SOLUTION

```
tibble::tibble(
  mu = seq(0.1,0.6,length.out = 101)
) %>%
  mutate(
    lok_lik <- log_binom_pmf_unnorm(player_hits, player_atbats, mu)
  )%>%
      ggplot(mapping = aes(x = mu, y =  lok_lik <- log_binom_pmf_unnorm(player_hits,
player_atbats, mu))) + geom_line() + geom_vline(xintercept = player_mle)
```



1f)

The un-normalized log-likelihood does not include normalizing constant terms. As discussed in lecture, the constant within the Binomial distribution is the Binomial coefficient. In `R` the Binomial coefficient is calculated by the function `choose()`. The input arguments are `n` and `k`, so the function can be read as "`n` choose `k`". There is also a function `lchoose()` which returns the log of the `choose()` function, and so serves as a short cut for writing `log(choose(n,k))`.

The code chunk below defines a function `log_binom_pmf()`. You will complete the function and include the log of the Binomial coefficient with the rest of the terms that you evaluated previously in the `log_binom_pmf_unnorm()` function.

Complete the function `log_binom_pmf()` in the code chunk below. Define the input arguments, `events`, `trials`, and `prob`, in the same order as used in `log_binom_pmf_unnorm()`.

SOLUTION

```
### set the input arguments!
log_binom_pmf <- function(events, trials, prob)
{
  log_lik <-  log((choose(trials, events) ) * (prob ^ events) * (1-prob)^(trials - ev
ents))
  return(log_lik)
}
```

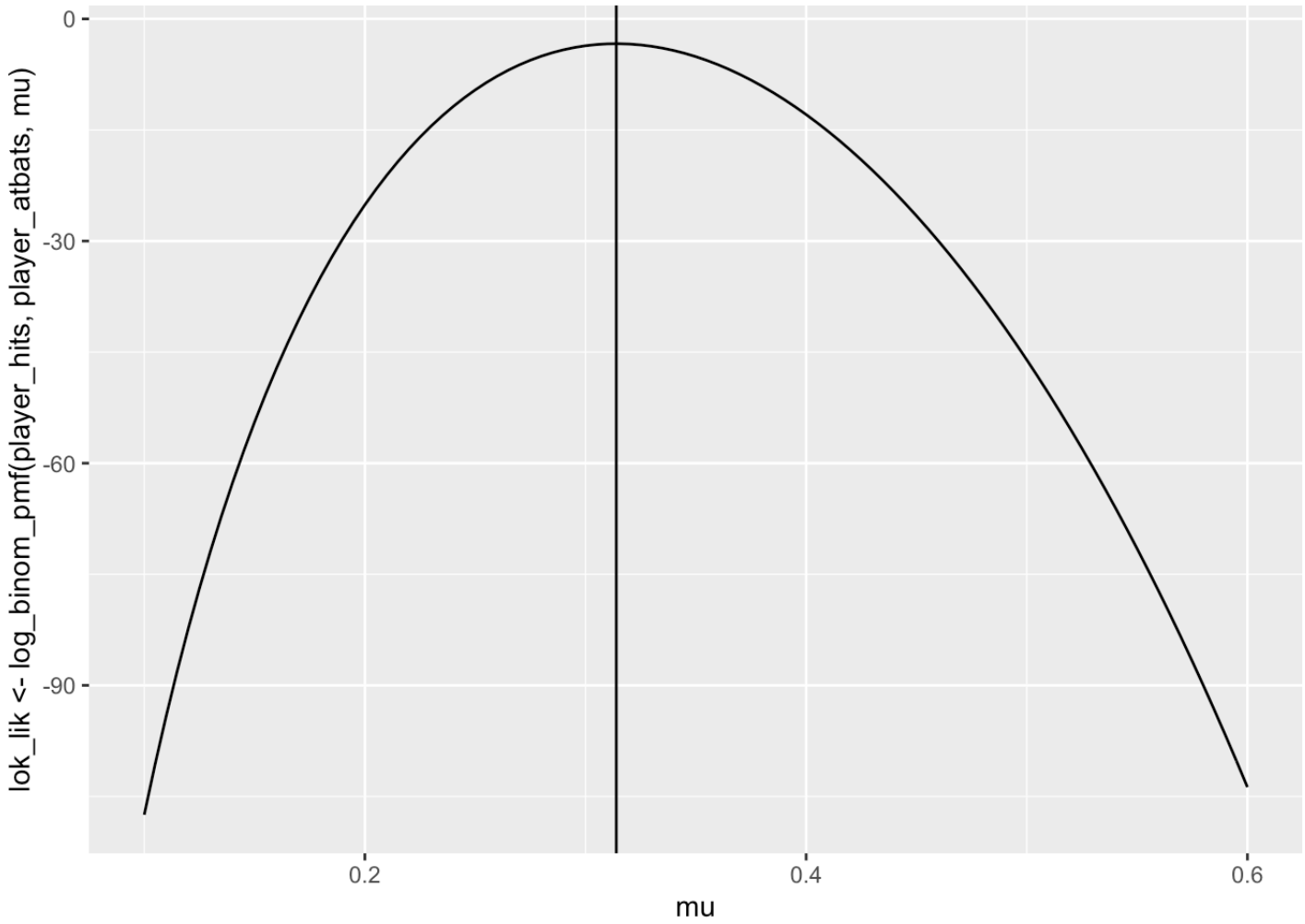
1g)

The un-normalized log-likelihood is all we needed when we were interested in finding the MLE. The constants do **not** impact the shape, because the constants drop out when we evaluate the derivative with respect to μ. To show that is indeed the case, recreate the visualization of the log-likelihood with respect to the probability μ. However, this time set the `log_lik` variable equal to the result of the `log_binom_pmf()` function instead of the un-normalized function.

Recreate the plot from Problem 1e), except set the variable `log_lik` equal to the result of the `log_binom_pmf()` function. Does the MLE correspond to the same location as it did with the un-normalized log-likelihood?

SOLUTION

```
tibble::tibble(
  mu = seq(0.1,0.6,length.out = 101)
) %>%
  mutate(
    lok_lik <- log_binom_pmf(player_hits, player_atbats, mu)
  )%>%
      ggplot(mapping = aes(x = mu, y =  lok_lik <- log_binom_pmf(player_hits, player_
atbats, mu) )) + geom_line() + geom_vline(xintercept = player_mle)
```



What do you think?
Yes, the MLE is still the mode of the curve, the same as the un-normalized log-likelihood.

Problem 02

Although we do not need to worry about normalizing constants when finding the MLE, we do need to include them when we wish to calculate probabilities. We have been working with the log of the Binomial PMF. We can use that PMF to answer questions such as "What is the probability of observing H hits out of AB at bats for a player with a true hit probability of 0.3?" We can therefore get an idea about the likelihood of the data we observed.

2a)

Use the `log_binom_pmf()` function to calculate the probability of observing the 189 hits out of 602 at bats, assuming the true hit probability was 0.3. It is important to note that `log_binom_pmf()` is the log-Binomial. Therefore, you must perform an action to convert from the log-scale back to the original probability scale.

Calculate the probability of observing 189 hits out of 602 at bats if the true probability is 0.3.

SOLUTION

```
prob <- log_binom_pmf(189,602,0.3)
exp(prob)
```

```
## [1] 0.02655379
```

2b)

It may seem like a lot of work in order to evaluate the Binomial distribution. However, you were told to write the function yourself. Luckily in `R` there are many different PMFs and PDFs predefined for you. Unless it is explicitly stated in the homework instructions, you will be allowed to use the predefined PMF and PDF functions throughout the semester.

For the Binomial distribution, the predefined function is `dbinom()`. It contains 4 input arguments: `x`, `size`, `prob`, and `log`. `x` is the number of observed events. `size` is the number of trials, so you can think of `size` as the Trial size. `prob` is the probability of observing the event. `log` is a Boolean, so it equals either `TRUE` or `FALSE`. It is a flag to specify whether to return the log of the probability, `log=TRUE`, or the probability `log=FALSE`. By default, if you do not specify `log` the `dbinom()` function assumes `log=FALSE`.

Check your result from Problem 2a) by using the `dbinom()` function to calculate the probability of observing 189 hits out of 602 at bats, assuming the probability of a hit is 0.3.

```
dbinom(player_hits, player_atbats, 0.3)
```

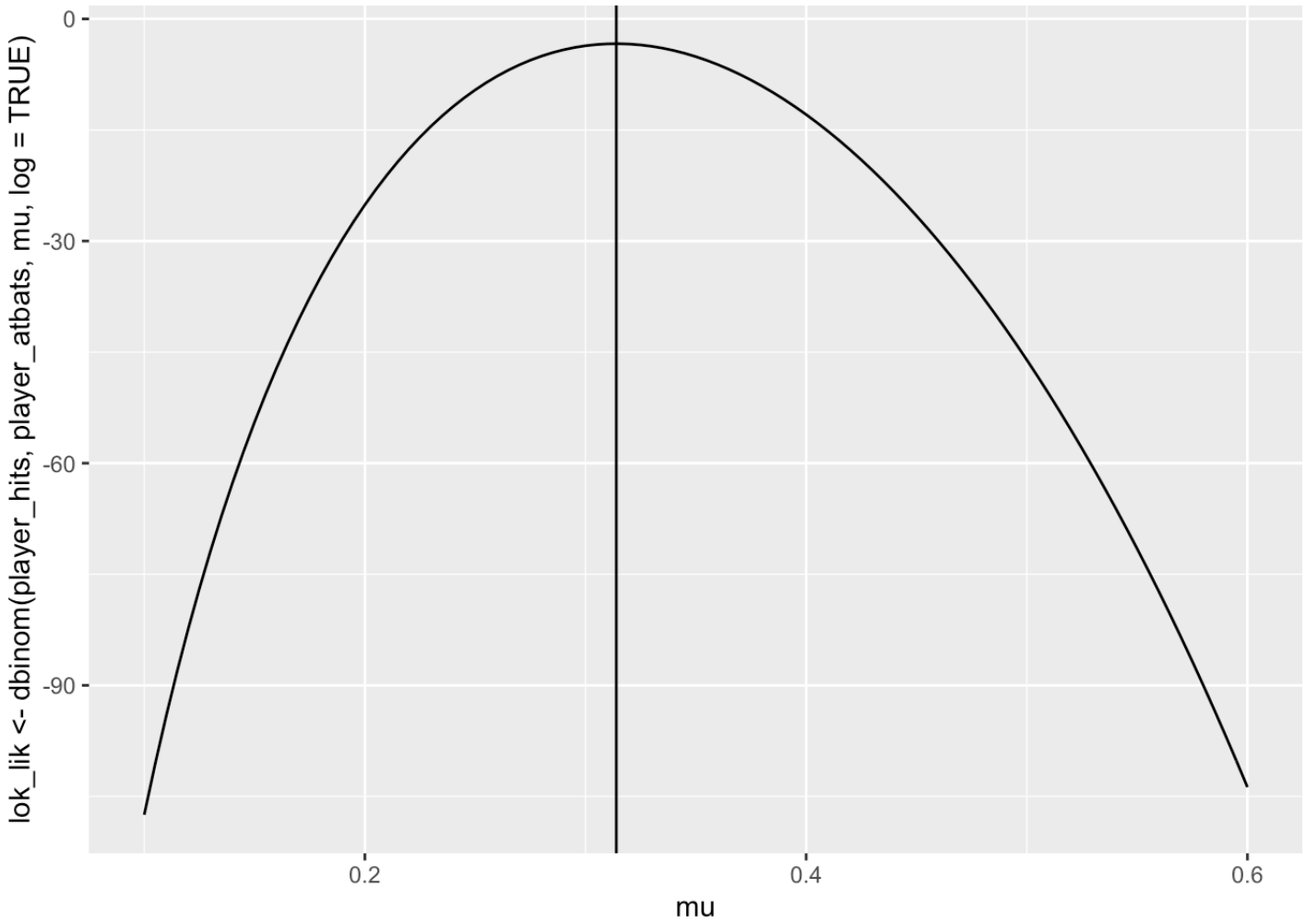
```
## [1] 0.02655379
```

2c)

Recreate the log-likelihood figure from Problem 1g), but this time use the `dbinom()` function instead of the `log_binom_pmf()`. Do not forget to set the `log` flag appropriately!

SOLUTION

```
tibble::tibble(
  mu = seq(0.1,0.6,length.out = 101)
) %>%
  mutate(
    lok_lik <- dbinom(player_hits, player_atbats, mu, log=TRUE)
  )%>%
      ggplot(mapping = aes(x = mu, y =  lok_lik <- dbinom(player_hits, player_atbats,
mu, log=TRUE) )) + geom_line() + geom_vline(xintercept = player_mle)
```



2d)

The `dbinom()` function evaluates the probability of observing **exactly** `x` events out of a `size` of trials. However, what if we were interested in the probability of observing at most a specified number of events? We would need to integrate, or sum, the probabilities of all events up to and including the max number of events.

To see how this works, consider a simple coin flip example. We will flip the coin 11 times. What is the probability of observing at most 5 heads if the probability of heads is 0.25 (so it is a biased coin). To perform this calculation we must first calculate the probability of observing exactly 0, 1, 2, 3, 4, and 5 heads out of 11 trials. The `dbinom()` function accepts vectors for the `x` argument. So all we have to do is pass in a vector from 0 to 5 as the `x` argument.

Calculate the probabilities of observing exactly 0 through 5 heads out of 11 trials assuming the probability of heads is equal to 0.25. Set the `x` argument in `dbinom()` to be a vector of 0 through 5 using the `:` operator. Assign the result to the variable `coin_probs`. Print the result to the screen and check the length of `coin_probs` by using the `length()` function. What does the first element in the `coin_probs` vector correspond to?

SOLUTION

```
coin_probs <- dbinom(0:5, 11, 0.25)

# print to screen
coin_probs
```

```
## [1] 0.04223514 0.15486217 0.25810361 0.25810361 0.17206907 0.08029890
```

```
# length?
length(coin_probs)
```

```
## [1] 6
```

?
The first element corresponds to getting no heads out of the 11 trials.

2e)

The probability of observing at most 5 heads out of 11 trials is then equal to the summation of all of the event probabilities. The `sum()` function will sum up all elements in a vector for us.

Use the `sum()` function to sum all of the elements of the `coin_probs` vector. What is the probability of observing at most, or up to and including, 5 heads out of 11 trials?

SOLUTION

```
sum(coin_probs)
```

```
## [1] 0.9656725
```

Probability of observing at most 5 heads is 0.966. ### 2f)

Alternatively, we can use the `pbinom()` function to perform this summation for us. The arguments are similar to `dbinom()`, except the first argument is referred to as `q` instead of `x`. The `q` argument corresponds to the value we are integrating up to. So in our coin example, we would set `q` to be 5.

Calculate the probability of observing at most, or up to and including, 5 heads out of 11 trials assuming the probability equals 0.25 with the `pbinom()` function. How does your result compare to the "manual" approach using `sum()` and `dbinom()`?

SOLUTION

```
pbinom(5,11,0.25)
```

```
## [1] 0.9656725
```

?
The answer is the same when using the pbinom() function compared to using sum() and dbinom().

2g)

With the `dbinom()` and `pbinom()` functions we can now work through many different types of probabilistic questions. Returning to the baseball example, let's consider the probability of observing between 175 hits and 195 hits out of 602 at bats, assuming the true hit probability is 0.3. We are now interested in the probability of an interval, rather than asking the probability of up to and including.

Calculate the probability of observing 175 to 195 hits out of 602 at bats if the true hit probability is 0.3. Also, calculate the probability of observing 165 to 205 hits out of 602 at bats if the true hit probability is 0.3

HINT: To calculate the probability of observing the number of hits between a specific interval, we have to take the difference of the summations.

SOLUTION

```
prob_175 <- pbinom(175, 602, 0.3)
prob_195 <- pbinom(195,602,0.3)
prob_175_195 <- prob_195 - prob_175
prob_175_195
```

```
## [1] 0.579966
```

```
prob_165 <- pbinom(165, 602, 0.3)
prob_205 <- pbinom(205, 602, 0.3)
prob_165_205 <- prob_205 - prob_165
prob_165_205
```

```
## [1] 0.8970417
```

Problem 3

The previous two questions focused on baseball. Moving forward in this assignment you will consider a generic situation of observed m events out of N trials. In this question, you will work through combining the Binomial likelihood with a Beta prior on the unknown event probability, μ.

3a)

You previously wrote out the un-normalized log-Binomial likelihood in terms of the baseball example. You will rewrite that expression, but this time with the generic variables for the number of events m out of a generic number of trials N.

Write out the un-normalized log-likelihood for the Binomial likelihood with m events out of N trials and unknown event probability μ.

SOLUTION

The equation block is started for you below.

$$\log(p\,(m \mid N, \mu)) \propto \log(\,\mu^m * (1-\mu)^{N-m}))$$

3b)

Write out the un-normalized log-density of the Beta distribution on the unknown event probability μ with hyperparameters a and b.

SOLUTION

The equation block is started for you below.

$$\log(p\,(\mu \mid a, b)) \propto \log(\mu^{a-1} * (1-\mu)^{b-1})$$

3c)

We already know that since the Beta is conjugate to the Binomial, the posterior distribution on the unknown event probability μ is also a Beta. You will practice working through the derivation of the updated hyperparameters a_{new} and b_{new}. The log-likelihood was written in Problem 3a) and the log-prior in Problem 3b). In this problem you must add the un-normalized log-likelihood to the un-normalized log-prior, then perform some algebra to derive a_{new} and b_{new}.

Derive the expressions for the updated or posterior Beta hyperparameters. You must show all steps in the derivation. You are allowed to use multiple equation blocks if that's easier for you to type with.

SOLUTION

Write out your derivation below. An equation block is started for you, but you can add as many as you feel are necessary.

$$ (^{a-1}*(1-)^{b-1})* + (^m (1-)^{N-m})) \ ()^{a-1} + ((1 -))^{b-1} + ()^m + (1-)^{N-m} \ ()^{a+m-1} + (1-)^{b+(N-m)-1}\ a_{new} -1 = a+m-1 ==> a_{new} = a+m \ b_{new} -1 = b + (N-m) -1 ==> b_{new} = b + (N-m)

$$

3d)

Since the posterior distribution on μ is a Beta, a formula exists for the posterior mode (Max a-posterior estimate). However, you will practice deriving the posterior mode through differentiation of the un-normalized log-posterior. You can always double check your answer with the known formula for the mode of a Beta!

Derive the derivative of the un-normalized log-posterior with respect to the unknown event probability μ. Write out the derivative in terms of the updated hyperparmaeters a_{new} and b_{new}.

SOLUTION

You may add as many equation blocks as you feel are necessary. One is started for you below.

$$ ()^{a+m} + (1-)^{b+(N-m)} \ = (a+m)*()^{a+m-1} \

1 - = x –> (1-) = (x) = f \ = * = ((x)) * (1-) = * -1 = \

(a+m)*()^{a+m-1} + (b+(N-m)) ^{b+(N-m)-1} \ + - +

$$

3e)

Set the derivative from your solution to Problem 3d) equal to zero and solve for the posterior mode of the unknown event probability. Denote the posterior mode as μ_{MAP}.

SOLUTION

You may add as many equation blocks as you feel are necessary. One is started for you below.

$$\frac{a}{\mu} + \frac{m}{\mu} - \frac{b}{1-\mu} + \frac{m-N}{1-\mu} = 0$$

$$\frac{a}{\mu} + \frac{m}{\mu} = \frac{b}{1-\mu} - \frac{m-N}{1-\mu}$$

$$a + m = \frac{\mu}{1-\mu}(b - (m-N))$$

$$\mu_{MAP} = \frac{\mu}{1-\mu} = \frac{a+m}{b-(m-N)}$$

Problem 4

Now that you have worked through the equations, it's time to study the behavior of the posterior under various conditions and assumptions. Specifically, you will examine how the posterior changes when the sample size is small, medium, and large using two different prior specifications on the unknown event probability.

The data are provided as the number of observed events, m, out of a specified number of trials, N. The small data set consists of $m = 0$ events out of $N = 7$ trials. The "medium" data set consists of $m = 9$ events out of $N = 100$ trials. The "big" data set contains $m = 103$ events out of $N = 1000$ trials. The data are assigned to the variables in the code chunk below.

```
m_small <- 0
N_small <- 7

m_medium <- 9
N_medium <- 100

m_big <- 103
N_big <- 1000
```

Two types of priors are considered for you to study. Both are Beta distributions, but one is considered to be "vague" less informative than the other more "informative" prior. The Beta priors are defined by their shape parameters, a and b. The shape parameters are provided for you in the code chunk below. The variable names state which prior type the shape parameters belong to.

```
a_vague <- 1
b_vague <- 1

a_inform <- 4.28
b_inform <- 48.63
```

This question is purposely open ended to also examine your programming style.

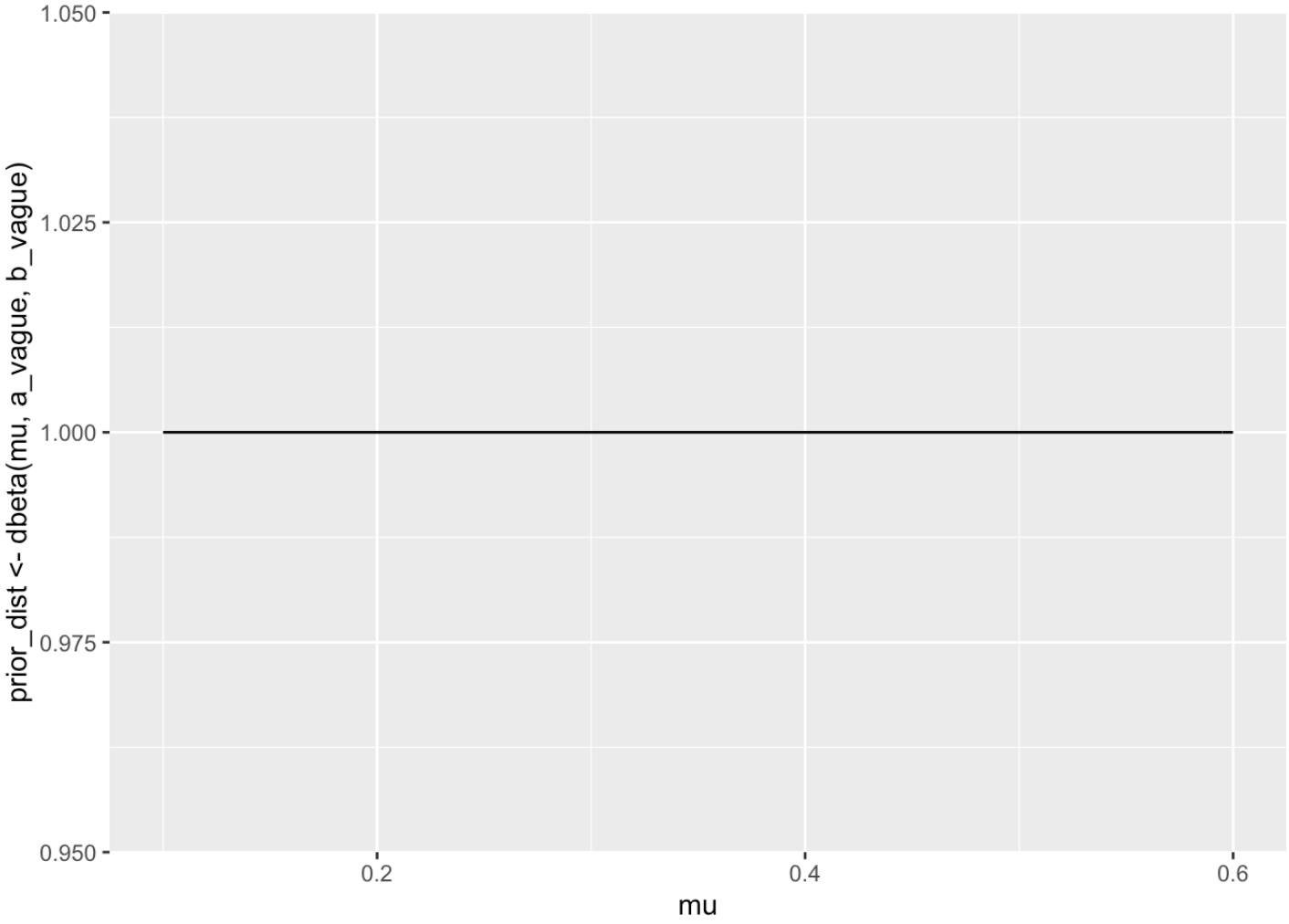
4a)

Plot the two prior distributions on the unknown event probability μ. The Beta pdf can be evaluated with the `dbeta()` function. You must plot the prior with `ggplot2`. The code chunk below is started for you.

SOLUTION

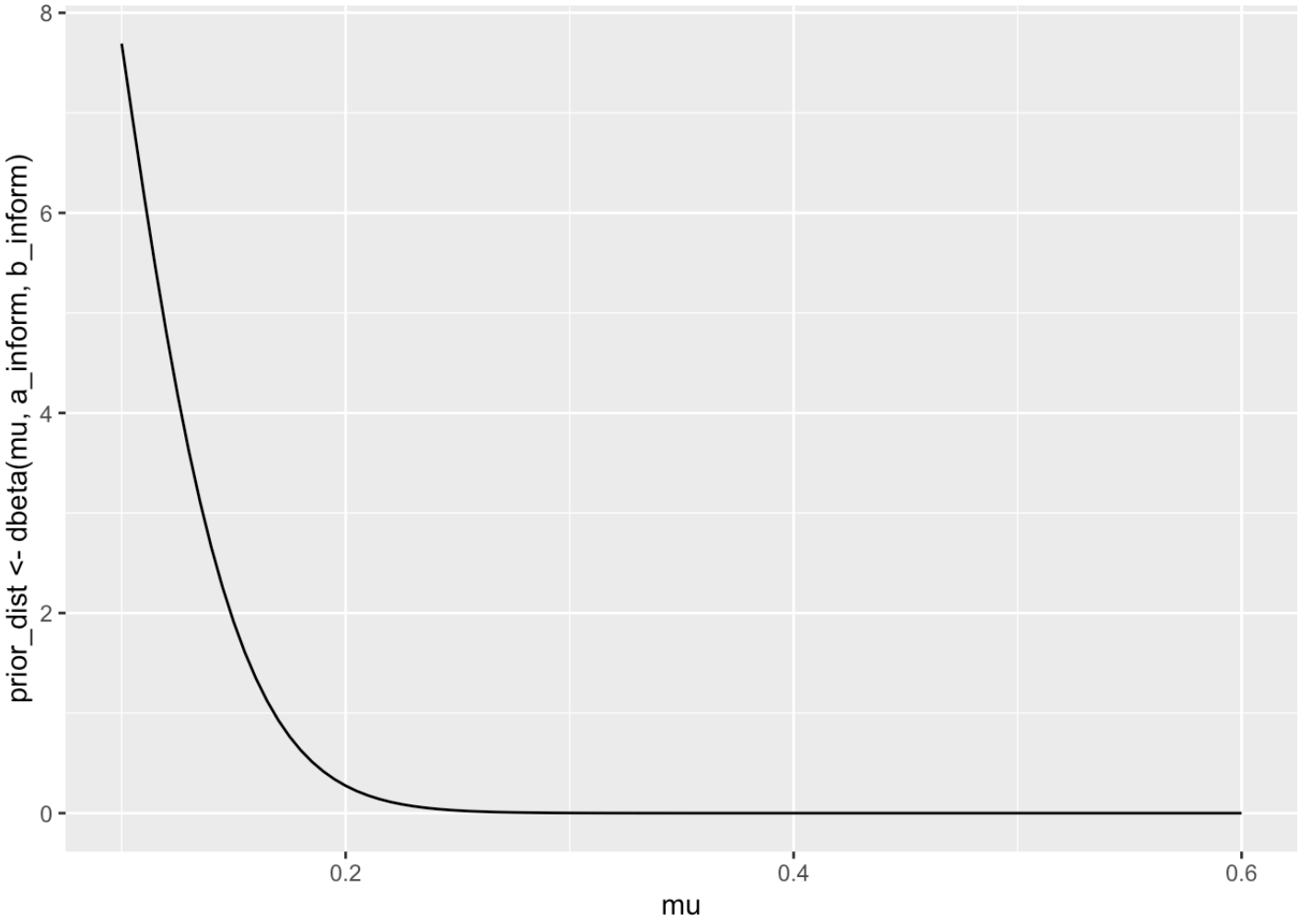
Plot the "vague" prior on μ.

```
tibble::tibble(
  mu = seq(0.1,0.6,length.out = 101)
) %>%
  mutate(
    prior_dist <- dbeta(mu, a_vague, b_vague)
  ) %>%
ggplot(mapping = aes(x = mu, y = prior_dist <- dbeta(mu, a_vague, b_vague)) ) + geom_
line()
```



Plot the informative prior on μ.

```
tibble::tibble(
  mu = seq(0.1,0.6,length.out = 101)
) %>%
  mutate(
    prior_dist <- dbeta(mu, a_inform, b_inform)
  ) %>%
ggplot(mapping = aes(x = mu, y = prior_dist <- dbeta(mu, a_inform, b_inform)) ) + geo
m_line()
```



4b)

There are 6 combinations in total between the 3 sample sizes and 2 prior types.

Calculate the updated or posterior hyperparameters, a_{new} and b_{new}, for each of the 6 combinations of prior type and sample size. Display the posterior hyperparameters as a table.

SOLUTION

You may use as many code chunks as you would like. A code chunk is started for you below.

```
vague_small_a <- a_vague + m_small
vague_small_b <- b_vague + (N_small - m_small)
vague_medium_a <- a_vague + m_medium
vague_medium_b <- b_vague + (N_medium - m_medium)
vague_big_a <- a_vague + m_big
vague_big_b <- b_vague + (N_big - m_big)
```

```
inform_small_a <- a_inform + m_small
inform_small_b <- b_inform + (N_small - m_small)
inform_medium_a <- a_inform + m_medium
inform_medium_b <- b_inform + (N_medium - m_medium)
inform_big_a <- a_inform + m_big
inform_big_b <- b_inform + (N_big - m_big)
```

```
hyperparameters <- data.frame("a" = c(vague_small_a, vague_medium_a, vague_big_a, inf
orm_small_a, inform_medium_a, inform_big_a),
                              "b" = c(vague_small_b,vague_medium_b, vague_big_b, info
rm_small_b,  inform_medium_b,  inform_big_b))
rownames(hyperparameters) <- c("vague_small", "vague_medium", "vague_big", "inform_sm
all", "inform_medium", "inform_big")
hyperparameters
```

```
##                     a      b
## vague_small      1.00   8.00
## vague_medium    10.00  92.00
## vague_big      104.00 898.00
## inform_small     4.28  55.63
## inform_medium   13.28 139.63
## inform_big     107.28 945.63
```

4c)

Calculate the posterior mean, mode, 5th percential (0.05 quantile), and 95th percentile (0.95 quantile) for each of the 6 combinations of prior type and sample size. You should use your results from Problem 2b) for the updated or posterior beta hyperparameters.

Display your results in a table.

NOTE: The `qbeta()` function allows calculating the quantiles associated with a particular probability of interest.

SOLUTION

You may use as many code chunks as you would like. A code chunk is started for you below.

```
mean_vague_s <-  (hyperparameters$a[1]) / (hyperparameters$a[1] +  hyperparameters$b[
1])
mode_vague_s <- (hyperparameters$a[1] -1) / (hyperparameters$a[1] +  hyperparameters$
b[1] -2)
vague_s_0.05 <- qbeta(0.05, hyperparameters$a[1], hyperparameters$b[1])
vague_s_0.95 <- qbeta(0.95, hyperparameters$a[1], hyperparameters$b[1])

mean_vague_m <-  (hyperparameters$a[2]) / (hyperparameters$a[2] +  hyperparameters$b[
2])
mode_vague_m<- (hyperparameters$a[2] -1) / (hyperparameters$a[2] +  hyperparameters$b
[2] -2)
vague_m_0.05 <- qbeta(0.05, hyperparameters$a[2], hyperparameters$b[2])
vague_m_0.95 <- qbeta(0.95, hyperparameters$a[2], hyperparameters$b[2])

mean_vague_b <-  (hyperparameters$a[3]) / (hyperparameters$a[3] +  hyperparameters$b[
3])
mode_vague_b<- (hyperparameters$a[3] -1) / (hyperparameters$a[3] +  hyperparameters$b
[3] -2)
vague_b_0.05 <- qbeta(0.05, hyperparameters$a[3], hyperparameters$b[3])
vague_b_0.95 <- qbeta(0.95, hyperparameters$a[3], hyperparameters$b[3])
```

```
mean_inform_s <-  (hyperparameters$a[4]) / (hyperparameters$a[4] +  hyperparameters$b
[1])
mode_inform_s <- (hyperparameters$a[4] -1) / (hyperparameters$a[4] +  hyperparameters
$b[1] -2)
inform_s_0.05 <- qbeta(0.05, hyperparameters$a[4], hyperparameters$b[4])
inform_s_0.95 <- qbeta(0.95, hyperparameters$a[4], hyperparameters$b[4])

mean_inform_m <-  (hyperparameters$a[5]) / (hyperparameters$a[5] +  hyperparameters$b
[5])
mode_inform_m<- (hyperparameters$a[5] -1) / (hyperparameters$a[5] +  hyperparameters$
b[5] -2)
inform_m_0.05 <- qbeta(0.05, hyperparameters$a[5], hyperparameters$b[5])
inform_m_0.95 <- qbeta(0.95, hyperparameters$a[5], hyperparameters$b[5])

mean_inform_b <-  (hyperparameters$a[6]) / (hyperparameters$a[6] +  hyperparameters$b
[6])
mode_inform_b<- (hyperparameters$a[6] -1) / (hyperparameters$a[6] +  hyperparameters$
b[6] -2)
inform_b_0.05 <- qbeta(0.05, hyperparameters$a[6], hyperparameters$b[6])
inform_b_0.95 <- qbeta(0.95, hyperparameters$a[6], hyperparameters$b[6])
```

```
summary <- data.frame("mean" = c(mean_vague_s, mean_vague_m,mean_vague_b, mean_inform
_s,mean_inform_m, mean_inform_b ),
                      "mode" = c(mode_vague_s, mode_vague_m, mode_vague_b,mode_inform
_s, mode_inform_m, mode_inform_b),
                      "0.05 quantile" = c(vague_s_0.05, vague_m_0.05, vague_b_0.05,in
form_s_0.05, inform_m_0.05, inform_b_0.05),
                      "0.95 quantile" = c(vague_s_0.95, vague_m_0.95, vague_b_0.95,in
form_s_0.95,inform_m_0.95,inform_b_0.95 ))
rownames(summary) <- c("vague_small", "vague_medium", "vague_big", "inform_small", "i
nform_medium", "inform_big")

summary
```

```
##                      mean      mode X0.05.quantile X0.95.quantile
## vague_small   0.11111111 0.0000000    0.006391151      0.3123440
## vague_medium  0.09803922 0.0900000    0.054705780      0.1503440
## vague_big     0.10379242 0.1030000    0.088413339      0.1200708
## inform_small  0.34853420 0.3190661    0.026381066      0.1327230
## inform_medium 0.08684847 0.0813730    0.052816042      0.1270271
## inform_big    0.10188905 0.1011314    0.087003093      0.1176350
```

4d)

The summary statistics calculated in Problem 4c) (the mean, mode, 5th percentile, and 95th percentile) provide summaries about the central tendency and the dispersion (uncertainty) in the unknown event probability.

Discuss the behavior of the summary statistics between the two prior assumptions (vague vs informative) when the data set is small compared to “big”.

SOLUTION

Your discussion here.
When the data set is small and the prior assumptions are vague, there appears to be the highest dispersion in the unknown event probability. This is most likely due to the fact that there is little data and with vague assumptions, the summary statistics do not capture the true central tendency or dispersion. However, when the prior assumption is informative, the mean and mode are much closer together, depicting a much more normal distribution. It appears that for the big dataset the informative and vague summary statistics depict the same amount of uncertaintiy.