Overview

This assignment is focused on the mathematics of likelihoods, priors, and posteriors. You will work with binomially distributed data in this assignment. You must perform calculations in R using for-loops, functions, and visualize your results using `ggplot2`. You must also perform derivations and type your expressions in LaTeX within equation blocks.

IMPORTANT!!!

Certain 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.

You are free to add more code chunks if you would like.

Load packages

You will ONLY use functions from the `tidyverse` in this assignment.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

Problem 01

Baseball has a rich history of quantitative analysis, even before the rise of advanced analytics techniques. Batting averages, slugging percentages, and other metrics have been used to evaluate player performance for over one hundred years. The batting average, BA, is calculated using the number of at bats, AB, and the successful number of hits, H. The batting average measures the proportion of at bats that a player successfully gets a hit. You can think of the number of hits as the number of **events** and the number of at bats as the number of **trials**.

You will work with a sequence of at bats of a real Major League Baseball (MLB) player. This sequence is a small sample from the 2022 MLB season. You are not told who this player is so that way you cannot know for certain if the player is a "good" or "bad" hitter! All you are provided with is the small sample size provided below.

The code chunk below populates the data using the encoding and format discussed in lecture. Each observation (element) of the vector `x` corresponds to an individual at bat. The result, hit or out, is recorded as 1 for hit (**event**) and 0 for out (**non-event**).

```
x <- c(0, 0, 0, 0,
       0, 1, 1, 0,
       1, 0, 0, 0,
       0, 1, 1, 0, 1,
       1, 0, 0, 1,
       0, 0, 1, 0,
       1, 1, 1, 0, 0,
       0, 0, 1, 1)
```

1a)

Calculate the average of the `x` vector.

Display the result to the screen.

SOLUTION Insert code chunks to answer the question.

```
aver_x <- mean(x)
print(aver_x)
```

```
## [1] 0.4117647
```

1b)

The player's batting results, hit or out, is a **binary outcome** which we will assume is a **Bernoulli** random variable. The likelihood function for each at bat (observation) is therefore the Bernoulli distribution. We will also assume the at bats are **independent**. The Bernoulli distribution consists of a single unknown parameter, μ, the **event probability**. In the context of this application, the event probability represents the probability the player gets a hit.

Having collected the player's data, you are tasked with estimating the player's hit probability. In lecture, we derived the Maximum Likelihood Estimate (MLE) for μ.

Without going through any mathematical derivations, what is the MLE for this player's hit probability, based on the data provided?

SOLUTION Insert code chunks to answer the question.

```
count_of_ones <- sum(x == 1)
length_of_x <- length(x)
print(count_of_ones/length_of_x)
```

```
## [1] 0.4117647
```

1c)

How does your result to 1a) compare to the result in 1b)?

SOLUTION What do you think?

The results in 1a) and 1b) are equal to each other. In 1b), we do (count of events)/(total number of trials). However, since we represent the event as x=1 and non event as x=0, the result in 1a) also gives the same value.

1d)

Let's now dive into the Bernoulli distribution in greater detail.

Write the natural log of the Bernoulli distribution for a single at bat (observation) x_n given the hit (event) probability, μ.

You MUST include at least several steps which simplify the expression using the properties of the natural log to receive full credit.

SOLUTION Let x_n represent a single observation and μ represent the event probability. Then we can express the following:

$$p(x_n|\mu) = (\mu)^{x_n}(1-\mu)^{1-x_n}$$

$$\Rightarrow \log[p(x_n|\mu)] = \log[(\mu)^{x_n}(1-\mu)^{1-x_n}]$$
$$\Rightarrow \log[p(x_n|\mu)] = x_n \log[\mu] + (1-x_n)\log[1-\mu]$$

which gives the desired result.

1e)

The `log_bernoulli_pmf()` function is started for you in the code chunk below. It consists of two input arguments, `xobs`, and `prob`. The `xobs` argument is a numeric vector of observations of a binary variable encoded as 0 and 1. The `prob` argument is the event probability.

Complete the code chunk below by correctly calculating the log of the Bernoulli PMF. The function must return a numeric vector the same length as the `xobs` argument.

```
log_bernoulli_pmf <- function(xobs, prob)
{
  log_pmf <- numeric(length(xobs))

  for (i in 1:length(xobs)) {
    if (xobs[i] == 1) {
      log_pmf[i] <- log(prob)
    } else {
      log_pmf[i] <- log(1 - prob)
    }
  }
  return(log_pmf)
}
```

SOLUTION

1f)

Let's check the operation of your `log_bernoulli_pmf()` function.

Use separate function calls to the `log_bernoulli_pmf()` function to calculate the values associated with the 1st, 2nd, 3rd, 4th, 5th, and 6th observations of the `x` vector. Therefore, you must provide a single `x` observation to the function and do so 6 times.

Use an event probability equal to 0.250 in each function call.

Display the results to the screen.

```
log_pmf1 <- log_bernoulli_pmf(x[1], 0.250)
log_pmf2 <- log_bernoulli_pmf(x[2], 0.250)
log_pmf3 <- log_bernoulli_pmf(x[3], 0.250)
log_pmf4 <- log_bernoulli_pmf(x[4], 0.250)
log_pmf5 <- log_bernoulli_pmf(x[5], 0.250)
log_pmf6 <- log_bernoulli_pmf(x[6], 0.250)
```

SOLUTION Here are the values associated with the 1st, 2nd, 3rd, 4th, 5th, and 6th observations of the `x` vector.

```
log_pmf1
```

```
## [1] -0.2876821
```

```
log_pmf2
```

```
## [1] -0.2876821
```

```
log_pmf3
```

```
## [1] -0.2876821
```

```
log_pmf4
```

```
## [1] -0.2876821
```

```
log_pmf5
```

```
## [1] -0.2876821
```

```
log_pmf6
```

```
## [1] -1.386294
```

1g)

The previous question focused on testing the function for a single observation. Let's now check the function works when multiple observations are provided.

Pass the first 6 elements of the `x` vector into the `log_bernoulli_pmf()` function. You must still use an event probability equal to 0.250.

Display the result to the screen.

What is the length of the returned result? How do the values compare to the previous question where you called the function separately for each observation?

SOLUTION Insert code chunks.

```
log_probs <- log_bernoulli_pmf(x[1:6], 0.250)
log_probs
```

```
## [1] -0.2876821 -0.2876821 -0.2876821 -0.2876821 -0.2876821 -1.3862944
```

The length of the returned vector is 6 and the individual values of the components are the same as in the previous question.

Problem 02

Now that you have practiced calculating the log Bernoulli PMF, let's consider the **joint distribution** of all observations. Remember that the joint distribution is the **likelihood function** for this application. It corresponds to the probability of the exact sequence of observed data given the assumptions. The likelihood of all N observations is written in vector notation for you in the equation block below:

$$p\left(x_1, x_2, ..., x_n, , ..., x_{N-1}, x_N \mid \mu\right) = p\left(\mathbf{x} \mid \mu\right)$$

2a)

Write the expression for the "complete" log-likelihood assuming the observations are independent.

SOLUTION We can derive the expression as follows,

$$p(\mathbf{x} \mid \mu) = \prod_{n=1}^{N} \left(\mu^{x_n}(1-\mu)^{1-x_n}\right)$$

$$\Rightarrow \log\left[p(\mathbf{x} \mid \mu)\right] = \log\left[\prod_{n=1}^{N} \left(\mu^{x_n}(1-\mu)^{1-x_n}\right)\right] = \sum_{n=1}^{N}\left(\log[\mu^{x_n}] + \log[(1-\mu)^{(1-x_n)}]\right)$$

$$= \log\mu \sum_{n=1}^{N}(x_n) + \log[1-\mu]\sum_{n=1}^{N}(1-x_n) = \log[\mu] \times M + \log[1-\mu] \times (N-M)$$

where M is the number of "events" and N is the total observation count.

2b)

Calculate the "complete" log-likelihood for all observations in x. You must continue to use an event probability of 0.25.

Display the result to the screen.

```r
cmplt_log_likelihood <- sum(log_bernoulli_pmf(x,0.25))
cmplt_log_likelihood
```

SOLUTION

```
## [1] -25.16176
```

2c)

You might be wondering, why are you using probabilities of 0.25 when you already calculated the MLE at the beginning of the assignment? We derived the MLE in lecture, but you will graphically find the MLE in this assignment. You must therefore calculate the log-likelihood for many candidate event probability values, graph the results, and visually identify the probability that maximizes the likelihood.

You will do this to reinforce optimization concepts such as why the MLE corresponds to the first derivative equal to zero. This exercise will also introduce visualizing curvature, an important concept we will discuss in greater depth later. Thus, the next few questions are laying the foundation for more complicated optimization problems such as training neural networks.

You must call the `log_bernoulli_pmf()` function for many potential probability **candidate values**. You will use for-loops to accomplish the iteration procedure. A for-loop is not the most efficient approach to accomplishing this task. We will see more efficient methods soon. For now, the basic for-loop will demonstrate the key concepts.

However, we need to setup the book keeping before we can iterate. We need the candidate probability values defined before anything else can happen.

Define a candidate grid of event probability values as a numeric vector using the `seq()` function. Specify the `from` argument to be 0.025 and the `to` argument to be `0.975`. Create the vector such that 251 evenly spaced values are between the bounds.

Assign the vector the `mu_grid` object.

```r
mu_grid <- seq(from=0.025, to=0.975,length.out=251)
```

SOLUTION

2d)

The basic structure of a for-loop in `R` is shown in the code chunk below. This simple for-loop simply prints the value of the *iterating variable* `n` to the screen. The `for` keyword is used to begin the for-loop. We must specify the iterating variable and the **sequence** the iterating variable is **in** within parentheses, `()`. The sequence in the example below is a vector starting at 1 and ending at 4.

```r
for( n in 1:4 ){
  print( n )
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
```

When we wish to calculate values and store them as elements within a larger object within a for-loop, it is best to first *initialize* the object with the appropriate size. This variable `example_vector` is initialized with `NA` (missing values) using the `rep()` function 10 times. Notice that the data type conversion function `as.numeric()` is used to ensure the initialized object is numeric.

```r
example_vector <- rep( as.numeric(NA), 10 )

example_vector %>% class()
```

```
## [1] "numeric"
```

To confirm the `example_vector` object contains only missing values.

```r
example_vector
```

```
##  [1] NA NA NA NA NA NA NA NA NA NA
```

We can create a sequence of integers from 1 to the length of `example_vector` using the `seq_along()` function, as shown below. The `seq_along()` function is a useful programmatic approach to creating a vector of integers useful for iteration.

```r
seq_along(example_vector)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

We can now iterate the elements of `example_vector` and populate the elements as desired. The simple example shown below simply sets each element of `example_vector` equal to the square of the element index.

```r
for( n in seq_along(example_vector) ){
  example_vector[n] <- n ^ 2
}
```

The `example_vector` object is displayed below to show it no longer contains missings.

```r
example_vector
```

```
##  [1]   1   4   9  16  25  36  49  64  81 100
```

Obviously this simple example does not require a for-loop. We could have reached the same result by doing the following:

```r
(1:10)^2
```

```
##  [1]   1   4   9  16  25  36  49  64  81 100
```

However, the point was to demonstrate the key ingredients of populating elements of an object within a for-loop. We must:

- initialize the object to the appropriate size
- iterate over the sequence of elements in the object
- perform the necessary calculation and assign result to the object's element

You will follow the above steps in order to calculate the log-likelihood associated with the `x` vector for all candidate event probability values contained in the `mu_grid` vector. You must assign the result to the `log_lik_xa` object and that object must have the same length as `mu_grid`.

You will still assume that all observations are independent.

NOTE: You must use a for-loop for this problem. We will make use of **functional programming** techniques to streamline this calculation later in the semester.

```r
log_lik_xa <- rep( as.numeric(NA), length(mu_grid) )

for( n in seq_along(log_lik_xa) ){
  log_lik_xa[n] <- sum(log_bernoulli_pmf(x, mu_grid[n]))
}
```

SOLUTION

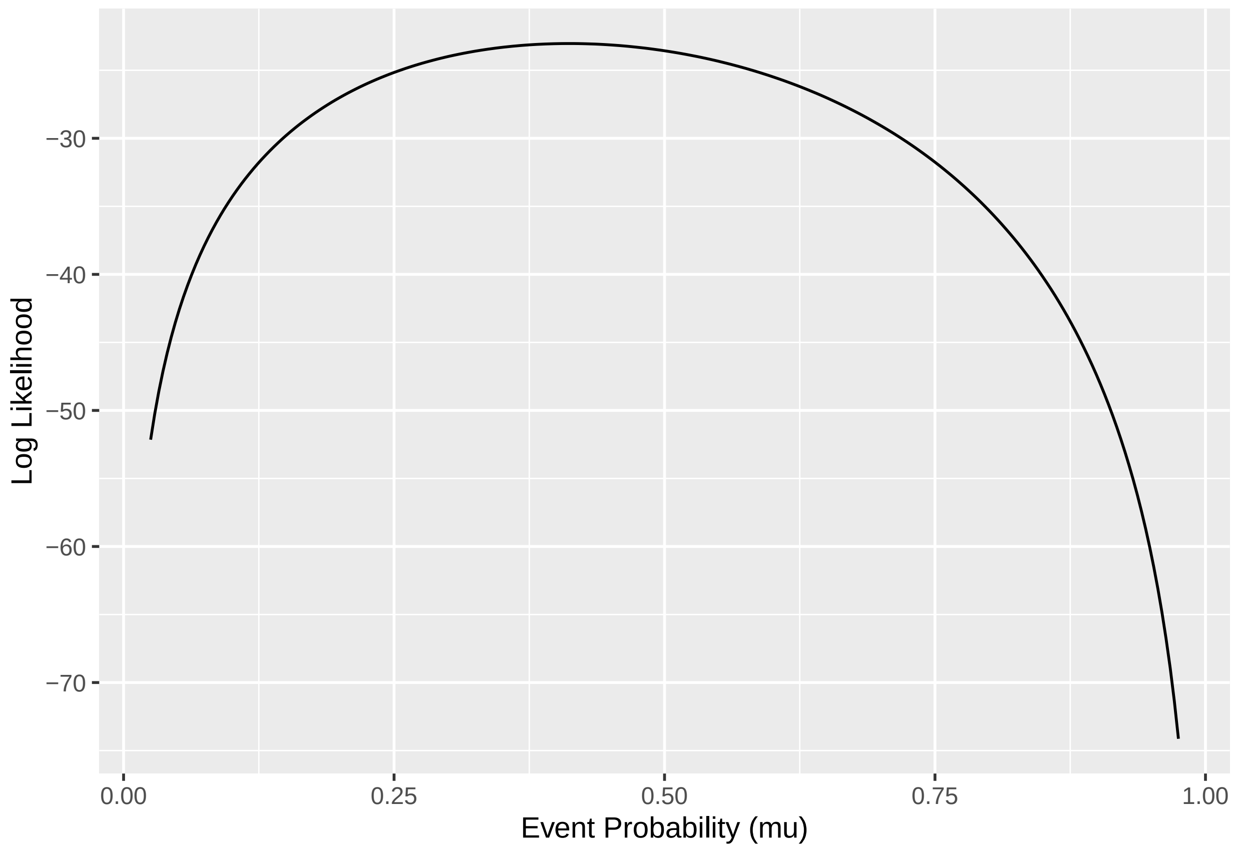
2e)

The code chunk below is completed for you. It assigns the `mu_grid` and `log_lik_xa` vectors as data variables (columns) within a tibble, `xa_results`. The code chunk below is not evaluated by default.

```r
xa_results <- tibble::tibble(
  mu = mu_grid,
  log_lik = log_lik_xa
)
```

Plot the log-likelihood with respect to the event probability using a line plot with `ggplot2`. The line should be created with the `geom_line()` function.

```r
ggplot(xa_results, aes(x=mu, y=log_lik)) +
  geom_line() +
  labs(x = "Event Probability (mu)", y = "Log Likelihood")
```

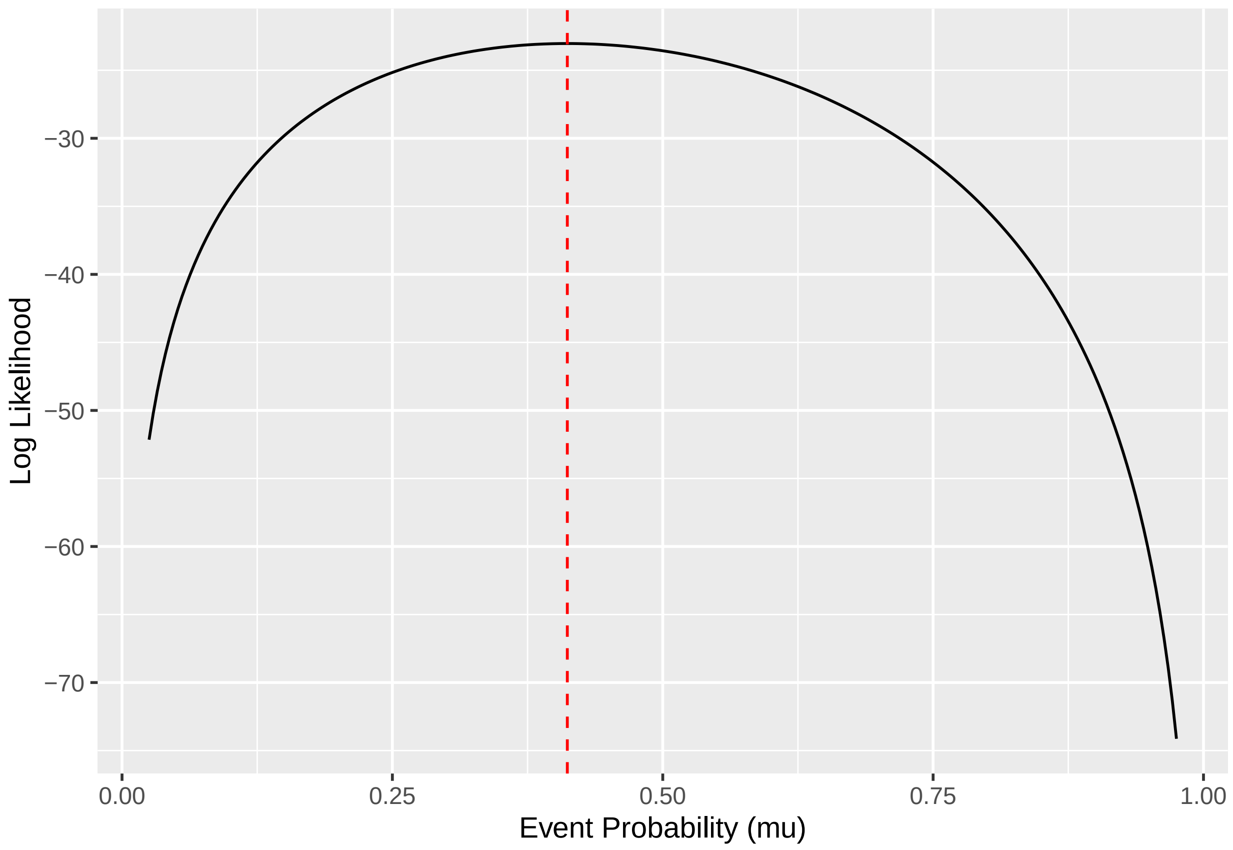


SOLUTION

2f)

Create the same line plot as in the previous question, but add an additional layer with `geom_vline()` to show a vertical reference line. Set the `xintercept` argument within `geom_vline()` to the MLE you calculated in 1b).

```r
ggplot(xa_results, aes(x=mu, y=log_lik)) +
  geom_line() +
  geom_vline(xintercept = sum(x == 1)/length(x), linetype="dashed", color="red") +
  labs(x = "Event Probability (mu)", y = "Log Likelihood")
```



SOLUTION

2g)

Describe the behavior of the log-likelihood with respect to the candidate event probability around the MLE in the plot shown in 2f). Does the curve look different near the MLE compared to other candidate values?

SOLUTION Yes, log-likelihood seems to get the maximum value around MLE, which is expected because it is the maximum value of the curve.

Problem 03

Let's now consider tackling the problem from the perspective of the more general Binomial distribution.

3a)

The previous problems worked with the observations stored as 0s and 1s in the vector `x`. You must now summarize the observations. The Binomial distribution requires the number of events, or Hits in this case, and the number of trials, or At Bats in this case.

Calculate the number of hits and number of at bats for the sequence of observations stored in the vector `x`. Assign the results to the corresponding variables defined in the code chunk below.

```r
player_hits <- sum(x)
player_atbats <- length(x)

player_hits
```

SOLUTION

```
## [1] 14
```

```
player_atbats
```

```
## [1] 34
```

3b)

You examined the behavior of the log-likelihood when we formulated the problem as a sequence of independent Bernoulli trials. As discussed in lecture, the Binomial distribution assumes the observations are independent Bernoulli trials! Thus, it should not matter if we analyze the problem with the Bernoulli formulation or the Binomial formulation. Let's confirm that is indeed true!

You will work with the log of the Binomial likelihood up to a normalizing constant. That means, you do not need to consider terms that do not directly involve the unknown event probability μ. Dropping or ignoring the constant terms is also referred to as the **un-normalized** likelihood.

Write out the expression for the Binomial log-likelihood up to a normalizing constant 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

$$\log\left(p\left(H \mid AB, \mu\right)\right) \propto H\log[\mu] + (AB - H)\log(1-\mu) \qquad (*)$$

This is because the formula for binomial distribution is:

$$p\left(H \mid AB, \mu\right) = \binom{AB}{H}\mu^{H}(1-\mu)^{AB-H} \qquad (**)$$

We can ignore the term $\binom{AB}{H}$ in $(**)$ above because it does not involve μ. Then, taking log of both sides in $(**)$ and using the properties of logarithms as in question 1d) we can get the result $(*)$ above.

3c)

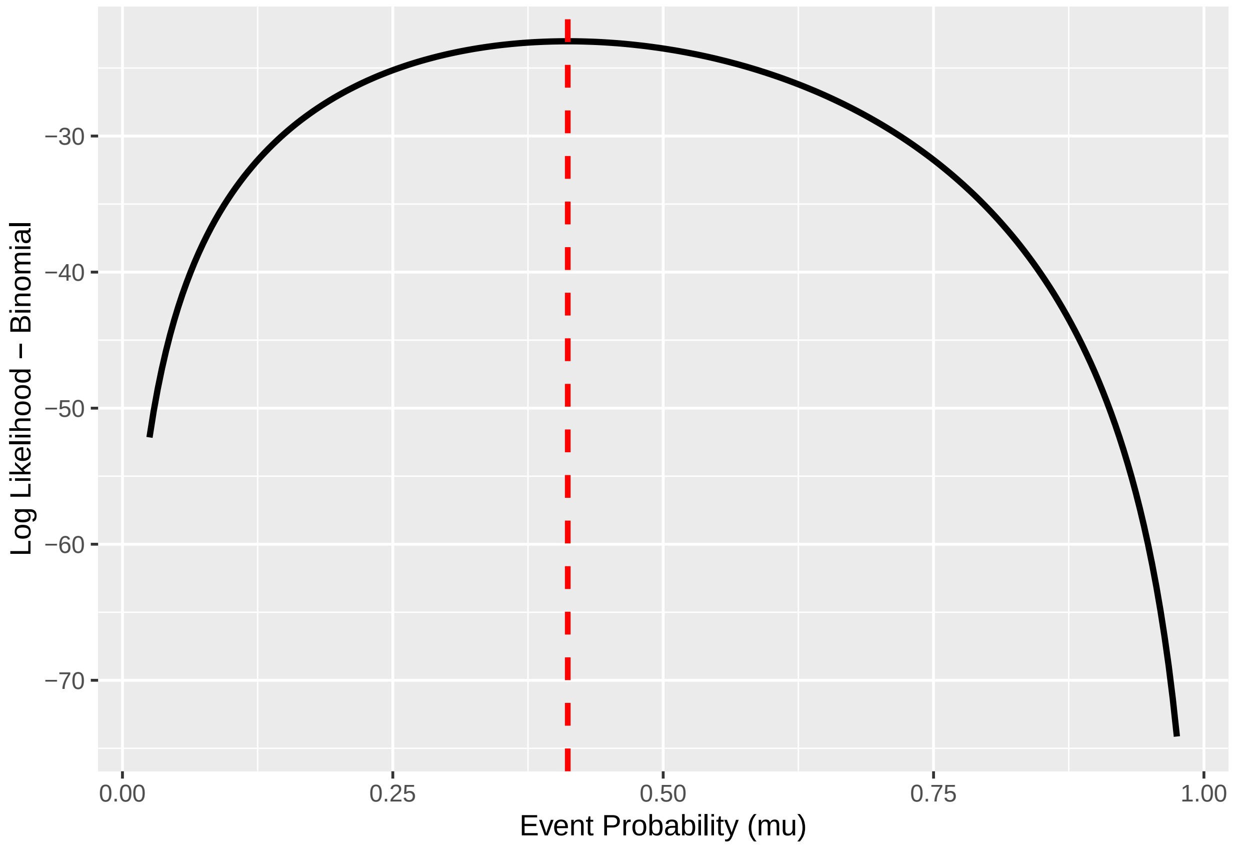
Regardless of the formulation (Bernoulli vs Binomial), our goal is to **learn the event probability**. Thus, we still need to find the maximum likelihood estimate (MLE) for μ. You graphically solved this for the Bernoulli formulation in Problem 02. Let's now graphically find the MLE with the Binomial formulation. However, you do not need to use for-loops to compile the data necessary to create the figure when using the Binomial formulation!

The code chunk below is started for you. A tibble (dataframe) is created with the data variable (column) `mu` assigned to the `mu_grid` object you created earlier. The tibble is passed to the `mutate()` function for you. You must create a new variable, `log_lik` within `mutate()` which is equal to the Binomial log-likelihood up to a normalizing constant. Thus, `log_lik` must equal the expression you wrote in 3a). 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 with `size` assigned to 1.1. As a reference, 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. Assign the `size`, `linetype`, and `color` arguments within `geom_vline()` to 1, 'dashed', and 'red', respectively.

```r
H = player_hits
AB = player_atbats
tibble::tibble(
  mu = mu_grid
) %>%
  mutate(
    log_lik = H * log(mu) + (AB - H) * log(1 - mu)
  ) %>%
  ggplot(aes(x = mu, y = log_lik)) +
  geom_line(size = 1.1) +
  geom_vline(xintercept = sum(x == 1)/length(x), size = 1, linetype="dashed", color="red") +
  labs(x = "Event Probability (mu)", y = "Log Likelihood - Binomial")
```

SOLUTION

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



3d)

How does your figure in 3c) compare to your figure in 2f)?

SOLUTION Figures in 3c) and 2f) are exactly the same because right hand side of (∗) in 3b) and the final expression in 2a) are the same.

3e)

The un-normalized Binomial likelihood you wrote in 3b) and programmed in 3c) is missing the Binomial coefficient. The Binomial coefficient properly normalizes the values of the Binomial distribution. It is not critical for the **shape** of the log-likelihood but the normalizing constant is critical for calculating probabilities.

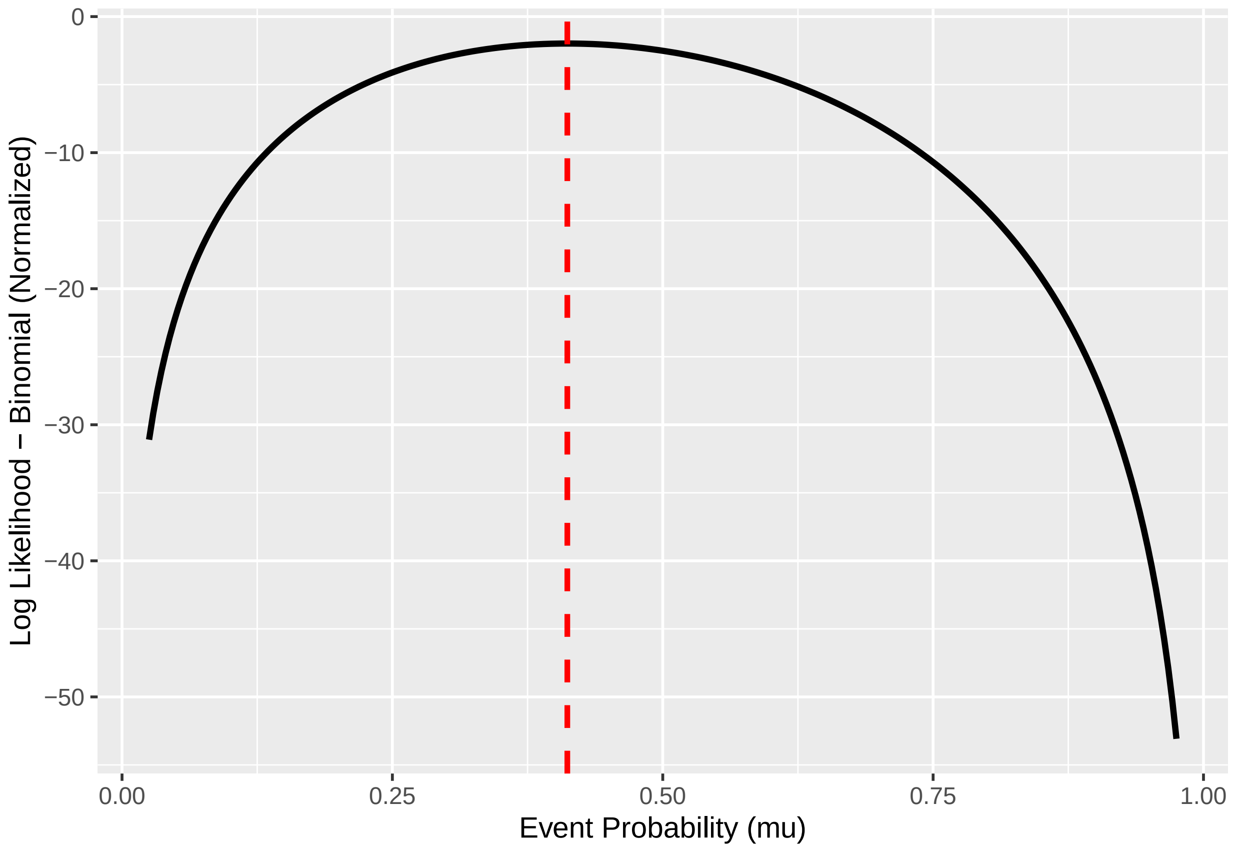
Unless specified otherwise, you are allowed to existing functions for evaluating properly normalized densities or probability mass functions. For the Binomial distribution, the predefined function is `dbinom()`. It contains 4 input arguments: `x`, `size`, `prob`, and `log`. `x` is the number of observed 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`.

You must use the `dbinom()` function to evaluate the log-Binomial likelihood for the player, similar to what you did in 3c). However, instead of manually typing the log-likelihood up to a normalizing constant, you may use the `dbinom()` function to properly evaluate the log-likelihood.

The code chunk below is started for you and is structured similar to that in 3c). A tibble (dataframe) is created with the data variable (column) `mu` assigned to the `mu_grid` object. The tibble is passed to the `mutate()` function for you. You must create a new variable, `log_lik` within `mutate()` which is equal to the Binomial log-likelihood. Use the `dbinom()` function to correctly calculate the Binomial log-likelihood. Pay close attention to the arguments of `dbinom()`. 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 with `size` assigned to 1.1. As a reference, 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. Assign the `size`, `linetype`, and `color` arguments within `geom_vline()` to 1, 'dashed', and 'red', respectively.

HINT: Do not forget to set the `log` flag appropriately!

```r
H = player_hits
AB = player_atbats
tibble::tibble(
  mu = mu_grid
) %>%
  mutate(
    log_lik = dbinom(H, size=AB, prob=mu, log=TRUE)
  ) %>%
  ggplot(aes(x=mu,y = log_lik)) +
  geom_line(size = 1.1) +
  geom_vline(xintercept = sum(x == 1)/length(x), size= 1, linetype="dashed",color="red") +
  labs(x = "Event Probability (mu)", y = "Log Likelihood - Binomial (Normalized)")
```



SOLUTION

3f)

How does your figure in 3e) compare to the figures in 3c) and 2f)?

SOLUTION The figure in 3e) is looking "similar" to the ones in 3c) and 2f) and they all give the same value for MLE. Since the binomial coefficient is involved in 3e), its graph has a scaling factor.

Problem 04

You estimated the event probability (probability of a hit) by maximizing the likelihood. It's now time to use Bayesian methods to learn a **posterior** distribution for the unknown parameter. This distribution will fully represent everything we know about the parameter, based on data and our assumptions. You will summarize this distribution to describe the uncertainty in the parameter, and representative values such as the posterior mean and most probable value (the posterior mode).

As discussed in lecture, Bayesian methods require **prior** distributions. These distributions represent what we believe about the unknowns. Priors enable combining expert opinion with the data in a controlled and consistent manner. The prior allows us to specify bounds or constraints on the parameter and thus prevents the learning process from being fooled by noise or small sample sizes.

Your goal is to learn the unknown event (hit) probability. You must therefore specify a prior belief about the probability that a professional baseball player gets a hit. You will use a **Beta** distribution to encode the prior belief on the event probability. The Beta **shape** parameters control the location, width (uncertainty), and skew (asymmetry) of the Beta distribution. Encoding our prior belief therefore comes down to specifying the shape parameter values.

Instead of focusing on how we should optimally decide those shape parameters, your task is to examine the influence of the prior belief on the posterior result. You will thus try out two different priors and compare the resulting posterior distributions. Determining the "most appropriate" prior is something we will discuss later in the semester.

The code chunk below defines two sets of shape parameters. The uniform set with both shape parameters equal to 1, and the "informative" set which to different values. Both sets refer to the first shape parameter as a and the second shape parameter as b. The R function `dbeta()` refers to a as the `shape1` argument and b as the `shape2` argument.

```
a_uniform <- 1
b_uniform <- 1

a_inform <- 10
b_inform <- 30
```

4a)

What is the prior number of trials associated with the two sets of shape parameters?

SOLUTION For the "uniform" set of shape parameters:

Prior number of trials = a_uniform + b_uniform = 2

```
a_uniform + b_uniform
```

```
## [1] 2
```

For the "informative" set of shape parameters:

Prior number of trials = a_inform + b_inform = 40

```
a_inform + b_inform
```

```
## [1] 40
```

4b)

What is the prior expected value (mean) for the event (hit) probability associated with the two sets of shape parameters?

SOLUTION For the "uniform" set of shape parameters, the mean is:

```
prior_mean_uniform <- a_uniform / (a_uniform+b_uniform)
prior_mean_uniform
```

```
## [1] 0.5
```

For the "informative" set of shape parameters, the mean is:

```
prior_mean_inform <- a_inform /(a_inform+b_inform)
prior_mean_inform
```

```
## [1] 0.25
```

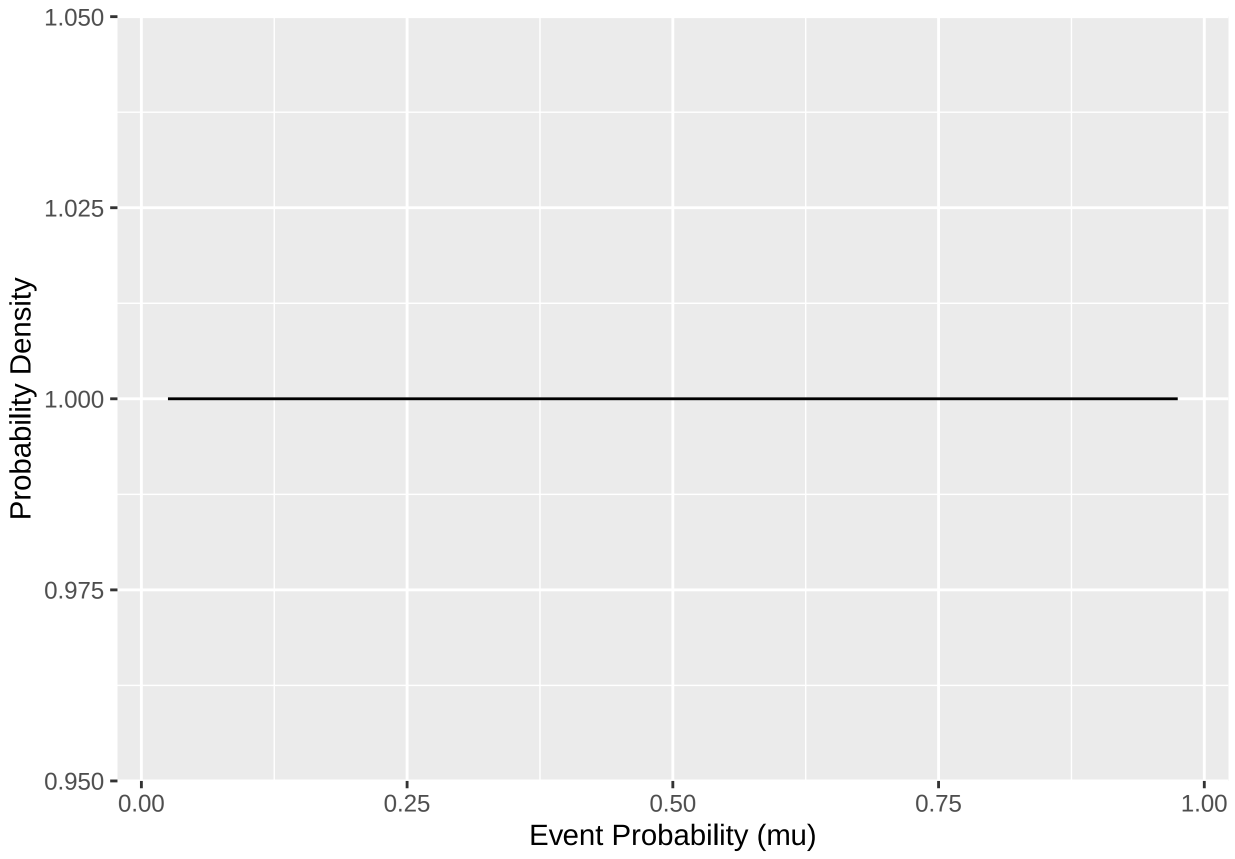
4c)

You will visualize the prior distributions and are allowed to calculate the Beta density with the `dbeta()` function. The first argument to `dbeta()` is the probability parameter, `x`. The second argument to `dbeta()` is `shape1`, the third argument to `dbeta()` is `shape2`. You do not need to set any other argument to `dbeta()` for this question.

Plot the two types of prior distributions on the unknown event (hit) probability μ. The Beta pdf can be evaluated with the `dbeta()` function. You must plot the prior with `ggplot2`. The code chunks below are started for you. The μ values are assigned to the `mu` column using the `mu_grid` object you defined earlier.

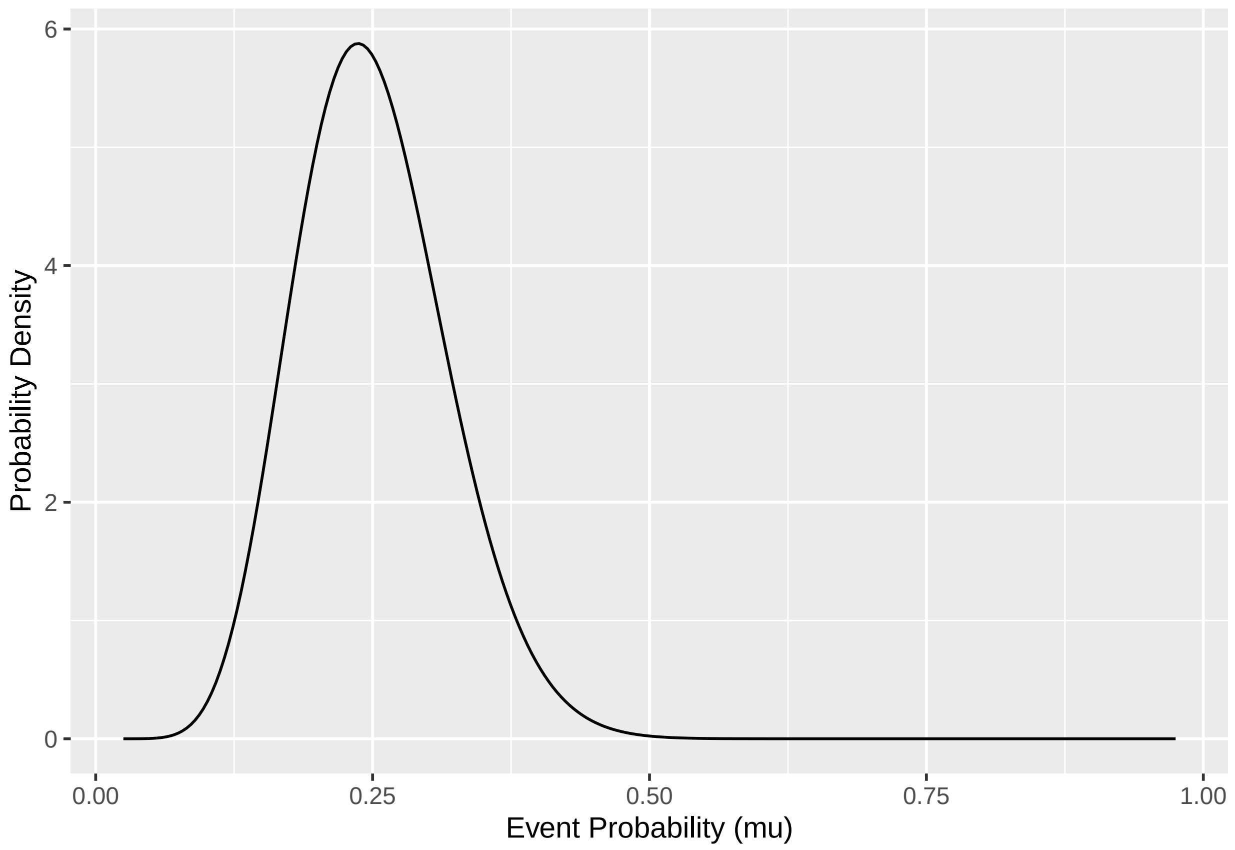
SOLUTION Plot the uniform prior on μ.

```r
tibble::tibble(
  mu = mu_grid
) %>%
  mutate(beta_pdf = dbeta(mu, shape1 = a_uniform, shape2 = b_uniform)) %>%
  ggplot(aes(x = mu, y = beta_pdf)) +
  geom_line() +
  labs(x = "Event Probability (mu)",y = "Probability Density")
```



Plot the informative prior on μ.

```r
tibble::tibble(
  mu = mu_grid
) %>%
  mutate(beta_pdf = dbeta(mu, shape1 = a_inform, shape2 = b_inform)) %>%
  ggplot(aes(x = mu, y = beta_pdf)) +
  geom_line() +
  labs(x = "Event Probability (mu)",y = "Probability Density")
```



4d)

How do the two priors compare? Are there event probability values that are “ruled out” by either of the priors?

SOLUTION Pdf for the “uniform” set of shape parameters is a constant function and thus ut can be stated that event probability values that are “ruled out” in that case.

Pdf for the “inform” set of shape parameters is a curve with maximum value close to 0.25.

Also, the area under the curve in both cases equals 1.

Problem 05

Now that you have practiced working with the likelihood and the prior, it is time to study the posterior! However, before executing the analysis for the current baseball problem, you will manipulate the expressions to get a better understanding of the posterior distribution in this application.

The previous problems in this assignment used notation consistent with the baseball example. However, you will use more generic nomenclature and syntax in this problem to be consistent with lecture. Thus, you will consider a Binomial likelihood with m events out of N trials. You are interested in learning the unknown event probability μ by combining the observations with the prior. You are using a Beta prior with prior shape parameters, a and b.

5a)

You previously wrote out the log-Binomial likelihood up to a normalizing constant in terms hits and at bats. 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

$$\log\left(p\left(m \mid N, \mu\right)\right) \propto m\log[\mu] + (N-m)\log(1-\mu) \qquad (*)$$

This is because the formula for binomial distribution is:

$$p\left(m \mid N, \mu\right) = \binom{N}{m}\mu^{m}(1-\mu)^{N-m} \qquad (**)$$

We can ignore the term $\binom{N}{m}$ in $(**)$ above because it does not involve μ. Then, taking log of both sides in $(**)$ and using the properties of logarithms as in question 1d) we can get the result $(*)$ above.

5b)

Write the log-density of the Beta distribution up to a normalizing constant on the unknown event probability μ with shape parameters a and b.

SOLUTION The equation block is started for you below.

$$\log\left(p\left(\mu \mid a, b\right)\right) \propto (a-1)\log(\mu) + (b-1)\log(1-\mu) \qquad (*)$$

This is because the formula for beta distribution is:

$$p(\mu \mid a, b) = \text{Beta}(\mu \mid a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\mu^{(a-1)}(1-\mu)^{(b-1)} \qquad (**)$$

We can ignore the term $\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}$ in $(**)$ above because it does not involve μ. Then, taking log of both sides in $(**)$ and using the properties of logarithms as in question 1d) we can get the result $(*)$ above.

5c)

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 must practice working through the derivation of the updated shape parameters a_{new} and b_{new}. The log-likelihood was written in Problem 5a) and the log-prior in Problem 5b). In this problem you must add the un-normalized log-likelihood to the un-normalized log-prior, then perform the required algebra to derive a_{new} and b_{new}.

Derive the expressions for the updated or posterior Beta shape parameters. 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 We know that

$$p(\mu \mid m, N) \propto \text{Binomial}(m \mid N, \mu)p(\mu) \qquad (1)$$

When we take log of both sides in (1) and use the results in 5a) and 5b), we get

$$\log[p(\mu \mid m, N)] \propto \log[\text{Binomial}(m \mid N, \mu)] + \log(\mu) = m\log[\mu] + (N-m)\log(1-\mu) + (a-1)\log[\mu] + (b-1)\log[1-\mu]$$

$$= (a+m-1)log[\mu] + (N-m+b-1)\log[1-\mu] \qquad (2)$$

Then, (2) implies that $a_{new} = a+m$ and $b_{new} = b+(N-m)$.

5d)

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 expression for the first derivative of the un-normalized log-posterior with respect to the unknown event probability μ. Write out the derivative in terms of the updated shape parameters a_{new} and b_{new}.

SOLUTION If we take the derivative of (2) in 5c), we get

$$\frac{d}{d\mu}\log[p(\mu \mid m, N)] \propto \frac{a_{new}-1}{\mu} - \frac{b_{new}-1}{1-\mu} \tag{1}$$

which is the desired result.

5e)

Set the derivative from your solution to Problem 5d) equal to zero and solve for the posterior mode of the unknown event probability. Denote the posterior mode as μ_{MAP}.

SOLUTION When we set r.h.s. of expression (1) equal to 0 in 5d) above, we get

$$\mu_{MAP} = \frac{a_{new}}{a_{new}+b_{new}}$$

Problem 06

Now that you've worked with the posterior Beta in greater detail, it is time to execute the Bayesian analysis for the baseball problem.

As a reminder, there are two sets of prior shape parameters. The uniform prior is defined by `a_uniform` and `b_uniform`. The informative prior is defined by `a_inform` and `b_inform`. The observations are stored in the variables `player_hits` and `player_atbats`.

6a)

Calculate the updated or posterior shape parameters, a_{new} and b_{new}, using the observations and the two sets of prior shape parameters.

This problem is open ended, you are free to make the calculations anyway you like. However, you must display the updated shape parameters to the screen. Your results should be clearly marked as to which prior the posterior shape parameters are associated with.

```r
a_new_uniform <- a_uniform + player_hits
b_new_uniform <- b_uniform + player_atbats - player_hits

a_new_inform <- a_inform + player_hits
b_new_inform <- b_inform + player_atbats - player_hits
```

```r
a_new_uniform
```

SOLUTION

```
## [1] 15
```

```
b_new_uniform
```

```
## [1] 21
```

```
a_new_inform
```

```
## [1] 24
```

```
b_new_inform
```

```
## [1] 50
```

6b)

Calculate the posterior mean, mode, 5th percentile (0.05 quantile), and 95th percentile (0.95 quantile) for the posteriors associated with the two prior types. You should use your results from Problem 6a) for the updated or posterior beta shape parameters

Display your results as a dataframe or tibble.

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

```
post_mean_uniform <- a_new_uniform / (a_new_uniform + b_new_uniform)
post_mode_uniform <- (a_new_uniform - 1) / (a_new_uniform + b_new_uniform - 2)
post_5th_uniform <- qbeta(0.05, a_new_uniform, b_new_uniform)
post_95th_uniform <- qbeta(0.95, a_new_uniform, b_new_uniform)

post_mean_inform <- a_new_inform / (a_new_inform + b_new_inform)
post_mode_inform <- (a_new_inform - 1) / (a_new_inform + b_new_inform - 2)
post_5th_inform <- qbeta(0.05, a_new_inform, b_new_inform)
post_95th_inform <- qbeta(0.95, a_new_inform, b_new_inform)

summary_results <- data.frame(
  Prior = c("Uniform", "Informative"),
  Mean = c(post_mean_uniform, post_mean_inform),
  Mode = c(post_mode_uniform, post_mode_inform),
  Percentile_5 = c(post_5th_uniform, post_5th_inform),
  Percentile_95 = c(post_95th_uniform, post_95th_inform))

summary_results
```

SOLUTION

```
##         Prior      Mean      Mode Percentile_5 Percentile_95
## 1     Uniform 0.4166667 0.4117647    0.2858488     0.5528245
## 2 Informative 0.3243243 0.3194444    0.2381307     0.4159531
```

6c)

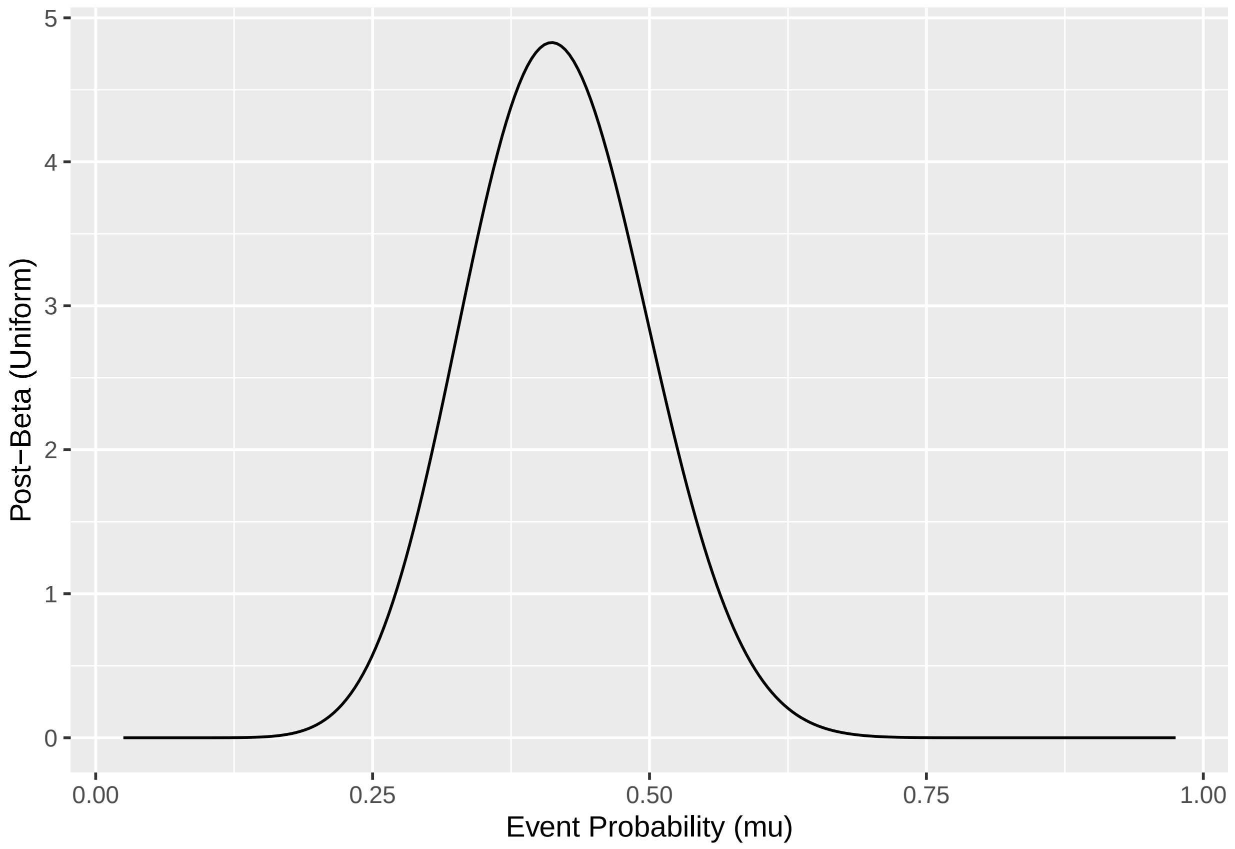
Problem 6b) required that you summarize the posterior Beta distribution. This problem requires that you visualize the posterior Beta density associated with the two prior types. You will create the visualizations similiar to what you did in 4c), but you must use the updated or posterior shape parameters, instead of the prior shape parameters.

Plot the two posterior distributions on the unknown event (hit) probability μ. The Beta pdf can be evaluated with the `dbeta()` function. You must plot the posterior with `ggplot2`. The

code chunks below are started for you. The μ values are assigned to the `mu` column using the `mu_grid` object you defined earlier. Use the posterior shape parameters you calculated in 6a) in the appropriate code chunk below.

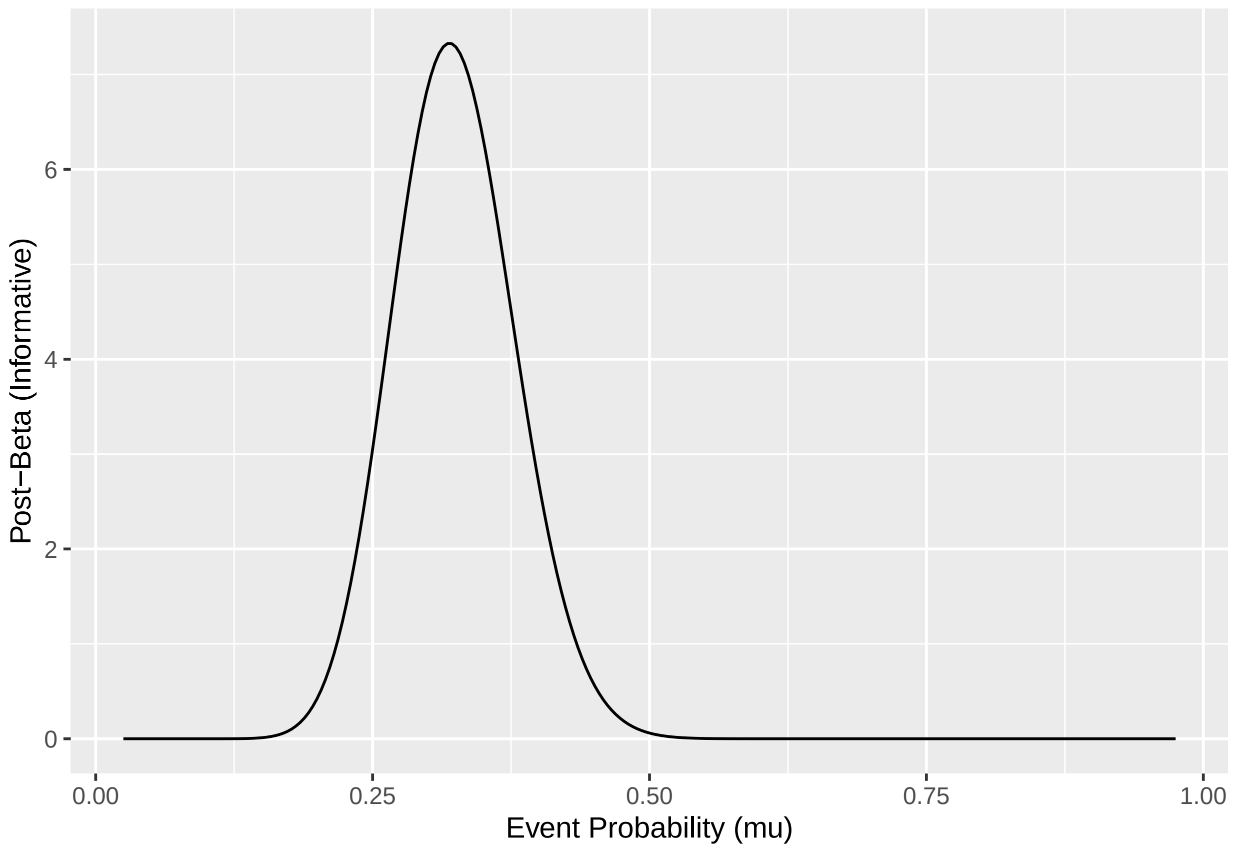
SOLUTION Plot the posterior Beta associated with the uniform prior.

```
tibble::tibble(
  mu = mu_grid
) %>%
  mutate(beta_pdf = dbeta(mu, shape1 = a_new_uniform, shape2 = b_new_uniform)) %>%
  ggplot(aes(x = mu, y = beta_pdf)) +
  geom_line() +
  labs(x = "Event Probability (mu)",y = "Post-Beta (Uniform)")
```



Plot the posterior Beta associated with the informative prior.

```
tibble::tibble(
  mu = mu_grid
) %>%
  mutate(beta_pdf = dbeta(mu, shape1 = a_new_inform, shape2 = b_new_inform)) %>%
  ggplot(aes(x = mu, y = beta_pdf)) +
  geom_line() +
  labs(x = "Event Probability (mu)",y = "Post-Beta (Informative)")
```



6d)

You have visualize the two posteriors and summarized them.

Based on your results, how would you describe the differences in the posterior belief based on the two sets of priors?

SOLUTION What do you think?

1) Mean represents the central tendancy. Mean value of the posterior belief associated with the uniform prior is greater than that of informative prior.
2) Middle 90% credible interval of the posterior belief associated with the uniform prior equals 0.5528245-0.2858488 = 0.2669757.
3) Middle 90% credible interval of the posterior belief associated with the informative prior equals 0.4159531-0.2381307 = 0.1778224. This value is less than the one above, which is expected because "informative" case contains more prior observations.
4) "Informative" case thinks it is very unlikely that $\mu > 0.4$ or $\mu < 0.23$.
5) "Uniform" case thinks it is very unlikely that $\mu > 0.55$ or $\mu < 0.28$.

Problem 07

The data provided to you at the beginning of the assignment is just a small sample of the at bats for this particular Major League Baseball player. The player has played in the MLB season June 2022 and has required many more at bats. You have evaluated the posterior based on the small sample size under two different prior assumptions. Let's now examine how the posterior behaves under larger sample sizes by using the data from the entire season.

The code chunk below provides the season total hits (number of events) and at bats (number of trials) for this player (at least up to the creation of this assignment).

```
season_hits <- 62

season_atbats <- 284
```

You will use the same two sets of prior shape parameters as in the previous problem. However, you will use the larger sample size observations for this question.

7a)

Calculate the updated or posterior shape parameters, a_{new} and b_{new}, using the season (larger sample size) observations and the two sets of prior shape parameters.

This problem is open ended, you are free to make the calculations anyway you like. However, you must display the updated shape parameters to the screen. Your results should be clearly marked as to which prior the posterior shape parameters are associated with.

```
a_new_uniform2 <- a_uniform + season_hits
b_new_uniform2 <- b_uniform + season_atbats - season_hits

a_new_inform2 <- a_inform + player_hits
b_new_inform2 <- b_inform + season_atbats - season_hits
```

```
a_new_uniform2
```

SOLUTION

```
## [1] 63
```

```
b_new_uniform2
```

```
## [1] 223
```

```
a_new_inform2
```

```
## [1] 24
```

```
b_new_inform2
```

```
## [1] 252
```

7b)

Calculate the posterior mean, mode, 5th percentile (0.05 quantile), and 95th percentile (0.95 quantile) for the posteriors associated with the two prior types. You should use your results from Problem 7a) for the updated or posterior beta shape parameters

Display your results as a dataframe or tibble.

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

```
post_mean_uniform2 <- a_new_uniform2 / (a_new_uniform2 + b_new_uniform2)
post_mode_uniform2 <- (a_new_uniform2 - 1) / (a_new_uniform2 + b_new_uniform2 - 2)
post_5th_uniform2 <- qbeta(0.05, a_new_uniform2, b_new_uniform2)
```

```
post_95th_uniform2 <- qbeta(0.95, a_new_uniform2, b_new_uniform2)

post_mean_inform2 <- a_new_inform2 / (a_new_inform2 + b_new_inform2)
post_mode_inform2 <- (a_new_inform2 - 1) / (a_new_inform2 + b_new_inform2 - 2)
post_5th_inform2 <- qbeta(0.05, a_new_inform2, b_new_inform2)
post_95th_inform2 <- qbeta(0.95, a_new_inform2, b_new_inform2)

summary_results2 <- data.frame(
  Prior = c("Uniform", "Informative"),
  Mean = c(post_mean_uniform2, post_mean_inform2),
  Mode = c(post_mode_uniform2, post_mode_inform2),
  Percentile_5 = c(post_5th_uniform2, post_5th_inform2),
  Percentile_95 = c(post_95th_uniform2, post_95th_inform2))

summary_results2
```

SOLUTION

```
##         Prior       Mean       Mode Percentile_5 Percentile_95
## 1     Uniform 0.22027972 0.21830986   0.18117530     0.2616117
## 2 Informative 0.08695652 0.08394161   0.06090699     0.1164100
```

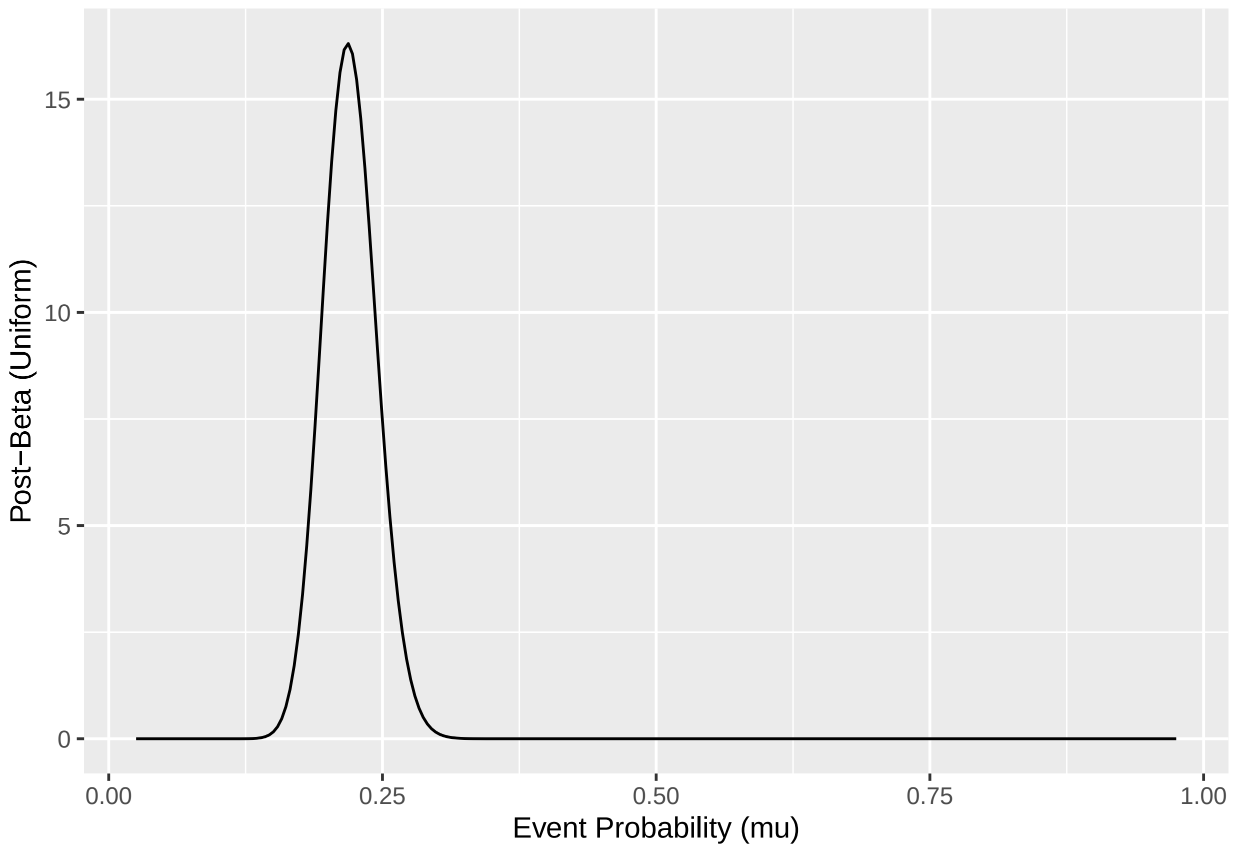
7c)

Problem 7b) required that you summarize the posterior Beta distribution. This problem requires that you visualize the posterior Beta density associated with the two prior types. You will create the visualizations similar to what you did in 4c), but you must use the updated or posterior shape parameters, instead of the prior shape parameters.

Plot the two posterior distributions on the unknown event (hit) probability μ. The Beta pdf can be evaluated with the `dbeta()` function. You must plot the posterior with `ggplot2`. The code chunks below are started for you. The μ values are assigned to the `mu` column using the `mu_grid` object you defined earlier. Use the posterior shape parameters you calculated in 7a) in the appropriate code chunk below.

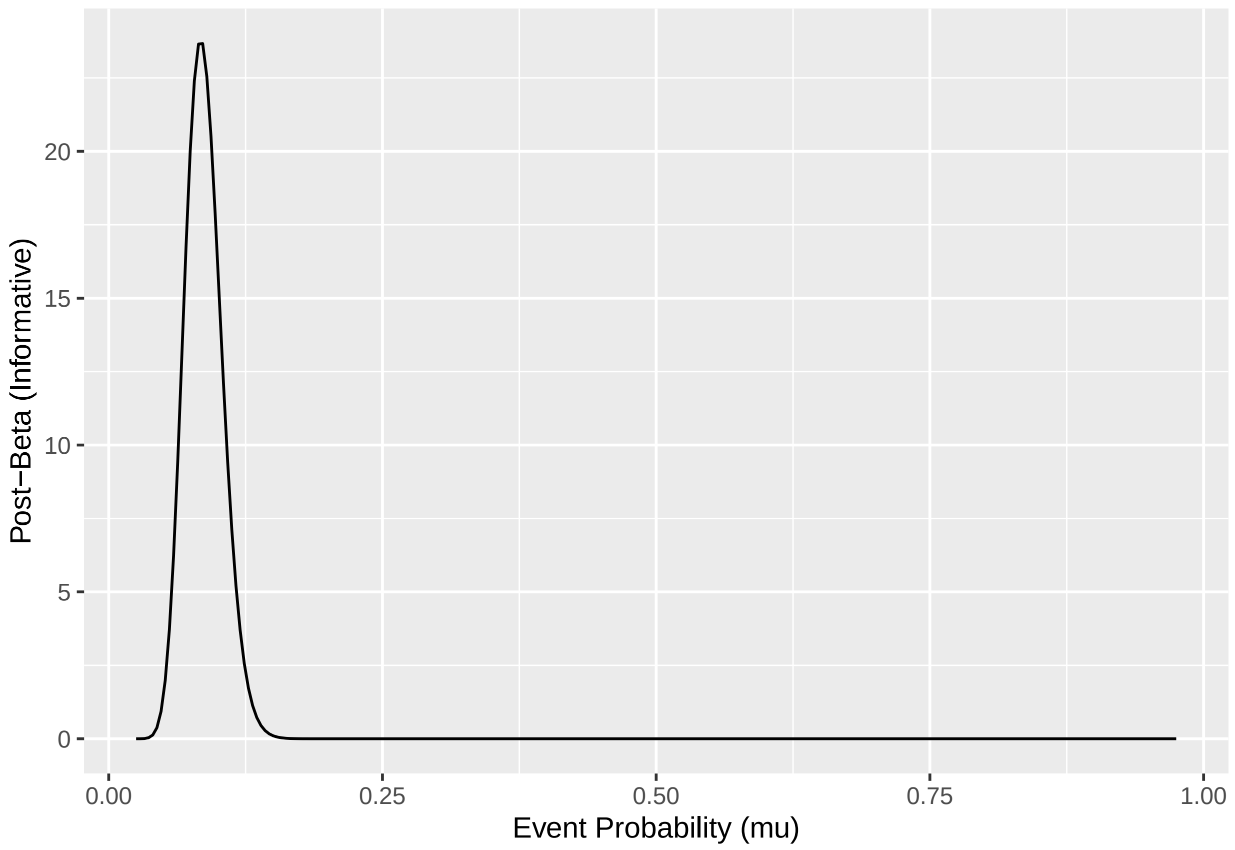
SOLUTION Plot the posterior Beta associated with the uniform prior.

```
tibble::tibble(
  mu = mu_grid
) %>%
  mutate(beta_pdf = dbeta(mu, shape1 = a_new_uniform2, shape2 = b_new_uniform2)) %>%
  ggplot(aes(x = mu, y = beta_pdf)) +
  geom_line() +
  labs(x = "Event Probability (mu)",y = "Post-Beta (Uniform)")
```



Plot the posterior Beta associated with the informative prior.

```
tibble::tibble(
  mu = mu_grid
) %>%
  mutate(beta_pdf = dbeta(mu, shape1 = a_new_inform2, shape2 = b_new_inform2)) %>%
  ggplot(aes(x = mu, y = beta_pdf)) +
  geom_line() +
  labs(x = "Event Probability (mu)",y = "Post-Beta (Informative)")
```



7d)

You examined the sensitivity of the posterior to two types of prior assumptions based on two sample sizes. One prior is uniform, while the other is "informative". One sample size was small, while the other was larger.

Describe the influence of the prior on the posterior when the sample size is small vs large.

SOLUTION What do you think?

1) Mean represents the central tendancy. In this 2nd case, mean value of the posterior belief associated with the uniform prior is greater than that of informative prior.
2) In this 2nd case, middle 90% credible interval of the posterior belief associated with the uniform prior equals 0.2616117-0.18117530 = 0.0804364.
3) In this 2nd case, middle 90% credible interval of the posterior belief associated with the informative prior equals 0.1164100-0.06090699 = 0.055503. This value is less than the one above, which is expected because "informative" case contains more prior observations.
4) When we compare 1st case and 2nd case, we observe that middle 90% credible intervals reduce for both uniform prior and informative prior, which indicates that adding larger sample size reduces the credible intervals.
5) In the case of uniform prior, the credible intervals reduces significantly from 1st to 2nd case, which shows that adding large sample size has more effect on small prior case.
6) In the 2nd case, "Informative" case thinks it is very unlikely that $\mu > 0.1164100$ or $\mu < 0.06090699$.
7) In the 2nd case, "Uniform" case thinks it is very unlikely that $\mu > 0.2616117$ or $\mu < 0.18117530$.