CS 1675 Homework: 02

Assigned: January 18, 2020; Due: February 1, 2020

Prayush Luitel

Submission time: February 1, 2020 at 7:00PM

Collaborators Include the names of your collaborators here.

Overview

This homework assignment focuses on the mathematics of likelihoods, priors, and posterior distributions. You will work with the Binomial and Normal likelihood throughout the assignment. You will also work on writing functions within `R`. You can see what the rendered document looks like at any time, by pressing the "Knit" button within RStudio. You can execute a code chunk by pressing the arrow button within that code chunk located to the upper right portion of the code chunk within the RStudio IDE. Alternatively, you can execute a line of code within a code chunk by selecting that line and pressing Ctrl+ENTER (Windows) or Command+ENTER (Mac).

Completing this assignment requires filling in missing pieces of information from existing code chunks, programming complete code chunks from scratch, typing discussions about results, and working with LaTeX style math formulas. A template .Rmd file is available to use as a starting point for this homework assignment. The template is available on CourseWeb.

IMPORTANT: Please pay attention to the `eval` flag within the code chunk options. Code chunks with `eval=FALSE` will **not** be evaluated (executed) when you Knit the document. You **must** change the `eval` flag to be `eval=TRUE`. This was done so that you can Knit (and thus render) the document as you work on the assignment, without worrying about errors crashing the code in questions you have not started. Code chunks which require you to enter all of the required code do not set the `eval` flag. Thus, those specific code chunks use the default option of `eval=TRUE`.

Load packages

This assignment uses the `dplyr` and `ggplot2` packages, which are loaded in the code chunk below. The assignment also uses the `tibble` package to create tibbles.

```r
library(dplyr)
library(ggplot2)
```

Problem 1

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.

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

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

PROBLEM Write out the formula for the Maxmimum 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:

$$\mu_{ML} = \frac{H}{AB}$$

For this particular example:

```r
### your code here
player_mle <- player_hits / player_atbats
player_mle
```

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

1b)

Let's check your answer in 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.

PROBLEM 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

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

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 $\mu = .6$.

PROBLEM **Set the variable `mu` equal to a vector consisting of 101 evenly spaced elements from 0.1 to 0.6. Pipe the `tibble` into a `mutate()` call 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 a `ggplot()` call where you set the `x` aesthetic equal to `mu` and the `y` aesthetic equal 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()` call. `geom_vline()` displays a vertical line at a specified `xintercept` value. You do not need to place `xintercept` within the `aes()` function.**

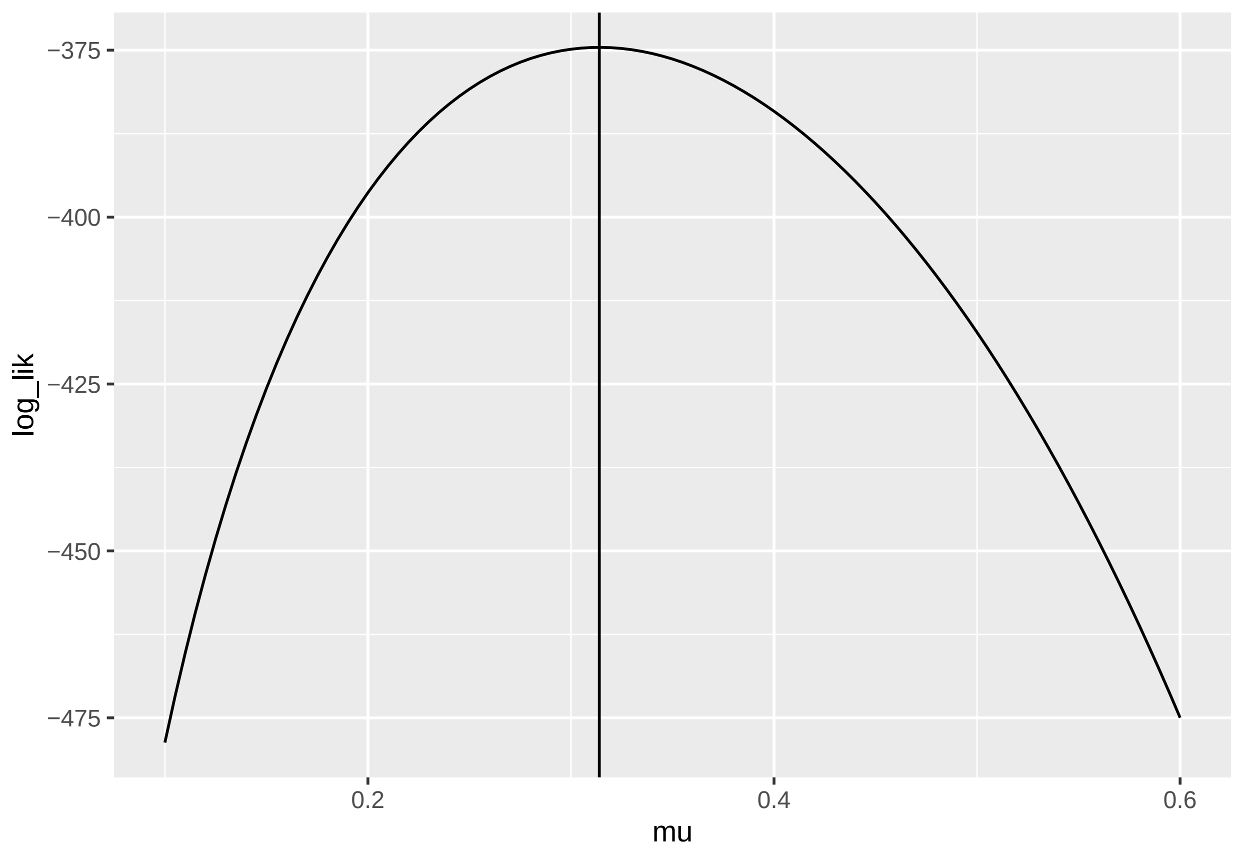
The MLE is corresponds to what on the plotted curve?

HINT: You used a function in homework 01 to create a vector of evenly spaced points between a defined interval.

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

SOLUTION ?

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



The MLE corresponds to the maximum value for the log-likelihood function.

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. In the code chunk 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.

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

```
## NULL
```

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

```
## NULL
```

Let's now create a function to calculate the un-normalized Binomrial 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`.

PROBLEM 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.

```r
### set the input arguments !!!
log_binom_pmf_unnorm <- function(events, trials, prob)
{
  # assign log_lik
  log_lik = events * log(prob) + (trials - events) * log(1-prob)

  # return log_lik
    return (log_lik)
}
```

SOLUTION

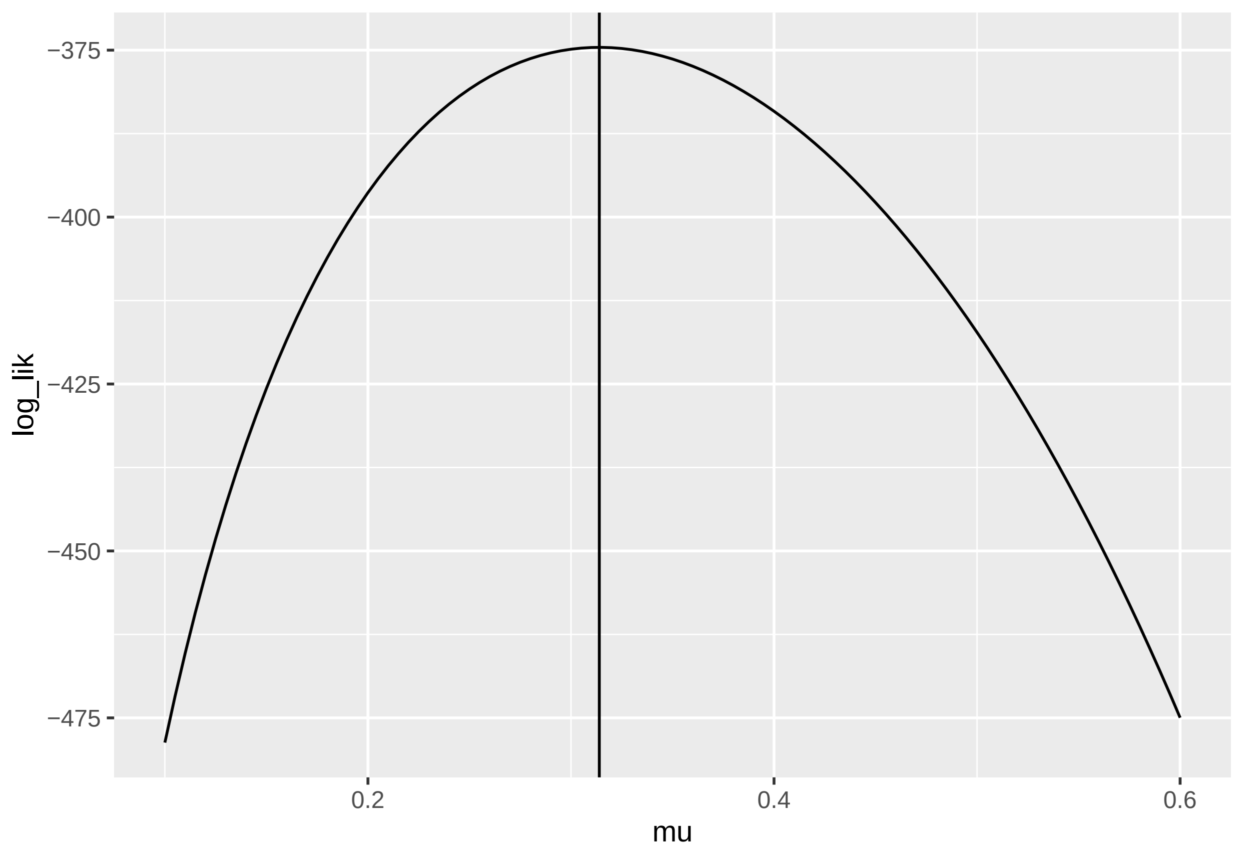
1e)

Let's now use the `log_binom_pmf_unnorm()` function to recreate the figure from Problem 1c).

PROBLEM 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 ?

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



1f)

The un-normalized log-likelihood does not include 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.

PROBLEM 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()`.

```r
### set the input arguments!
log_binom_pmf <- function(events, trials, prob)
{
  log_lik = lchoose(trials,events) * (events * log(prob) + (trials - events) * log(1-prob))
  # your code here
  return (log_lik)
}
```

SOLUTION

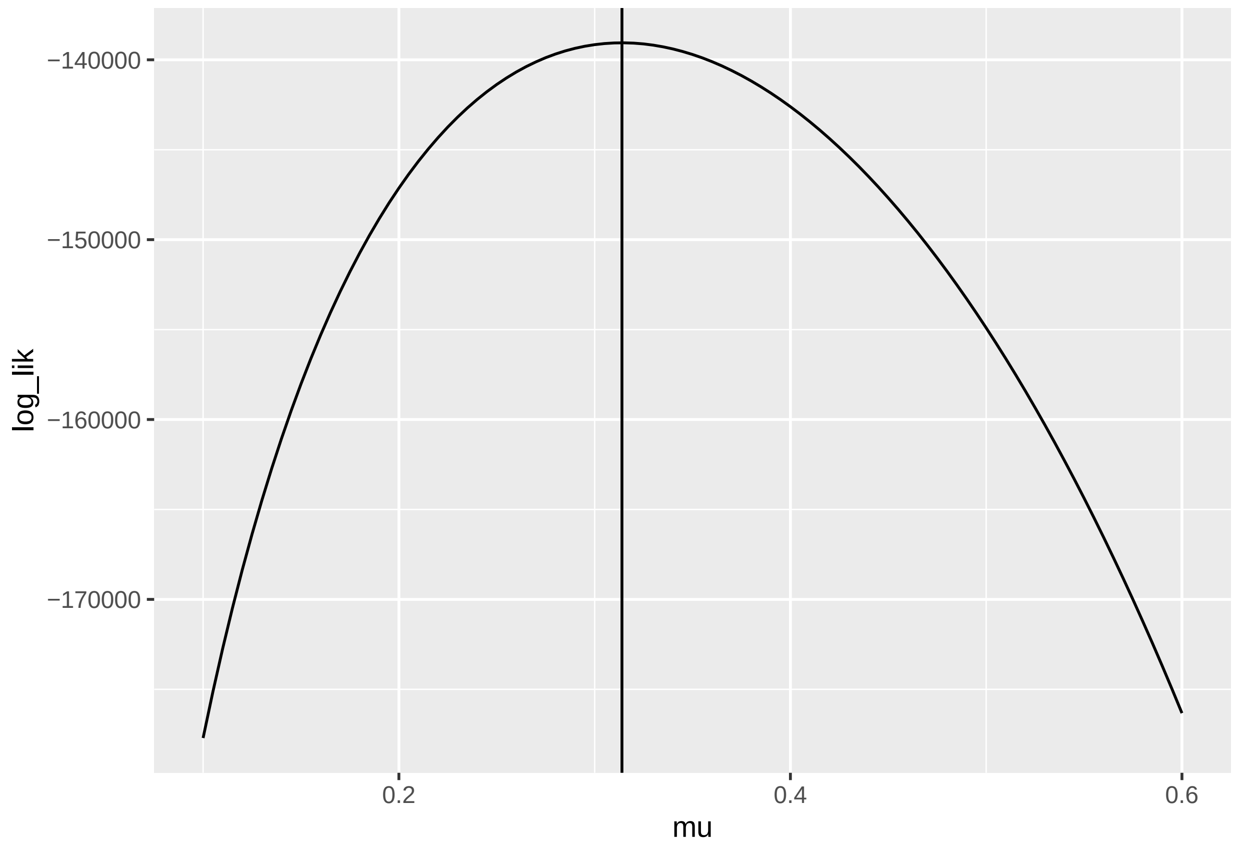
1g)

The un-normalized log-likelihood is all we needed when we were interested in finding the MLE. The constants do not impact the shape, when we evaluate the derivative with respect to μ the constant terms drop out. 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.

PROBLEM 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 ?

```r
tibble::tibble(
  mu = seq(from = 0.1, to = 0.6, length.out = 101)
) %>%
  mutate(log_lik = log_binom_pmf(player_hits,player_atbats,mu)) %>%
  ggplot(mapping = aes(x = mu, y = log_lik)) +
  geom_line() +
  geom_vline(mapping = aes(xintercept = player_mle))
```



No the MLE does not correspond to the same location for the normalized and unnormalized log likelihoods as the value for the unnomalized likelihood is around -375 and the normalized log likelihood has a value around -140000.

Problem 2

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.

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

```
### your code here
exp(log_binom_pmf(player_hits, player_atbats,0.30) - log_binom_pmf(player_hits, player_atbats, player_ml
```

SOLUTION

```
## [1] 2.478929e-45
```

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`.

PROBLEM 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.

```r
### your code here
dbinom(x = 189, size = 602, prob =0.3)
```

SOLUTION

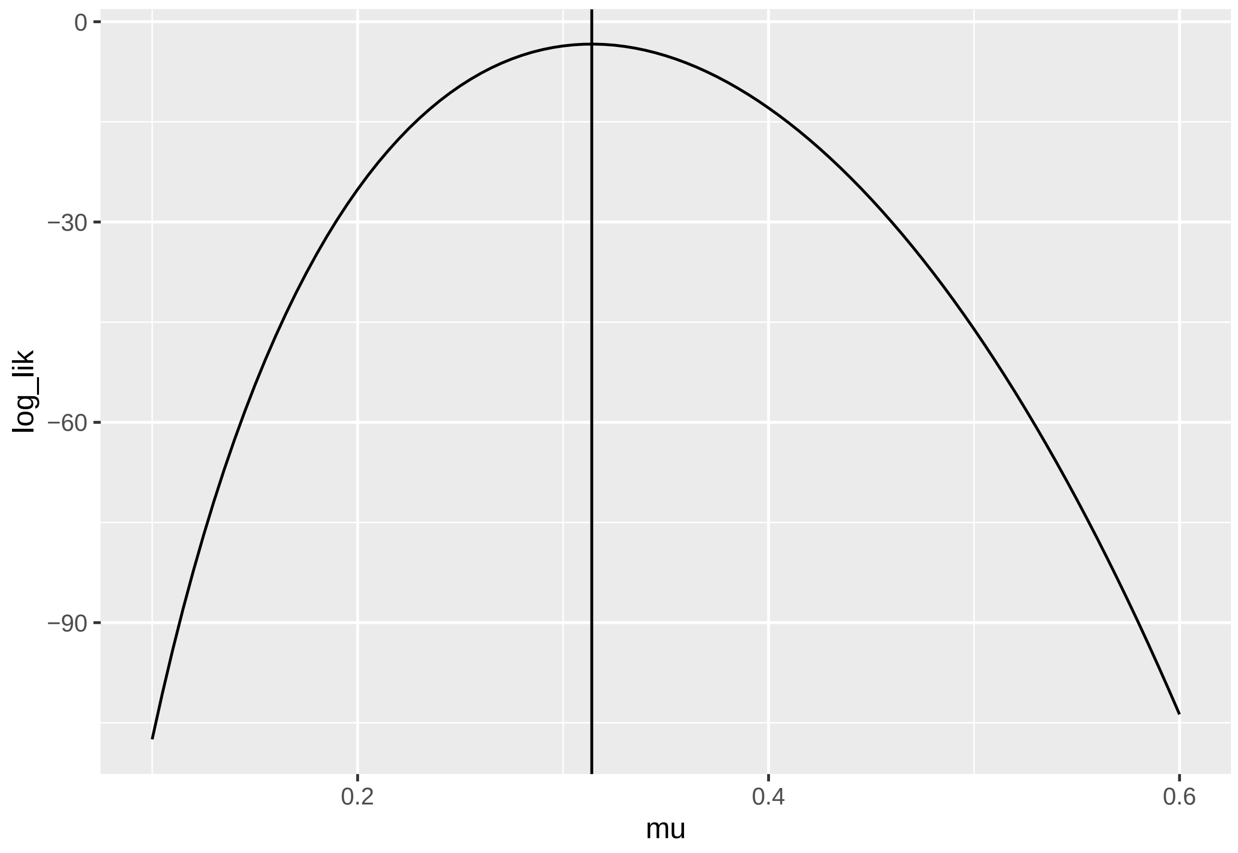
```
## [1] 0.02655379
```

2c)

PROBLEM 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 ?

```r
tibble::tibble(
  mu = seq(from = 0.1, to = 0.6, length.out = 101)
) %>%
  mutate(log_lik = dbinom(player_hits,player_atbats,mu,log =TRUE)) %>%
  ggplot(mapping = aes(x = mu, y = log_lik)) +
  geom_line() +
  geom_vline(mapping = aes(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 up, 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.

PROBLEM 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?

```r
coin_probs <- dbinom(x = 0:5, size = 11, prob = 0.25)

# print to screen
coin_probs
```

SOLUTION

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

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

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

The first element in the vector of coin probs is the probability of finding exactly 0 heads in a trial of size 11 attempts.

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.

PROBLEM 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?

```r
### your code here
sum(coin_probs)
```

SOLUTION

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

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.

PROBLEM 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()`?

```r
### your code here
pbinom(q =5, size = 11, prob=.25)
```

SOLUTION

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

The result form the pbinom function is the same as the result form the manual approach of 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.

PROBLEM 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

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

```r
### your code here
##probability of observing 175-195 hits
pbinom(q =195, size = 602,prob = 0.3) - pbinom(q =175, size = 602,prob = 0.3)
```

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

```r
## probability of observing 165-205 hits
pbinom(q =205, size = 602,prob = 0.3) - pbinom(q =165, size = 602,prob = 0.3)
```

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

?

Problem 3

The goal of Problem 2 was to get you to start thinking *probabilistically*. A player with a true ability of successfully getting a hit 30% of the time may not exactly achieve 30 hits out of 100 at bats. Assuming the number of hits is Binomially distributed, there is approximately a 72% probability that player will achieve between 25 to 35 hits out of 100 at bats. This may not sound like a big deal, but that's the difference of being considered less than average to one of the best "hitters" in the sport.

When estimating the probability of getting a hit (or more generally the probability of the event), we need to consider the uncertainty in the probability. This is especially true when the sample size is small. In this class, we will typically represent uncertainty within a Bayesian framework. We will update a prior belief based on observations, to yield our posterior belief.

In our baseball example, although the player ended the season with a good "batting average", he started out "in a slump". Initially, he had 6 hits out of 30 at bats. However, I have followed this player's development. I feel his initial performance is not representative of his true ability. As we discussed in lecture, we can encode our prior belief about a probability with a Beta distribution. The beta distribution consists of two hyperparameters, a and b. These two hyperparameters allow us to control our prior expected value on the probability, as well as to encode our prior uncertainty on the probability.

3a)

PROBLEM Write out the un-normalized log-density of the Beta distribution with hyperparameters a and b. The equation block below is started for you. Notice that the log-prior density is written as proportional to rather than equal to the right hand side. This denotes you can drop the constant terms.

SOLUTION

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

3b)

We already know that since the Beta is conjugate to the Binomial, the posterior distribution is a Beta. You will practice working through the derivation of the updated hyperparameters a_{new} and b_{new}. Way back in Problem 1b), you wrote the un-normalized log-likelihood for observing H hits out of AB at bats, given the probability μ. 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}.

PROBLEM 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 to help with the rendering of the expressions in the rendered PDF document.

SOLUTION Write out your derivation in multiple equation blocks.

$$
(H + a)\log(\mu) = a_{new}\log(\mu)
$$
$$
a_{new} = H + a
$$
$$
(AB - H + b)\log(1 - \mu) = b_{new} log(1 - \mu)
$$
$$
b_{new} = (AB - H + b)
$$

3c)

Since I have been following this player's career, I feel I can specify a prior distribution representative of his batting ability. After some thought, I settle on $a = 41.76$ and $b = 108.32$. Those hyperparameters are defined as the variables `a_prior` and `b_prior` for you in the code chunk below. The code chunk also defines the early number of hits and at bats as `start_hits` and `start_atbats`, respectively. You will use these variables to compare my prior belief to the results of the initial 30 at bats in the season.

```
a_prior <- 41.76
b_prior <- 108.32
start_hits <- 6
start_atbats <- 30
```

PROBLEM Calculate the MLE to the probability of a hit, based on the first 30 at bats. Calculate the prior expected value (mean) on the probability of a hit using the defined hyperparameters to the Beta prior. How many "prior at bats" does my prior belief represent?

Assign the starting MLE to the variable `start_mle`. Assign the prior expected value (mean) to the variable `prior_mean`.

SOLUTION The MLE based on the first 30 at bats is:

```
### your code here
start_mle <- start_hits / start_atbats
start_mle
```

```
## [1] 0.2
```

The prior mean on the probability of a hit is:

```
### your code here
prior_mean <- a_prior / (a_prior + b_prior)
prior_mean
```

```
## [1] 0.2782516
```

The prior number of "at bats" based on the prior hyperparameters is:

```
### your code here
b_prior + a_prior
```

```
## [1] 150.08
```

?

3d)

Let's now study the prior uncertainty in my belief. Remember that the Beta distribution is a continuous probability density function (PDF). The probability of observing an exact value is 0. We must therefore consider probabilities the value is within intervals by integrating the PDF. As with the Binomial distribution, a function already exists in `R` to evaluate that integral or cumulative density function (CDF) for the Beta distribution. The function is `pbeta()`.

Alternatively, rather than calculating the probability the variable is less than a specific value (the result of `pbeta()`), we could ask what specific value corresponds to a particular probability. This quantity is known as a quantile. For example, the median is the 0.5 quantile (or as I usually say the 50th quantile or percentile). It represents that 50% of the probability is less than the median and 50% of the probability is greater than the median. The 0.95 quantile is the value that has 95% of the probability less than it.

To calculate the the quantile of a Beta distribution you can use `qbeta()` function. The first argument is `p`, the probability of interest (for example 0.5, or 0.95, or 0.05). The second and third arguments are the hyperparameters, `shape1` and `shape2`. The `shape1` argument is our a hyperparameter and the `shape2` argument is our b hyperparameter.

PROBLEM Calculate the 0.05 and 0.95 quantiles (5th and 95th percentiles) for the defined prior on the probability of a hit. How does the MLE based on the first 30 at bats compare to the 0.05 quantile? How does the MLE based on all 602 at bats in the season compare to both calculated quantiles?

SOLUTION The 0.05 and 0.95 quantiles of the Beta prior are:

```r
### your code here
qbeta(0.05,a_prior, b_prior)
```

```
## [1] 0.2199853
```

```r
qbeta(0.95,a_prior, b_prior)
```

```
## [1] 0.3398887
```

The MLE based on the first 30 bats (0.2) is smaller than the 5th quantile of the Beta Prior(0.2199), which suggests that attaining an MLE of 0.2 is very unlikely. However, the MLE based on all 602 bats(0.31395) is smaller and decently close to the the 95th quantile of the Beta Prior(0.3398), which suggests that attainging an MLE of 0.31395 is more common.

3e)

Let's now calculate the updated or posterior hyperparameters for the posterior Beta distribution on the probability of a hit. Perform this calculation using the starting data based on the first 30 at bats, and for the complete season data based on all 602 at bats.

The code chunk below defines two sets of variables. The first set are `a_post_start` and `b_post_start`, which correspond to the posterior results after the first 30 at bats. The second set are `a_post_player` and `b_post_player`, which correspond to the posterior results after all 602 at bats.

PROBLEM Calculate the updated a and b hyperparameters based on the first 30 at bats and all 602 at bats. The code chunk below defines two sets of variables, you must complete calculations.

```
a_post_start <- a_prior + start_hits
b_post_start <- b_prior + (start_atbats-start_hits)

a_post_player <- a_prior + player_hits
b_post_player <- b_prior + (player_atbats-player_hits)
```

SOLUTION

3f)

Our updated hyperparameters define the posterior Beta distribution which combines the observed data with our prior belief.

PROBLEM Calculate the posterior expected value (mean) on the probability of a hit based on the first 30 at bats and all 602 at bats. Assign the results to the variables `post_mean_start` and `post_mean_player`. How do the posterior means compare to their corresponding MLEs?

SOLUTION Calculate the posterior means for the two conditions below.

```
post_mean_start <- a_post_start / (a_post_start + b_post_start)
post_mean_start
```

```
## [1] 0.2652155
```

```
post_mean_player <- a_post_player/ (a_post_player + b_post_player)
post_mean_player
```

```
## [1] 0.3068291
```

The posterior mean for the dataset of 30 bats(0.2652) is much larger than the MLE of the same dataset(0.2). Alternatively, the posterior mean for the dataset of 602 bats(0.3068) is smaller than the MLE of the same dataset(0.31395). This suggests that the likelihood is being affected by the Beta Prior.

3g)

Let's now consider our posterior uncertainty in the probability of a hit. You will use the `qbeta()` function to calculate quantiles in this problem.

PROBLEM Calculate the posterior 0.05 and 0.95 quantiles for the case of just 30 at bats and based on all 602 at bats. Are the MLEs contained within the middle 90% uncertainty intervals?

SOLUTION Create your own code chunks to calculate the 0.05 and 0.95 quantiles for BOTH conditions and add in your discussion.

The 0.05 and 0.95 quantiles for the first 30 bats are:

```r
qbeta(0.05, a_post_start, b_post_start)
```

```
## [1] 0.212761
```

```r
qbeta(0.95, a_post_start, b_post_start)
```

```
## [1] 0.3206426
```

The 0.05 nad 0.95 quantiles for all 602 bats are:

```r
qbeta(0.05, a_post_player, b_post_player)
```

```
## [1] 0.2794793
```

```r
qbeta(0.95, a_post_player, b_post_player)
```

```
## [1] 0.3347633
```

For the dataset with 30 bats, the MLE is not contained within the 90% uncertainty interval since 0.2< 0.212. However, for the dataset with 602 bats, the MLE is contained within the 90% uncertainty interval since $0.279 < 0.3139 < 0.3347$.

3h)

In Major League Baseball, a player is traditionally considered to be a very good hitter if they get hits greater than 30% of the time. We can use our prior and our posterior to study this condition probabilistically. The `pbeta()` function allows us to calculate the probability a Beta random variable is less than a particular quantity, `q`. Here `q` is the first argument to the `pbeta()` function. The second and third arguments are `shape1` and `shape2`, just like the `qbeta()` function.

PROBLEM Calculate the prior probability that the player's hit probability is greater than 0.3. Calculate the posterior probability that the player's hit probability is greater than 0.3, based on all 602 at bats. Would you feel comfortable saying the player is a good hitter based on the data from the complete season?

SOLUTION Create your own code chunk to perform the necessary calculation and include your discussion.

The prior probability that the player's hit probability is greater than 0.3 is:

```r
1 - pbeta(0.3, a_prior, b_prior)
```

```
## [1] 0.270441
```

The posterior probability that the player's hit probability is greater than 0.3 is:

```r
1 - pbeta(0.3, a_post_player, b_post_player)
```

```
## [1] 0.6544916
```

Based on the data from the whole season, I would be comfortable saying that the player is a good hitter since the prior probability suggests a low chance of hitting above 0.3(<50%), however the posterior still shows a higher chance of hitting above 0.3(>50%). This means that the player is very capable of hitting above 0.3.

Problem 4

In lecture we discussed estimating the unknown mean of a normal likelihood, given a series of observations. We derived the MLE and studied the Bayesian formulation through the conjugate normal prior. In this problem, you will still work with a normal likelihood, but now we will consider the situation that the mean is known, while the standard deviation (or likelihood noise) is unknown. The N observations will still be denoted as a vector, $\mathbf{x}$. The known mean is μ and the unknown likelihood noise is σ. The posterior on σ given the observations and the known mean is proportional to:

$$p(\sigma \mathbf{x}, \mu) \propto p(\mathbf{x} \mu, \sigma) p(\sigma)$$

We will continue to assume all observations are conditionally independent given the parameters. The likelihood therefore factors into the product of N normal likelihoods:

$$p\left(\sigma \mid \mathbf{x}, \mu\right) \propto \prod_{n=1}^{N} \left(\text{normal}\left(x_n \mid \mu, \sigma\right)\right) \times p\left(\sigma\right)$$

Compared to the example in lecture, we now need to specify our prior belief on the unknown likelihood noise, σ. We will use an Exponential distribution, which consists of a single hyperparameter λ. The hyperparameter is typically referred to as the rate parameter. The Exponential distribution has support only over positive values. The shorthand notation for the Exponential distribution is usually Exp. I will always use a captial E to denote the exponential distribution compared to the exponential function, exp. The probability density function for the Exponential distribution is:

$$\text{Exp}\left(\sigma \mid \lambda\right) = \lambda \exp\left(-\lambda \sigma\right)$$

The posterior distribution on the unknown likelihood noise, σ, is therefore:

$$p\left(\sigma \mid \mathbf{x}, \mu\right) \propto \prod_{n=1}^{N} \left(\text{normal}\left(x_n \mid \mu, \sigma\right)\right) \times \log(\text{Exp}\left(\sigma \mid \lambda\right)$$

4a)

PROBLEM You must write out the un-normalized log-posterior on σ. By focusing on the un-normalized log-posterior you can drop constant terms. You do not need to simplify the summation term involving the quadratic portion within the likelihood.

SOLUTION Use multiple equation blocks to complete the derivation.

$$p\left(\sigma \mid \mathbf{x}, \mu\right) \propto \sum_{n=1}^{N} \left(-\frac{1}{2\sigma^2}(x_n - \mu)^2\right) \times \left(-\lambda \sigma\right)$$

4b)

When we discussed the normal-normal model for the unknown mean, we talked about the sufficient statistic being the sample average. This situation, an unknown standard deviation with a known mean, also has a sufficient statistic, but it is not the sample average. The sufficient statistic is defined for you in the equation block below:

$$v = \frac{1}{N} \sum_{n=1}^{N} \left((x_n - \mu)^2 \right)$$

PROBLEM Simplify your result from Problem 4a), by making use of the sufficient statistic, v. Why do you think the quantity v is referred to as being "sufficient"? Does your expression for the log-posterior depend on the individual observations after making use of v?

SOLUTION Write out the simplified expression and include your discussion.

$$p(\sigma \mid \mathbf{x}, \mu) \propto -\frac{v}{2\sigma^2 N} \times (-\lambda\sigma)$$

The quantity v is defined as sufficient because the infromation content can be represented by the difference of the sample mean and the population mean. After simplifying by including the sufficient statistic, the expression for the log-posterior no longer depends on the individual observations.

4c)

Let's see how the un-normalized log-posterior behaves under a particular set of assumptions. The code chunk below defines a sample size of $N = 21$, and a sufficient statistic of $v = 4.2$.

```r
N <- 21
v <- 4.2
```

In this problem, you will define a function `log_post_sigma_unnorm()` which has as input arguments, `sigma`, `N`, `v`, and `lambda`. The argument `sigma` is the σ value, `N` is the sample size, `v` is the sufficient statistic, and `lambda` is the prior rate parameter, λ.

PROBLEM Define the `log_post_sigma_unnorm()` function using the aguments and order of the arguments given in the problem statement.

```r
### set input arguments !!!
log_post_sigma_unnorm <- function(sigma, N, v, lambda)
{
  # your code here
  log_post <- (-v/(2*(sigma)^2*N))*(-lambda*sigma)
  return (log_post)
}
```

SOLUTION

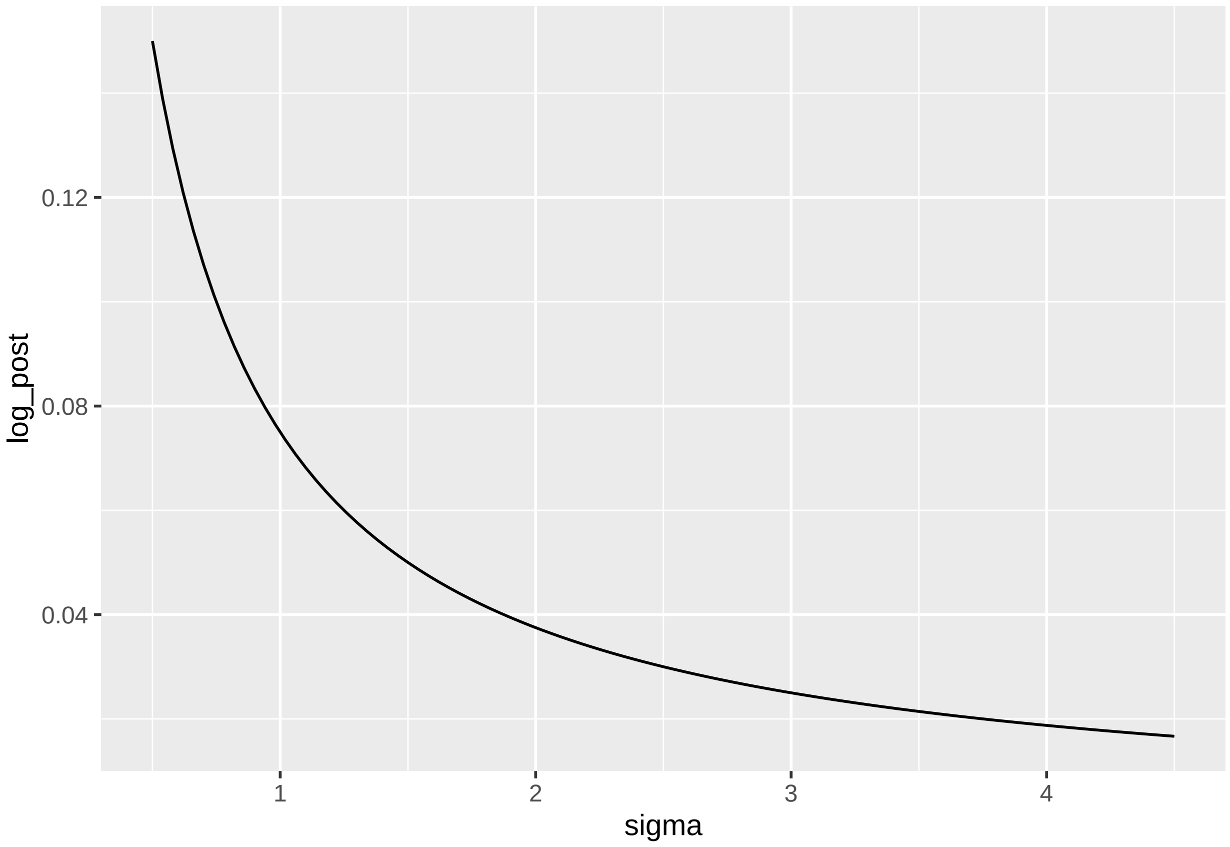
4d)

Let's visualize the log-posterior with respect σ. You will use the created `N` and `v` variables, and assume that the prior rate parameter is equal to 0.75.

PROBLEM Complete the code chunk below. Assign a vector of 101 evenly spaced values between 0.5 and 4.5 to the variable `sigma` within the `tibble`. Pipe the result to a `mutate()` call and calculate the log-posterior using the `log_post_sigma_unnorm()` function. The log-posterior should be saved to the variable `log_post`. Pipe the result to `ggplot()` and plot the `sigma` and `log_post` variables as the `x` and `y` aesthetics, respectively. Use a `geom_line()` geometric object.

Does the log-posterior have one or several modes? Does the shape of the log-posterior look like a parabola?

```
tibble::tibble(
  sigma = seq(from = 0.5, to = 4.5, length.out = 101)
) %>%
  mutate(log_post = log_post_sigma_unnorm(sigma, N, v, 0.75)) %>%
  ggplot(mapping = aes(x = sigma, y = log_post))+
  geom_line()
```



SOLUTION

Based on the graph, the log_posterior seems to have multiple values for the mode. The shape of the graph is also not very much parabolic as much as a strictly decreasing function.

4e)

The quadratic or Laplace approximation approximates the posterior distribution as a Gaussian. Although this can be a useful approximation, it is not appropriate for this type of situation. The likelihood noise σ has a natural lower bound of 0. A Gaussian distribution has no such constraints, and so allows all values from negative infinity to positive infinity. Therefore, even though the prior Exponential distribution respects the bound on σ the Laplace approximation allows the lower bound to be violated!

We need to make use of a transformation in order to overcome this limitation. Since the likelihood noise is lower bounded at 0, we will apply a log-transformation. The transformed variable, φ, will be equal to the log of σ. Thus, the transformation or link function, $g\left(\right)$, is equal to the log function:

$$\varphi = g\left(\sigma\right) = \log\left(\sigma\right)$$

The change-of-variables formula, written in terms of the log-density, is given below:

$$\log\left(p_{\varphi}\left(\varphi \mid \mathbf{x}, \mu\right)\right) = \log\left(p_{\sigma}\left(g^{-1}\left(\varphi\right) \mid \mathbf{x}, \mu\right)\right) + \log\left(\left|\frac{d}{d\varphi}\left(g^{-1}\left(\varphi\right)\right)\right|\right)$$

This may seem like an intimidating expression, but it simply says plug in the expression for σ in terms of φ (the inverse of the link function) into the log-posterior expression on σ. Then add in the log of the derivative of the inverse link function with respect to φ.

Let's tackle this problem by first determining the log of the derivative of the inverse link function.

PROBLEM The link function is the log. What is the inverse link function? Write down the expression for calculating σ as a function of φ. Then calculate the derivative of the inverse link function with respect to φ. Finally, write down the log of the derivative of the inverse link function.

SOLUTION First, the function for sigma in terms of varphi is:

$$\sigma = e^{\varphi}$$

Second, the derivative for the inverse link cuntion with respect to varphi is:

$$\frac{d}{d\varphi}\left(g^{-1}\left(\varphi\right)\right) = e^{\varphi}$$

lastly, the log of the derivative of the inverse link funciton is:

$$log(\frac{d}{d\varphi}\left(g^{-1}\left(\varphi\right)\right)) = \varphi$$

Use multiple equation blocks to answer each of the derivations.

4f)

To handle the substitution portion, you will define a new function `log_post_varphi_unnorm()` which accepts as input arguments, `varphi`, `N`, `v`, and `lambda`. The last three arguments are identical to those in `log_post_sigma_unnorm()`, while the first argument is the value of the transformed variable φ. Within the function, you must calculate `sigma` based on the supplied value of `varphi` using the correct inverse link function. With the value of `sigma` calculated, you can evaluate the log-likelihood just as you did in previously. You must then account for the log-derivative adjustment by correctly adding in the log of the derivative you calculated in Problem 4e).

PROBLEM Complete the code chunk below by calculating the quantities specified in the comments.

```r
### set the input arguments !!!
log_post_varphi_unnorm <- function(varphi, N, v, lambda)
{
  # back-calculate sigma given varphi
  sigma <- exp(varphi)

  # calculate the unnormalized log-posterior on sigma
  log_post <- log_post_sigma_unnorm(sigma, N, v, lambda) + varphi

  # account for the derivative adjustmet
  return (log_post)
}
```

SOLUTION

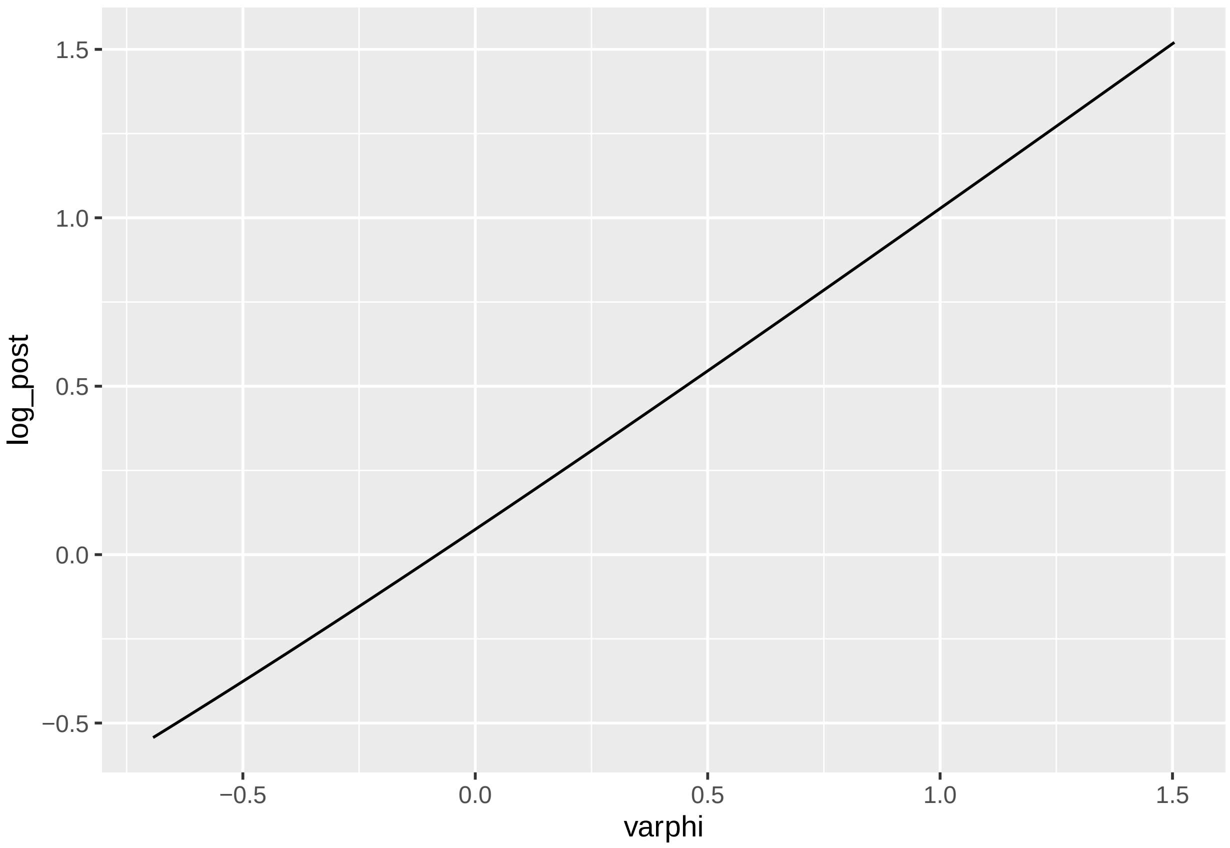
4g)

Now visualize the un-normalized log-posterior on the transformed variable φ. The code chunk below is similar to that used Problem 4d), except now you must define a vector of φ values instead of σ values. In Problem 4d), you calculated an evenly spaced vector of points between 0.5 and 4.5, in the σ space. Transform those bounds to the appropriate values in φ space and define a vector of 101 evenly spaced points between the transformed bounds. Pipe the result into `mutate()` and calculate the un-normalized log-posterior using the `log_post_varphi_unnorm()` function and save the result to the `log_post` variable. Pipe the result into `ggplot()` and visualize `varphi` and `log_post` as the `x` and `y` aesthetics with a `geom_line()`.

You will continue to use a prior rate parameter value of 0.75.

PROBLEM Visualize the un-normalized log-posterior on the transformed φ variable, following the instructions in the problem statement. Does the log-posterior appear more parabolic now?

```r
tibble::tibble(
  varphi = seq(from = log(0.5), to = log(4.5), length.out = 101)
) %>%
  mutate(log_post = log_post_varphi_unnorm(varphi, N, v, 0.75)) %>%
  ggplot(mapping = aes(x = varphi, y = log_post))+
  geom_line()
```



SOLUTION
The log__posterior does look more parabolic now after the transformation than before.