

# ISYE 6420: Midterm

aelhabr3

## 1. Emily, Car, Stock Market, Sweepstakes, Vacation and Bayes.

### Instructions

Emily is taking Bayesian Analysis course...

### Response

Below I give my response in terms of exact calculation, using R to perform calculations. In the notation and code that follows, I use `a`, `b`, `c` for Emily's grade; `bull` and `bear` to indicate bullish or bearish market; `yes` or `no` to indicate whether Emily's uncle buys here a car and whether she goes on vaca tion (to Redington Shores); and `win` or `lose` to indicate if she wins the lott ery (i.e. sweepstakes).

**a**

We want to find  $\Pr(\text{car}=\text{yes}|\text{vaca}=\text{yes})$ . This can be calculated using Bayes' theorem as follows.

$$\begin{aligned}\Pr(\text{car}=\text{yes}|\text{vaca}=\text{yes}) &= \frac{\Pr(\text{vaca}=\text{yes}|\text{car}=\text{yes}) \Pr(\text{car}=\text{yes})}{\Pr(\text{vaca}=\text{yes})} \\ &= \frac{\Pr(\text{vaca}=\text{yes},\text{car}=\text{yes})}{\Pr(\text{vaca}=\text{yes})}\end{aligned}$$

In the code that follows, I delineate all possible scenarios in a manner, calculating the joint probability for each. This is essentially like drawing a tree diagram or notating all scenarios such as follows.

Scenario A: `lott=win,grade=a,market=bull,car=yes,vaca=yes`  
Scenario B: `lott=win,grade=a,market=bear,car=yes,vaca=yes`  
...

```
library(tidyverse)
```

```
p_a <- 0.6
p_b <- 0.3
p_c <- 0.1
p_bull <- 0.5
p_bear <- 0.5
p_car_given_a_bull <- 0.8
p_car_given_b_bull <- 0.5
p_car_given_c_bull <- 0.2
p_car_given_a_bear <- 0.5
p_car_given_b_bear <- 0.3
p_car_given_c_bear <- 0.1
p_vaca_given_car <- 0.7
p_vaca_given_no_car <- 0.2
p_lott <- 0.001
p_vaca_given_lott <- 0.99
```

```

states <-
  crossing(
    lott = c('win', 'lose'),
    grade = c('a', 'b', 'c'),
    market = c('bull', 'bear'),
    car = c('yes', 'no'),
    vaca = c('yes', 'no')
  ) %>%
  mutate_at(
    vars(lott),
    list(p_lott = ~case_when(
      . == 'win' ~ p_lott,
      . == 'lose' ~ 1 - p_lott
    ))
  ) %>%
  mutate_at(
    vars(grade),
    list(p_grade = ~case_when(
      . == 'a' ~ 0.6,
      . == 'b' ~ 0.3,
      . == 'c' ~ 0.1
    ))
  ) %>%
  mutate_at(
    vars(market),
    list(p_market = ~case_when(
      . == 'bull' ~ 0.5,
      . == 'bear' ~ 0.5
    ))
  ) %>%
  mutate_at(
    vars(market),
    list(p_market = ~0.5)
  ) %>%
  mutate_at(
    vars(car),
    list(p_car = ~case_when(
      grade == 'a' & market == 'bull' & . == 'yes' ~ p_car_given_a_bull,
      grade == 'a' & market == 'bull' & . == 'no' ~ 1 - p_car_given_a_bull,
      grade == 'a' & market == 'bear' & . == 'yes' ~ p_car_given_a_bear,
      grade == 'a' & market == 'bear' & . == 'no' ~ 1 - p_car_given_a_bear,
      grade == 'b' & market == 'bull' & . == 'yes' ~ p_car_given_b_bull,
      grade == 'b' & market == 'bull' & . == 'no' ~ 1 - p_car_given_b_bull,
      grade == 'b' & market == 'bear' & . == 'yes' ~ p_car_given_b_bear,
      grade == 'b' & market == 'bear' & . == 'no' ~ 1 - p_car_given_b_bear,
      grade == 'c' & market == 'bull' & . == 'yes' ~ p_car_given_c_bull,
      grade == 'c' & market == 'bull' & . == 'no' ~ 1 - p_car_given_c_bull,
      grade == 'c' & market == 'bear' & . == 'yes' ~ p_car_given_c_bear,
      grade == 'c' & market == 'bear' & . == 'no' ~ 1 - p_car_given_c_bear
    ))
  ) %>%
  mutate_at(
    vars(vaca),

```

```

list(p_vaca = ~case_when(
  lott == 'win' & car == 'yes' & . == 'yes' ~ p_vaca_given_lott,
  lott == 'win' & car == 'yes' & . == 'no' ~ 1 - p_vaca_given_lott,
  lott == 'win' & car == 'no' & . == 'yes' ~ p_vaca_given_lott,
  lott == 'win' & car == 'no' & . == 'no' ~ 1 - p_vaca_given_lott,
  lott == 'lose' & car == 'yes' & . == 'yes' ~ p_vaca_given_car,
  lott == 'lose' & car == 'yes' & . == 'no' ~ 1 - p_vaca_given_car,
  lott == 'lose' & car == 'no' & . == 'yes' ~ p_vaca_given_no_car,
  lott == 'lose' & car == 'no' & . == 'no' ~ 1 - p_vaca_given_no_car
))
) %>%
mutate(
  p = p_lott * p_grade * p_market * p_car * p_vaca
) %>%
mutate_at(vars(lott), ~factor(., levels = c('lose', 'win')))) %>%
mutate_at(vars(grade), ~factor(.)) %>%
mutate_at(vars(market), ~factor(., levels = c('bull', 'bear')))) %>%
mutate_at(vars(car, vaca), ~factor(., levels = c('yes', 'no')))) %>%
  arrange(lott, grade, market, car, vaca)
states

```

```
## # A tibble: 48 x 11
##   lott grade market car vaca p_lott p_grade p_market p_car p_vaca
##   <fct> <fct> <fct> <fct> <fct> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 lose a bull yes yes 0.999 0.6 0.5 0.8 0.7
## 2 lose a bull yes no 0.999 0.6 0.5 0.8 0.3
## 3 lose a bull no yes 0.999 0.6 0.5 0.2000 0.2
## 4 lose a bull no no 0.999 0.6 0.5 0.2000 0.8
## 5 lose a bear yes yes 0.999 0.6 0.5 0.5 0.7
## 6 lose a bear yes no 0.999 0.6 0.5 0.5 0.3
## 7 lose a bear no yes 0.999 0.6 0.5 0.5 0.2
## 8 lose a bear no no 0.999 0.6 0.5 0.5 0.8
## 9 lose b bull yes yes 0.999 0.3 0.5 0.5 0.7
## 10 lose b bull yes no 0.999 0.3 0.5 0.5 0.3
## 11 lose b bull no yes 0.999 0.3 0.5 0.5 0.2
## 12 lose b bull no no 0.999 0.3 0.5 0.5 0.8
## 13 lose b bear yes yes 0.999 0.3 0.5 0.3 0.7
## 14 lose b bear yes no 0.999 0.3 0.5 0.3 0.3
## 15 lose b bear no yes 0.999 0.3 0.5 0.7 0.2
## 16 lose b bear no no 0.999 0.3 0.5 0.7 0.8
## 17 lose c bull yes yes 0.999 0.1 0.5 0.2 0.7
## 18 lose c bull yes no 0.999 0.1 0.5 0.2 0.3
## 19 lose c bull no yes 0.999 0.1 0.5 0.8 0.2
## 20 lose c bull no no 0.999 0.1 0.5 0.8 0.8
## 21 lose c bear yes yes 0.999 0.1 0.5 0.1 0.7
## 22 lose c bear yes no 0.999 0.1 0.5 0.1 0.3
## 23 lose c bear no yes 0.999 0.1 0.5 0.9 0.2
## 24 lose c bear no no 0.999 0.1 0.5 0.9 0.8
## 25 win a bull yes yes 0.001 0.6 0.5 0.8 0.99
## 26 win a bull yes no 0.001 0.6 0.5 0.8 0.01
## 27 win a bull no yes 0.001 0.6 0.5 0.2000 0.99
## 28 win a bull no no 0.001 0.6 0.5 0.2000 0.01
## 29 win a bear yes yes 0.001 0.6 0.5 0.5 0.99
## 30 win a bear yes no 0.001 0.6 0.5 0.5 0.01
## 31 win a bear no yes 0.001 0.6 0.5 0.5 0.99
## 32 win a bear no no 0.001 0.6 0.5 0.5 0.01
## 33 win b bull yes yes 0.001 0.3 0.5 0.5 0.99
## 34 win b bull yes no 0.001 0.3 0.5 0.5 0.01
## 35 win b bull no yes 0.001 0.3 0.5 0.5 0.99
## 36 win b bull no no 0.001 0.3 0.5 0.5 0.01
## 37 win b bear yes yes 0.001 0.3 0.5 0.3 0.99
## 38 win b bear yes no 0.001 0.3 0.5 0.3 0.01
## 39 win b bear no yes 0.001 0.3 0.5 0.7 0.99
## 40 win b bear no no 0.001 0.3 0.5 0.7 0.01
## 41 win c bull yes yes 0.001 0.1 0.5 0.2 0.99
## 42 win c bull yes no 0.001 0.1 0.5 0.2 0.01
## 43 win c bull no yes 0.001 0.1 0.5 0.8 0.99
## 44 win c bull no no 0.001 0.1 0.5 0.8 0.01
## 45 win c bear yes yes 0.001 0.1 0.5 0.1 0.99
## 46 win c bear yes no 0.001 0.1 0.5 0.1 0.01
## 47 win c bear no yes 0.001 0.1 0.5 0.9 0.99
## 48 win c bear no no 0.001 0.1 0.5 0.9 0.01
## # ... with 1 more variable: p <dbl>
```

The data.frame states describes all possible scenarios (24 total) and probabilities. (Note that the p\_\* columns describe the probabilities, and the p column is the product of these.) This data.frame is essentially equivalent to a probability tree diagram partitioning out the entire sample space.

In case the reader has any doubts about the probabilities in the data.frame, we can do a quick check of the code and show that the probabilities of all scenarios sum up to 1.

```
states %>% summarise_at(vars(p), sum) %>% pull(p)

## [1] 1
```

*Also, before moving on, I should note that we can disregard the case where Emily goes on vacation after winning the lottery because it is independent of the non-lottery scenarios. (“Independently, Emily may be a lucky winner...”.)*

Now, to begin filling in numbers for our Bayes’ equation above, let’s begin by finding  $\Pr(\text{vaca}|\text{yes})$ . Filtering for just the vaca == 'yes' scenarios, we find that there are 24 records.

```
states %>% filter(vaca == 'yes')

## # A tibble: 24 x 11
##   lott grade market car vaca p_lott p_grade p_market p_car p_vaca
##   <fct> <fct> <fct> <fct> <fct> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 lose a bull yes yes 0.999 0.6 0.5 0.8 0.7
## 2 lose a bull no yes 0.999 0.6 0.5 0.2000 0.2
## 3 lose a bear yes yes 0.999 0.6 0.5 0.5 0.7
## 4 lose a bear no yes 0.999 0.6 0.5 0.5 0.2
## 5 lose b bull yes yes 0.999 0.3 0.5 0.5 0.7
## 6 lose b bull no yes 0.999 0.3 0.5 0.5 0.2
## 7 lose b bear yes yes 0.999 0.3 0.5 0.3 0.7
## 8 lose b bear no yes 0.999 0.3 0.5 0.7 0.2
## 9 lose c bull yes yes 0.999 0.1 0.5 0.2 0.7
## 10 lose c bull no yes 0.999 0.1 0.5 0.8 0.2
## # ... with 14 more rows, and 1 more variable: p <dbl>
```

And we can view the marginal probabilities of the going on vacation or not by summing by group.

```
states_vaca <-
  states %>%
  group_by(vaca) %>%
  summarise_at(vars(p), sum) %>%
  ungroup()
states_vaca
```

```
## # A tibble: 2 x 2
##   vaca      p
##   <fct>   <dbl>
## 1 yes    0.4630
## 2 no     0.5370
```

```
p_vaca <-
  states_vaca %>%
  filter(vaca == 'yes') %>%
  pull(p)
p_vaca
```

```
## [1] 0.4630275
```

Summing up the probabilities for the scenarios where Emily does go on vacation gives us  $P(\text{vaca}|\text{yes}) = 0.4630$ , which is the denominator of the Bayes' equation noted above.

Next, we can get the joint probabilities of Emily getting a car and going on vacation.

```
p_car_and_vaca <-
  states %>%
  filter(car == 'yes', vaca == 'yes') %>%
  summarise_at(vars(p), sum) %>%
  pull(p)
p_car_and_vaca
```

```
## [1] 0.3676522
```

We see that the probability of  $\Pr(\text{car}=\text{yes}, \text{vaca}=\text{yes}) = 0.368$ , which is a term we need for the numerator of our Bayes' equation.

Finally, since we already have  $\Pr(\text{vaca}=\text{yes}|\text{car}=\text{yes}) = 0.7$ , we can now “plug-and-chug” to calculate  $\Pr(\text{car}=\text{yes}|\text{vaca}=\text{yes})$ .

```
p_car_given_vaca <- p_car_and_vaca / p_vaca
p_car_given_vaca
```

```
## [1] 0.7940182
```

**Thus, we find that  $\Pr(\text{car}=\text{yes}|\text{vaca}=\text{yes}) = 0.7940$ .**

$$\begin{aligned}\Pr(\text{car}=\text{yes}|\text{vaca}=\text{yes}) &= \frac{\Pr(\text{vaca}=\text{yes}, \text{car}=\text{yes})}{\Pr(\text{vaca}=\text{yes})} \\ &= \frac{(0.3676522)}{0.4630} \\ &= 0.7940\end{aligned}$$

**b**

We want to find  $\Pr(\text{lott}=\text{win}|\text{vaca}=\text{yes})$ . This can be calculated using Bayes' theorem as follows.

$$\begin{aligned}\Pr(\text{lott}=\text{win}|\text{vaca}=\text{yes}) &= \frac{\Pr(\text{vaca}=\text{yes}|\text{lott}=\text{win}) \Pr(\text{lott}=\text{win})}{\Pr(\text{vaca}=\text{yes})} \\ &= \frac{\Pr(\text{vaca}=\text{yes}, \text{lott}=\text{win})}{\Pr(\text{vaca}=\text{yes})}\end{aligned}$$

```
p_vaca_and_lott <-
  states %>%
  mutate(rn = row_number()) %>%
  filter(vaca == 'yes', lott == 'win') %>%
  summarise_at(vars(p), sum) %>%
  pull(p)
p_vaca_and_lott
```

```
## [1] 0.00099
```

```
p_lott_given_vaca <- p_vaca_and_lott / p_vaca
p_lott_given_vaca
```

```
## [1] 0.002138102
```

$$\begin{aligned}\Pr(\text{lott}=\text{win}|\text{vaca}=\text{yes}) &= \frac{\Pr(\text{vaca}=\text{yes}, \text{lott}=\text{win})}{\Pr(\text{vaca}=\text{yes})} \\ &= \frac{(9.9 \times 10^{-4})}{(0.4630)} \\ &= 0.0021.\end{aligned}$$

**Thus, we have shown that  $\Pr(\text{lott}=\text{win}|\text{vaca}=\text{yes}) = 0.0021$ .**

**c**

We want to find  $\Pr(\text{grade}=\text{b}|\text{vaca}=\text{yes})$ . This can be calculated using Bayes' theorem as follows.



$$\Pr(\text{grade}=\text{b}|\text{vaca}=\text{yes}) = \frac{\Pr(\text{vaca}=\text{yes}|\text{grade}=\text{b}) \Pr(\text{grade}=\text{b})}{\Pr(\text{vaca}=\text{yes})}$$

We can calculate marginal probabilities for each possibility of `grade` given `vaca` in a similar way done before in part a (for just `vaca` )

```
p_vaca_given_grade <-
  states %>%
  group_by(vaca, grade) %>%
  summarise_at(vars(p), sum) %>%
  ungroup() %>%
  group_by(grade) %>%
  mutate_at(vars(p), ~(. / sum(.))) %>%
  ungroup() %>%
  arrange(grade, vaca)
p_vaca_given_grade
```

```
## # A tibble: 6 x 3
##   vaca grade      p
##   <fct> <fct> <dbl>
## 1 yes   a      0.5255
## 2 no    a      0.4745
## 3 yes   b      0.4006
## 4 no    b      0.5994
## 5 yes   c      0.2757
## 6 no    c      0.7243
```

and isolate just the marginal probability that we are interested in.

```
p_vaca_given_b <-
  p_vaca_given_grade %>%
  filter(vaca == 'yes', grade == 'b') %>%
  pull(p)
p_vaca_given_b
```

```
## [1] 0.40059
```

We see that the probability of  $\Pr(\text{vaca}=\text{yes}|\text{grade}=\text{b}) = 0.4$ , which is a term we need for the numerator of our Bayes' equation.

Finally, since we already have  $\Pr(\text{grade}=\text{b}) = 0.3$ , we can now calculate  $\Pr(\text{grade}=\text{b}|\text{vaca}=\text{yes})$ .

```
p_b_given_vaca <- (p_vaca_given_b * p_b) / p_vaca
p_b_given_vaca
```

```
## [1] 0.2595461
```

Thus, we find that  $\Pr(\text{grade}=\text{b}|\text{vaca}=\text{yes}) = \mathbf{0.2595}$ .

$$\begin{aligned}\Pr(\text{grade}=\text{b}|\text{vaca}=\text{yes}) &= \frac{\Pr(\text{vaca}=\text{yes}|\text{grade}=\text{b}) \Pr(\text{grade}=\text{b})}{\Pr(\text{vaca}=\text{yes})} \\ &= \frac{(0.4)(0.3)}{(0.4630)} \\ &= 0.2595\end{aligned}$$

**d**

We want to find  $\Pr(\text{market}=\text{bear}|\text{vaca}=\text{yes})$ . This can be calculated using Bayes' theorem as follows.

$$\Pr(\text{market}=\text{bear}|\text{vaca}=\text{yes}) = \frac{\Pr(\text{vaca}=\text{yes}|\text{market}=\text{bear}) \Pr(\text{market}=\text{bear})}{\Pr(\text{vaca}=\text{yes})}$$

```
states_bear <-  
  states %>%  
  group_by(vaca, market) %>%  
  summarise_at(vars(p), sum) %>%  
  ungroup() %>%  
  group_by(market) %>%  
  mutate_at(vars(p), ~(. / sum(.))) %>%  
  ungroup() %>%  
  arrange(market, vaca)  
states_bear
```

```
## # A tibble: 4 x 3  
##   vaca market      p  
##   <fct> <fct>   <dbl>  
## 1 yes   bull    0.5255  
## 2 no    bull    0.4745  
## 3 yes   bear    0.4006  
## 4 no    bear    0.5994
```

```
p_vaca_given_bear <-  
  states_bear %>%  
  filter(vaca == 'yes', market == 'bear') %>%  
  pull(p)  
p_vaca_given_bear
```

```
## [1] 0.40059
```

We see that the probability of  $\Pr(\text{vaca}=\text{yes}|\text{market}=\text{bear}) = 0.4$ , which is a term we need for the numerator of our Bayes' equation.

Finally, since we already have  $\Pr(\text{market}=\text{bear}) = 0.5$ , we can now calculate  $\Pr(\text{car}=\text{yes}|\text{vaca}=\text{yes})$ .

```
p_bear_given_vaca <- (p_vaca_given_bear * p_bear) / p_vaca
p_bear_given_vaca
```

```
## [1] 0.4325769
```

**Thus, we find that  $\Pr(\text{market}=\text{bear}|\text{vaca}=\text{yes}) = 0.4326$ .**

$$\begin{aligned}\Pr(\text{market}=\text{bear}|\text{vaca}=\text{yes}) &= \frac{\Pr(\text{vaca}=\text{yes}|\text{market}=\text{bear}) \Pr(\text{market}=\text{bear})}{\Pr(\text{vaca}=\text{yes})} \\ &= \frac{(0.4)(0.5)}{(0.4630)} \\ &= 0.4326\end{aligned}$$

## 2. Trials until Fourth Success.

### Instructions

*The number of failures until the fourth success in a series of independent trials ...*

### Response

Given  $X = (X_1, \dots, X_n)$  is a sample from  $\mathcal{NB}(m, p)$  and  $p \sim \mathcal{Be}(a, b)$ , the posterior for  $p$  follows a  $\mathcal{Be}(a + mn, b + \sum_{i=1}^n x_i)$  distribution. The proof of this relationship is provided in an exercise from earlier in the semester. It goes as follows.

*Proof.*

Recall that the pmf of the negative binomial distribution  $\mathcal{NB}(m, p)$  (which model the number of failures before the  $m$ th success in  $n$  Bernoulli experiments) is given by

$$f(x) = \binom{m+x-1}{x} p^m (1-p)^x, x = 0, 1, 2, \dots$$

and that the pdf of the Beta distribution  $\mathcal{Be}(a, b)$  is proportional to

$$f(x) = \frac{1}{B(x)} x^{a-1} (1-x)^{b-1}, 0 \leq x \leq 1$$

(where  $B$  is the beta function).

If the random variable  $X$  comes from  $\mathcal{NB}(m, p)$ , then the likelihood  $\mathcal{L}(x)$  is proportional to  $p^{mn} (1-p)^{\sum_{i=1}^n x_i}$ ; and if the prior  $\pi(p)$  comes from  $\mathcal{Be}(a, b)$ , then  $\pi(p) \propto p^{a-1} (1-p)^{b-1}$ . This results in a posterior proportional to  $p^{mn+a-1} (1-p)^{\sum_{i=1}^n x_i+b-1}$ .

$$\begin{array}{ccccc} \text{likelihood} & \times & \text{prior} & \propto & \text{posterior} \\ p^{mn} (1-p)^{\sum_{i=1}^n x_i} & \times & p^{a-1} (1-p)^{b-1} & \propto & p^{mn+a-1} (1-p)^{\sum_{i=1}^n x_i+b-1}. \end{array}$$

This is a kernel of the beta distribution  $\mathcal{Be}(a + mn, b + \sum_{i=1}^n x_i)$ .  $\square$

For this problem set-up, note that  $m = 4$ ,  $n = 11$ , and  $x = [5, 2, 2, 0, 1, 4, 3, 1, 2, 5, 0, 7, 1]$ .

In the subsections that follow, I calculate (1) the Bayes estimator  $p_{\text{Bayes}}$ , (2) the 95% credible set for  $p$ , and (3) the posterior probability of hypothesis  $H : p \geq 0.8$  (along with several other things not explicitly required, such as the Bayes factor  $B_{01}$ ). Each of these concepts deserve some brief discussion beforehand.

Regarding (1), the Bayes estimator  $p_{\text{Bayes}}$  is simply the expected value of  $p$  under the posterior distribution, i.e. the posterior mean. The expected value of arbitrary random variable  $x$  for a Beta distribution is  $E[x] = \frac{a}{a+b}$ . Thus, for this setup,  $p_{\text{Bayes}} = E[p_{\text{posterior}}] = \frac{a+mn}{b+\sum_{i=1}^n x_i}$ .

Regarding (2), a credible set  $C = [L, U]$  defines the parameter space of the posterior distribution with  $1 - \alpha$  credibility. (In this problem, we let  $\alpha = 0.05$ ). An equitailed credible set is just one form of a credible set. Formally, it is calculated as follows.

$$\begin{array}{l} \int_{-\infty}^L \pi(p|x) dp \leq \frac{\alpha}{2}, \int_U^{+\infty} \pi(p|x) dp \leq \frac{\alpha}{2} \\ \text{s.t. } \Pr(p \in [L, U] | X) \geq 1 - \alpha \end{array}$$

Regarding (3), to test the one-sided (null) hypothesis  $H_0 : p \geq p_{\text{split}}$  (here,  $p_{\text{split}} = 0.8$ —against the (alternative) alternative  $H_1 : p < p_{\text{split}}$ , we simply choose the hypothesis with larger posterior probability. We can go further by calculating the Bayes Factor  $B_{01}$  in favor of  $H_0$  (or the Bayes Factor  $B_{10}$  in favor of  $H_1$ ) and calibrate our deduction in favor or/against  $H_0$  using tables like the one shown in lecture.

Value	Evidence against $H_0$
$0 \leq \log_{10} B_{10} \leq 0.5$	Poor
$0.5 < \log_{10} B_{10} \leq 1$	Substantial
$1 < \log_{10} B_{10} \leq 1.5$	Strong
$1.5 < \log_{10} B_{10} \leq 2$	Very Strong
$\log_{10} B_{10} > 2$	Decisive

The following code implements all parts of the instructions. Specifically `do_q2()` calls other functions to compute the required components—(1) `compute_beta_mu()` for  $p_{\text{Bayes}}$ , `compute_equi_credible_set_beta()` for the 95% credible set, and (3) `compute_p_h1()` for the hypothesis test. Discussion of the findings for each set of priors is included in the following subsections.

```

set.seed(42)
m <- 4
x <- c(5, 2, 2, 0, 1, 4, 3, 5, 0, 7, 1)
n <- length(x) # This is 11.
p_split <- 0.8
# p_split <- 0.6

.compute_a_post <- function(a, m, n) {
  a + m * n
}

compute_a_post <- function(a, .m = m, .n = n) {
  .compute_a_post(a, .m, .n)
}

.compute_b_post <- function(b, x) {
  b + sum(x)
}

compute_b_post <- function(b, .x = x) {
  .compute_b_post(b = b, x = .x)
}

compute_beta_mu <- function(a, b) {
  a / (a + b)
}

compute_equi_credible_set_beta <- function(a, b, level = 0.95) {
  q_buffer <- (1 - level) / 2
  q_l <- (1 - level) - q_buffer
  q_u <- level + q_buffer
  res <-
    c(
      l = qbeta(q_l, a, b),
      u = qbeta(q_u, a, b)
    )
  res
}

# .compute_equi_credible_set <- function(x, level = 0.95) {
#   q_buffer <- (1 - level) / 2
#   q_l <- (1 - level) - q_buffer
#   q_u <- level + q_buffer
#   res <-
#     c(
#       l = quantile(x, q_l),
#       u = quantile(x, q_u)
#     )
#   res
# }
#
compute_equi_credible_set <- function(.x = x) {

```

```

# .compute_equi_credible_set(x = .x)
# }

.compute_p_h1 <- function(a, b, p) {
  pbeta(p, a, b, lower.tail = TRUE)
}

compute_p_h1 <- function(a, b, .p_split = p_split) {
  .compute_p_h1(a, b, .p_split)
}

do_q2 <- function(a_0, b_0) {
  a_1 <- compute_a_post(a_0)
  b_1 <- compute_b_post(b_0)
  mu_prior <- compute_beta_mu(a_0, b_0)
  mu_post <- compute_beta_mu(a_1, b_1)
  # credible_set <- compute_equi_credible_set()
  credible_set_beta <- compute_equi_credible_set_beta(a_1, b_1)
  p_prior_h1 <- compute_p_h1(a_0, b_0)
  p_prior_h0 <- 1 - p_prior_h1
  p_post_h1 <- compute_p_h1(a_1, b_1)
  p_post_h0 <- 1 - p_post_h1
  b_01_num <- p_post_h0 / p_post_h1
  b_01_den <- p_prior_h0 / p_prior_h1
  b_01 <- b_01_num / b_01_den
  b_10_num <- p_post_h1 / p_post_h0
  b_10_den <- p_prior_h1 / p_prior_h0
  b_10 <- b_10_num / b_10_den
  res <-
    list(
      # a_prior = a_0,
      # b_prior = b_0,
      a_post = a_1,
      b_post = b_1,
      mu_prior = mu_prior,
      mu_post = mu_post,
      credible_set_l = credible_set_beta['l'],
      credible_set_u = credible_set_beta['u'],
      p_prior_h0 = p_prior_h0,
      p_prior_h1 = p_prior_h1,
      p_post_h0 = p_post_h0,
      p_post_h1 = p_post_h1,
      # b_01 = b_01,
      # b_10 = b_10,
      b_01_log10 = log10(b_01),
      b_10_log10 = log10(b_10)
    )
  res
}

```

The results in this subsection are just for the priors  $a = b = 1$ . (The following subsections show the results for the other sets of priors.)

```
a_0_a <- 1
b_0_a <- a_0_a
```

```
res_a <- do_q2(a_0_a, b_0_a)
res_a
```

```
## $a_post
## [1] 45
##
## $b_post
## [1] 31
##
## $mu_prior
## [1] 0.5
##
## $mu_post
## [1] 0.5921053
##
## $credible_set_l
##      1
## 0.4803705
##
## $credible_set_u
##      u
## 0.6992464
##
## $p_prior_h0
## [1] 0.2
##
## $p_prior_h1
## [1] 0.8
##
## $p_post_h0
## [1] 1.996161e-05
##
## $p_post_h1
## [1] 0.99998
##
## $b_01_log10
## [1] -4.097736
##
## $b_10_log10
## [1] 4.097736
```

The R package {R2OpenBUGS} was also used to computer the posterior probabilities. See the code and output below.



```
model <- function() {
  p ~ dbeta(a, b)
  for(i in 1:n){
    x[i] ~ dnegbin(p, m)
  }
  p_h_0 <- step(p - 0.8)
  p_h_1 <- step(0.8 - p)
}
data <- list(a = 0.5, b = 0.5, m = 4, n = 11, x = c(5, 2, 2, 0, 1, 4, 3, 5, 0, 7, 1))
inits <- NULL
params <- c('p_h_0', 'p_h_1', 'p')
res_sim <-
  R2OpenBUGS::bugs(
    data = data,
    inits = inits,
    model.file = model,
    parameters.to.save = params,
    DIC = FALSE,
    # debug = TRUE,
    n.chains = 1,
    n.iter = 10000,
    n.burnin = 1000
  )
res_sim$summary
```

	mean	sd	2.5%	25%	50%	75%	97.5%
p_h_0	0.0000000	0.0000000	0.0000	0.0000	0.0000	0.0000000	0.0000000
p_h_1	1.0000000	0.0000000	1.0000	1.0000	1.0000	1.0000000	1.0000000
p	0.5922603	0.0561095	0.4783	0.5552	0.5936	0.630625	0.7009025

**For (1), we find that  $p_{\text{Bayes}} = 0.5921$  (corresponding to `mu_post` in `res_a`).**  
**For (2), we find that the 95% credible set  $C = [0.4804, 0.6992]$  (corresponding to `credible_set_l` and `credible_set_u` respectively).**  
**For (3), we find that the posterior probability of the hypothesis  $H_1 : p < 0.8$  is  $p_{H_1} = 1.0000$  (corresponding to `p_post_h1`), which is much larger than that for the the hypothesis  $H_0 : p \geq 0.8$ ,  $p_{H_0} = 0.0000$  (corresponding to `p_post_h0`).**

Furthermore, we note that the log of the Bayes Factor  $B_{10}$  (in favor of  $H_1$  compared to  $H_0$ ) provides decisive evidence against  $H_0$ —or, equivalently, in favor of  $H_1$ —because  $\log_{10}(B_{10}) > 2$  (and because  $\log_{10}(B_{01}) < 0.5$ ). Also, we observe that the upper bound of the equitailed 95% credible interval—0.6992—is smaller than the hypothesis test probability, so it is not unsurprising that the conclusion to be made is so definitive.

See the code and output below for the same calculations performed in part a for priors  $a = b = 0.5$

```
a_0_b <- 0.5  
b_0_b <- a_0_b
```

```
res_b <- do_q2(a_0_b, b_0_b)  
res_b
```

```
## $a_post  
## [1] 44.5  
##  
## $b_post  
## [1] 30.5  
##  
## $mu_prior  
## [1] 0.5  
##  
## $mu_post  
## [1] 0.5933333  
##  
## $credible_set_l  
##      1  
## 0.4808762  
##  
## $credible_set_u  
##      u  
## 0.7010734  
##  
## $p_prior_h0  
## [1] 0.2951672  
##  
## $p_prior_h1  
## [1] 0.7048328  
##  
## $p_post_h0  
## [1] 2.479207e-05  
##  
## $p_post_h1  
## [1] 0.9999752  
##  
## $b_01_log10  
## [1] -4.227659  
##  
## $b_10_log10  
## [1] 4.227659
```

Again, {R2openBUGS} was used to check the results. (The code is the same as in part a, with the exception of `a` and `b` variables in the data.)

	mean	sd	2.5%	25%	50%	75%	97.5%
p_h_0	0.0000000	0.0000000	0.0000000	0.0000	0.0000	0.0000	0.000000
p_h_1	1.0000000	0.0000000	1.0000000	1.0000	1.0000	1.0000	1.000000
p	0.5936116	0.0563999	0.4795975	0.5563	0.5949	0.6323	0.702805

For (1), we find that  $p_{\text{Bayes}} = 0.5933$ .

For (2), we find that the 95% credible set  $C = [0.4809, 0.7011]$ .

F For (3), we find that the posterior probability of the hypothesis  $H_1$  is  $p_{H_1} = 1.0000$ , which is much larger than that for the the hypothesis  $H_0$ ,  $p_{H_0} = 0.0000$ .

The additional observations made about the Bayes Factor and the upper credible set bound in part a also apply here.

### c

See the code and output below for the same calculations performed in part a for priors  $a = 9$  and  $b = 1$ .

```
a_0_c <- 9
b_0_c <- 1
```

```
res_c <- do_q2(a_0_c, b_0_c)
res_c
```

```
## $a_post
## [1] 53
##
## $b_post
## [1] 31
##
## $mu_prior
## [1] 0.9
##
## $mu_post
## [1] 0.6309524
##
## $credible_set_l
##      1
## 0.5257093
##
## $credible_set_u
##      u
## 0.7302874
##
## $p_prior_h0
## [1] 0.8657823
##
## $p_prior_h1
## [1] 0.1342177
##
## $p_post_h0
## [1] 0.0001926464
##
## $p_post_h1
## [1] 0.9998074
##
## $b_01_log10
## [1] -4.524754
##
## $b_10_log10
## [1] 4.524754
```

And the output from {R2openBUGS} is shown below.

	mean	sd	2.5%	25%	50%	75%	97.5%
p_h_0	0.0001111	0.0105409	0.0000000	0.0000	0.00000	0.0000	0.0000000
p_h_1	0.9998889	0.0105409	1.0000000	1.0000	1.00000	1.0000	1.0000000
p	0.6312816	0.0521394	0.5269975	0.5965	0.63235	0.6676	0.7291025

**For (1), we find that  $p_{\text{Bayes}} = 0.6310$ .**

**For (2), we find that the 95% credible set  $C = [0.5257, 0.7303]$ .**

**For (3), we find that the posterior probability of the hypothesis  $H_1$  is  $p_{H_1} = 0.9998$ , which is much larger than that for the the hypothesis  $H_0$ ,  $p_{H_0} = 0.0002$ .**

Again, the additional observations made about the Bayes Factor and the upper credible set bound in part a also apply here.

Thus, for all three prior sets, we find that evidence in favor of  $H_1 : p < 0.8$  is decisive. Really, this is unsurprising. Note that the observed mean is 0.5946.

```
n_success <- n * m
n_fail <- sum(x)
n_total <- n_success + n_fail
mu_actual <- n_success / n_total
mu_actual
```

```
## [1] 0.5945946
```

Then, given the flat priors of (a) and (b), we should have expected that the posterior mean would not be much different. And the prior set for (c) is not all that “strong” either, so we should have expected similar results.

### 3. Penguins.

#### Instructions

*A researcher is interested in testing ...*

#### Response

In what follows I derive the full conditional expressions for  $\mu$  and  $\tau$ , closely following the guide provided by the Gibbs handout from class.

Per the instructions, we make the assumption that the measurements  $y_1 \dots y_n$  come from the normal distribution  $\mathcal{N}(\mu, 1/\tau)$ . (Note that  $n = 14$  here.) Furthermore, we assume  $\mu \sim \mathcal{N}(\mu_0, 1/\tau_0)$  (where  $\mu_0 = 45$  and  $\tau_0 = 4$ , per the instructions), and the precision parameter  $\tau \sim \mathcal{Ga}(a_0, b_0)$  (where  $a_0 = 4$  and  $b_0 = 2$ ).

Now, in preparation of expressing the joint distribution, we define the likelihood of  $\mu$  as

$$\mathcal{L}(\mu|y_1, \dots, y_n) = \prod_{i=1}^n f(y_i|\mu, \tau) \propto \tau^{n/2} \exp\left\{-\frac{\tau}{2} \sum_{i=1}^n (y_i - \mu)^2\right\}.$$

Thus, the joint distribution is

$$\begin{aligned} f(y, \mu, \tau) &= \left\{ \prod_{i=1}^n f(y_i | \mu, \tau) \right\} \pi(\mu) \pi(\tau) \\ &\propto \tau^{n/2} \exp \left\{ -\frac{\tau}{2} \sum_{i=1}^n (y_i - \mu)^2 \right\} \exp \left\{ -(\mu - \mu_0)^2 / 2 \right\} \tau^{a_0-1} \exp \{-b_0 \tau\} \end{aligned}$$

After selecting the terms from the joint distribution expression that contain  $\mu$  and normalizing, we find that

$$\begin{aligned} \pi(\mu | \tau, y) &\propto \exp \left\{ -\frac{\tau}{2} \sum_{i=1}^n (y_i - \mu)^2 \right\} \exp \left\{ -\frac{\tau_0}{2} (\mu - \mu_0)^2 \right\} \\ &\propto \exp \left\{ -\frac{1}{2} (\tau_0 + n\tau) \left( \mu - \frac{\tau \sum y_i + \tau_0 \mu_0}{\tau_0 + n\tau} \right)^2 \right\} \end{aligned}$$

which is the kernel for the  $\mathcal{N} \left( \frac{\tau \sum y_i + \tau_0 \mu_0}{\tau_0 + n\tau}, \frac{1}{\tau_0 + n\tau} \right)$  distribution. Plugging in the provided numbers for  $y_i, \dots, y_n, \mu_0, \tau_0$  and  $n$ , we can write the full conditional for  $\mu$  explicitly as

$$\begin{aligned} \mu | \tau, y &\sim \mathcal{N} \left( \frac{\tau(616) + (2^{-2})(45)}{(2^{-2}) + (14)\tau}, \frac{1}{(2^{-2}) + (14)\tau} \right) \\ &\sim \mathcal{N} \left( \frac{616\tau + 11.25}{0.25 + 14\tau}, \frac{1}{0.25 + 14\tau} \right) \end{aligned}$$

Next, we can derive the full conditional for  $\tau$  as follows.

$$\begin{aligned} \pi(\tau | \mu, y) &\propto \tau^{n/2} \exp \left\{ -\frac{\tau}{2} \sum_{i=1}^n (y_i - \mu)^2 \right\} \tau^{a_0-1} \exp \{-b_0 \tau\} \\ &= \tau^{n/2+a_0-1} \exp \left\{ -\tau \left[ b + \frac{1}{2} \sum_{i=1}^n (y_i - \mu)^2 \right] \right\} \end{aligned}$$

which is a kernel for the  $\mathcal{Ga} \left( a_0 + \frac{n}{2}, b_0 + \frac{1}{2} \sum_{i=1}^n (y_i - \mu)^2 \right)$  distribution. We can write this explicitly as

$$\tau|\mu, y \sim \mathcal{Ga}\left((4) + \frac{(14)}{2}, (2) + \frac{1}{2} \sum_{i=1}^{(14)} (y_i - \mu)^2\right)$$

$$\sim \mathcal{Ga}\left(18, 2 + \frac{1}{2} \sum_{i=1}^{14} (y_i - \mu)^2\right)$$

Please see the code and its output below for the implementation of a Gibbs sampler for this problem. Discussion of the output follows. **IMPORTANT: Note that I use variable suffixes `_1` and `_2` (starting with `do_mcmc_gibbs*()` to differentiate between calculations where I define  $\tau$  as the rate parameter in R's `rgamma()` function (which seems to make a lot more sense to me given the prior and despite the instructions indicating that it is the scale parameter) and calculations where  $\tau$  is defined as the scale parameter (which seems to make less sense to me).**

```
set.seed(42)
y <- c(41, 44, 43, 47, 43, 46, 45, 42, 45, 45, 43, 45, 47, 40)
y_sum <- sum(y)
y_sum
```

```
## [1] 616
```

```
n_obs <- length(y)
n_obs
```

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

```
n_burnin <- 1000L
n_mcmc <- 10000L + n_burnin

mu_0 <- 45
tau_0 <- 1 / (2^2)
a_0 <- 4
b_0 <- 2
```

```

.compute_mu_new <- function(mu_i, tau_i, mu_0, tau_0, y_sum, n_obs) {
  mu_tau_0 <- mu_0 * tau_0
  mu_rnorm_num <- tau_i * y_sum + mu_tau_0
  mu_rnorm_den <- tau_0 + n_obs * tau_i
  mu_rnorm <- mu_rnorm_num / mu_rnorm_den
  sigma2_rnorm <- 1 / (tau_0 + n_obs * tau_i)
  sigma_rnorm <- sqrt(sigma2_rnorm)
  rnorm(1, mu_rnorm, sigma_rnorm)
}

n_obs <- length(y)
compute_mu_new <-
  function(mu_i,
           tau_i,
           .mu_0 = mu_0,
           .tau_0 = tau_0,
           .y_sum = y_sum,
           .n_obs = n_obs) {
    .compute_mu_new(
      mu_i,
      tau_i,
      mu_0 = .mu_0,
      tau_0 = .tau_0,
      y_sum = .y_sum,
      n_obs = .n_obs
    )
  }

.compute_tau_new <- function(mu_new, y, a_0, b_0, n_obs, scale = TRUE) {
  shape_rgamma <- a_0 + 0.5 * n_obs
  z_rgamma <- b_0 + 0.5 * sum((y - mu_new) ^ 2)
  if(scale) {
    res <- rgamma(1, shape = shape_rgamma, scale = z_rgamma)
  } else {
    res <- rgamma(1, shape = shape_rgamma, rate = z_rgamma)
  }
  res
}

compute_tau_new <-
  function(mu_new,
           .y = y,
           .a_0 = a_0,
           .b_0 = b_0,
           .n_obs = n_obs,
           scale = TRUE) {
    .compute_tau_new(
      mu_new = mu_new,
      y = .y,
      a_0 = .a_0,
      b_0 = .b_0,
      n_obs = .n_obs,

```



```
        scale = scale
      )
    }

mu_init <- mean(y)
tau_init <- 1 / sd(y)
```

```

is_likeinteger <- function(x, tol = .Machine$double.eps^0.5) {
  abs(x - round(x)) < tol
}

.do_mcmc_gibbs <-
  function(n_mcmc,
           n_burnin,
           mu_init,
           tau_init,
           scale = TRUE) {
    stopifnot(is_likeinteger(n_mcmc))
    if (!is.null(n_burnin)) {
      stopifnot(is_likeinteger(n_burnin))
      stopifnot(n_burnin < n_mcmc)
    }
    stopifnot(is.numeric(mu_init))
    stopifnot(is.numeric(tau_init))
    cols_mat_mcmc <- c('mu', 'tau')
    mat_mcmc <- matrix(nrow = n_mcmc, ncol = length(cols_mat_mcmc))

    colnames(mat_mcmc) <- cols_mat_mcmc
    mu_i <- mu_init
    tau_i <- tau_init
    y_sum <- sum(y)
    n_obs <- length(y)
    for (i in 1:n_mcmc) {
      mu_new <- compute_mu_new(mu_i = mu_i, tau_i = tau_i)
      tau_new <- compute_tau_new(mu_new = mu_new, scale = scale)
      mat_mcmc[i,] <- c(mu_new, tau_new)
      mu_i <- mu_new
      tau_i <- tau_new
    }
    res_mcmc <-
      mat_mcmc %>%
      as_tibble() %>%
      mutate(idx = row_number()) %>%
      select(idx, everything())

    if (!is.null(n_burnin)) {
      idx_slice <- (n_burnin + 1):nrow(mat_mcmc)
      res_mcmc <- res_mcmc %>% slice(idx_slice)
    }

    res_mcmc
  }

do_mcmc_gibbs_1 <-
  function(...,
           .n_mcmc = n_mcmc,
           .n_burnin = n_burnin,
           .mu_init = mu_init,
           .tau_init = tau_init,

```

```

      .scale = FALSE) {
    .do_mcmc_gibbs(
      n_mcmc = .n_mcmc,
      n_burnin = .n_burnin,
      mu_init = .mu_init,
      tau_init = .tau_init,
      scale = .scale,
      ...
    )
  }

do_mcmc_gibbs_2 <-
  function(...,
    .n_mcmc = n_mcmc,
    .n_burnin = n_burnin,
    .mu_init = mu_init,
    .tau_init = tau_init,
    .scale = TRUE) {
    .do_mcmc_gibbs(
      n_mcmc = .n_mcmc,
      n_burnin = .n_burnin,
      mu_init = .mu_init,
      tau_init = .tau_init,
      scale = .scale,
      ...
    )
  }

```

```

# Treating tau as `rate` parameter.
res_mcmc_gibbs_1 <- do_mcmc_gibbs_1()
res_mcmc_gibbs_1

```

```

## # A tibble: 10,000 x 3
##   idx    mu    tau
##   <int> <dbl> <dbl>
## 1  1001 43.66 0.3438
## 2  1002 44.17 0.4340
## 3  1003 43.84 0.3517
## 4  1004 44.19 0.2368
## 5  1005 44.24 0.2940
## 6  1006 44.21 0.4355
## 7  1007 44.42 0.4610
## 8  1008 44.07 0.4270
## 9  1009 43.73 0.2546
## 10 1010 43.91 0.4731
## # ... with 9,990 more rows

```

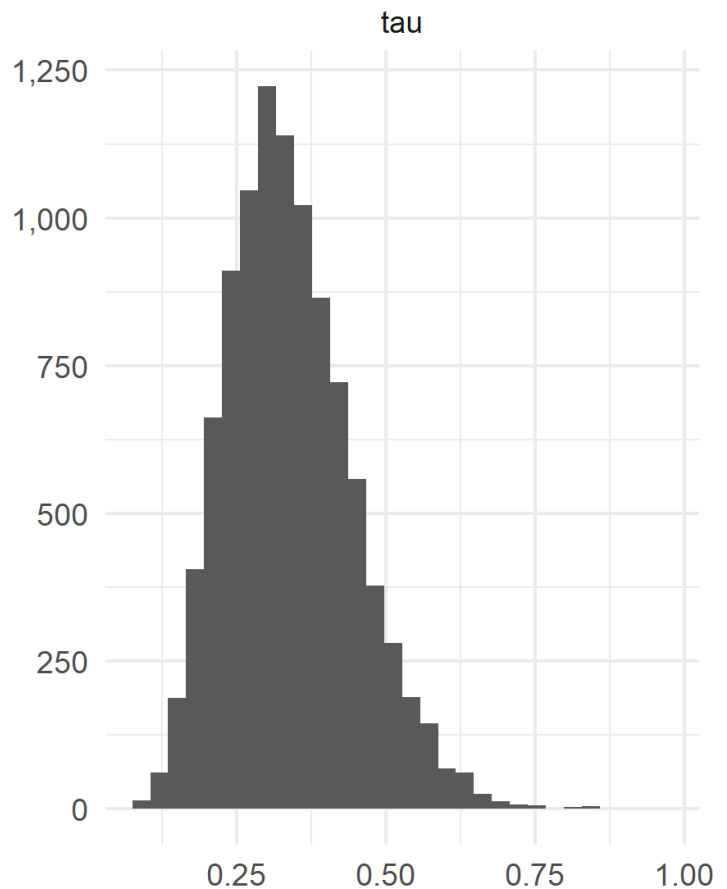
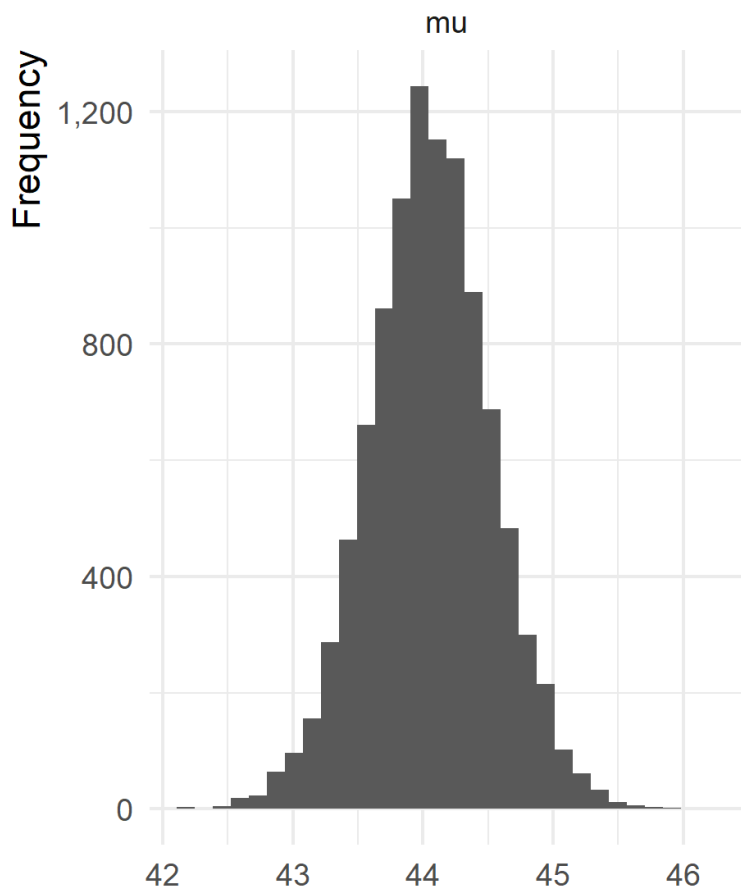
```
# Treating tau as `scale` parameter.
res_mcmc_gibbs_2 <- do_mcmc_gibbs_2()
res_mcmc_gibbs_2
```

```
## # A tibble: 10,000 x 3
##   idx    mu    tau
##   <int> <dbl> <dbl>
## 1  1001 43.99 177.3
## 2  1002 44.02 390.9
## 3  1003 44.02 277.1
## 4  1004 44.02 221.4
## 5  1005 44.00 240.8
## 6  1006 44.02 189.3
## 7  1007 44.01 399.1
## 8  1008 44.01 525.6
## 9  1009 44.00 238.8
## 10 1010 43.99 235.0
## # ... with 9,990 more rows
```

```
theme_custom <- function(...) {
  theme_minimal() +
    theme(
      legend.position = 'bottom',
      legend.title = element_blank(),
      axis.title.x = element_text(hjust = 1),
      axis.title.y = element_text(hjust = 1),
      ...
    )
}
```

```
viz_mcmc_gibbs_1 <-
  res_mcmc_gibbs_1 %>%
  gather(key = 'key', value = 'value', -idx) %>%
  ggplot() +
  aes(x = value) +
  geom_histogram() +
  scale_y_continuous(labels = scales::comma) +
  theme_custom() +
  facet_wrap(~key, scales = 'free') +
  labs(
    caption = 'tau treated as `rate` parameter in R\'s `rgamma()` function.',
    x = NULL,
    y = 'Frequency'
  )
```

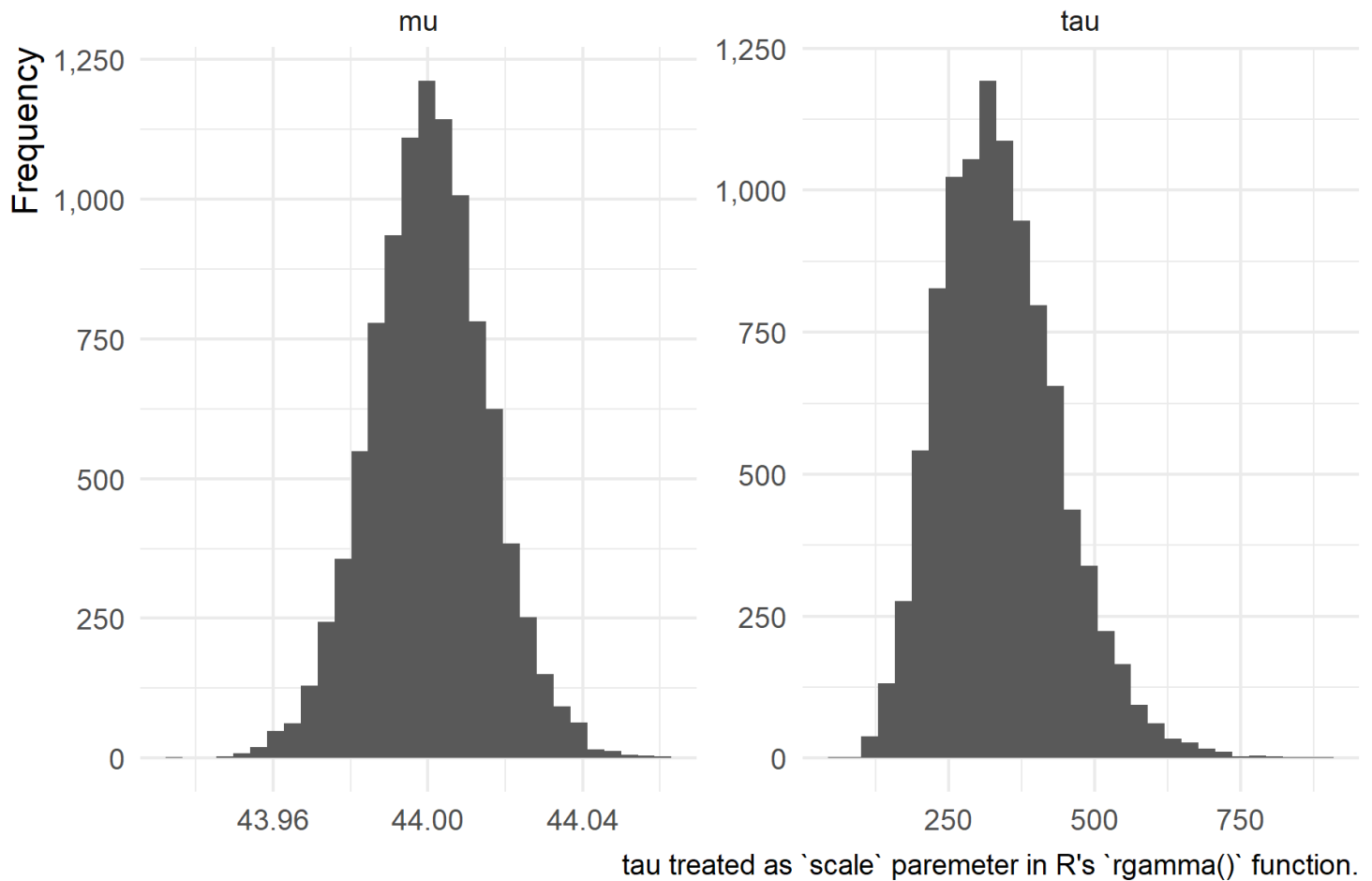
```
viz_mcmc_gibbs_1
```



$\tau$  treated as 'rate' parameter in R's 'rgamma()' function.

```
viz_mcmc_gibbs_2 <-
  res_mcmc_gibbs_2 %>%
  gather(key = 'key', value = 'value', -idx) %>%
  ggplot() +
  aes(x = value) +
  geom_histogram() +
  scale_y_continuous(labels = scales::comma) +
  theme_custom() +
  facet_wrap(~key, scales = 'free') +
  labs(
    caption = 'tau treated as `scale` parameter in R\'s `rgamma()` function.',
    x = NULL,
    y = 'Frequency'
  )
```

```
viz_mcmc_gibbs_2
```



As a quick check of the validity of our results, we can compare the posterior mean  $\hat{\mu}$  (from the Gibbs sampling results) with those of the observations  $y$  and the prior  $\mu_0$ .

```
mcmc_gibb_mu_mean_1 <-
  res_mcmc_gibbs_1 %>%
  summarise_at(vars(mu), ~mean(.)) %>%
  pull(mu)
mcmc_gibb_mu_mean_1
```

```
## [1] 44.05479
```

```
mcmc_gibb_mu_mean_2 <-
  res_mcmc_gibbs_2 %>%
  summarise_at(vars(mu), ~mean(.)) %>%
  pull(mu)
mcmc_gibb_mu_mean_2
```

```
## [1] 44.00036
```

```
y_mean <- mean(y)
y_mean
```

```
## [1] 44
```

We find that  $\hat{\mu} = 44.0548$ , which is between the observed mean  $\hat{y} = 44$  and the prior  $\mu_0 = 45$ . This is what we should have expected. (Note that  $\hat{\mu} = 44.0004$  in the second set of calculations, which is a big reason why I believe that tau should not be treated as the `scale` parameter for R's `rgamma()` function.)

## a

To approximate the posterior probability  $p_{H_0}$  of the hypothesis  $H_0 : \mu < 45$ , we calculate the fraction of samples where this condition is met. (Note that  $\tau$  does not matter here, so we don't show calculations for the two methods of calculating the posterior of  $\tau$ .)

```
mu_split <- 45
```

```
h_0_1 <-
  res_mcmc_gibbs_1 %>%
  mutate(h_0 = ifelse(mu < mu_split, 1, 0)) %>%
  summarise(n_h_0 = sum(h_0), n = n()) %>%
  mutate(frac = n_h_0 / n) %>%
  select(n_h_0, frac)
h_0_1
```

```
## # A tibble: 1 x 2
##   n_h_0   frac
##   <dbl> <dbl>
## 1  9767 0.9767
```

**We find that the posterior probability is  $p_{H_0} = 0.977$  (i.e. 97.7%).** This very large probability is illustrated by the histogram from above, which shows a large majority of  $\mu$  samples with values less than 45.

Additionally, we may calculate the Bayes Factor  $B_{01}$  in favor of hypothesis  $H_0$ .

```
p_h_0_1 <- h_0_1 %>% pull(frac)
p_h_1_1 <- 1 - p_h_0_1
b_01_1 <- p_h_0_1 / p_h_1_1
b_01_1_log10 <- log10(b_01_1)
b_01_1_log10
```

```
## [1] 1.622405
```

We deduce that there is strong evidence against the alternative hypothesis  $H_1 \geq 45$ —or equivalently, in favor of  $H_0$ —because  $1.5 < \log_{10}(B_{01}) = 1.6224 \leq 2$ .

## b

We can calculate the 95% equitailed credible set for  $\tau$  (trying both methods, as described before) as follows.

```
equi_credible_set_1 <-  
  res_mcmc_gibbs_1 %>%  
  summarise_at(  
    vars(tau),  
    list(  
      l = ~quantile(., 0.025),  
      u = ~quantile(., 0.975)  
    ))  
equi_credible_set_1
```

```
## # A tibble: 1 x 2  
##       l       u  
##   <dbl> <dbl>  
## 1 0.1639 0.5709
```

```
equi_credible_set_2 <-  
  res_mcmc_gibbs_2 %>%  
  summarise_at(  
    vars(tau),  
    list(  
      l = ~quantile(., 0.025),  
      u = ~quantile(., 0.975)  
    ))  
equi_credible_set_2
```

```
## # A tibble: 1 x 2  
##       l       u  
##   <dbl> <dbl>  
## 1 167.9 565.2
```

**We find that the 95% equitailed credible set is [0.1639, 0.5709].** (For the second set of  $\tau$  calculations, this is [167.8636, 565.1571].) If we had exact posterior estimates for  $a$  and  $b$  for the  $\mathcal{G}a$  posterior distribution, we could have possibly used R's `qgamma()` function to compute the interval.