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`, whether she goes on `vaca`tion (to Redington Shores), and wins the `lott`ery (i.e. sweepstakes).

a

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

$$\Pr(\text{car=yes}|\text{vaca=yes}) = \frac{\Pr(\text{vaca=yes}|\text{car=yes})\,\Pr(\text{car=yes})}{\Pr(\text{vaca=yes})}$$

Next, we could delineate all possible scenarios in a manner, such as follows,

$$\begin{aligned}&\text{Scenario A: grade=a,market=bull,car=yes,vaca=yes}\\&\text{Scenario B: grade=a,market=bear,car=yes,vaca=yes}\\&\ldots,\end{aligned}$$

or more directly such as follows,

$$\begin{aligned}\Pr_1 &= \Pr(\text{vaca=yes}|\text{car=yes}) \times \Pr(\text{car=yes}|\text{market=bull}) \times \textstyle\sum_g^{a,b,c} \Pr(\text{car=yes}|\text{grade=g})\\ \Pr_2 &= \ldots,\end{aligned}$$

or even with a probability tree diagram. However, because noting all scenarios and probabilities in such a manner can be tedious, I think it might be easier to first show the code defining these things and provide an explanation afterwards.

```
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
```

```
states_wo_lott <-
  crossing(
    grade = c('a', 'b', 'c'),
    market = c('bull', 'bear'),
    car = c('yes', 'no'),
    vaca = c('yes', 'no')
  ) %>%
  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(
      car == 'yes' & . == 'yes' ~ p_vaca_given_car,
      car == 'yes' & . == 'no' ~ 1 - p_vaca_given_car,
      car == 'no' & . == 'yes' ~ p_vaca_given_no_car,
      car == 'no' & . == 'no' ~ 1 - p_vaca_given_no_car
    ))
  ) %>%
  mutate(
```

```
    # p_car_cum = p_grade * p_market * p_car,
    p = p_grade * p_market * p_car * p_vaca
  ) %>%
  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(grade, market, car, vaca)
states_wo_lott
```

```
## # A tibble: 24 x 9
##    grade market car   vaca  p_grade p_market  p_car p_vaca       p
##    <fct> <fct>  <fct> <fct>   <dbl>    <dbl>  <dbl>  <dbl>   <dbl>
##  1 a     bull   yes   yes       0.6      0.5 0.8       0.7 0.1680
##  2 a     bull   yes   no        0.6      0.5 0.8       0.3 0.072
##  3 a     bull   no    yes       0.6      0.5 0.2000    0.2 0.01200
##  4 a     bull   no    no        0.6      0.5 0.2000    0.8 0.04800
##  5 a     bear   yes   yes       0.6      0.5 0.5       0.7 0.105
##  6 a     bear   yes   no        0.6      0.5 0.5       0.3 0.045
##  7 a     bear   no    yes       0.6      0.5 0.5       0.2 0.03
##  8 a     bear   no    no        0.6      0.5 0.5       0.8 0.12
##  9 b     bull   yes   yes       0.3      0.5 0.5       0.7 0.0525
## 10 b     bull   yes   no        0.3      0.5 0.5       0.3 0.0225
## 11 b     bull   no    yes       0.3      0.5 0.5       0.2 0.015
## 12 b     bull   no    no        0.3      0.5 0.5       0.8 0.06
## 13 b     bear   yes   yes       0.3      0.5 0.3       0.7 0.0315
## 14 b     bear   yes   no        0.3      0.5 0.3       0.3 0.0135
## 15 b     bear   no    yes       0.3      0.5 0.7       0.2 0.021
## 16 b     bear   no    no        0.3      0.5 0.7       0.8 0.084
## 17 c     bull   yes   yes       0.1      0.5 0.2       0.7 0.007
## 18 c     bull   yes   no        0.1      0.5 0.2       0.3 0.003
## 19 c     bull   no    yes       0.1      0.5 0.8       0.2 0.008
## 20 c     bull   no    no        0.1      0.5 0.8       0.8 0.032
## 21 c     bear   yes   yes       0.1      0.5 0.1       0.7 0.0035
## 22 c     bear   yes   no        0.1      0.5 0.1       0.3 0.0015
## 23 c     bear   no    yes       0.1      0.5 0.9       0.2 0.009
## 24 c     bear   no    no        0.1      0.5 0.9       0.8 0.036
```

The data.frame `states_wo_lott` 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_wo_lott %>% summarise_at(vars(p), sum) %>% pull(p)
```

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

Also, before moving on, I should note that we can disregarding 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 ther are 12 records.

```
states_wo_lott %>%
  filter(vaca == 'yes') %>%
  select(grade, market, car, vaca, p_grade, p_market, p_car, p_vaca, p)
```

```
## # A tibble: 12 x 9
##    grade market car   vaca  p_grade p_market  p_car p_vaca       p
##    <fct> <fct>  <fct> <fct>   <dbl>    <dbl>  <dbl>  <dbl>   <dbl>
##  1 a     bull   yes   yes       0.6      0.5 0.8       0.7 0.1680
##  2 a     bull   no    yes       0.6      0.5 0.2000    0.2 0.01200
##  3 a     bear   yes   yes       0.6      0.5 0.5       0.7 0.105
##  4 a     bear   no    yes       0.6      0.5 0.5       0.2 0.03
##  5 b     bull   yes   yes       0.3      0.5 0.5       0.7 0.0525
##  6 b     bull   no    yes       0.3      0.5 0.5       0.2 0.015
##  7 b     bear   yes   yes       0.3      0.5 0.3       0.7 0.0315
##  8 b     bear   no    yes       0.3      0.5 0.7       0.2 0.021
##  9 c     bull   yes   yes       0.1      0.5 0.2       0.7 0.007
## 10 c     bull   no    yes       0.1      0.5 0.8       0.2 0.008
## 11 c     bear   yes   yes       0.1      0.5 0.1       0.7 0.0035
## 12 c     bear   no    yes       0.1      0.5 0.9       0.2 0.009
```

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

```
states_wo_lott_vaca <-
  states_wo_lott %>%
  group_by(vaca) %>%
  summarise_at(vars(p), sum) %>%
  ungroup()
states_wo_lott_vaca
```

```
## # A tibble: 2 x 2
##   vaca       p
##   <fct>  <dbl>
## 1 yes   0.4625
## 2 no    0.5375
```

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

```
## [1] 0.4625
```

Summing up the probabilities for the scenarios where Emily does go on vacation gives us $P(\text{vaca}|\text{yes})$ = 0.4625, which is the denominator of the Bayes' equation noted above.

Next, we can view the marginal probabilities of Emily getting a car or not

```
states_wo_lott_car_bad <-
  states_wo_lott %>%
  group_by(vaca, car) %>%
  summarise_at(vars(p), sum) %>%
  ungroup() %>%
  group_by(car) %>%
  mutate_at(vars(p), ~(. / sum(.))) %>%
  ungroup() %>%
  arrange(car, vaca)
states_wo_lott_car_bad
```

```
## # A tibble: 4 x 3
##   vaca  car       p
##   <fct> <fct> <dbl>
## 1 yes   yes     0.7
## 2 no    yes     0.3
## 3 yes   no      0.2
## 4 no    no      0.8
```

```
states_wo_lott_car <-
  states_wo_lott %>%
  group_by(car) %>%
  summarise_at(vars(p), sum) %>%
  ungroup() %>%
  arrange(car)
states_wo_lott_car
```

```
## # A tibble: 2 x 2
##   car       p
##   <fct> <dbl>
## 1 yes   0.525
## 2 no    0.475
```

and isolate just the value we need, i.e. when `car == 'yes'`.

```
p_car <-
  states_wo_lott_car %>%
  filter(car == 'yes') %>%
  pull(p)
p_car
```

```
## [1] 0.525
```

We see that the probability of $\Pr(\text{car=yes})$ = 0.525, which is a term we need for the numerator of our Bayes' equation.

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

```
p_car_given_vaca <- (p_vaca_given_car * p_car) / p_vaca
p_car_given_vaca
```

```
## [1] 0.7945946
```

Thus, we find that $\Pr(\text{car=yes}|\text{vaca=yes})$ = 0.7946.

b

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

$$\Pr(\text{lott=yes}|\text{vaca=yes}) = \frac{\Pr(\text{vaca=yes}|\text{lott=yes})\,\Pr(\text{lott=yes})}{\Pr'(\text{vaca=yes})}$$

We are already given two of the quantities that we'll need—namely, the terms in the numerator, $\Pr(\text{vaca=yes}|\text{lott=yes}) = 0.99$ and $\Pr(\text{lott=yes}) = 0.001$. The denominator term $\Pr'(\text{vaca=yes})$ is equivalent to the sum of the product of the numerator terms and $\Pr(\text{vaca=yes})$ found in part a. (Note the distinction of the $\Pr(\text{vaca=yes})$ from part a and the one here with the prime symbol $'$.)

$$\begin{aligned}
\Pr(\text{lott=yes}|\text{vaca=yes}) &= \frac{\Pr(\text{vaca=yes}|\text{lott=yes})\,\Pr(\text{lott=yes})}{\Pr'(\text{vaca=yes})} \\
&= \frac{\Pr(\text{vaca=yes}|\text{lott=yes})\,\Pr(\text{lott=yes})}{\Pr(\text{vaca=yes}|\text{lott=yes})\,\Pr(\text{lott=yes}) + \Pr(\text{vaca=yes})} \\
&= \frac{(0.99)(0.001)}{(0.99)(0.001) + (0.4625)} \\
&= 0.0021.
\end{aligned}$$

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

c

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

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

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_wo_lott %>%
  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     0.525
## 2 no    a     0.475
## 3 yes   b     0.4
## 4 no    b     0.6
## 5 yes   c     0.275
## 6 no    c     0.725
```

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] 0.4
```

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

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

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

```
## [1] 0.2594595
```

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

d

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

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

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

```
## # A tibble: 4 x 3
##   vaca  market      p
##   <fct> <fct>   <dbl>
## 1 yes   bull   0.5250
## 2 no    bull   0.475
## 3 yes   bear   0.4
## 4 no    bear   0.6
```

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

```
## [1] 0.4
```

We see that the probability of $\Pr(\text{vaca=yes}|\text{market=bear})$ = 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] 0.4324324
```

Thus, we find that $\Pr(\text{market}{=}\text{bear}|\text{vaca}{=}\text{yes})$ = 0.4324.

2. Trials until Fourth Success.

Instructions

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

Response

Given $X = (X_1, \ldots, X_n)$ is a sample from $\mathcal{NB}(m, p)$ and $p \sim \mathcal{B}e(a, b)$, the posterior for p follows a $\mathcal{B}e(a + mn, b + \sum_{i=1}^n x_i)$ distribution. The proof of this relationship is provided in an 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 mth success in n Bernoulli experiments) is given by

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

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

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

(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 form $\mathcal{B}e(a, b)$, 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 $\mathcal{B}e(a + mn, b + \sum_{i=1}^n x_i)$. $\square$

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_{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_{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 $\mathrm{E}[x] = \frac{a}{a+b}$. Thus, for this setup, $p_{\text{Bayes}} = \mathrm{E}[p_{\text{posterior}}] = \frac{a+mn}{b+\sum_{i=1}^{n} x_i}$.

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.

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

Regarding (3), to test the one-sided (null) hypothesis $H_0 : p \geq p_{split}$ (here, $p_{split} = 0.8$–against the (alternative) alternative $H_1 : p < p_{split}$, 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 $\boldsymbol{H_0}$
$0 \leq \log_{10} B_{10} \leq 0.5$	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 `compute_q2()` calls other functions to compute the required components—(1) `compute_beta_mu()` for p_{Bayes}, `compute_equi_credible_set_b()` 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) {
```

```r
#   .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)
}

compute_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_0 = a_0,
      b_0 = b_0,
      a_1 = a_1,
      b_1 = 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
}
```

a

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 <- compute_q2(a_0_a, b_0_a)
res_a
```

```
## $a_0
## [1] 1
##
## $b_0
## [1] 1
##
## $a_1
## [1] 45
##
## $b_1
## [1] 31
##
## $mu_prior
## [1] 0.5
##
## $mu_post
## [1] 0.5921053
##
## $credible_set_l
##         l
## 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
## [1] 7.984803e-05
##
## $b_10
## [1] 12523.79
##
## $b_01_log10
## [1] -4.097736
##
## $b_10_log10
## [1] 4.097736
```

For (1), we find that p_{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.

b

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 <- compute_q2(a_0_b, b_0_b)
res_b
```

```
## $a_0
## [1] 0.5
##
## $b_0
## [1] 0.5
##
## $a_1
## [1] 44.5
##
## $b_1
## [1] 30.5
##
## $mu_prior
## [1] 0.5
##
## $mu_post
## [1] 0.5933333
##
## $credible_set_l
##         l
## 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
## [1] 5.920269e-05
##
## $b_10
## [1] 16891.12
##
## $b_01_log10
## [1] -4.227659
##
## $b_10_log10
## [1] 4.227659
```

For (1), we find that p_{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 <- compute_q2(a_0_c, b_0_c)
res_c
```

```
## $a_0
## [1] 9
##
## $b_0
## [1] 1
##
## $a_1
## [1] 53
##
## $b_1
## [1] 31
##
## $mu_prior
## [1] 0.9
##
## $mu_post
## [1] 0.6309524
##
## $credible_set_l
##         l
## 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
## [1] 2.987073e-05
##
## $b_10
## [1] 33477.59
##
## $b_01_log10
## [1] -4.524754
##
## $b_10_log10
## [1] 4.524754
```

For (1), we find that p_{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 μ and τ, closely following the guide provided by the Gibbs handout from class.

Per the instructions, we make the assumption that the measurements $y_i \ldots y_n$ 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 μ_0 = 45 and τ_0 = 4, per the instructions), and the precision parameter $\tau \sim \mathcal{G}a(a_0, b_0)$ (where a_0 = 4 and b_0 = 2).

Now, in preparation of expressing the joint distribution, we define the likelihood of μ as

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

Thus, the joint distribution is

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

After selecting the terms from the joint distribution expression that contain μ and normalizing, we find that

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

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 $y_i, \ldots, y_n$, μ_0, τ_0 and n, we can write the full conditional for μ 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 τ as follows.

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

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

$$
\begin{aligned}
\tau|\mu,y &\sim \mathcal{G}a\left((4)+\frac{(14)}{2}, (2)+\frac{1}{2}\sum_{i=1}^{(14)}(y_i-\mu)^2\right) \\
&\sim \mathcal{G}a\left(18, 2+\frac{1}{2}\sum_{i=1}^{14}(y_i-\mu)^2\right)
\end{aligned}
$$

Please see the code and its output below for the implementation of a Gibbs sampler for this problem. Discussion of the output follows.

```
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 <- 100L
n_mcmc <- 10000L + n_burnin

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

```r
.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) {
  shape_rgamma <- a_0 + 0.5 * n_obs
  rate_rgamma  <- b_0 + 0.5 * sum((y - mu_new) ^ 2)
  rgamma(1, shape = shape_rgamma, rate = rate_rgamma)
}

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

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) {
    # n_burnin <- 100L
    # n_mcmc <- 10000L + n_burnin
    # mu_init <- 0
    # tau_init <- 0
    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)
      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 <-
  function(...,
           .n_mcmc = n_mcmc,
           .n_burnin = n_burnin,
```

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

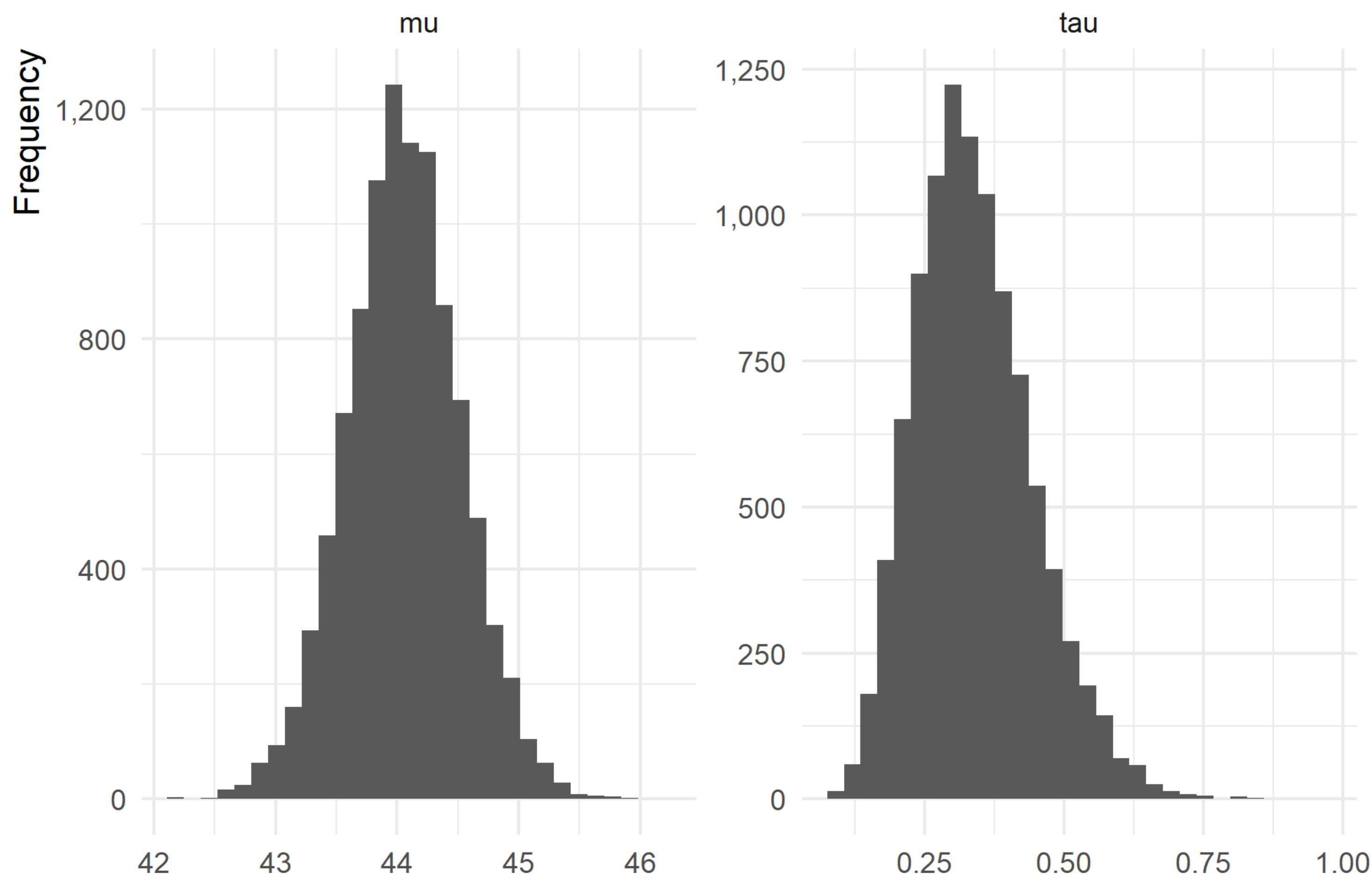
```
res_mcmc_gibbs <- do_mcmc_gibbs()
res_mcmc_gibbs
```

```
## # A tibble: 10,000 x 3
##      idx    mu    tau
##    <int> <dbl>  <dbl>
##  1   101 43.95 0.2732
##  2   102 44.57 0.3069
##  3   103 43.93 0.2987
##  4   104 44.85 0.2835
##  5   105 43.60 0.2575
##  6   106 43.50 0.3081
##  7   107 43.41 0.5439
##  8   108 44.39 0.3245
##  9   109 43.82 0.5443
## 10   110 43.60 0.3466
## # ... 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 <-
  res_mcmc_gibbs %>%
  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(
    x = NULL,
    y = 'Frequency'
  )
```

```
viz_mcmc_gibbs
```



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 μ_0.

```
mcmc_gibb_mu_mean <-
  res_mcmc_gibbs %>%
  summarise_at(vars(mu), ~mean(.)) %>%
  pull(mu)
mcmc_gibb_mu_mean
```

```
## [1] 44.05354
```

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

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

We find that $\hat{\mu}$ = 44.0535, which is between the observed mean $\hat{y}$ = 44 and the prior μ_0 = 45. This is what we should have expected.

a

To approximate the posterior probability p_{H_0} of the hypothesis H_0 : μ < 45, we calculate the fraction of samples where this condition is met.

```
mu_split <- 45
```

```
h_0 <-
  res_mcmc_gibbs %>%
  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
```

```
## # A tibble: 1 x 2
##   n_h_0  frac
##   <dbl> <dbl>
## 1  9770 0.977
```

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 μ samples with values less than 45.

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

```
p_h_0 <- h_0 %>% pull(frac)
p_h_1 <- 1 - p_h_0
b_01 <- p_h_0 / p_h_1
b_01_log10 <- log10(b_01)
b_01_log10
```

```
## [1] 1.628167
```

We deduce that there is strong evidence against the alternative hypothesis $H_1 \geq$ 45—or equivalently, in favor of H_0—because $1.5 < \log_{10}(B_{01})$ = 1.6282 ≤ 2.

b

We can calculate the 95% equitailed credible set for τ as follows.

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

```
## # A tibble: 1 x 2
##        l      u
##    <dbl>  <dbl>
## 1 0.1654 0.5710
```

We find that the 95% equitailed credible set is [0.1654, 0.5710]. (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.)