Assignment 1 Solutions

Julia Haaf & Nicle Cruz

```
library(brms)
```

```
## Loading required package: Rcpp
```

```
## Loading 'brms' package (version 2.22.0). Useful instructions
## can be found by typing help('brms'). A more detailed introduction
## to the package is available through vignette('brms_overview').
```

```
##
## Attaching package: 'brms'
```

```
## The following object is masked from 'package:stats':
##
##     ar
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(ggplot2)
library(ggpubr)
```

Exercise 1

The French mathematician Pierre-Simon Laplace (1749-1827) was the first person to show definitively that the proportion of female births in the French population was less then 0.5, in the late 18th century, using a Bayesian analysis based on a uniform prior distribution.

Suppose you were doing a similar analysis but you had more definite prior beliefs about the ratio of male to female births. In particular, if θ represents the proportion of female births in a given population, you are willing to place a Beta(100,100) prior distribution on θ.

i. Show that this means you are more than 95% sure that θ is between 0.4 and 0.6, although you are ambivalent as to whether it is greater or less than 0.5.

ii. Now you observe that out of a random sample of $1,000$ births, 511 are boys. What is your posterior probability that $\theta > 0.5$?

Solution to exercise 1

i. The probability that θ lies between 0.6 and 0.4:

```
pbeta(0.6,shape1=100,shape2=100)-pbeta(0.4,shape1=100,shape2=100)
```

```
## [1] 0.9956798
```

ii. The data: $n = 1000$ people, 511 boys. (Assuming a binary gender here because back then they only recognized two genders.) So, $k = 1000 - 511 = 489$ girls.

The posterior will be Beta($\alpha = 100 + 489, \beta = 100 + 511$). (See lecture slides and book if it is not clear why). That means that the proportion of female births is:

```
a<-100+489
b<-100+1000-489
## The mean of a Beta distribution:
a/(a+b)
```

```
## [1] 0.4908333
```

The probability that θ is larger than 0.5:

```
pbeta(0.5,shape1=100+489,shape2=100+1000-489,lower.tail=FALSE)
```

```
## [1] 0.2626087
```

Now let's do the same in `brms`. First, we need to figure out the prior:

```
p_sim <- rbeta(100000, 100, 100)
theta_sim <- log(p_sim / (1 - p_sim))
c(mean(theta_sim), sd(theta_sim))
```

```
## [1] -0.0002132136  0.1415308695
```

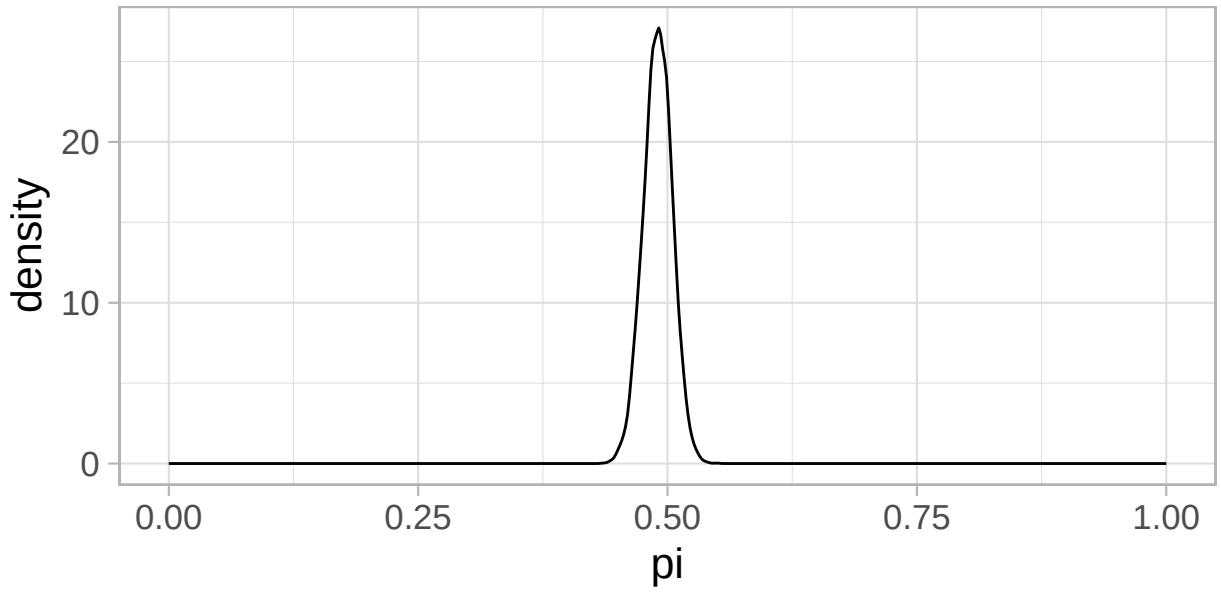
```
library(brms)
girlsdata <- data.frame(y = c(rep(1, 489), rep(0, 511)))
fit <- brm(data = girlsdata
           , family = bernoulli(link = "logit")
           , y  ~ 0 + Intercept
           , prior = c(prior(normal(0, 0.14)
                             , coef = Intercept))
           , iter = 2000
           , warmup = 700
           , silent = 2
           , refresh = 0)
```

```
## Trying to compile a simple C file
```

```
## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## using C compiler: 'gcc (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0'
## gcc -I"/usr/share/R/include" -DNDEBUG   -I"/home/juliahaaf/R/x86_64-pc-linux-gnu-library/4.4/Rcpp/in
## In file included from /home/juliahaaf/R/x86_64-pc-linux-gnu-library/4.4/RcppEigen/include/Eigen/Core
##                  from /home/juliahaaf/R/x86_64-pc-linux-gnu-library/4.4/RcppEigen/include/Eigen/Dens
##                  from /usr/lib/R/site-library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp:22,
##                  from <command-line>:
## /home/juliahaaf/R/x86_64-pc-linux-gnu-library/4.4/RcppEigen/include/Eigen/src/Core/util/Macros.h:679
##   679 | #include <cmath>
##       |          ^~~~~~
## compilation terminated.
## make: *** [/usr/lib/R/etc/Makeconf:195: foo.o] Error 1
```

```r
post <- as_draws_df(fit)
post %>%
  mutate(pi = exp(b_Intercept) / (1 + exp(b_Intercept))) -> post

ggplot(post, aes(pi)) +
  geom_density() +
  xlim(c(0, 1)) +
  theme_light(base_size = 16)
```




```r
mean(post$pi > 0.5)
```

```
## [1] 0.2517308
```

Exercise 2: Is cilantro soapy?

We all know that cilantro (coriander) is a delicious herb that should be added to virtually any meal. However, there seems to be a minority of voices on the internet claiming that cilantro tastes like soap. We want to investigate the prevalence of this claim.

i. Suppose we assume that about 25% of the broader (European) public are part of the tastes-like-soap population. Consequently, we pick a prior for the probability that any one person thinks cilantro tastes like soap, θ, as $\theta \sim \text{Beta}(5, 15)$. Estimate the posterior distribution for three different samples: 1. 4 out of 50 participants think cilantro tastes like soap, 2. 40 out of 500 participants think cilantro tastes like soap, and 3. 400 out of 5000 participants think cilantro tastes like soap. Interpret the effect of sample size on the posterior.

ii. Compare your results from i. to the results if you choose a different prior, $\theta \sim \text{Beta}(1, 1)$. Interpret the differences.

Solution to exercise 2

```r
bayes_binomial <- function(successes, failures, prior_alpha, prior_beta){
  # Parameter of the Posterior
  aprime <- prior_alpha + successes
  bprime <- prior_beta + failures
```

```
  # Estimator for theta
  schaetzer <- aprime / (aprime + bprime)
  ci <- qbeta(c(0.025, 0.975), aprime, bprime)

  # Plot
  cols <- hcl(h = seq(15, 375
                      , length = 3)
              , l = 65, c = 100)[1:2]
  p <- ggplot(data.frame(x = 1), aes(x = x)) +
    xlim(c(0, 1)) +
    stat_function(fun = dbeta
                  , args = list(prior_alpha, prior_beta)
                  , geom = "area", alpha = 0.35, aes(fill = 'Prior')) +
    stat_function(fun = dbeta
                  , args = list(aprime, bprime)
                  , geom = "area", alpha = 0.35, aes(fill = 'Posterior')) +
    scale_fill_manual(name='Distribution',
                      breaks=c('Prior', 'Posterior'),
                      values=c('Prior' = cols[1], 'Posterior' = cols[2])) +
    xlab(expression("Parameter" ~ theta)) +
    ylab("Probability Density") +
  theme_light(base_size = 14)

  return(list("estimate" = schaetzer, "ci" = ci, "p" = p))
}
```

i.

```
res1 <-
  bayes_binomial(successes = 4
                 , failures = 50 - 4
                 , prior_alpha = 5
                 , prior_beta = 15)

res1$estimate; res1$ci
```

```
## [1] 0.1285714
```

```
## [1] 0.0614171 0.2157325
```

```
res2 <-
  bayes_binomial(successes = 40
                 , failures = 500 - 40
                 , prior_alpha = 5
                 , prior_beta = 15)

res2$estimate; res2$ci
```

```
## [1] 0.08653846
```

```
## [1] 0.06394548 0.11214052
```

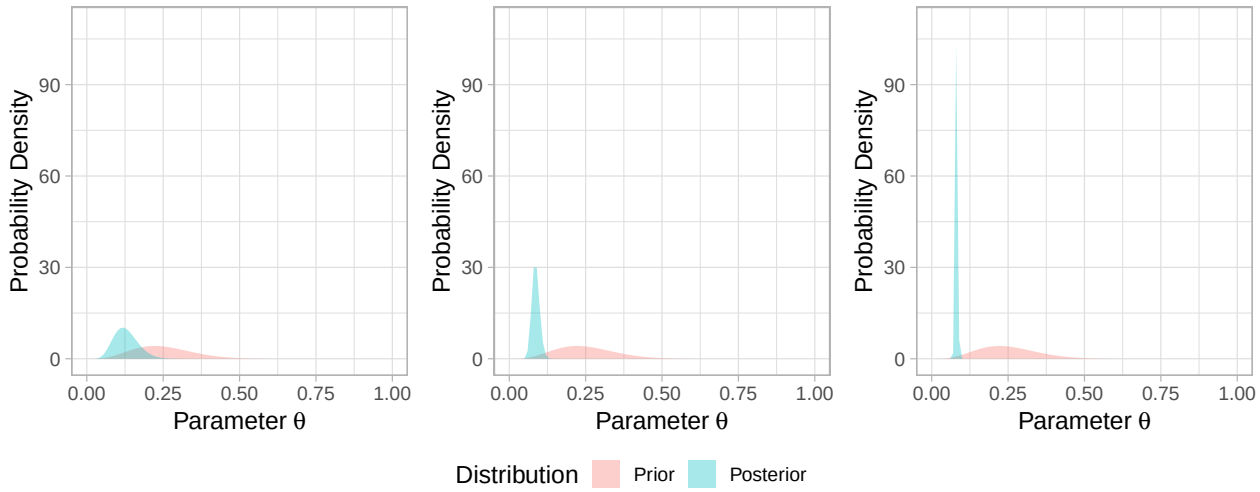
```
res3 <-
  bayes_binomial(successes = 400
                 , failures = 5000 - 400
                 , prior_alpha = 5
                 , prior_beta = 15)
```

```
res3$estimate; res3$ci
```

```
## [1] 0.08067729
```

```
## [1] 0.07330416 0.08836685
```

```
ggarrange(res1$p + ylim(c(0, 110))
          , res2$p+ ylim(c(0, 110))
          , res3$p+ ylim(c(0, 110))
          , nrow=1, common.legend = TRUE, legend="bottom")
```



→ Posterior is getting more informed as the number of observation grows.

ii.

```
res1 <-
  bayes_binomial(successes = 4
                 , failures = 50 - 4
                 , prior_alpha = 1
                 , prior_beta = 1)

res1$estimate; res1$ci
```

```
## [1] 0.09615385
```

```
## [1] 0.03260649 0.18880601
```

```
res2 <-
  bayes_binomial(successes = 40
                 , failures = 500 - 40
                 , prior_alpha = 1
                 , prior_beta = 1)

res2$estimate; res2$ci
```

```
## [1] 0.08167331
```

```
## [1] 0.05936542 0.10713441
```

```
res3 <-
  bayes_binomial(successes = 400
                 , failures = 5000 - 400
                 , prior_alpha = 1
```

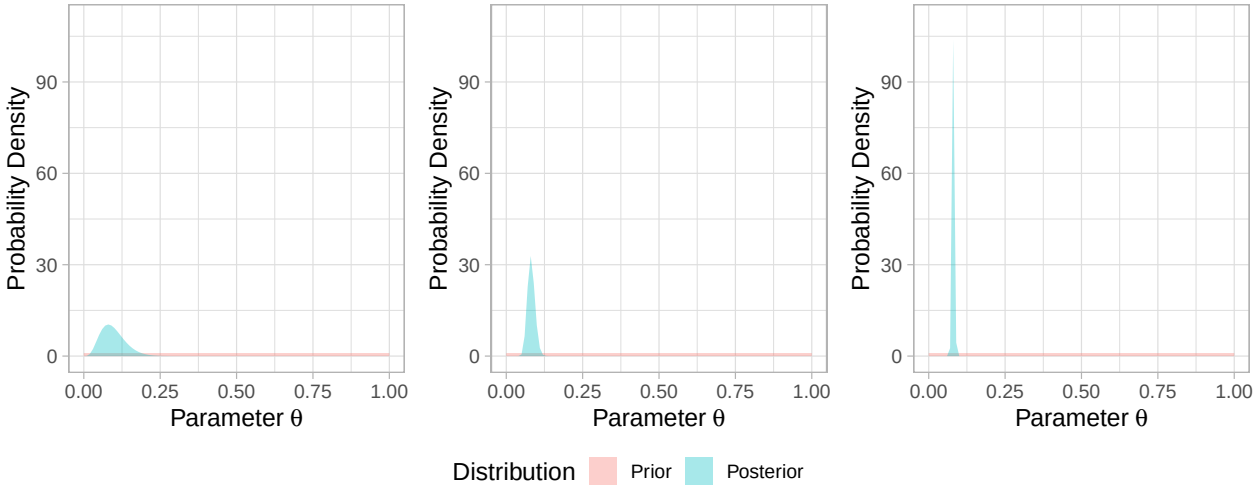
```
                  , prior_beta = 1)

res3$estimate; res3$ci
```

```
## [1] 0.08016793
```

```
## [1] 0.07280381 0.08785000
```

```
ggarrange(res1$p + ylim(c(0, 110))
          , res2$p+ ylim(c(0, 110))
          , res3$p+ ylim(c(0, 110))
          , nrow=1, common.legend = TRUE, legend="bottom")
```



→ less bias, even with fewer observations; overall less information.

Bonus Exercise 3

Check out the example code in `Intro_mcmc_sampling.R`, which shows a Metropolis Hastings algorithm to estimate θ for a Binomial model with a Beta prior. Try to understand the code, and run the algorithm a few times with different settings for `proposalSD`.

Bonus Exercise 4 (in case you finish early and/or are bored)

Suppose that 1 in 1000 people in a population is expected to get HIV. Suppose a test is administered on a suspected HIV case, where the test has a true positive rate (the proportion of positives that are actually HIV positive) of 95% and true negative rate (the proportion of negatives are actually HIV negative) 98%. Use Bayes' theorem to find out the probability that a patient testing positive actually has HIV.

Solution to bonus exercise 3

Need to find: Prob(HasHIV | TestedPositive)

We know:

- Prob(HasHIV) = 1/1000. Therefore, Prob(¬HasHIV) = 999/1000.
- Prob(TestedPositive | HasHIV) = 0.95
- Prob(¬TestedPositive | ¬ HasHIV) = 0.98. This implies that Prob(TestedPositive | ¬ HasHIV) = 1-0.98 = 0.02

Notice that, from the law of total probability (section 1.3 of book draft),

$$Prob(TestPos) = Prob(TestPos|HasHIV)Prob(HasHIV) + Prob(TestPos|\neg HasHIV)Prob(\neg HasHIV)$$

Here, TestPos=TestedPositive.

So,

$$Prob(TestedPositive) = 0.95 \times 1/1000 + 0.02 \times 999/1000 = 0.021$$

Now, Bayes' rule is:

$$Prob(A|B) = \frac{Prob(B|A)Prob(A)}{Prob(B)}$$

Let A=HasHIV, B=TestedPositive. We can rewrite the above in terms of our research question:

$$Prob(HasHIV|TestedPositive) = \frac{Prob(TestedPositive|HasHIV)Prob(HasHIV)}{Prob(TestedPositive)}$$

Plugging in the probabilities:

$$Prob(HasHIV|TestedPositive) = \frac{0.95 \times (1/1000)}{0.03} = 0.045$$

So, the probability of having HIV given that one has tested positive in this situation is, surprisingly, 0.045.

This has to do with the low base rate of HIV (1/1000). Suppose that HIV was rampant, and the probability of having HIV was 999/1000 (obviously a ridiculously unrealistic example, but just to illustrate the effect of base rates).

Now,

$$Prob(TestedPositive) = 0.95 \times 999/1000 + 0.02 \times 1/1000 = 0.949$$

This implies that

$$Prob(HasHIV|TestedPositive) = \frac{0.95 \times (999/1000)}{0.949} = 1$$

Now, the probability of having HIV given that one has tested positive is 1.

In other words, the base rate (the prior probability of something happening) matters a lot in interpreting the posterior probability given some data. This makes intuitive sense: if some event is highly unlikely a priori, then seeing any evidence that event has happened should not shift your prior belief much that the event happened. By contrast, if something is highly likely to happen a priori, then finding evidence that the event happened should lead you to believe that that event actually did happen.

One practical example of this is the infamous Bem paper:

Bem, D. J. (2011). Feeling the future: experimental evidence for anomalous retroactive influences on cognition and affect. Journal of personality and social psychology, 100(3), 407.

It doesn't really matter that he got significant effects of pre-cognition; the prior probability of people being able to tell the future accurately is near 0. So any evidence he reports of pre-cognition doesn't shift my belief at all.

Here is another example, looking at frequentist null hypothesis testing and how it shifts our beliefs about the null hypothesis:

Daniel J. Schad and Shravan Vasishth. The posterior probability of a null hypothesis given a statistically significant result. Quantitative Methods for Psychology, 2022.

https://danielschad.shinyapps.io/probnull/