Bayes Assignment 4 of 2025

Sean van der Merwe

2025-03-12

*Activate the navigation pane (View menu) for ease of navigation through the document.*

# Instructions

In this assignment you will conduct a simulation study to compare two priors in terms of their performance on a particular task.

The scenario is that a set of proportions are measured and a decision must be made whether they are close enough to uniform or adhering to a beta distribution with parameter values away from 1. The procedure is that a beta distribution is fitted to the proportions, and if the posterior median estimates of the parameters fall in the region where are close to 1 then the conclusion is made that they are ‘close enough to uniform’. If you incorrectly conclude they are close enough then the loss is ; while if you incorrectly conclude they are too far away then the loss is . Correct conclusions make a profit of .

From the history, it is known that samples come from either a (uniform) distribution with probability , or a distribution with probability .

The two priors being compared are **(A)** the objective prior and **(B)** the subjective prior and .

Given the hyperparameter values next to your student number below, which prior provides the minimum risk (expected loss)? Answer this question on the basis of simulated samples.

options(scipen = 12)  
library(knitr)  
library(tidyverse)  
library(devEMF)  
opts\_chunk$set(dev='emf', fig.ext='emf')

# Generate samples that are different for each student  
# This is the code used to generate the data, for interest only.  
set.seed(202503)  
students <- as.character(c(2014095653,2017159092,2017418365,2018006516,2018395968,2019369780,2020231664,2021603747,2024180487,2028830517, 2021234567, 2022345678, 2023456789))  
nn <- length(students)  
hyper\_pars <- data.frame(  
 Student\_num = students,  
 h1 = runif(nn, 0.91, 0.97) |> round(2),  
 h2 = runif(nn, 1.03, 1.11) |> round(2),  
 h3 = runif(nn, 0.92, 0.97) |> round(2),  
 h4 = runif(nn, 1.03, 1.09) |> round(2),  
 h5 = runif(nn, 200, 400) |> ceiling(),  
 h6 = runif(nn, 300, 500) |> ceiling(),  
 h7 = runif(nn, 80, 160) |> ceiling(),  
 h8 = runif(nn, 0.3, 0.7) |> round(2),  
 h9 = runif(nn, 0.85+0.2\*(seq\_len(nn)%%2), 0.95+0.2\*(seq\_len(nn)%%2)) |> round(2),  
 h10 = runif(nn, 0.8+0.3\*(seq\_len(nn)%%2), 0.9+0.3\*(seq\_len(nn)%%2)) |> round(2),  
 h11 = runif(nn, 0.8, 2.2) |> round(1),  
 h12 = runif(nn, 0.8, 2.2) |> round(1),  
 n = runif(nn, 30, 50) |> ceiling(),  
 M = (runif(nn, 10, 16) |> ceiling())\*100  
)

hyper\_pars[,1:8] |> kable()

| Student\_num | h1 | h2 | h3 | h4 | h5 | h6 | h7 |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2014095653 | 0.96 | 1.08 | 0.96 | 1.09 | 315 | 442 | 112 |
| 2017159092 | 0.93 | 1.07 | 0.95 | 1.06 | 263 | 365 | 144 |
| 2017418365 | 0.95 | 1.06 | 0.97 | 1.06 | 388 | 326 | 151 |
| 2018006516 | 0.97 | 1.07 | 0.92 | 1.04 | 267 | 382 | 143 |
| 2018395968 | 0.94 | 1.03 | 0.96 | 1.08 | 236 | 424 | 107 |
| 2019369780 | 0.95 | 1.06 | 0.92 | 1.04 | 270 | 483 | 130 |
| 2020231664 | 0.93 | 1.06 | 0.97 | 1.03 | 295 | 335 | 104 |
| 2021603747 | 0.91 | 1.03 | 0.94 | 1.04 | 328 | 431 | 96 |
| 2024180487 | 0.97 | 1.11 | 0.94 | 1.07 | 324 | 400 | 96 |
| 2028830517 | 0.94 | 1.10 | 0.93 | 1.05 | 366 | 397 | 133 |
| 2021234567 | 0.94 | 1.06 | 0.96 | 1.08 | 324 | 429 | 152 |
| 2022345678 | 0.94 | 1.07 | 0.93 | 1.09 | 207 | 314 | 101 |
| 2023456789 | 0.95 | 1.04 | 0.93 | 1.04 | 398 | 480 | 98 |

hyper\_pars[,c(1, 9:15)] |> kable()

| Student\_num | h8 | h9 | h10 | h11 | h12 | n | M |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2014095653 | 0.32 | 1.09 | 1.16 | 2.0 | 1.7 | 50 | 1300 |
| 2017159092 | 0.40 | 0.91 | 0.86 | 1.5 | 2.0 | 43 | 1500 |
| 2017418365 | 0.62 | 1.09 | 1.13 | 1.8 | 1.9 | 36 | 1500 |
| 2018006516 | 0.42 | 0.92 | 0.88 | 2.0 | 1.9 | 41 | 1600 |
| 2018395968 | 0.34 | 1.14 | 1.17 | 1.9 | 0.9 | 41 | 1500 |
| 2019369780 | 0.46 | 0.86 | 0.81 | 1.9 | 1.7 | 37 | 1200 |
| 2020231664 | 0.36 | 1.07 | 1.18 | 1.9 | 1.2 | 37 | 1300 |
| 2021603747 | 0.67 | 0.88 | 0.85 | 0.8 | 1.1 | 46 | 1500 |
| 2024180487 | 0.70 | 1.10 | 1.18 | 1.0 | 1.8 | 49 | 1300 |
| 2028830517 | 0.40 | 0.88 | 0.81 | 1.3 | 1.3 | 45 | 1600 |
| 2021234567 | 0.68 | 1.09 | 1.15 | 2.1 | 1.5 | 45 | 1100 |
| 2022345678 | 0.69 | 0.89 | 0.85 | 1.6 | 1.5 | 46 | 1500 |
| 2023456789 | 0.48 | 1.13 | 1.19 | 0.9 | 2.0 | 44 | 1600 |

# Memorandum

student\_number <- '2023456789'  
s <- hyper\_pars |> filter(Student\_num %in% student\_number)

The approach to solving such a problem is to break it into small steps. The first step is to simulate a single sample, fit the model, and calculate the observed loss. The second step is to then embed the process in a function, which can then be called repeatedly. The final step is to calculate the risks and draw a conclusion.

## Single sample simulation and fit

The first step in simulating a sample is to select a ground truth or state of nature, using the given probability. Then a sample is generated given this ground truth.

is\_uniform\_truth <- runif(1) <= s$h8  
if (is\_uniform\_truth) {  
 x <- rbeta(s$n, 1, 1)  
} else {  
 x <- rbeta(s$n, s$h9, s$h10)  
}

Now we fit the model using Stan.

library(rstan)  
mycores <- 3  
options(mc.cores = mycores)

First the model must be defined. Either we define two models, one for each prior, or we include a prior switching mechanism in the model.

data {  
 int n; // sample size  
 real x[n]; // sample  
 real h11; // hyperprior value  
 real h12; // hyperprior value  
 int prior; // prior switch: 0 for objective or 1 for subjective  
}  
parameters {  
 real<lower=0> a;  
 real<lower=0> b;  
}  
model {  
 x ~ beta(a, b);  
 target += prior\*(lognormal\_lpdf(a | 0, h11) +   
 lognormal\_lpdf(b | 0, h12)) -   
 (1-prior)\*(log(a) + log(b));  
}

The model is fit using each prior.

post\_fit\_obj <- beta\_model |> sampling(  
 list(n = s$n, x = x, h11 = s$h11, h12 = s$h12, prior = 0),  
 chains = mycores)  
post\_fit\_sbj <- beta\_model |> sampling(  
 list(n = s$n, x = x, h11 = s$h11, h12 = s$h12, prior = 1),  
 chains = mycores)

From the fit we obtain the posterior median estimates and check whether they meet the given criteria. In general we would calculate the estimates from the simulations, but in this case it has already been done and we can obtain them from the model summary.

(post\_fit\_obj |> summary())$summary |> kable(digits = 3)

|  | mean | se\_mean | sd | 2.5% | 25% | 50% | 75% | 97.5% | n\_eff | Rhat |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | 0.863 | 0.005 | 0.169 | 0.561 | 0.745 | 0.853 | 0.968 | 1.215 | 1044.087 | 1.001 |
| b | 0.896 | 0.006 | 0.177 | 0.585 | 0.770 | 0.886 | 1.007 | 1.286 | 1035.975 | 1.001 |
| lp\_\_ | -0.736 | 0.032 | 1.058 | -3.483 | -1.099 | -0.412 | -0.001 | 0.255 | 1107.333 | 1.001 |

(post\_fit\_sbj |> summary())$summary |> kable(digits = 3)

|  | mean | se\_mean | sd | 2.5% | 25% | 50% | 75% | 97.5% | n\_eff | Rhat |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | 0.873 | 0.005 | 0.168 | 0.577 | 0.755 | 0.862 | 0.976 | 1.231 | 1190.603 | 1.001 |
| b | 0.899 | 0.005 | 0.174 | 0.588 | 0.780 | 0.888 | 1.004 | 1.270 | 1165.133 | 1.002 |
| lp\_\_ | -3.173 | 0.028 | 1.022 | -5.982 | -3.581 | -2.847 | -2.440 | -2.187 | 1331.563 | 1.000 |

post\_fit\_obj\_ests <- (post\_fit\_obj |> summary())$summary[,"50%"]  
post\_fit\_sbj\_ests <- (post\_fit\_sbj |> summary())$summary[,"50%"]

To check the criteria and calculate the loss we can define a function as this simplifies the code.

get\_loss <- function(ests, s, is\_uniform\_truth) {  
 inside <- (s$h1 < ests[1]) & (ests[1] < s$h2) & (s$h3 < ests[2]) & (ests[2] < s$h4)  
 loss <- (inside & (!is\_uniform\_truth))\*s$h5 + ((!inside) & is\_uniform\_truth)\*s$h6 -  
 (inside & is\_uniform\_truth)\*s$h7 - ((!inside) & (!is\_uniform\_truth))\*s$h7  
}  
loss\_obj <- post\_fit\_obj\_ests |> get\_loss(s, is\_uniform\_truth)  
loss\_sbj <- post\_fit\_sbj\_ests |> get\_loss(s, is\_uniform\_truth)

In this case the losses are -98 and -98.

To calculate the expected losses in general, the above procedure must be repeated times.

## Repetition

A function is defined to do the above steps.

simulate\_losses <- function(sample\_number, chains = mycores) {  
 is\_uniform\_truth <- runif(1) <= s$h8  
 if (is\_uniform\_truth) {  
 x <- rbeta(s$n, 1, 1)  
 } else {  
 x <- rbeta(s$n, s$h9, s$h10)  
 }  
 post\_fit\_obj <- beta\_model |> sampling(  
 list(n = s$n, x = x, h11 = s$h11, h12 = s$h12, prior = 0),  
 chains = chains, cores = chains, verbose = FALSE)  
 post\_fit\_sbj <- beta\_model |> sampling(  
 list(n = s$n, x = x, h11 = s$h11, h12 = s$h12, prior = 1),  
 chains = chains, cores = chains, verbose = FALSE)  
 post\_fit\_obj\_ests <- (post\_fit\_obj |> summary())$summary[,"50%"]  
 post\_fit\_sbj\_ests <- (post\_fit\_sbj |> summary())$summary[,"50%"]  
 loss\_obj <- post\_fit\_obj\_ests |> get\_loss(s, is\_uniform\_truth)  
 loss\_sbj <- post\_fit\_sbj\_ests |> get\_loss(s, is\_uniform\_truth)  
 data.frame(Smpl\_Num = sample\_number,  
 Objective\_Loss = loss\_obj,  
 Subjective\_Loss = loss\_sbj)  
}

The function is run repeatedly and the results combined.

We first do a test run to check the expected run time though.

system.time({  
 test\_df <- seq\_len(5) |> lapply(simulate\_losses) |> bind\_rows()  
})

user system elapsed   
 3.60 4.09 51.55

We note that the process is slow and inefficient due to wasting a lot of time creating and closing child R processes. To speed it up we can reduce the number of chains to 1. This is only an option if we are certain the simulation process is running reliably (must do extensive checks, like seeing that the Rhat values are **always** close to 1).

system.time({  
 test\_df <- seq\_len(10) |> lapply(simulate\_losses, 1) |> bind\_rows()  
})

Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quantiles may be unreliable.  
Running the chains for more iterations may help. See  
https://mc-stan.org/misc/warnings.html#tail-ess

Another option for gaining speed is to run the processes in parallel.

library(parallel)  
cl <- makeCluster(mycores)  
cl |> clusterEvalQ(library(rstan)) |> invisible()  
cl |> clusterExport(c('s', 'get\_loss', 'beta\_model')) |> invisible()  
results\_df <- cl |> parLapplyLB(seq\_len(s$M), simulate\_losses, chains = 1) |> bind\_rows()  
cl |> stopCluster()

Now we can calculate the expected losses by averaging over the simulations.

Objective\_Prior\_Risk <- mean(results\_df$Objective\_Loss)  
Subjective\_Prior\_Risk <- mean(results\_df$Subjective\_Loss)

In this case the risks are 176.6 for the objective prior and 176.5 for the subjective prior.