STAT 8004: Homework 2

Srikar Katta

3/6/2022

Q1. Priors: $\mathbb{P}(\theta_A) = \mathbb{P}(\theta_B) = Gamma(2,1)$ **Data:** $\sum_{i=1}^{10} Y_i^A = 217, \sum_{i=1}^{10} Y_i^B = 66$

For notational convenience, let $\Sigma_A = \sum_{i=1}^{10} Y_i^A = 217$ and $\Sigma_B = \sum_{i=1}^{10} Y_i^B = 66$. Additionally, assume that $Y_i^A \overset{\text{iid}}{\sim} Poisson(\theta_A)$ and $Y_i^B \overset{\text{iid}}{\sim} Poisson(\theta_B)$.

a. What is $\mathbb{P}(\theta_A > \theta_B|\Sigma_A, \Sigma_B)$? Recall that $\mathbb{P}(\theta_A > \theta_B|\Sigma_A, \Sigma_B)$ is equivalent to

$$\int_0^\infty \int_0^{\theta_A} \mathbb{P}(\theta_A, \theta_B|\Sigma_A, \Sigma_B) d\theta_B d\theta_A. \tag{1}$$

Since θ_A and θ_B are independent, Equation 1 simplifies to

$$\int_0^\infty \int_0^{\theta_A} \mathbb{P}(\theta_A|\Sigma_B)\mathbb{P}(\theta_B|\Sigma_B) d\theta_B d\theta_A. \tag{2}$$

Because $\theta_A, \theta_B \sim Gamma(2,1)$, and the Gamma is a conjugate prior with the Poisson likelihood, we have the following posteriors:

$$\mathbb{P}(\theta_A|\Sigma_A) = Gamma(2 + \Sigma_A, 1 + 10) = Gamma(219, 11)$$
$$\mathbb{P}(\theta_B|\Sigma_B) = Gamma(2 + \Sigma_B, 1 + 10) = Gamma(68, 11).$$

So, we finally have the following equality:

$$\mathbb{P}(\theta_A > \theta_B|\Sigma_A, \Sigma_B) = \int_0^\infty \int_0^{\theta_A} Gamma(219, 11) Gamma(68, 11) d\theta_B d\theta_A. \tag{3}$$

Unfortunately, we cannot compute Equation 3 analytically and therefore must rely on Monte Carlo simulations: $\mathbb{P}(\theta_A > \theta_B|\Sigma_A, \Sigma_B) = 0$ for all sample sizes between 100 and 100000 (see Figure 1).

```
q1a_fn <- function(n) {
  # goal: run monte carlo sims to estimate P(theta_a > theta_b | Y_A, Y_B)
  # inputs
    # n: numeric w/ number of monte carlo simulations
```

```
  # returns
    # numeric with proportion of simulated theta_a > theta_b

  theta_a <- rgamma(n = n, shape = 219, rate = 11)
  theta_b <- rgamma(n = n, shape = 68, rate = 11)

  prop <- sum(theta_a > theta_b) / n

  return(prop)
}
```

b. What is $\mathbb{P}(\tilde{Y}_A > \tilde{Y}_B|\Sigma_A, \Sigma_B)$, given the known posterior predictive distribution? Following the same mathematical steps as part a, we can show that

$$\mathbb{P}(\tilde{Y}_A > \tilde{Y}_B|\Sigma_A, \Sigma_B) = \int_0^\infty \int_0^{\tilde{Y}_A} \mathbb{P}(\tilde{Y}_A|\Sigma_A)\mathbb{P}(\tilde{Y}_B|\Sigma_B)d\tilde{Y}_B d\tilde{Y}_A. \tag{4}$$

Notice that $\mathbb{P}(\tilde{Y}_A|\Sigma_A)$ and $\mathbb{P}(\tilde{Y}_B|\Sigma_B)$ are the posterior predictive distributions. When we have a $\theta \sim Gamma(\alpha, \beta)$ prior and $Y \overset{\text{iid}}{\sim} Poisson(\theta)$ likelihood, the posterior predictive is $NegBin\left(\frac{n+\beta}{n+\beta+1}, \sum_{i=1}^{n} Y_i + \alpha\right)$, where n is the number of observations. Then,

$$\mathbb{P}(\tilde{Y}_A|\Sigma_A) = NegBin\left(\frac{10+1}{10+1+1}, \Sigma_A + 2\right) = NegBin\left(\frac{11}{12}, 219\right) \tag{5}$$

$$\mathbb{P}(\tilde{Y}_B|\Sigma_B) = NegBin\left(\frac{10+1}{10+1+1}, \Sigma_B + 2\right) = NegBin\left(\frac{11}{12}, 68\right). \tag{6}$$

```
q1b_fn <- function(n) {
  # goal: run monte carlo sims to estimate P(Y_tilde_a > Y_tilde_b | Y_A, Y_B)
  # inputs
    # n: numeric w/ number of monte carlo simulations
  # returns
    # numeric with proportion of simulated Y_tilde_a > Y_tilde_b

  y_tilde_a <- rnbinom(n = n, size = 219, p = 11/12)
  y_tilde_b <- rnbinom(n = n, size = 68, p = 11/12)

  prop <- sum(y_tilde_a > y_tilde_b) / n

  return(prop)

}
```

c. What is $\mathbb{P}(\tilde{Y}_A > \tilde{Y}_B|\Sigma_A, \Sigma_B)$ when the posterior predictive distribution is unknown? If we could not identify the posterior predictive distribution's analytic form, then we would need to utilize a sampling algorithm. First, let us expand expand $\mathbb{P}(\tilde{Y}|\sum Y)$ for the Poisson-Gamma likelihood-prior combination.

$$\begin{aligned}\mathbb{P}(\tilde{Y}|\sum Y) &= \int_\Theta \mathbb{P}(\tilde{Y}, \theta|\sum Y)d\theta \\ &= \int_\Theta \mathbb{P}(\tilde{Y}|\theta)\mathbb{P}(\theta|\sum Y)d\theta \\ &= \int_0^\infty Poisson(\tilde{Y};\theta)Gamma\left(\theta|\sum Y;\alpha,\beta\right)d\theta.\end{aligned}$$

We can utilize the following Monte Carlo algorithm:

1. For $i = 1, \ldots, N$
   (a) $\theta_{A_i} \sim Gamma(219, 11)$ and $\theta_{B_i} \sim Gamma(68, 11)$
   (b) $\tilde{Y}_{A_i} \sim Poisson(\theta_{A_i})$ and $\tilde{Y}_{B_i} \sim Poisson(\theta_{B_i})$
2. Compare $(\tilde{Y}_{A_1}, \ldots, \tilde{Y}_{A_N})$ with $(\tilde{Y}_{B_1}, \ldots, \tilde{Y}_{B_N})$ to approximate $\mathbb{P}(\tilde{Y}_A > \tilde{Y}_B|\Sigma_A, \Sigma_B)$

```
q1c_fn <- function(n) {
  # goal: run monte carlo sims to estimate P(Y_tilde_a > Y_tilde_b | Y_A, Y_B): post pred unknown
  # inputs
    # n: numeric w/ number of monte carlo simulations
  # returns
    # numeric with proportion of simulated Y_tilde_a > Y_tilde_b

  theta_a <- rgamma(n = n, shape = 219, rate = 11)
  theta_b <- rgamma(n = n, shape = 68, rate = 11)

  y_tilde_a <- sapply(theta_a, function(lambda) rpois(n = 1, lambda = lambda))
  y_tilde_b <- sapply(theta_b, function(lambda) rpois(n = 1, lambda = lambda))

  prop <- sum(y_tilde_a > y_tilde_b) / n

  return(prop)
}
```

Q2

a. Sample 10,000 samples from a standard normal distribution truncated at [0,1]

1) Use accept-reject algorithm and report the mean, variance, median, and time to completion. Report the acceptance rate. Because R's for/while-loops require significant overhead time/space, I will

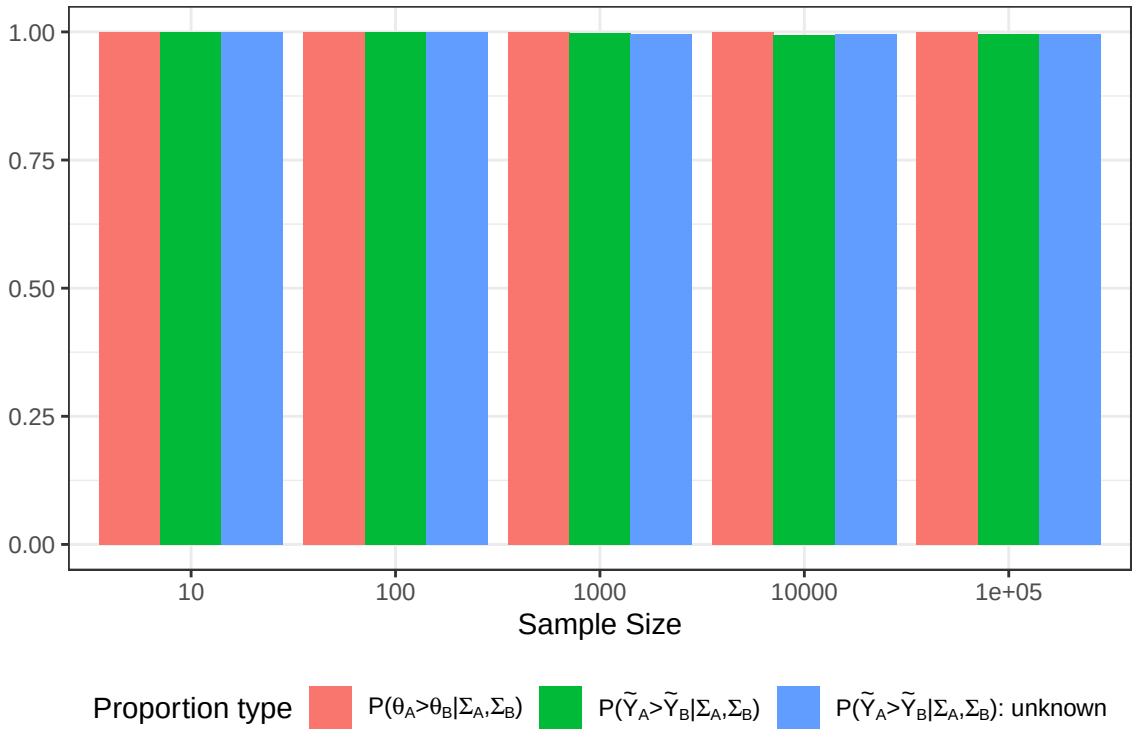


Figure 1: Q1 plots

sample 10,000 values from the standard normal distribution using R's vectorized sampling functions until I have a sample of 10,000.

```
q2a1_fn <- function() {
  # goal: run simulation using accept-reject algorithm to find 10,000 samples from Normal(0,1) truncate
  # inputs
  # returns
    # table with mean, variance, median, acceptance rate, and completion time of process

  n <- 0

  samples_0_1 <- c()
  start <- Sys.time()
  while(length(samples_0_1) < 10000) {
    samples <- rnorm(n = 10000, mean = 0, sd = 1)
    samples_0_1 <- c(samples_0_1, samples[(0 <= samples & samples <= 1)])
    n <- n + 10000
  }

  end <- Sys.time()
  stats <- tibble(Mean = mean(samples_0_1),
                  Variance = var(samples_0_1),
                  Median = median(samples_0_1),
```

```
                  `Acceptance Rate (%)` = length(samples_0_1)/n * 100,
                  `Completion time (sec)` = as.numeric(end - start))
  return(stats)
}

q2a1_fn() %>%
  pander::pander(round = 3, caption = 'Question 2, part a.1 results')
```

Table 1: Question 2, part a.1 results

Mean	Variance	Median	Acceptance Rate (%)	Completion time (sec)
0.459	0.08	0.441	34.38	0.003

2) Use method of inversion and report the mean, variance, median, and time to completion. Let X be a random variable with density equivalent to the standard Normal distribution truncated on $[0, 1]$. Then,

$$f_X(x) \propto \frac{1}{\sqrt{2\pi}} exp\left(-\frac{1}{2}x^2\right), 0 \leq x \leq 1. \tag{7}$$

We can find the normalizing constant by dividing integrating over the support of Equation 7,

$$\int_0^1 \frac{1}{\sqrt{2\pi}} exp\left(-\frac{1}{2}x^2\right) dx = \int_{-\infty}^1 \frac{1}{\sqrt{2\pi}} exp\left(-\frac{1}{2}x^2\right) dx - \int_{-\infty}^0 \frac{1}{\sqrt{2\pi}} exp\left(-\frac{1}{2}x^2\right) dx,$$

which is the same as $\Phi(1) - \Phi(0)$, where Φ represents the CDF of the standard Normal. Therefore,

$$f_X(x) = \frac{1}{\Phi(1) - \Phi(0)} \left(\frac{1}{\sqrt{2\pi}} exp\left(-\frac{1}{2}x^2\right)\right). \tag{8}$$

Now, we need to find the cumulative distribution function of X,

$$\begin{aligned} F_X(x) &= \int_0^x f_X(s)ds & (9)\\ &= \int_0^x \frac{1}{\Phi(1) - \Phi(0)} \left(\frac{1}{\sqrt{2\pi}} esp\left(-\frac{1}{2}s^2\right)\right) ds & (10)\\ &= \frac{1}{\Phi(1) - \Phi(0)} \int_0^x \left(\frac{1}{\sqrt{2\pi}} esp\left(-\frac{1}{2}s^2\right)\right) ds & (11)\\ &= \frac{\Phi(x) - \Phi(0)}{\Phi(1) - \Phi(0)}. & (12) \end{aligned}$$

In order to implement the method of inversion, we need to find a function $F^{-1} : [0, 1] \to [0, 1]$ such that $F^{-1}(F_X(x)) = x$. I claim that $F^{-1}(x) = \Phi^{-1}\left((\Phi(1) - \Phi(0))x + \Phi(0)\right)$, where Φ^{-1} represents the inverse CDF of the standard normal distribution.

Notice,

$$\begin{aligned} F^{-1}(F_X(x)) &= \Phi^{-1}\left(\Phi(1) - \Phi(0))F_X(x) + \Phi(0)\right) \\ &= \Phi^{-1}\left(\Phi(1) - \Phi(0))\frac{\Phi(x) - \Phi(0)}{\Phi(1) - \Phi(0)} + \Phi(0)\right) \\ &= \Phi^{-1}\left(\Phi(x) - \Phi(0) + \Phi(0)\right) \\ &= \Phi^{-1}\left(\Phi(x)\right) \\ &= x, \end{aligned}$$

our desired result. We can further simplify $F^{-1}(x)$ by recalling that $\Phi(0) = \frac{1}{2}$, so

$$F^{-1}(x) = \frac{\Phi(x) - 0.5}{\Phi(1) - 0.5}. \tag{13}$$

We can then utilize Equation 13 in our Method of Inversion algorithm.

```
q2a2_fn <- function() {
  # goal: utilize method of inversion to obtain 10,000 samples from truncated normal dist
  # inputs:
  # returns
    # table with mean, variance, median, acceptance rate, and completion time of process

  # helper function to compute inverse of uniform samples
  inverse_cdf <- function(u) {
    return((pnorm(u) - 1/2)/(pnorm(1) - 1/2))
  }
  start <- Sys.time()
  unif_samples <- runif(n = 10000, min = 0, max = 1)
  samples_0_1 <- sapply(unif_samples, inverse_cdf)
  end <- Sys.time()

  stats <- tibble(Mean = mean(samples_0_1),
                  Variance = var(samples_0_1),
                  Median = median(samples_0_1),
                  `Completion time (sec)` = as.numeric(end - start))
  return(stats)
}

q2a2_fn() %>%
  pander::pander(round = 3, caption = 'Question 2, part a.2 results')
```

Table 2: Question 2, part a.2 results

Mean	Variance	Median	Completion time (sec)
0.542	0.086	0.565	0.041

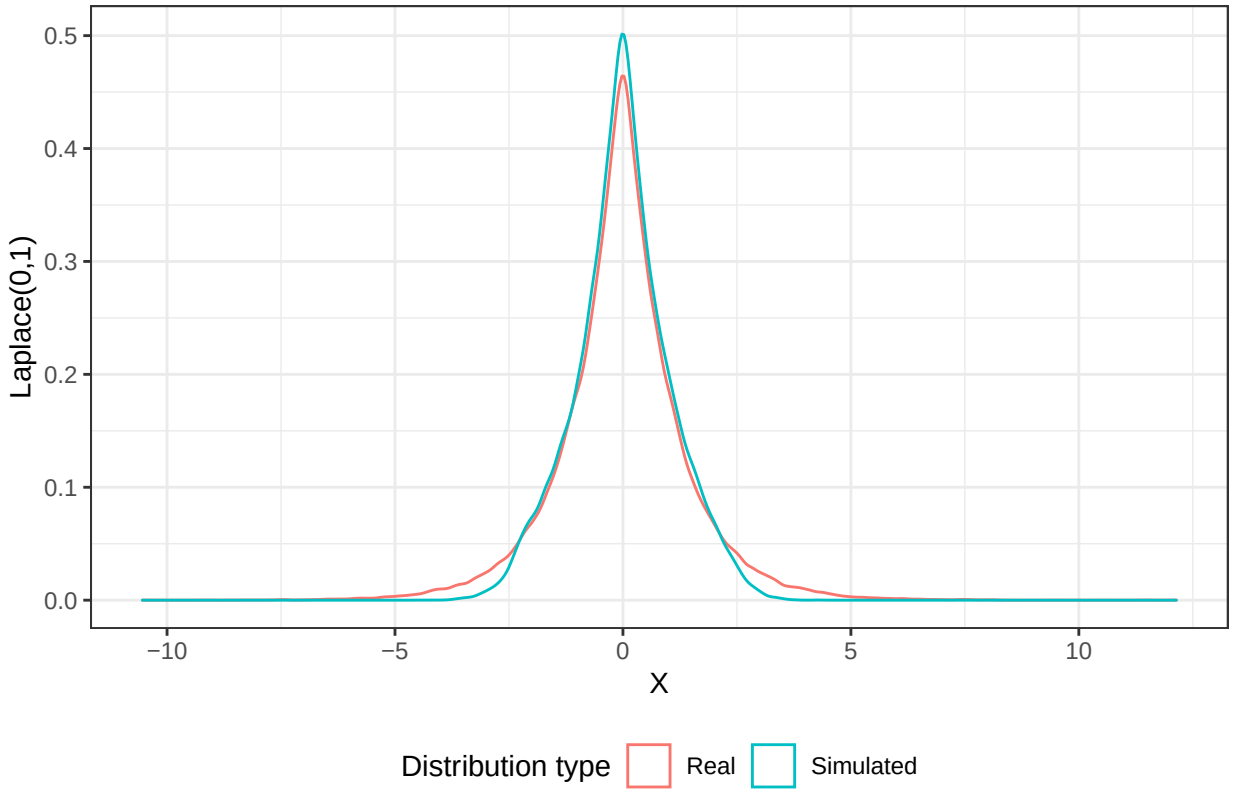
b. Use the standard normal distribution (for samples 100000) as an envelope to obtain samples from the Laplace(0,1) distribution (rejection sampling). Plot the empirical and theoretical distribution of the Laplace, and the rejection rate. In our rejection sampling algorithm, we need to decide on a hyperparameter M that scales the envelope distribution such $f_X(x) \leq Mg_X(x)$, where f_X is the distribution of interest and g_X is the proposed envelope distribution. I propose using $M^* = 1.5$ as the scaling factor since the Laplace(0,1) and standard Normal distributions are fairly similar.

```
q2b_fn <- function(M) {
  # goal: use rejection sampling to sample from Laplace distribution
  # inputs:
    # M: numeric scaling factor btw [1, infty)
  # outputs:
    # table with simulation results
  # helper function to calculate density of Laplace
  laplace_prob_calc <- function(x) {
    return((1/2) * exp(-abs(x)))
  }

  tibble(
    Simulated = rnorm(n = 100000, mean = 0, sd = 1),
    unif_samples = runif(n = 100000, min = 0, max = 1),
    Real = VGAM::rlaplace(n = 100000, location = 0, scale = 1)
  ) %>%
    mutate(norm_probs = dnorm(Simulated, mean = 0, sd = 1),
           laplace_probs = sapply(Simulated, laplace_prob_calc)) %>%
    pivot_longer(cols = c('Simulated', 'Real'), names_to = 'dist', values_to = 'sample') %>%
    filter((dist == 'Real') | (unif_samples <= laplace_probs/(norm_probs * M))) %>%
    return()
}

q2b_tbl <- q2b_fn(1.5)
q2b_rej <- 100 - sum(q2b_tbl$dist == 'Simulated')/100000 * 100
ggplot(q2b_tbl) +
  geom_density(aes(x = sample, color = dist)) +
  # geom_label(aes(x = 10, y = 0.4), label = paste0('Rejection rate: ', round(q2b_rej, 3))) +
  labs(x = TeX('$X$'),
       y = 'Laplace(0,1)',
       caption = paste0('Rejection rate: ', round(q2b_rej, 3)),
```

```
       color = 'Distribution type')
```



Rejection rate: 38.041

```
q2c_fn <- function(n) {
  # goal: find mean of gumbel(0,1) via importance sampling
  # inputs:
    # n: numeric representing sampling size
  # returns
    # table with simulation results
  dgumbel <- function(x) {
    exp(-(x + exp(-x))) %>% return()
  }
  tibble(
    norm_samples = rnorm(n = n, mean = 0, sd = 1),
    n_samples = n,
  ) %>%
    mutate(norm_probs = dnorm(norm_samples, mean = 0, sd = 1),
           gumbel_probs = dgumbel(norm_samples),
           imp_weight = gumbel_probs/norm_probs,
           gumbel_mean = sum(imp_weight * norm_samples)/n,
           gumbel_samples = VGAM::rgumbel(n = n, location = 0, scale = 1)
```

```
            ) %>%
    return()
}

q2c_tbl <- lapply(c(10, 100, 1000, 10000, 100000), q2c_fn) %>%
  bind_rows()
```

Sample from a standard normal distribution (for samples 10, 100, 1000, 10000, 100000) to obtain the mean of a standard Gumbel distribution (importance sampling). Plot the two distributions, the importance weights, and the mean over the # of samples.