Project 1 - TMA4300 Computer Intensive Statistical Methods

Jostein Aastebøl Aanes, Frederick Nilsen

February 11th 2022

Problem A

Sampling from an exponential distribution

We wish to develop a method for pulling samples from an exponential distribution using probability integral transform. The CDF of the exponential distribution is

$$F(x) = 1 - e^{-\lambda x}.$$

Let $F(x) = u$, then the inverse is given by

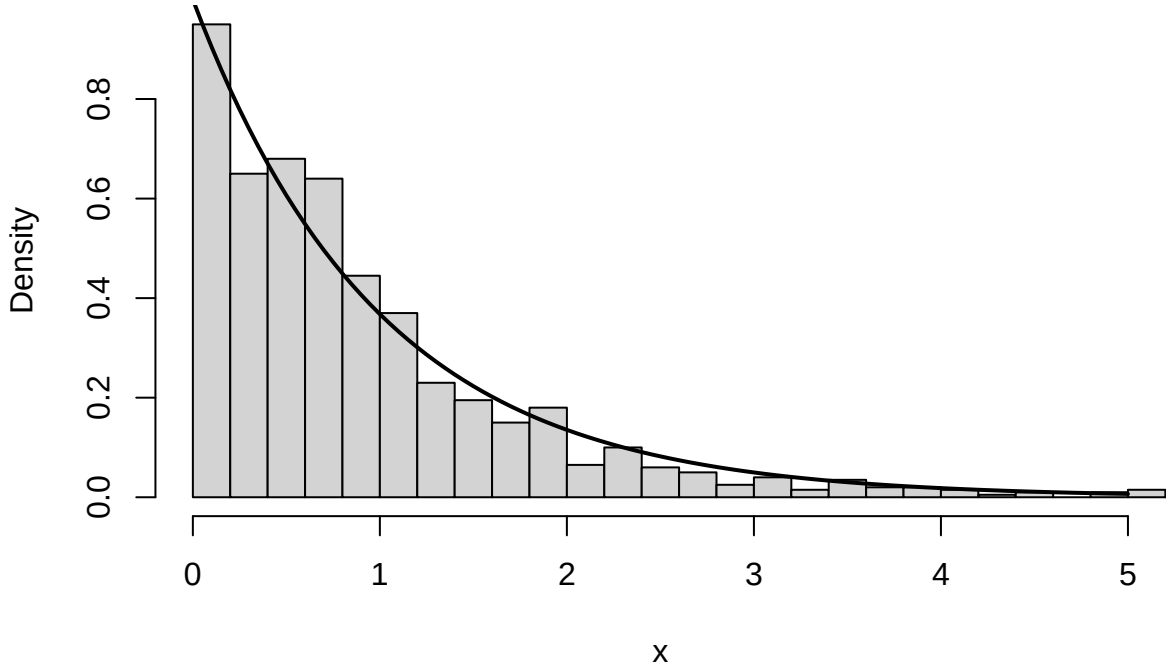
$$1 - u = e^{-\lambda x} \iff x = \frac{-1}{\lambda} \ln(1 - u).$$

Since we have that $u \sim \mathcal{U}[0, 1]$, so is $1 - u$. Implementing this in R, we do the following

```
sample_exp <- function(lam, n){
  u <- runif(n)
  #Here, using the inversion method, we get x from the exponential distribution.
  return(-1/lam * log(1-u))
  }
```

We can determine its accuracy by plotting some realizations of this function compared to the theoretical sample. Figure A.1 shows 1000 realizations of the sampling with $\lambda = 1$.

Figure A.1: Histogram of sampled values from exponential distribution versus theoretical density, n = 1000 , lambda = 1



As we can see, the sampling closely follows the theoretical density with only 1000 samples, indicating that our sampling algorithm is correct.

Integral transform of a discontinuous distribution

We now consider a probability density function given by

$$g(x) = \begin{cases} cx^{\alpha-1}, & 0 < x < 1, \\ ce^{-x}, & 1 \leq x, \\ 0, & \text{Otherwise} \end{cases}$$

where $\alpha \in (0, 1)$, and c is a normalizing constant. The cumulative distribution function $G(\tau)$ is, assuming $\tau \geq 1$,

$$\begin{aligned} \int_{-\infty}^{\tau} g(x)dx &= c\left(\int_0^1 x^{\alpha-1}dx + \int_1^{\tau} e^{-x}dx\right) \\ &= c\left(\frac{1}{\alpha}1^{\alpha} - e^{-\tau} + e^{-1}\right) \end{aligned}$$

We must also consider the case $0 < \tau < 1$, in which the CDF becomes

$$\int_0^{\tau} cx^{\alpha-1}dx = \frac{c}{\alpha}(\tau^{\alpha} - 0) = \frac{c\tau^{\alpha}}{\alpha}$$

The normalizing constant c is determined by solving for $\int_{-\infty}^{\infty} g(x)dx = 1$, i.e.

$$c\left(\frac{1}{\alpha} - 0 + e^{-1}\right) = 1 \iff c = \frac{1}{\frac{1}{\alpha} + \frac{1}{e}} = \frac{e\alpha}{\alpha + e}$$

Furthermore, the inverse of this CDF is found by solving $G(\tau) := u$.

For $1 \leq \tau$:

$$\begin{aligned} u &= c\left(\frac{1}{\alpha} - e^{-\tau} + e^{-1}\right) \\ \frac{u}{c} - \frac{1}{\alpha} &= e^{-1} - e^{-\tau} \\ \frac{u(\alpha+e) - e - a}{e\alpha} &= -e^{-\tau} \\ \frac{-e\alpha}{(\alpha+e)(u-1)} &= e^{\tau} \\ \therefore \tau &= -\ln\left(\frac{-e\alpha}{(\alpha+e)(u-1)}\right), \end{aligned}$$

and similarly for $0 < \tau < 1$,

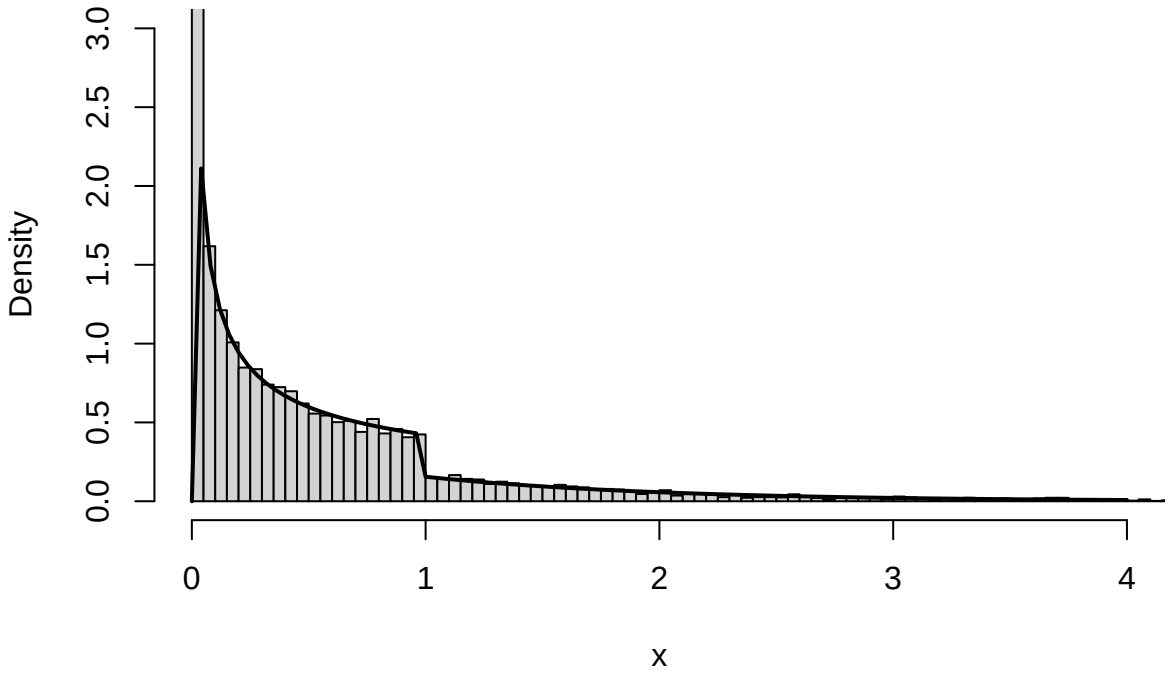
$$u = \frac{c\tau^{\alpha}}{\alpha} \iff \tau = \left(\frac{u\alpha}{c}\right)^{1/\alpha}$$

We can apply the inversion method, now that we have the inverses of the CDF to sample from $g(x)$. Note that since $g(x)$ is discontinuous at $x = 1$, the inverse of the CDF is also discontinuous.

```r
sample_gfunc <- function(n, alpha){
  # Function that samples out of the given distribution g(x)
  # n: int, number of samples
  # alpha: float, parameter in (0,1)
  # returns: n-sizes vector of samples

  u <- runif(n)
  c <- alpha*exp(1)/(alpha+exp(1))

  inv_cdf <- ifelse(
    u < c/alpha,
    (u*alpha/c)^(1/alpha), # First part of inverse CDF
    -log(1/alpha + exp(-1) - u/c) # Second part of inverse CDF
  )
  return(inv_cdf)
}

# Given density g(x)
g <- function(x, alpha){
  # Given density function g(x)
  # x: (vector of) x-values
  # alpha: parameter between 0 and 1
  c <- (exp(1)*alpha)/(alpha+exp(1)) # Normalizing constant
  return(
    ifelse(x <= 0, 0,
           ifelse(x < 1, c*x^(alpha-1),
                  c*exp(-x)
                  )
    ))
}
```

After sampling 10000 realizations of $g(x)$ choosing $\alpha = 0.5$, we obtain the histogram as shown in Figure A.2. We also plot the theoretical distribution of $g(x)$ over the histogram to determine its performance.

Figure A.2: Histogram of sampled values from g(x) versus theoretical density, n = 10000 alpha = 0.5



We see that the sampled values closely follows the analytic density for most of the domain. There is some deviation around $x \approx 1$, and a relatively large deviation when $x \approx 0$. The error can be explained by the discontinuity, since it at $x = 0$, $g(x)$ should be exactly 0 but as $x \to 0^+$, $g(x)$ has relatively high values for higher α.

Problem A3

We now consider the density

$$f(x) = \frac{ce^{\alpha x}}{(1+e^{\alpha x})^2},$$

where $-\infty < x < \infty, \quad \alpha > 0$ and c is a normalizing constant. We determine c by setting $\int_{-\infty}^{\infty} f(x)dx = 1$. Using the substitution $u = 1 + e^{\alpha x}$,

$$\frac{du}{dx} = \alpha e^{\alpha x} \iff dx = \frac{1}{\alpha e^{\alpha x}} du$$

$$\implies \int f(x)dx = \int \frac{cu}{\alpha u \cdot u^2} du = \int \frac{c}{\alpha u^2} du = \frac{c}{\alpha}\left(-\frac{1}{u}\right) + C = \frac{-c}{\alpha(1+e^{\alpha x})} + C$$

$$\implies \int_{-\infty}^{\infty} f(x)dx = \frac{c}{\alpha}\left(\lim_{x\to\infty}\left(\frac{-1}{1+e^{\alpha x}}\right) - \lim_{x\to-\infty}\left(\frac{-1}{1+e^{\alpha x}}\right)\right)$$

$$= \frac{c}{\alpha}\left(0 - \left(\frac{-1}{1+0}\right)\right) = \frac{c}{\alpha} = 1 \iff c = \alpha.$$

Thus, for $\int_{-\infty}^{\infty} f(x)dx = 1$ to hold c must be equal to α.

Furthermore, we wish to find the CDF, F:

$$F(x) = \int_{\infty}^{\tau} f(x)dx = \frac{-1}{1+e^{\alpha\tau}} + 1 := u,$$

where solving for τ gives the inverse of the CDF, namely

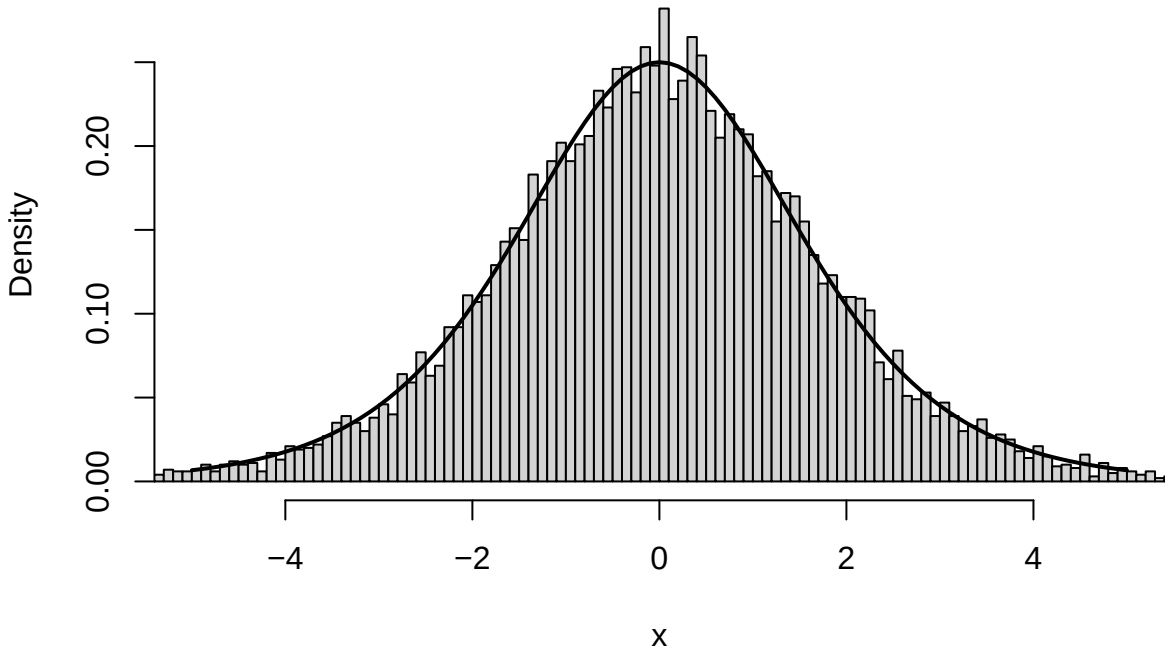
$$(u-1)(1+e^{\alpha\tau}) = -1 \iff 1+e^{\alpha\tau} = \frac{-1}{u-1} \iff \alpha\tau = \ln\left(\frac{-u}{u-1}\right) \iff \tau = \frac{1}{\alpha}\ln\left(\frac{-u}{u-1}\right)$$

We now generate samples from $f(x)$ using the inversion method:

```
sample_f_func <- function(n, alpha){
  # Samples from f(x) using on its inverse CDF
  u <- runif(n)
  return(1/alpha * log(-u/(u-1)))
}

f <- function(x, alpha){
  # Density function f(x)
  return(alpha*exp(alpha*x)/(1+exp(alpha*x))^2)
}
```

In Figure A.3, we see a histogram of the sampled values compared to the theoretical density $f(x)$. The plot uses $\alpha = 1$ and 10000 samples.

Figure A.3: Histogram of sampled values of f(x) versus theoretical density, n = 10000 alpha = 1



Although the histogram shows some deviations from the theoretical density, it still closely follows the density.

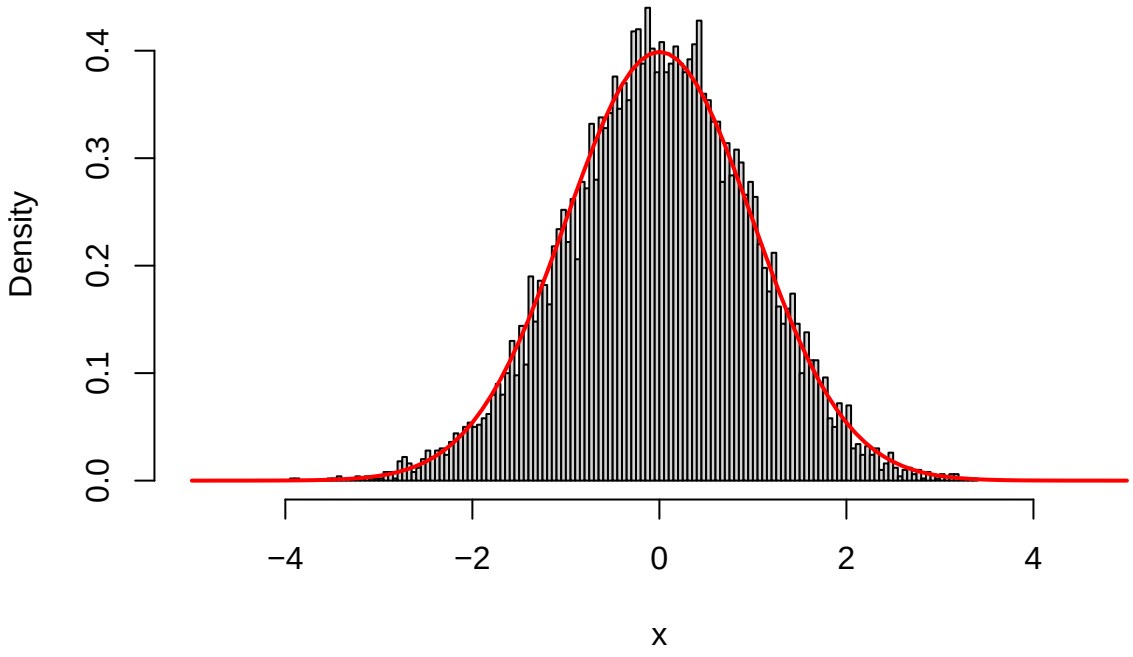
Box-Muller algorithm

In many cases, it is of interest to be able to generate samples from a standard normal distribution, $\mathcal{N}(0,1)$. We will now implement the Box-Muller algorithm in order to generate such samples.

```
sample_From_Normal_Distribution <- function(n){
  # A function that generates n samples from the standard Normal distribution
  x1 = runif(n, min = 0, max = (2*pi)) # n samples uniform(0,2 pi)

  # Generate random samples from exponential distribution with rate 1/2
  R <- sqrt(-2*log(runif(n)))

  return(R*cos(x1)) #Returns sample from standard normal distribution.
}
```

Figure A.4 shows the result of the Box-Muller algorithm using 10000 samples. The samples and theoretical distribution coincide well.

Figure A.4: Histogram of standard normal distributed samples versu theoretical density



Multivariate normal distribution

We now wish to sample from a multivariate normal distribution with an arbitrary mean and covariance. We can use the Box-Muller algorithm to produce n independent standard normal distributed variables. Using linearity and scaling, we can modify the samples to represent any mean and covariance.

```
generate_mvn <- function(d, mean, cov){
  # Function that generates realizations of MVN-distribution
  # d: int, number of dimensions
  # mean: mean vector
```

```
  # cov: covariance matrix
  # returns: d-dimensional vector of realizations

  # Draw d-length vector of N(0,1) realizations
  z <- sample_From_Normal_Distribution(d)
  # Cholesky-decompose cov-matrix
  A <- chol(cov)
  # Return the transformed N(mean, cov) realizations
  x <- A %*% z + mean
  return(x)
}
```

We can test the implementation by comparing the theoretical mean and covariance to its empirical values in a test case. Using

$$d = 2, \quad \mu = (1, 2)^T, \quad \Sigma = \begin{pmatrix} 5 & 1 \\ 1 & 5 \end{pmatrix},$$

we get the following results

```
# Test case values
test_mu <- c(1, 2)
test_Sigma <- matrix(c(5,1,1,5),2,2)

# Sampling from the normal distribution
sampled_1d <- c()
sampled_2d <- c()
for(i in 1:10000){ # Run 10 000 samples
  samp <- generate_mvn(2, test_mu, test_Sigma)
  sampled_1d <- append(sampled_1d, samp[1])
  sampled_2d <- append(sampled_2d, samp[2])
}
```

Compared to the theoretical mean $\mu = (1, 2)^T$, our sampled mean is

```
c(mean(sampled_1d), mean(sampled_2d))
```

```
## [1] 1.022325 1.999053
```

and the sampled covariance matrix is

```
matrix(c(
  var(sampled_1d), var(sampled_1d, sampled_2d),
  var(sampled_2d, sampled_1d), var(sampled_2d)),
  2,2)
```

```
##          [,1]     [,2]
## [1,] 5.219054 1.001841
## [2,] 1.001841 4.867260
```

In summary, we see about 4% deviation in the mean, but the covariance in this run has an error of about 15%. The relatively small error indicates that the Box-Muller algorithm can also be used to generate multivariate normal distributions with arbitrary mean and covariance.

Problem B

Gamma distribution and rejection sampling for $0 < \alpha < 1$

For this problem we wish to sample from a Gamma distribution where $\alpha \in (0, 1)$ and $\beta = 1$. We will be using rejection sampling, where we propose samples from

$$g(x) = d \cdot \begin{cases} x^{\alpha-1}, & 0 < x < 1 \\ e^{-x}, & 1 \leq x \\ 0, & \text{Otherwise} \end{cases}$$

where d is a normalizing constant equal to $\frac{1}{1/\alpha+1/e}$ as calculated previously in the section *Integral transform of a discontinuous distribution*.

To begin, we will find an expression for the acceptance probability. We have that the density of this gamma distribution is given as

$$f(x) = \begin{cases} \frac{1}{\Gamma(\alpha)} x^{\alpha-1} e^{-x}, & 0 < x \\ 0, & \text{Otherwise.} \end{cases}$$

The acceptance probability is given as $\gamma = \frac{1}{c} \frac{f(x)}{g(x)}$. Thus, the acceptance probability γ is

$$\gamma = \frac{1}{cd\Gamma(\alpha)} \begin{cases} e^{-x}, & 0 < x < 1 \\ x^{\alpha-1}, & 1 \leq x \\ 0, & \text{Otherwise.} \end{cases}$$

Now, we wish to find an expression for c. We require that c is the smallest value that still satisfies $\max_x \frac{f(x)}{g(x)} \leq c, \quad c > 1, \quad f(x) > 0$. Therefore we must find

$$\max_x \frac{f(x)}{g(x)} = \max_x \begin{cases} \frac{1}{\Gamma(\alpha)d} e^{-x}, & 0 < x < 1 \\ \frac{1}{\Gamma(\alpha)d} x^{\alpha-1}, & 1 \leq x \\ 0, & \text{Otherwise.} \end{cases}$$

Since both $\frac{1}{\Gamma(\alpha)d} e^{-x}$ and $\frac{1}{\Gamma(\alpha)d} x^{\alpha-1}$ are greater than 0 in their respective domains, we don't need to consider the case $x \leq 0$. Now consider that $\frac{d}{dx} e^{-x} < 0, \quad 0 < x < 1$, this means that

$$\max_{0<x<1} \frac{1}{\Gamma(\alpha)d} e^{-x} = \frac{1}{\Gamma(\alpha)d} e^{0} = \frac{1}{\Gamma(\alpha)d}$$

And now consider that $\frac{d}{dx} x^{\alpha-1} < 0, \quad \alpha < 1, \quad 1 \leq x$, this means that

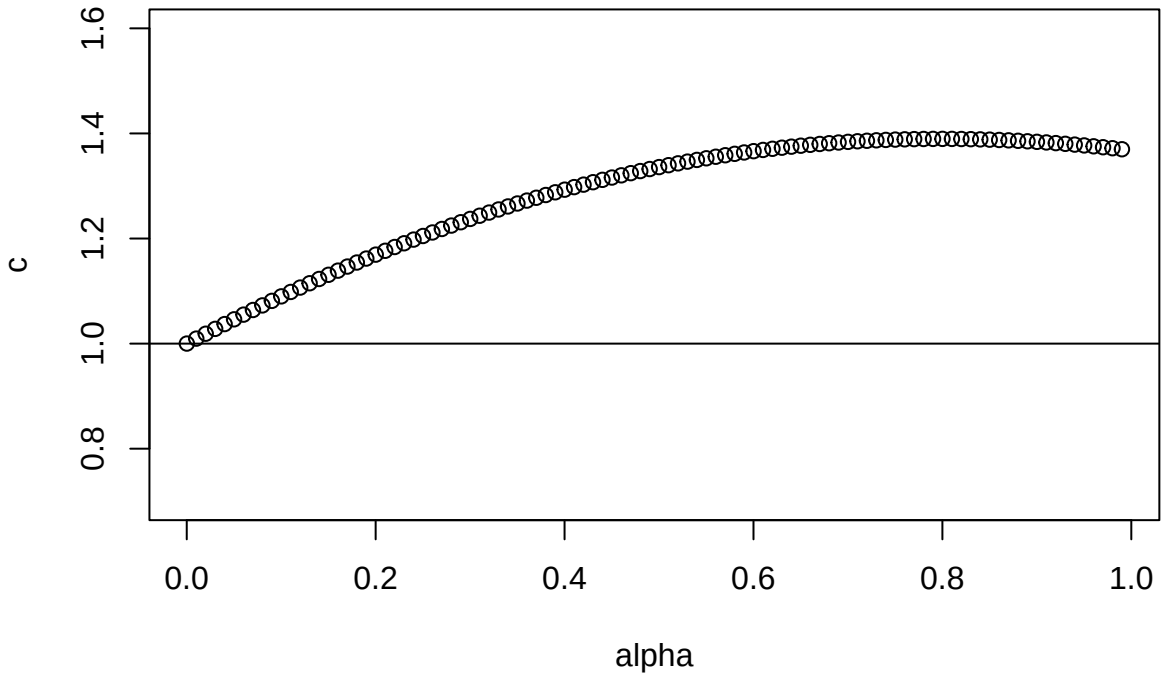
$$\max_{1 \leq x} \frac{1}{\Gamma(\alpha)d} x^{\alpha-1} = \frac{1}{\Gamma(\alpha)d}$$

Thus, we get that

$$c = \frac{1}{\Gamma(\alpha)d} = \frac{\alpha + e}{\Gamma(\alpha)\alpha e}.$$

We can see graphically from Figure B.1 that c is larger than 1 for all $\alpha \in (0, 1)$.

Figure B.1: c plotted against alpha



Implementation of rejection sampling

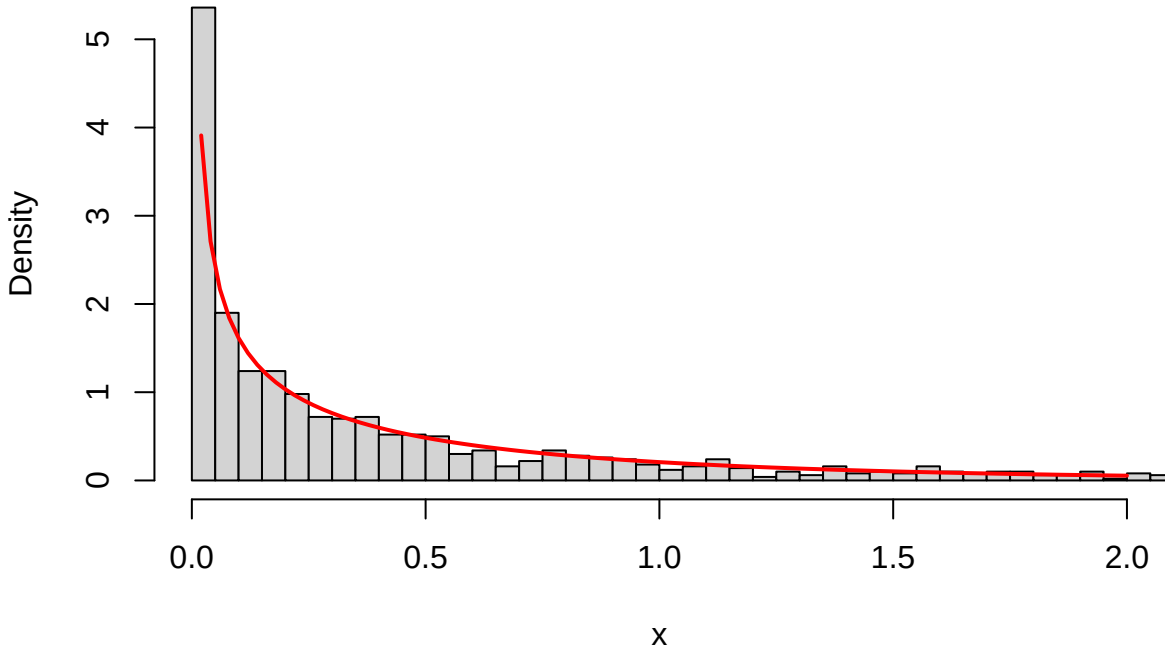
We will now use the result above to implement a rejection sampling algorithm to sample from the Gamma distribution with $0 < \alpha < 1$ and $\beta = 1$.

```
gamma_distribution <- function(x, alpha){
  #Function that returns actual values for the density of the
  #    gamma distribution with beta=1
  return ( exp(  (alpha-1)*log(x) -x - lgamma(alpha)  ) )
}

acceptance_probability <- function(x, alpha){
  # Function that calculates acceptance probability

  d <- alpha*exp(1)/(alpha+exp(1))
  c <- (alpha+exp(1))/(gamma(alpha)*alpha*exp(1))

  acceptance_probability = 1/(gamma(alpha)*c*d)*ifelse(
    x < 1,
    exp(-x),
    x^(alpha-1)
  )

  return(acceptance_probability)
}
sample_gamma_small_alpha <- function(n, alpha){
```

```
  # Rejection sampling algorithm
  x = sample_gfunc(n, alpha)
  u = runif(n)
  gamma = acceptance_probability(x, alpha)

  samples = x[u<gamma] # Acceptance criteria

  # Check if we have wanted number of samples. If not, call function again
  if(length(samples)<n){
    return(c(samples,sample_gamma_small_alpha(n-length(samples), alpha)))
  }

  return(samples)
}
```

Figure B.2 shows the histogram of the samples from the gamma distribution with $\alpha = 0.5$, compared to the theoretical density. The samples follow the theoretical density for the most part, but with larger errors as x approaches 0. This is due to the fact that $x^{\alpha-1} \to \infty$ as $x \to 0$. Thus, it is expected to see some higher deviations in this area.

Figure B.2: Histogram of samples from Gamma−distribution versus theoretical density, n = 10000, alpha = 0.5



Ratio of uniforms for Gamma distribution

For this problem, we wish to sample from the Gamma distribution with $\alpha > 1$ and $\beta = 1$, using ratio-of-uniforms method to generate the samples. We define

$$C_f = \left\{ (x_1, x_2) \ : \ 0 \leq x_1 \leq \sqrt{f^*\left(\frac{x_2}{x_1}\right)} \right\}, \quad f^*(x) = \begin{cases} x^{\alpha-1}e^{-x}, & 0 < x \\ 0, & \text{Otherwise.} \end{cases}$$

Since f^* is non-negative and obviously $\int_{-\infty}^{\infty} f^*(x)dx < \infty$, we get that C_f is a finite area and that if $y = \frac{x_2}{x_1}$, then

$$f(y) = \frac{f^*(y)}{\int_{-\infty}^{\infty} f^*(x)dx}$$

which means that $y \sim Gamma(\alpha, 1)$. Now, we can find a square $[0, a] \times [b_-, b_+]$ such that $C_f \subset [0, a] \times [b_-, b_+]$. Then, we can generate samples from within this square, and see if they land within C_f, if they do, then it is a sample from the desired Gamma distribution. Therefore, we have to calculate a, b_- and b_+. For $C_f \subset [0, a] \times [b_-, b_+]$ to hold, we must impose certain requirements on a, b_- and b_+, and they are as follows:

$$a = \sqrt{\sup_x f^*(x)}, \quad b_- = -\sqrt{\sup_{x \leq 0}(x^2 f^*(x))} \quad \text{and} \quad b_+ = \sqrt{\sup_{x \geq 0}(x^2 f^*(x))}.$$

Starting with a: Since $f^*(x) = \begin{cases} x^{\alpha-1}e^{-x}, & 0 < x \\ 0, & \text{Otherwise} \end{cases}$, we clearly know that $x^{\alpha-1}e^{-x} > 0, x > 0$. Therefore we only have to focus on $\sup_{x>0}(x^{\alpha-1}e^{-x})$. Now, we take the derivative of this, and get

$$\frac{d}{dx}(x^{\alpha-1}e^{-x}) = e^{-x}(\alpha-1)x^{\alpha-2} - e^{-x}x^{\alpha-1} = e^{-x}x^{\alpha-1}\left((\alpha-1)x^{-1} - 1\right).$$

Again, since $e^{-x}x^{\alpha-1} > 0, x > 0$, we look at $(\alpha-1)x^{-1} - 1$. This is equal to zero when $x = \alpha - 1$. The supremum is thus $(\alpha-1)^{\alpha-1}e^{-\alpha+1}$. Hence, we get that $a = \sqrt{(\alpha-1)^{\alpha-1}e^{-\alpha+1}}$.

Now, b_-:

$$b_- = -\sqrt{\sup_{x \leq 0}(x^2 f^*(x))} = -\sqrt{\sup_{x \leq 0}\left(x^2 \cdot \begin{cases} x^{\alpha-1}e^{-x}, & 0 < x \\ 0, & \text{Otherwise} \end{cases}\right)} = -\sqrt{0} = 0.$$

Lastly, b_+:

$$b_+ = \sqrt{\sup_{x \geq 0}(x^2 f^*(x))} = \sqrt{\sup_{x \geq 0}\left(x^2 \cdot \begin{cases} x^{\alpha-1}e^{-x}, & 0 < x \\ 0, & \text{Otherwise} \end{cases}\right)}.$$

Since $x^2 \cdot x^{\alpha-1}e^{-x} > 0, x > 0$, we get that $b_+ = \sqrt{\sup_{x>0} x^{\alpha+1}e^{-x}}$, where we consider the term inside the square root, i.e.

$$\sup_{x>0} x^{\alpha+1}e^{-x}.$$

We find the roots of the derivative

$$\frac{d}{dx}(x^{\alpha+1}e^{-x}) = e^{-x}x^{\alpha}(\alpha + 1 - x) = 0.$$

This then yields $x = \alpha + 1$. Hence, $b_+ = \sqrt{(\alpha+1)^{\alpha+1}e^{-\alpha-1}}$.

Implementation

In order to actually sample from $f(x)$ for $\alpha \in (0, 2000]$, we have to do some log-scaling. First, we have to log-scale the uniform distribution.

Since we have that $X \sim \mathcal{U}[0, a] = a \cdot \mathcal{U}[0, 1]$, we can log-scale the uniform distribution as $\ln(x) = \ln(a) + \ln(u)$, where $u \sim \mathcal{U}[0, 1]$.

When checking the condition $x_1 \leq \sqrt{f^*\left(\frac{x_2}{x_1}\right)}$, we must make sure to log-scale the condition and the distribution $f^*(x)$. Starting with the condition itself, we square both sides, and get that $x_1^2 \leq f^*\left(\frac{x_2}{x_1}\right)$, thus we get that $2 \cdot \ln(x_1) \leq \ln\left(f^*\left(\frac{x_2}{x_1}\right)\right)$. Now we compute the log-scaled version of $f^*(x)$. This is straight forward, $\ln(f^*(x)) = \ln(x^{\alpha-1}e^{-x}) = (\alpha - 1)\ln(x) - x$. Now, we can't compute either x_1 or x_2 itself, we need to get the ratio $\frac{x_2}{x_1}$ directly using log scales. We have the value $\frac{\ln(x_2)}{\ln(x_1)} = k$. Thus

$$\frac{\ln(x_2)}{\ln(x_1)} = k \implies k \cdot \ln(x_1) = \ln(x_2) \implies x_1^k = x_2 \implies x_1^{k-1} = \frac{x_2}{x_1}.$$

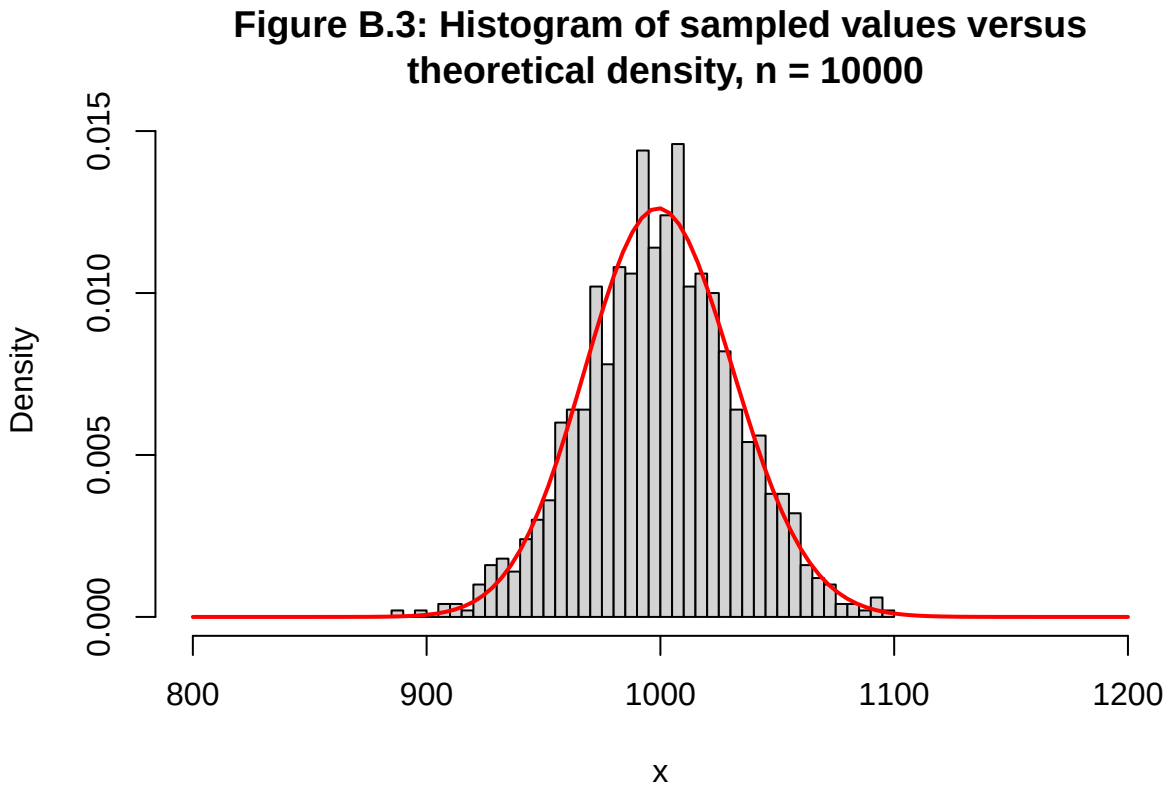
We now see that we can calculate $\frac{x_2}{x_1}$ from x_1^{k-1}. But, x_1^{k-1} will still produce NaN, so we have to do some more adjustments. Observe that $x_1^{k-1} = e^{(k-1)\ln(x_1)}$ which we can compute. Thus, we get that $y = e^{(k-1)\ln(x_1)}$, $k = \frac{\ln(x_2)}{\ln(x_1)}$.

Now, we can implement this as R code, which can be seen below. We wish to create 1000 samples for several $\alpha \in (1, 2000]$ and see how the number of attempts to create a 1000 samples changes as α increases.

```
f_star_log <- function(x, alpha){
  return((alpha-1)*log(x)-x)
}
sample_gamma_large_alpha <- function(n,alpha){
  #Function that samples from a gamma distribution with beta=1
  #    and for alpha larger than 1.
  #Sample from [0,a]x[b_-, b_+] and remove points where x_1>sqrt(f^*(x_2/x_1))

  #Defining intervals on log-scale
  a_log     = (alpha-1)/2 * (log(alpha-1)-1)
  b_min_log = 1
  b_max_log = (alpha+1)/2 * (log(alpha+1)-1)


  samples = rep(0,n) #Stores accepted samples
  i = 1
  counter=0
  while (i<=n) {
    counter = counter+1

    #Simulating x_1 and x_2
    x_1_log = a_log + log(runif(1))
    x_2_log = b_max_log + log(runif(1)) #sample_logscaled_uniform(1,b_max)
    k = x_2_log/x_1_log

    y = exp((k-1)*x_1_log)

    if(2*x_1_log <= f_star_log(y,alpha)){ #Accept because lays in C_f
      samples[i]=y
      i = i+1
```

```
    }

    if(counter>1e6){ #If number of attempts becomes too large
      break
    }


  }
  return(list(counter = counter, samples = samples))

}
```

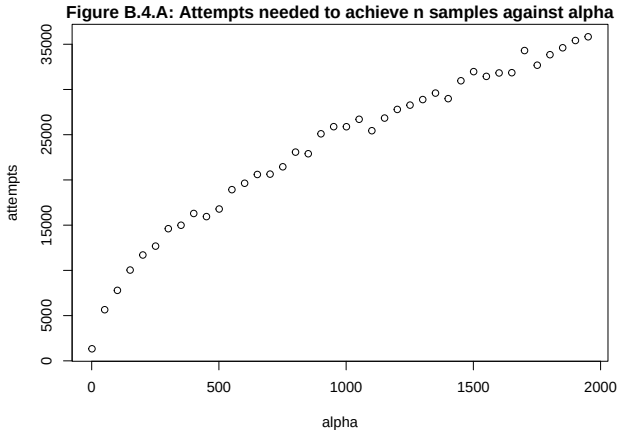
We can confirm that the method produces samples from a Gamma distribution in Figure B.3.

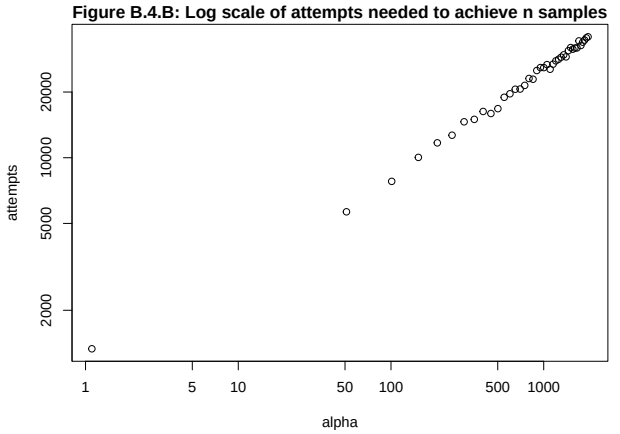


Now, we see how the number of attempts change as α increases from 1 to 2000.

```
alphas = seq(1.1, 2000, by = 50)
attempts = rep(0, length(alphas))
i = 1
for(alpha in alphas){
  samples = sample_gamma_large_alpha(1000, alpha)
  attempts[i] = samples$counter
  i = i + 1
}
par(mar=c(4,4,1.5,1.1), cex=0.9)
plot(alphas, attempts,
     main="Figure B.4.A: Attempts needed to achieve n samples against alpha", xlab="alpha")
```

```
plot(alphas, attempts, log='xy',
     main="Figure B.4.B: Log scale of attempts needed to achieve n samples", xlab="alpha")
```





Looking at figures B.4.A and B.4.B, the results make a lot of sense. As α increases, a and b_+ increases on the form α^α. This means that the area of the box increases more than the area of C_f, i.e. C_f makes out a smaller fraction of the square $[0, a] \times [b_-, b_+]$. Thus, it takes an increasing amount of attempts to generate the desired amount of samples.

We also note that the relationship between α and number of attempts are linear in the log-log plot in Figure B.4.B, indicating an exponential increase in attempts as α increases.

Sampling without parameter constraints

For this problem, we wish to sample from a Gamma distribution with $\alpha > 0$ and $\beta > 0$. We already know how to sample from a Gamma distribution with $\alpha \in (0, 1)$ and for $\alpha \in (1, \infty)$ (with $\beta = 1$). For $\alpha = 1$ we get an exponential distribution, which we also know how to sample from. In other words, to generate a Gamma distribution for $\alpha > 0$ and $\beta = 1$ is something we already know, we just have to use a conditional structure (if, else if, else), to check for which area α is in. Since β is an inverse scale parameter, all we have to do to get an arbitrary Gamma distribution is to take the samples from $Gamma(\alpha, 1)$ and divide by β. This is implemented below.

```
sample_Gamma <- function(n, alpha, beta){

  if(alpha<1){
    samples = sample_gamma_small_alpha(n, alpha)
  } else if(alpha==1){
    #Need exponential distribution right here
    samples = sample_exp(1, n)
  } else{
    result = sample_gamma_large_alpha(n, alpha)
    samples = result$samples
  }

  #Since beta is inverse scale parameter for Gamma distribution:
  samples = samples/beta
  return(samples)
}
```

The implementation is tested for three different alphas and betas. In Figure B.5.A, $\alpha = 0.5, \beta = 2$, in Figure B.5.B $\alpha = 1, \beta = 3$, and in Figure B.5.C $\alpha = 100, \beta = 1$. As observed in all three plots, the sample density coincide with the theoretical density, which is expected as it's a combination of previous results.

Figure B.5.A: alpha=0.5, beta=2

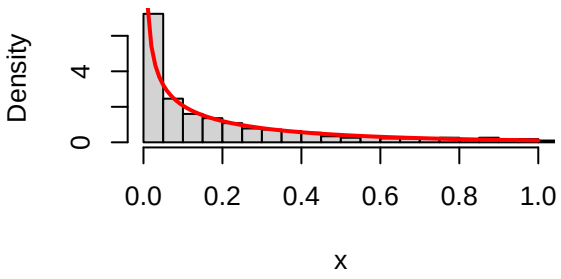


Figure B.5.B: alpha=1, beta=3

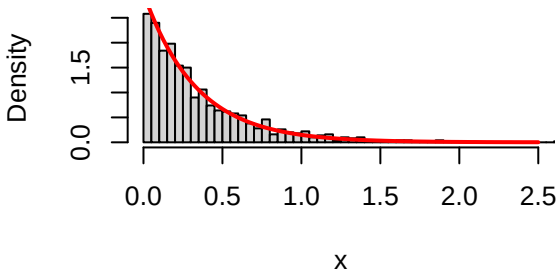
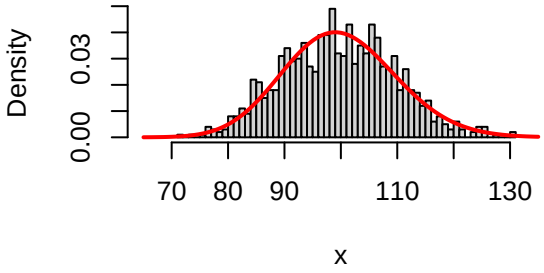


Figure B.5.C: alpha=100, beta=1



Relation to beta distribution

For this problem we define $X \sim \text{Gamma}(\alpha, 1)$ and $Y \sim \text{Gamma}(\beta, 1)$ whom are independent. We wish to show that $Z = \frac{X}{X+Y} \sim \text{Beta}(\alpha, \beta)$.

We have that the moment generating function for $X + Y$ is given as

$$M_{X+Y}(t) = E\left[e^{t(X+Y)}\right] = E\left[e^{tX}e^{tY}\right] = E\left[e^{tX}\right]E\left[e^{tY}\right],$$

where the last step is because X and Y are independent. This means that $M_{X+Y}(t) = M_X(t)M_Y(t)$. Hence,

$$M_{X+Y}(t) = \left(\frac{1}{1-t}\right)^{\alpha}\left(\frac{1}{1-t}\right)^{\beta} = \left(\frac{1}{1-t}\right)^{\alpha+\beta} \iff X + Y \sim Gamma(\alpha+\beta, 1).$$

Now, define $Z = \frac{X}{X+Y}$ and $T = X + Y$. Note that $T \sim \text{Gamma}(\alpha + \beta, 1)$. We know that

$$(Z, T) = g(X, Y) = \left(\frac{X}{X+Y},\ X+Y\right) \iff (X, Y) = g^{-1}(Z, T) = (ZT,\ T(1-Z)).$$

We use multivariate change-of-variables technique and get that the joint distribution for Z and T is given by

$$f_{Z,T}(z, t) = f_{X,Y}(g^{-1}(z, t)) \cdot |det(J)|,$$

where $f_{X,Y}$ is the joint distribution of X and Y, and J is the Jacobian of the transformation. We have that since X and Y are independent, that $f_{X,Y}(x, y) = f_X(x)f_Y(y)$. Calculating the determinant of the Jacobian,

$$det(J) = det\begin{pmatrix} \frac{\partial x}{\partial z} & \frac{\partial x}{\partial t} \\ \frac{\partial y}{\partial z} & \frac{\partial y}{\partial t} \end{pmatrix} = det\begin{pmatrix} t & z \\ -t & (1-z) \end{pmatrix} = t(1-z) + tz = t.$$

Thus, we get that

$$f_{Z,T}(z,t) = f_{X,Y}(g^{-1}(z,t)) \cdot t = f_X(zt) f_Y(t(1-z)) \cdot t.$$

Since $f_X(zt) = f_X(x)$ and $f_Y(t(1-z)) = f_Y(y)$, we get that

$$f_{Z,T}(z,t) = \frac{t}{\Gamma(\alpha)\Gamma(\beta)}(zt)^{\alpha-1}e^{-zt} \cdot (t(1-z))^{\beta-1}e^{-t}e^{zt} = \frac{1}{\Gamma(\alpha)\Gamma(\beta)}t^{\alpha+\beta-1}e^{-t}z^{\alpha-1}(1-z)^{\beta-1}.$$

Since the above term is a product of a marginal distribution of t and a marginal distribution of z, this proves that Z and T are independent. Hence, we get that $f_{Z,T}(z,t) = f_Z(z)f_T(t)$. We know that $T \sim \text{Gamma}(\alpha+\beta, 1)$, therefore we end up with

$$f_Z(z) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}z^{\alpha-1}(1-z)^{\beta-1}.$$

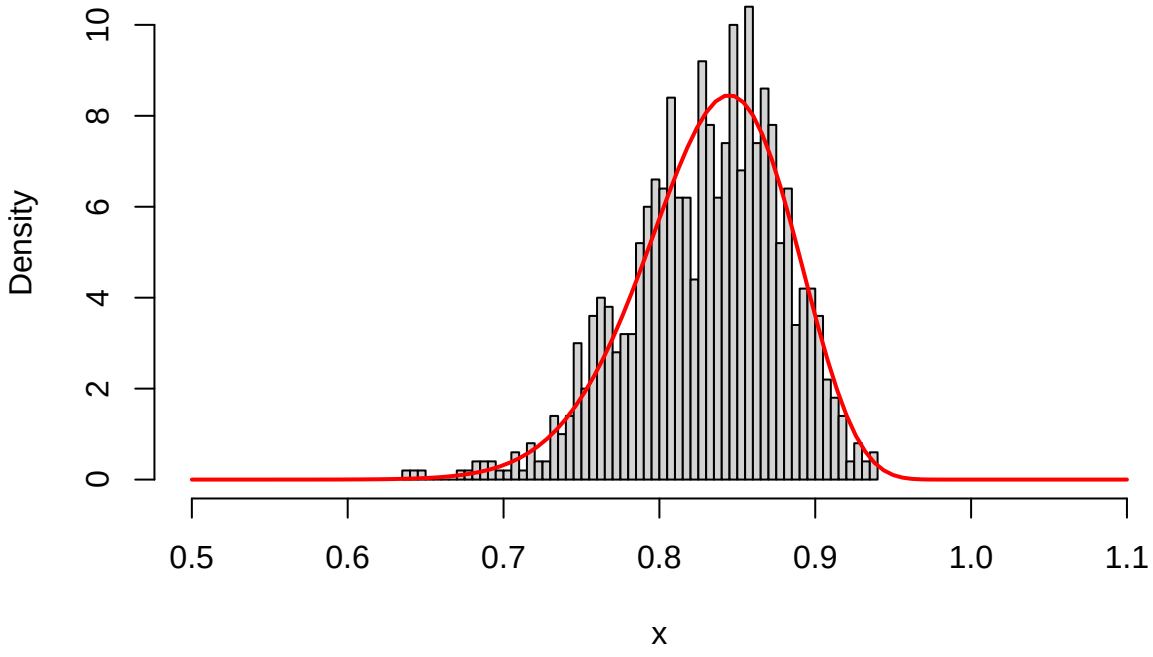
This concludes the proof that $Z = \frac{X}{X+Y} \sim Beta(\alpha, \beta)$ if $X \sim Gamma(\alpha, 1)$ and $Y \sim Gamma(\beta, 1)$ independently.

It is of interest to sample from a beta distribution. Below, we implement a way of sampling from a beta distribution using the existing methods for sampling from the gamma distribution and the relation proved above.

```
sample_beta_distribution <- function(n, alpha, beta){
  x = sample_Gamma(n, alpha, 1)
  y = sample_Gamma(n, beta, 1)
  return(x/(x+y))
}
```

We also produce a fitness plot as shown in Figure B.6. The samples seem to coincide well with some deviations around the peak of the distribution.

Figure B.6: Histogram of sampled values versus theoretical density, n = 1000, alpha = 50, beta = 10



Problem C

Monte Carlo integration and variance reduction

For this problem, we wish to estimate $\theta = \text{Prob}(X \geq 4)$ where $X \sim \mathcal{N}(0,1)$. For our first estimate, we will just use Monte Carlo integration. Our Monte Carlo estimate for θ then becomes $\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} I(x_i > 4) \approx P(X > 4)$, where $I(x > 4) = 1$ if $x_i > 4$ and 0 otherwise, and the x_i's are n independent samples from a standard normal distribution. We implement this as shown below.

```
set.seed(96)
estimate_theta <- function(n){
  #Generating n samples from standard normal distribution:
  y = sample_From_Normal_Distribution(n)

  #Estimate theta using Monte Carlo integration.
  #    Also return all the samples from N(0,1)
  return(list(theta_hat = (1/n)*sum(y>4), normal_distr_samples = y))
}

n = 100000 #Number of samples desired
result = estimate_theta(n) #Estimate theta

theta_theoretical = 1-pnorm(4,0,1) #Calculate the theoretical value of theta
```

From the code above, we get that $\hat{\theta} = 4 \times 10^{-5}$, meaning that only four samples from x were greater than 4. The theoretical value is 3.1671242×10^{-5}, which gives a relative error of 0.208219. This shows that our Monte Carlo estimate is decent, but that it is quite rare for samples from standard normal distribution to be larger than 4, thus we would need quite a large number of samples to get a good estimate, or as we are going to do, would need to adjust our estimation method.

Now, we calculate a 95% confidence interval. We use here that whether or not $X > 4$ will behave as a Bernoulli process. Since n is quite large, following the central limit theorem, this will approximate to a normal distribution. Thus we get that the 95% confidence interval is given by $\hat{\theta} \pm z_{\alpha/2}\sqrt{\frac{S^2}{n}}$, where S^2 is the empirical variance and $z_{\alpha/2}$ is the critical value in the standard normal distribution for $\alpha/2$. This is implemented below.

```
empirical_variance <- function(x, x_hat, n){
  #Function that calculates the empirical variance
  return(1/(n-1)*sum((x-x_hat)^2))
}

#Calculate the 95% confidence interval
empirical_Variance_theta = empirical_variance((result$normal_distr_samples),
                                              result$theta_hat, n)
CI_theta = c(result$theta_hat-1.96*sqrt(empirical_Variance_theta/n),
             result$theta_hat+1.96*sqrt(empirical_Variance_theta/n))
```

This results in a empirical variance of 1.0056959 and the following confidence interval $[-0.0061757, 0.0062557]$. We can see that the actual value of theta is within the confidence interval, which is good. However, due to a relatively large variance, the confidence interval is rather large.

Importance sampling

As mentioned above, we wish to adjust our sampling to obtain a better estimate. For this we will use importance sampling via

$$g(x) = \begin{cases} cx \cdot e^{-\frac{1}{2}x^2}, & x > 4 \\ 0, & \text{Otherwise} \end{cases}$$

where c is normalizing constant. One can use inversion to sample from $g(x)$, and thus, we must find the CDF $G(x)$ and then its inverse.

$$G(x) = \int_{-\infty}^{x} g(z)dz = \int_{4}^{x} cze^{-\frac{1}{2}z^2}dz = \int_{16}^{x^2} \frac{c}{2}e^{-\frac{1}{2}u}du = \left[-ce^{-\frac{1}{2}u}\right]_{16}^{x^2} = \left[-ce^{-\frac{1}{2}x^2}\right]_{4}^{x} = c(e^{-8} - e^{-\frac{1}{2}x^2}).$$

We now find c by setting $\int_{-\infty}^{\infty} g(z)dz = 1$. From above we have that

$$\int_{-\infty}^{\infty} g(z)dz = c \cdot (e^{-8} - \lim_{x\to\infty} e^{-\frac{1}{2}x^2}) = c \cdot e^{-8} \implies c = e^{8}.$$

Now we calculate G^{-1}.

$$U = c(e^{-8} - e^{-x^2/2})$$
$$e^{-x^2/2} = e^{-8} - U/c$$
$$-\frac{x^2}{2} = \ln(e^{-8} - U/c).$$

Hence, we get that

$$G^{-1}(U) = \sqrt{-2\ln(e^{-8}(1-U))}.$$

Our importance sampling estimator is given by $\frac{1}{N}\sum h(x_i)\frac{f(x_i)}{g(x_i)}$, where f is the pdf of the standard normal distribution, $h = I(x_i > 4)$, the x_i's are n independent samples from $g(x)$, and g is as given above. Thus, our estimator becomes

$$\hat{\theta}_{IS} = \frac{1}{n}\sum_{i=1}^{n} \frac{I(x_i > 4)}{\sqrt{2\pi}e^{8}x_i}$$

This is implemented below.

```
G_inverse <- function(u){
  return(sqrt(-2*log(exp(-8)-u/exp(8)))) #Returns the inverse of G
}


IS_estimator <- function(x, n){
  1/n*sum((x>4)*1/(sqrt(2*pi)*exp(8)*x))
}

set.seed(5001)
estimate_theta_IS <- function(n){
  #Function that utilizes importance sampling to estimate theta
  u = runif(n)
  x = G_inverse(u) #Use inversion method
  theta_hat_IS = IS_estimator(x, n) #Importance sampling estimator of theta

  return(list(theta_estimate_IS = theta_hat_IS , g_samples = x))
}


result = estimate_theta_IS(n)
```

From the above implementation, we get that our importance sampling estimator for θ is 3.1659865×10^{-5} and that the relative error is $-3.5933626 \times 10^{-4}$. The improvement in the estimator is due to importance sampling weighting certain areas more. Now, we calculate the 95% confidence interval the same way we did for our Monte Carlo integration estimate.

```
#Calculate empirical variance
empirical_variance_2 = empirical_variance(IS_estimator(result$g_samples,1),
                                          result$theta_estimate_IS, n)
#Calculate 95% CI
CI_theta_2 = c(result$theta_estimate_IS-1.96*sqrt(empirical_variance_2/n),
               result$theta_estimate_IS+1.96*sqrt(empirical_variance_2/n))
```

From above we get that the confidence interval is $[-3.039316 \times 10^{-5}, 9.3712891 \times 10^{-5}]$, and we have an empirical variance of 1.002337×10^{-4}. Comparing these values to the ones from Monte Carlo integration, we see that the empirical variance is reduced considerably and that the confidence interval is more precise in accordance with importance sampling being a variance reduction method.

To find out how many more samples our Monte Carlo estimator would need in order to be equally as precise as our importance sampling estimator, we look at the expression for the confidence intervals. We have that $CI = [\hat{\theta} - z_{\alpha/2} \cdot \sqrt{S^2/n}, \hat{\theta} + z_{\alpha/2} \cdot \sqrt{S^2/n}]$, where $S^2 = \frac{1}{n}\sum(x - \bar{x})^2$. This means that the confidence interval shrinks with a factor of $1/n$. So, since the width of our Monte Carlo estimators confidence interval is equal to 0.0124314, while the width of our Importance sampling estimators confidence interval is 1.2410605×10^{-4}, we get that we need a factor of 100.1674092 more samples for our Monte Carlo estimate to be equally precise. This means that we need approximately $n = 10^7$ samples to get an equally precise estimate for our Monte Carlo estimation.

Antithetic values

Now we will use that $u \sim \mathcal{U}[0, 1]$, which means that their antithetic value $1 - u$ also follows $\mathcal{U}[0, 1]$. Therefore, we will sample n random numbers u from $\mathcal{U}[0, 1]$ and calculate their antithetic value. We will then use u and $1 - u$ to estimate θ, using importance sampling.

```
set.seed(5003)
estimate_theta_IS_antithetic <- function(n){
  # Function that utilizes importance sampling to estimate theta
  u = runif(n)
  # Generate the antithetic values of u, and make a
  #    vector containing these values and u
  u = c(u, 1-u)

  #Use inversion method to get 2n samples from distribution g(x):
  x = G_inverse(u)
  theta_hat_IS_antithetic = IS_estimator(x, 2*n) #Use importance sampling

  return(list(theta_hat_IS_antithetic = theta_hat_IS_antithetic, samples_g = x))
}


n = 50000
result = estimate_theta_IS_antithetic(n)
```

From the above code we get that our estimate for θ is 3.1666379×10^{-5}, and this estimate has a relative error of $-1.5356115 \times 10^{-4}$, which is quite similar to that of our estimate using regular importance sampling.

Now we calculate a 95% confidence interval as before.

```r
#Calculate the empirical variance
empirical_variance_3 = empirical_variance(IS_estimator(result$samples_g, 1),
                                          result$theta_hat_IS_antithetic, 2*n)

#Calculate 95% CI
CI_3 = c(result$theta_hat_IS_antithetic - 1.96*sqrt(empirical_variance_3/(2*n)),
         result$theta_hat_IS_antithetic + 1.96*sqrt(empirical_variance_3/(2*n)))
```

This yields the confidence interval $[-3.0399414 \times 10^{-5}, 9.3732172 \times 10^{-5}]$. This is more or less equally precise as that of our confidence interval using regular importance sampling. Here, we have to use $n = 50000$ because we double the amount of samples from the uniform distribution when adding the antithetic values.

Problem D

Rejection sampling and importance sampling

We wish to examine the multinomial mass function and its posterior density, given by

$$f(\theta|\boldsymbol{y}) \propto (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4}, \quad \boldsymbol{y} = [y_1, y_2, y_3, y_4]^T$$

We wish to be able to sample from $f(\theta|\boldsymbol{y})$ only using the uniform $\mathcal{U}(0,1)$ density.

This can be done using rejection sampling by using $\mathcal{U}(0,1) := g(\theta)$ as the proposal density. We must then determine c expressed by

$$\frac{f(\theta|\boldsymbol{y})}{g(\theta)} \leq c, \; c > 1 \;\; \forall \theta \text{ st. } f(\theta|\boldsymbol{y}) > 0,$$

i.e. finding the roots of $\frac{\partial f(\theta|\boldsymbol{y})}{\partial \theta}$:

$$\frac{\partial f(\theta|\boldsymbol{y})}{\partial \theta} = y_1(2+\theta)^{y_1-1}(1-\theta)^{y_2+y_3}\theta^{y_4} + (y_2+y_3)(1-\theta)^{y_2+y_3-1}(-1)(2+\theta)^{y_1}\theta^{y_4} + y_4\theta^{y_4-1}(2+\theta)^{y_1}(1-\theta)^{y_2+y_3}$$

$$= \theta^{33}(2+\theta)^{124}(1-\theta)^{37}(125(1-\theta) - 38\theta(2+\theta) + 34(2+\theta)(1-\theta))$$

Studying the terms of the factorized expression, only the last factor can have roots given the constraint $0 < \theta < 1$. Thus,

$$\begin{aligned} &\implies 125(1-\theta) - 38\theta(2+\theta) + 34(2+\theta)(1-\theta) \\ &= \theta^2(-125-38-34) + \theta(125-76-34) + 68 = 0 \\ &= -197\,\theta^2 + 15\theta + 68 = 0 \end{aligned}$$

Ignoring the negative root since $\theta > 1$, we get with the standard quadratic formula

$$\theta^* = \frac{\sqrt{53809}+15}{394} \approx 0.63 \iff c := f(\theta^*|\boldsymbol{y})$$

We now implement the rejection sampling to sample from $f(\theta|\boldsymbol{y})$ below.

```r
sample_multinomial <- function(n, y, theta_star){
  # Rejection sampling algorithm for multinomial distribution
  # n          : number of samples
  # y          : (y_1, y_2, y_3, y_4)^T, vector of size 4
  # theta_star : local maximum in distribution
  # -----------------------------------------------------------

  # Start by initializing lists and constants
  c = density_multinomial( y, theta_star ) # Upper bound described above
```

```
  # Container initialization
  sampled <- c()
  count <- 0

  # Do rejection sampling until we have wnated num. of points (n)
  while(length(sampled) < n){
    u <- runif(n - length(sampled)) # Draw uniform distributions
    alpha <- density_multinomial(y, u)/c # Acceptance probability
    samples_label <- ifelse(runif(length(u)) <= alpha, u, NA)
    # We now place the accepted values in sampled and count (accepted+rejected)
    #    in a count-variable
    sampled <- append(sampled, samples_label[!is.na(samples_label)])
    count <- count + length(u)
  }
  return(list(
    sampled = sampled,
    count = count
  ))
}
density_multinomial <- function(y, theta){
  # Unscaled density f(theta | y)
  # y     : vector of (y_1, y_2, y_3, y_4)^T
  # theta : theta, must be 0 < theta < 1
  return( (2+theta)^y[1] * (1-theta)^(y[2]+y[3]) * theta^y[4] )
}
# Globally defined variables
y_vector <- c(125, 18, 20, 34)
theta_star <- ( sqrt(53809)+15)/394
```

We allow ourselves to find the normalization constant of $f(\theta|\boldsymbol{y})$ numerically. We have

$$\int_{-\infty}^{\infty} kf(\theta|\boldsymbol{y})d\theta = \int_{0}^{1} kf(\theta|\boldsymbol{y})d\theta = 1 \iff k = \frac{1}{\int_0^1 kf(\theta|\boldsymbol{y})d\theta}$$

```
norm_const <- 1 / integrate(density_multinomial, 0, 1, y=y_vector)$value
norm_const
```

```
## [1] 4.24143e-29
```

Posterior mean and fit of samples

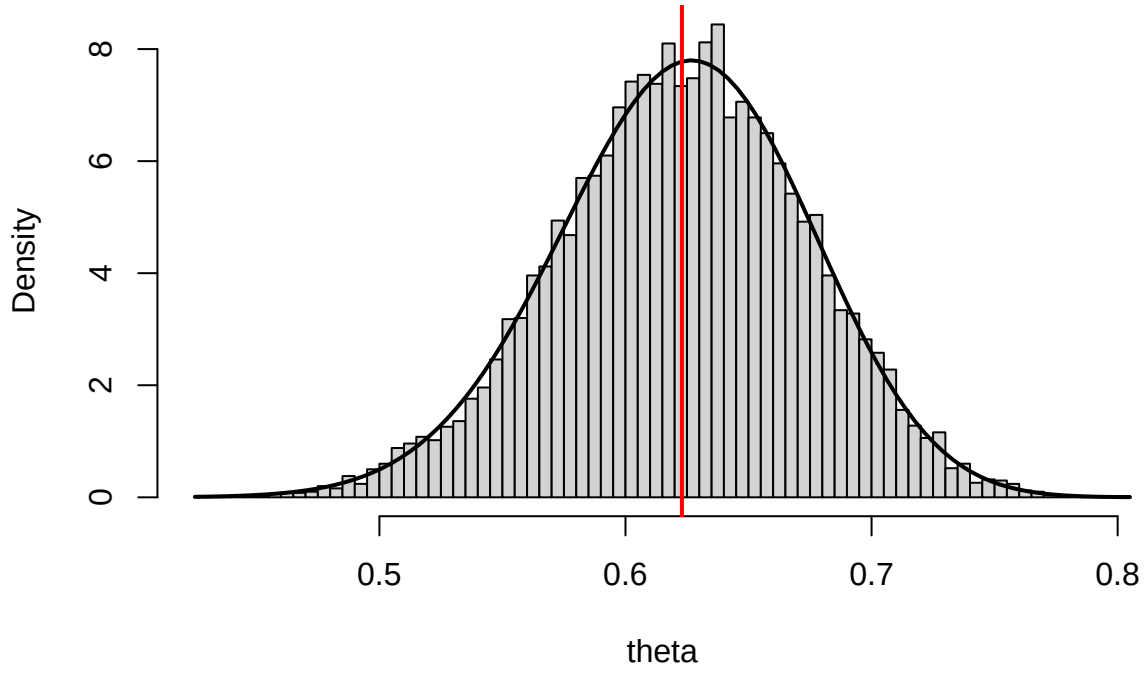
Before we determine the fit of the sampling algorithm, we will try to approximate the posterior mean $E(\theta|\boldsymbol{y})$ using Monte Carlo integration. Since we want to find the posterior mean, we have that $h(x) = x$, such that our Monte Carlo estimator becomes $\widehat{\theta|\boldsymbol{y}} = \frac{1}{n}\sum\theta_i$ where θ_i are the sampled values for $i = 1, ..., n$.

```
monte_carlo_multinomial <- function(samples){
  # Monte Carlo algorithm for E[f(theta | y)]
  N = length(samples)
  return(
    sum(samples)/
      length(samples))
}

# Define some necessary consts
```

```r
nsamples <- 10000 # number of samples
theta <- seq(0.01,0.99, length=100)
# Draw samples
res <- sample_multinomial(nsamples, y_vector, theta_star)
real_values <- density_multinomial(y_vector, theta) * norm_const
# Calculate mean
approx_mean <- monte_carlo_multinomial(res$sampled)
```

We can now determine fit of samples with a histogram, which is shown in Figure D.1. In Figure D.1 we see that the sampled multinomial distribution closely follows the theoretical distribution. The red vertical line represents the posterior mean, and as we can see, it is quite close to the theoretical mean.

Figure D.1: Sampled versus theoretical values for f(theta | y), n = 10000



We see that the estimated mean is 0.6229375. We find the theoretical mean by integrating the normalized $\theta f(\theta|\boldsymbol{y})$.

```r
theoretical_mean <- integrate(
  # Note that we don't integrate f(theta|y) but k*theta f(theta|y)
  #    where k is normalization constant
  (function(y, theta) theta*norm_const*density_multinomial(y, theta)),
  lower=0, upper=1, y=y_vector)$value
```

Thus, the theoretical mean is 0.6228061 and the relative error is 0.0210966 %, which is indeed very small.

Efficiency of rejection sampling algorithm

The rejection sampling is implemented such that it returns both the samples, as well as a count of number of generated $\mathcal{U}(0,1)$, i.e. both accepted and rejected values. Thus, on average, for each `sample` you need `count/length(sample)` random numbers, resulting in

```
res$count/length(res$sampled)
```

```
## [1] 7.7333
```

This shows that you need on average 7-8 random numbers to generate one sample.

Posterior mean with alternative prior

We wish to examine how the posterior mean would change if we had used a different prior than the original Beta(1,1) prior. We find the alternative density expression using a Beta(1,5) prior and find the new posterior mean using importance sampling weights, using the same sampels as generated in the section *Posterior mean and fit of samples*. The implementation is shown below.

```
alt_density_multinomial <- function(y, theta){
  # Alternative multinomial density
  # input: y-vector and theta
  # output: (pdf of new beta distr) * (multinomial density)
  return(dbeta(theta, 1, 5) * density_multinomial(y, theta))
}

# We have to normalize the new density like previously
alt_norm_const <- 1/(integrate(alt_density_multinomial,0,1, y=y_vector)$value)


IS_estimate <- function(samples){
  # Importance sampling estimate
  IS_weights <- (
    alt_norm_const * alt_density_multinomial(y_vector, samples)/(
    norm_const     * density_multinomial(y_vector, samples))
  )
  N <- length(samples)

  return(1/N * sum(samples * IS_weights))
}
```

We can now apply this method to our sampled values from before. In addition, we solve the integral $\int_0^1 \theta f_{new}(\theta|y)$ as a theoretical value for the posterior mean.

```
# Create a list of sampled estimate ($sampled) and a 'theoretical' with R's
#    `integrate'-function
list(sampled=IS_estimate(res$sampled),
     theoretical=integrate(
       (function(y, theta) theta*alt_norm_const*(
         alt_density_multinomial(y, theta)
         )),
       lower=0, upper=1, y=y_vector)$value
     )
```

```
## $sampled
## [1] 0.5950037
##
## $theoretical
## [1] 0.5959316
```

We see that the relative error is 0.252143 %, which is considerably low. Note also that this posterior mean of about 0.59 is smaller than the posterior mean when using the Beta$(1,1)$ distribution as a prior.