Project 2

Computer Intensive Statistical Methods

Erling og Christian

03 02 2022

Contents

Problem A: Stochastic simulation by the probabilty integral transform and bivariate techniques **1**

1. 1

2 2

b 4

3 5

4. 7

5. 8

Problem B: The gamma distribution **9**

1. 9

a) 9

2.) 11

(a) 11

3. 14

4. 18

a) 18

(b) 19

Problem C: Monte Carlo integration and variance reduction **20**

1 Monte Carlo integration 20

2 Importance sampling 22

3 Antithetic Sampling 24

Problem D: Rejection sampling and importance sampling **25**

2. 26

3. 27

4. Beta$(1, 5)$ as prior. 28

Problem A: Stochastic simulation by the probabilty integral transform and bivariate techniques

1.

We are going to generate samples from an exponential distribution with rate parameter λ, and the number of samples is n. Let $X \sim Exp(\lambda)$. This gives pdf and cdf

$$f(x) = \lambda \exp(-\lambda x)$$

$$F(x) = 1 - \exp(-\lambda x)$$

The inversion method can be used for generate samples from the exponential distribution. First random variable U is generated from the standard uniform distribution in interval $[0, 1]$. Then $X = F^{-1}(U)$. The algorithm is then

$$u \sim U[0, 1]$$

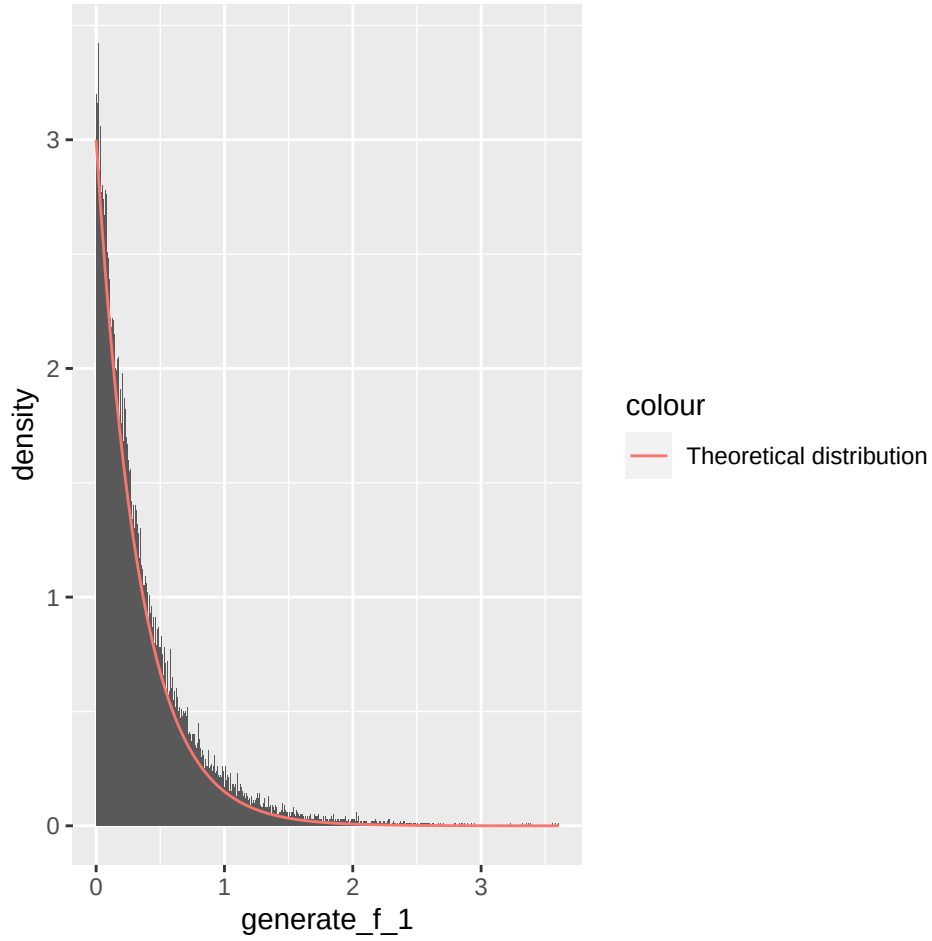
$$x = -\frac{1}{\lambda} \log(u)$$

return x

```
generate_exponential <- function(n, lambda) {
    u <- runif(n)
    x <- -(1/lambda) * log(u)
    return(x)
}
```

Below the function is used with $n = 10000$ and $\lambda = 3$. The result is plotted against the theoretical distribution.

```
library(ggplot2)

theoretical_f_1 <-function(x,lambda)
{
  return(lambda*exp(-lambda*x))
}
set.seed(2)
n=100000
lambda=3
generate_f_1=generate_exponential(n, lambda)


ggplot()+
  geom_histogram(
    data=as.data.frame(generate_f_1),
    mapping=aes(x=generate_f_1,y=..density..),
    binwidth=0.001

  )+
  stat_function(
  fun=theoretical_f_1,
  args=list(lambda=lambda),
  aes(col="Theoretical distribution")
)
```



2

a)

We want to find the cumulative distribution function and the inverse of the cumulative distribution function when the probability density function is

$$g(x) =$$

The cumulative distribution $G(x)$ is given by

$$G(x) = \int_{-\infty}^{x} g(t)dt$$

Thus, for $0 < x < 1$, the cdf becomes

$$G(x) = \int_{0}^{x} ct^{\alpha-1} = c \cdot \left[\frac{1}{\alpha}t^{\alpha}\right]_{0}^{x} = \frac{c}{\alpha}x^{\alpha}.$$

For $1 \leq x$, the cdf is given by

$$G(x) = \int_{0}^{1} ct^{\alpha-1} + \int_{1}^{x} ce^{-t}dt = c \cdot \left[\frac{1}{\alpha}t^{\alpha}\right]_{0}^{1} + c \cdot \left[-e^{-t}\right]_{1}^{x} = \frac{c}{\alpha} - ce^{-x} + e^{-1} = c \cdot \left(\frac{1}{\alpha} - e^{-x} + \frac{1}{e}\right)$$

The constant c can be found by solving the following equation for c,

$$\int_0^1 ct^{\alpha-1} + \int_1^\infty ce^{-t}dt = 1$$
$$\frac{c}{\alpha} + c \cdot \left[-e^{-t}\right]_1^\infty = \frac{c}{\alpha} + \frac{c}{e}$$
$$\frac{c}{\alpha} + \frac{c}{e} = 1 \implies c = \frac{\alpha e}{e+\alpha}.$$

The inverse of this cumulative distribution function can be found by solving the following equation for x,

$$y = G(x).$$

For $0 < x < 1$, we have

$$y = \frac{ex^\alpha}{e+\alpha} \implies y(e+\alpha) = ex^\alpha \implies x = \left(\frac{u(e+\alpha)}{e}\right)^{\frac{1}{\alpha}}$$

Thus, the inverse of the cumulative distribution function is

$$G^{-1}(y) = \left(\frac{y(e+\alpha)}{e}\right)^{\frac{1}{\alpha}}$$

for $0 < G^{-1}(y) < 1 \implies 0 < y < \frac{e}{e+\alpha}$. For $1 \leq x$, the following eqaution is solved for x

$$y = 1 - \frac{\alpha e^{-x+1}}{e+\alpha} \implies x = 1 - \ln\left(\frac{(1-y)(e+\alpha)}{\alpha}\right) = \ln\left(\frac{\alpha e}{(1-y)(e+\alpha)}\right)$$
$$\implies G^{-1}(y) = \ln\left(\frac{\alpha e}{(1-y)(e+\alpha)}\right)$$

For $x = 1$, we have

$$1 = \ln\left(\frac{e\alpha}{(1-y)(e+\alpha)}\right) \implies e = \frac{e\alpha}{(1-y)(e+\alpha)} \implies y = 1 - \frac{\alpha}{\alpha+e}$$

When $x = \infty$, $y = 1$. Therefore the inverse cumulative function is

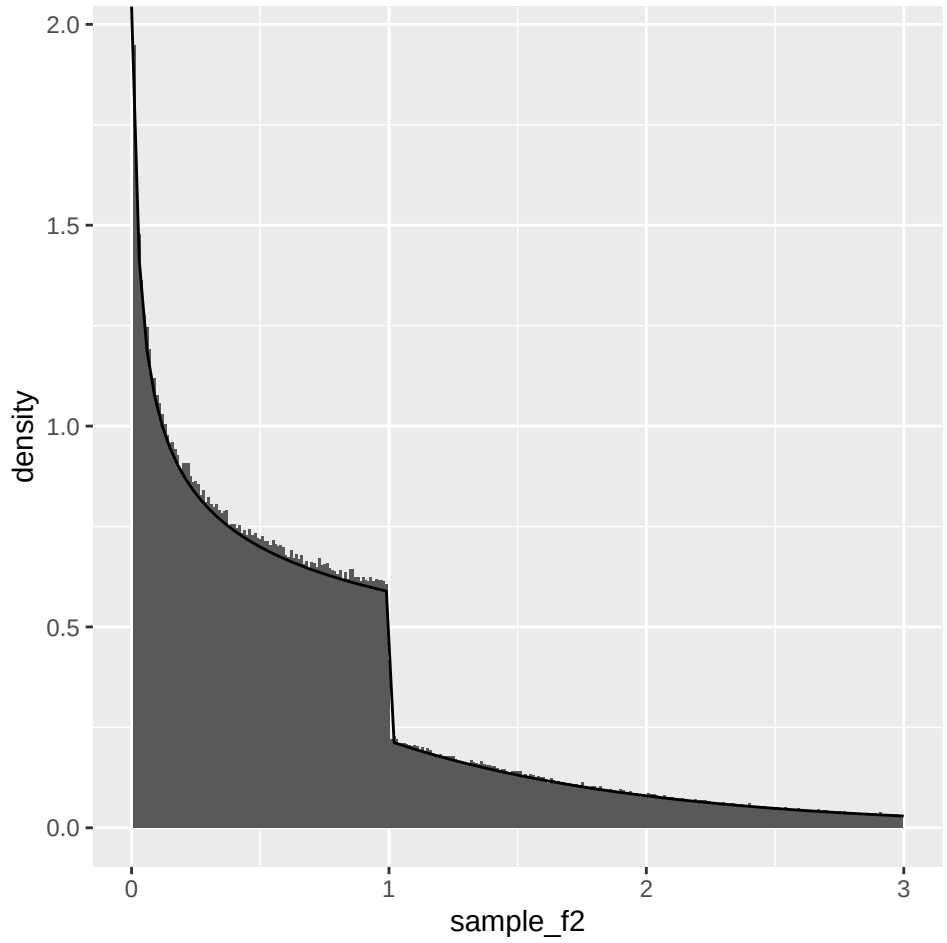
b

The inversion method is used to generate samples from g.

```
sample_g = function(n, alpha) {
    c = (alpha * exp(1))/(alpha + exp(1))
    u = runif(n)
    samples = vector(length = n)
    samples[u < c/alpha] = (alpha/c * u[u < c/alpha])^(1/alpha)

    samples[u >= c/alpha] = log(c/(1 - u[u >= c/alpha]))
    return(samples)
}
```

The result for $\alpha = 2$ and $n = 10000$ is plotted against the theoretical distribution.

```
  density_g = function(x, alpha) {
    # Normalizing constant c:
    c = alpha*exp(1)/(alpha + exp(1))

    # Create an empty vector of same length as x:
    density = vector(length = length(x))

    # All elements corresponding to x < 1:
    density[x < 1.] = c*x[x<1.]^(alpha-1)

    # All elements corresponding to x >= 1:
    density[x >= 1.] = c*exp(-x[x>=1.])
    return(as.double(density))
  }


n=1000000
alpha=0.75
sample_f2<-sample_g(n,alpha)
ggplot()+
  geom_histogram(
    data=as.data.frame(sample_f2),
    mapping=aes(x=sample_f2,y=..density..),
    binwidth=0.01

  )+
  stat_function(fun=density_g,args=list(alpha=alpha))+
  xlim(0,3)
```



3

We consider the probability density function

$$f(x) = \frac{ce^{\alpha x}}{(1+e^{\alpha x})^2}, \quad \infty < x < \infty, \quad \alpha > 0$$

a) To find the normalizing constant, we consider the integral I of the pdf over R

$$I = \int_{-\infty}^{\infty} f(x)dx = 1 \implies \int_{-\infty}^{\infty} \frac{ce^{\alpha x}}{(1+e^{\alpha x})^2}dx = 1$$

Let $u = 1 + e^{\alpha x}$. This means that $\frac{du}{dx} = \alpha e^{\alpha x}$ and $u(-\infty) = 1$ and $u(\infty) = \infty$. By using variable change the integral becomes

$$I = \int_1^{\infty} \frac{c}{\alpha} u^{-2} du = \frac{c}{\alpha} \left[-u^{-1}\right]_1^{\infty} = \frac{c}{\alpha}$$

$$I = 1 \implies \frac{c}{\alpha} = 1 \implies c = \alpha$$

The pdf is therefore

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

The cumulative distribution function is given by

$$F(x) = \int_{-\infty}^{x} \frac{\alpha e^{\alpha t}}{(1+e^{\alpha t})^2} dt.$$

By using $u = 1 + e^{\alpha t}$, we get

$$F(x) = \int_{-\infty}^{1+e^{\alpha x}} u^{-2} du = \left[-u^{-1}\right]_{1}^{1+e^{\alpha x}} = \frac{-1}{1+e^{\alpha x}} + 1 = \frac{e^{\alpha x}}{1+e^{\alpha x}}$$

The inverse cumulative distribution is found by solving $y = F(x)$ for x.

$$y = \frac{e^{\alpha x}}{1+e^{\alpha x}} \implies e^{\alpha x} = \frac{y}{1-y} \implies x = \frac{1}{\alpha} \ln\left(\frac{y}{1-y}\right)$$

This means that the inverse cumulative distribution function is

$$F^{-1}(y) = \frac{1}{\alpha} \ln\left(\frac{y}{1-y}\right)$$

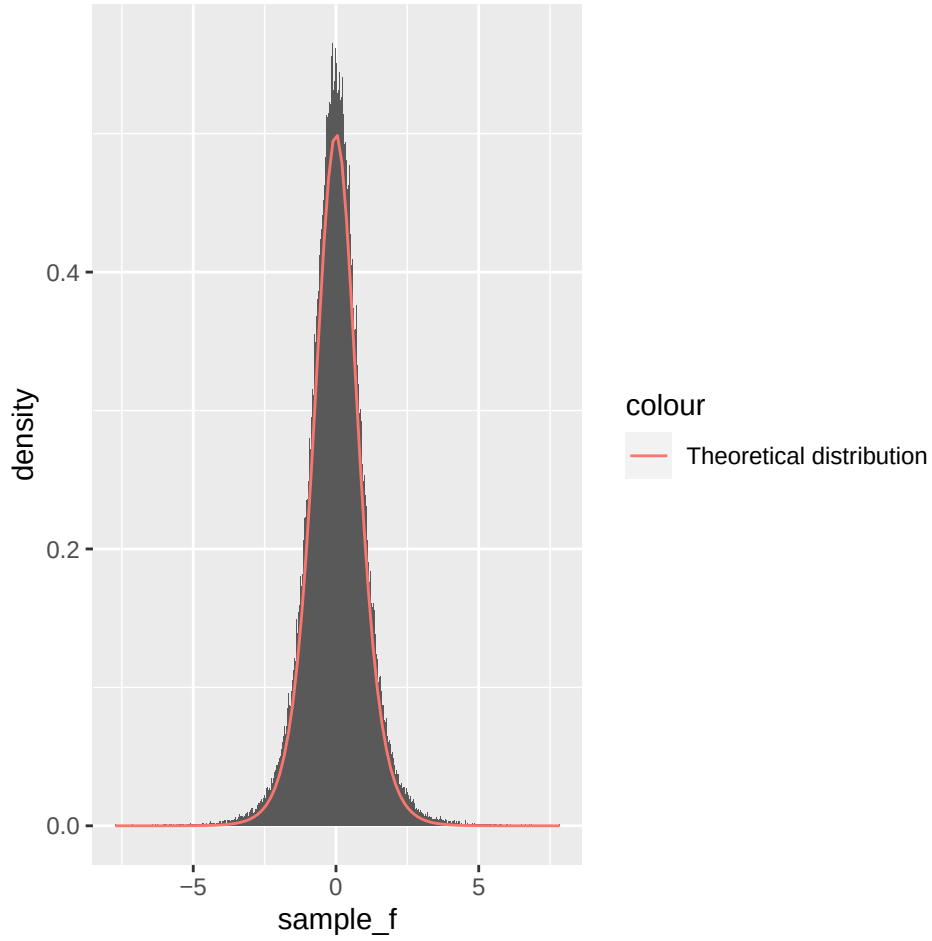
c) In the following chunk there is code for a function generating samples form f by using the inversion method

```
generate_f <- function(n, alpha) {
    u <- runif(n)
    x <- (1/alpha) * log(u/(1 - u))
    return(x)
}
```

To check that the function works properly, an example with using the function $\alpha = 2$ and $n = 1000000$ is plotted against the theoretical distribution.

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

```
library(ggplot2)
n = 1e+06
alpha = 2
sample_f <- generate_f(n, alpha)
ggplot() + geom_histogram(data = as.data.frame(sample_f), mapping = aes(x = sample_f,
    y = ..density..), binwidth = 0.001, ) + stat_function(fun = theoretical_f, args = list(alpha = alpha
    aes(col = "Theoretical distribution"))
```



4.

We use the Box-Muller algorithm to represent independent variables which are standard normal distributed. Let $X \sim N(0,1)$ and $Y \sim N(0,1)$ be independent. The joint distribution of these two variables is

$$f(x,y) = f_X(x) \cdot f_Y(y) = \frac{1}{2\pi} e^{-\frac{x^2+y^2}{2}}$$

By using polar coordinates where $x^2 + y^2 = r^2$, the joint distribution becomes

$$f(r) = \frac{1}{2\pi} e^{-\frac{r^2}{2}}.$$

This is a joint distribution of $r^2 \sim \exp(1/2)$ and $X_1 \sim \text{Unif}(0, 2\pi)$. This means that

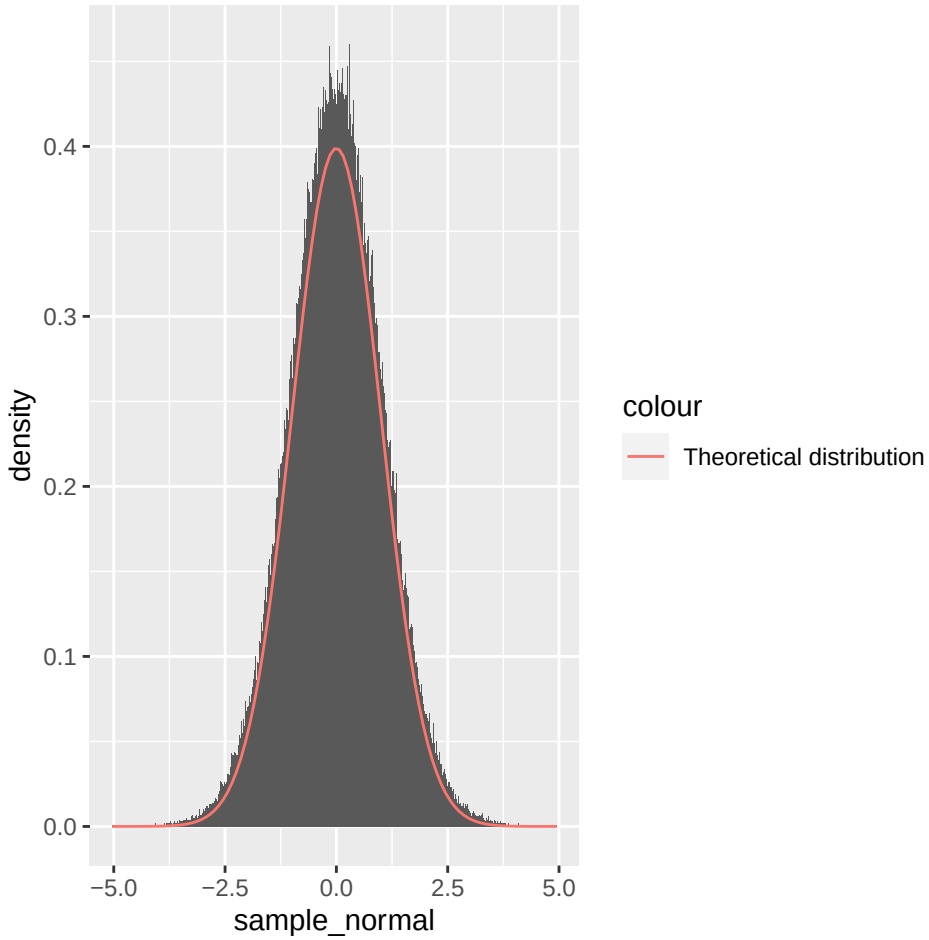
$$X = r\cos(X_1)$$

$$Y = r\sin(X_1)$$

are normal distributed. $r \sim \sqrt{-2\log(\text{Unif}(0,1))}$ and $X_2 \sim 2\pi\text{Unif}(0,1)$

In the following chunk, the Box-Muller algorithm is implemented. We first draw two samples from $\text{Unif}(0,1)$. We then calculate r and X_1 and at last return $X = r\cos(X_1)$ which is standard normal distributed.

```r
generate_from_normal <- function(n) {
    u1 <- runif(n)
    u2 <- runif(n)
    r <- sqrt(-2 * log(u1))
    x_1 <- 2 * pi * u2
    x = r * cos(x_1)
    return(x)
}
n = 1e+06
sample_normal <- generate_from_normal(n)
ggplot() + geom_histogram(data = as.data.frame(sample_normal), mapping = aes(x = sample_normal,
    y = ..density..), binwidth = 0.001) + stat_function(fun = dnorm, args = list(mean = 0,
    sd = 1), aes(col = "Theoretical distribution"))
```



5.

We want to to simulate from a d-variate normal distribution with mean vector μ and covariance matrix Σ. . Let $x \sim \text{Normal}(0, I_d)$, where I_d is the identity matrix. Then

$$y = \mu + DZ \sim Normal(\mu, DD^T)$$

Thus, we have to find D such that $\Sigma = DD^T$. The Cholesky decomposition can be used to find D.

```
generate_d_normal <- function(n, mu, cov, d) {
    # D is the cholesky decomposition of the covariance matrix
    D <- t(chol(cov))
    x <- generate_from_normal(n)
    y <- mu + D %*% x
    return(y)
}
```

An example of this a covariance matrix is

$$\begin{pmatrix} 1 & 3 & 5 \\ 3 & 2 & 2 \\ 5 & 2 & 3 \end{pmatrix}$$

To test whether the function works, we use $\mu = [1, 7, 2]^T$ and Σ

```
n <- 10000
mu <- c(3, 4, 5)
cov_mat <- cbind(c(2, -1, 0), c(-1, 2, -1), c(0, -1, 2))
cov_mat
sample_normal_d <- generate_d_normal(n, mu, cov_mat, 3)
```

Problem B: The gamma distribution

1.

The gamma distribution with parameters $\alpha \in (0, 1)$ and $\beta = 1$ has probability density function

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

a)

The acceptance probability γ in the rejection sampling algorithm is given by

$$\gamma = c^{-1} \cdot \frac{f(x)}{g(x)}$$

where $g(x)$ is the proposal density. We use the density in problem A.2.

For $0 < x < 1$,

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

This means that the acceptance probability is

$$\gamma = c^{-1} \frac{1}{\Gamma(\alpha)} x^{\alpha-1} e^{-x} \cdot \frac{\alpha + e}{\alpha e \cdot x^{\alpha-1}} = \frac{e^{-x}(\alpha + e)}{c \alpha e \Gamma(\alpha)}$$

For $x \geq 1$, we have

$$g(x) = \frac{\alpha e}{\alpha + e} e^{-x}$$

which give the acceptance probability

$$\gamma = \frac{x^{\alpha-1}(\alpha + e)}{c\Gamma(\alpha)\alpha e}$$

To sample from f, we need to find an efficient bound c such that

$$\frac{f(x)}{g(x)} \leq c, \forall x$$

We need to choose the smallest possible value for c.

For $0 < x < 1$

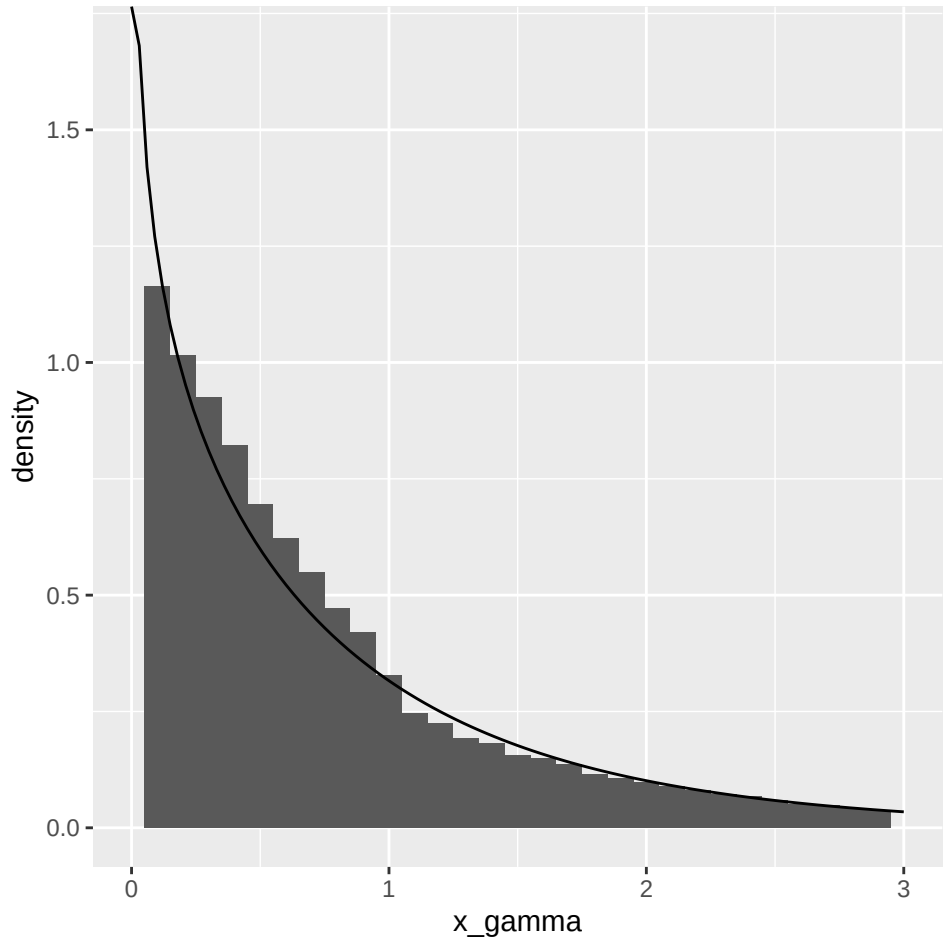
$$c \geq \frac{(\alpha + e)}{\alpha e \Gamma(\alpha)} \geq \frac{e^{-x}(\alpha + e)}{\alpha e \Gamma(\alpha)} = \frac{f(x)}{g(x)}$$

For $x \geq 1$

$$c \geq \frac{(\alpha + e)}{\alpha e \Gamma(\alpha)} \geq \frac{x^{\alpha-1}(\alpha + e)}{\Gamma(\alpha)\alpha e} = \frac{f(x)}{g(x)}.$$

We can therefore choose $c = \frac{(\alpha+e)}{\alpha e \Gamma(\alpha)}$. (b) The rejection sampling algorithm is used to generate a vector of n independent samples from f.

```
sample_gamma <- function(n, alpha) {
    x <- vector(mode = "numeric", length = n)
    c <- (alpha + exp(1))/(alpha + exp(1))
    for (i in 1:n) {
        finished = 0
        while (finished == 0) {
            xi <- sample_g(1, alpha)
            u <- runif(1)
            f <- dgamma(xi, alpha)
            g <- density_g(xi, alpha)
            gamma <- (1/c) * (f/g)
            if (u <= gamma) {
                x[i] = xi
                finished = 1
            }
        }
    }
    return(x)
}

x_gamma <- sample_gamma(1e+05, 0.8)
ggplot() + geom_histogram(data = as.data.frame(x_gamma), mapping = aes(x = x_gamma,
    y = ..density..), binwidth = 0.1) + stat_function(fun = dgamma, args = list(shape = 0.8)) +
    xlim(0, 3)
```



2.)

(a)

$a = \sqrt{\sup_x f^*(x)}$, where

$$f(x) = \begin{cases} x^{\alpha-1}e^{-x} & \text{if } 0 < x, \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

We start to find $\sup_x f^*(x)$. If $x < 0$,

$$\frac{\partial}{\partial x} x^{\alpha-1}e^{-x} = (\alpha-1)x^{\alpha-2}e^{-x} - x^{\alpha-1}e^{-x}$$

Put this equation equal to 0 and we get

$$(\alpha-1)x^{\alpha-2}e^{-x} - x^{\alpha-1}e^{-x} = 0 \implies e^{-x}x^{\alpha-1} \cdot ((\alpha-1)x^{-1} - 1)$$

$$\implies x = \alpha - 1.$$

For $\alpha > 0$, the supremum is given by

$$\sup_x f^*(x) = (\alpha-1)^{\alpha-1}e^{\alpha-1} \implies a = \sqrt{(\alpha-1)^{\alpha-1}e^{-\alpha+1}}.$$

The constant b_+ is given by

$$b_+ = \sqrt{\sup_{x\geq 0}(x^2 f^*(x))}.$$

$$\frac{\partial}{\partial x}x^2 f^*(x) = \frac{\partial}{\partial x}x^{\alpha+1}e^{-x} = (\alpha+1)x^\alpha e^{-x} - e^{-x}x^{\alpha+1}$$

By setting this equal to zero, we get

$$0 = (\alpha+1)x^\alpha e^{-x} - e^{-x}x^{\alpha+1} \implies x = \alpha+1$$

Thus, the supremum is

$$\sup_x x^2 f^*(x) = \begin{cases} (\alpha+1)^{\alpha-1}e^{-\alpha-1} & \text{if } 0 < x, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

,

which means that the constant b_+ is given by

$$b_+ = \sqrt{(\alpha+1)^{(\alpha+1)}e^{-\alpha-1}}$$

and

$$b_- = 0$$

b.)

The algorithm to generate n samples from f has to be implemented on log-scale. The log-transformations are

$$X_1 \sim \text{Uniform}(0,a) \implies \log(X_1) = \log(a\cdot U_1) = \log(a) + log(U_1)$$

where $U_1 \sim \text{Uniform}(0,1)$

$$X_2 \sim \text{Uniform}(b_-,b_+) = \text{Uniform}(0,b_+) \implies \log(X_2) = \log(b_+\cdot U_2) = \log(b_+) + \log(U_2)$$

where $U_2 \sim \text{Uniform}(0,1)$. We have

$$\frac{x_2}{x_1} = \exp\left(\log\left(\frac{x_2}{x_1}\right)\right) = \exp\left(\log(x_1) + \log(x_2)\right)$$

$$f^*\left(\frac{x_2}{x_1}\right) = \begin{cases} \left(\frac{x_2}{x_1}\right)^{\alpha-1} e^{-(x_2/x_1)} & \text{if } 0 < x_2/x_1, \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

$$\implies$$

$$\log f^*\left(\frac{x_2}{x_1}\right) = \begin{cases} (\alpha-1)\log(x_2/x_1) - (x_2/x_1) & \text{if } 0 < x_2/x_1, \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

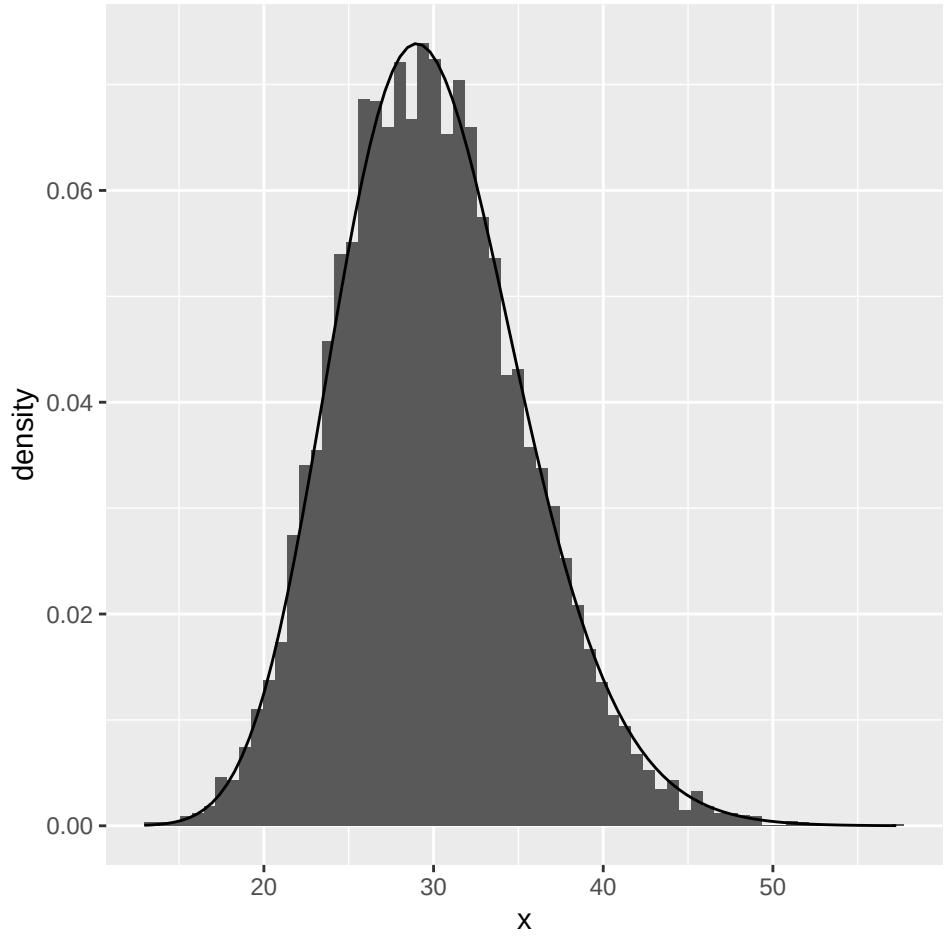
We have

$$0 \leq x_1 \leq \sqrt{f^*(x_2/x_1)} \implies \log(x_1) \implies \frac{1}{2}\log(f^*(x_2/x_1))$$

```r
log_f_star <- function(x, alpha) {
    if (x <= 0) {
        return(0)
    } else if (x > 0) {
        return(log(x^(alpha - 1) * exp(-x)))
    }
}


# Function generating samples from f
sample_gamma_2 <- function(n, alpha) {
    a <- sqrt((alpha - 1)^(alpha - 1) * exp(-alpha + 1))
    b <- sqrt((alpha + 1)^(alpha + 1) * exp(-alpha - 1))
    loga <- log(a)
    logb <- log(b)
    x <- vector()
    count <- 0
    for (i in 1:n) {
        finished = 0
        while (finished == 0) {
            log_x1 <- loga + log(runif(1))
            log_x2 <- logb + log(runif(1))
            fun <- log_f_star(exp(log_x2 - log_x1), alpha)
            if (log_x1 <= (1/2) * fun) {
                x[i] = exp(log_x2 - log_x1)
                finished = 1
            }
            count <- count + 1
        }
    }
    return(list(x = x, count = count))
}
```

We plot the results

```r
n <- 10000
alpha <- 30
x <- sample_gamma_2(n, alpha)$x

ggplot() + geom_histogram(data = as.data.frame(x), mapping = aes(x = x, y = ..density..),
    binwidth = 0.7) + stat_function(fun = dgamma, args = list(shape = alpha))
```



3.

We want to write an R function that generates a vector of n independent samples from a gamma distribution with parameters α and β. So far we have made functions that generate from a gamma distribution with $\alpha \in (0,1)$ and $\alpha \in (1,\infty)$ and $\beta = 1$. The parameter β is an inverse scale parameter. This means that if $X \sim \text{Gamma}(\alpha, 1)$, then $\frac{1}{\beta}X \sim \text{Gamma}(\alpha, \beta)$. Thus, when we sample from $\text{Gamma}(\alpha, 1)$, we can multiply the samples with $1/\beta$. The parameter α can also be 1. We have already made a function that generate samples from the exponential distribution. If $X \sim \text{Gamma}(1, \beta)$, the X has pdf

$$f(x) = \beta e^{-\beta x} \implies X \sim \exp(\beta)$$

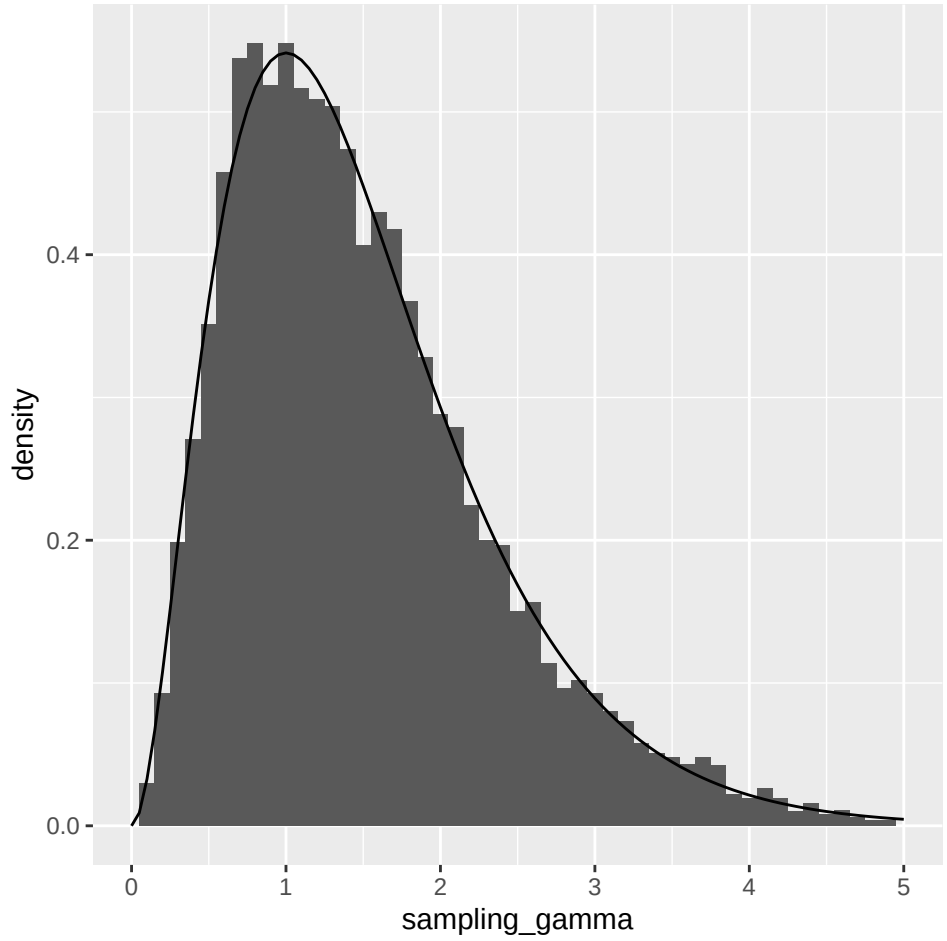
Therefore, we can use the function generating from the exponential distribution when $\alpha = 1$.

```
sample_gamma_final<-function(n, alpha,beta)
{
  #If alpha=1, we generate from the exponential distribution
  if(alpha==1)
  {
    x<-generate_exponential(n,beta)
  }
  #If alpha>1 we use the function from B.2
  else if(alpha>1)
```

```
  {
    x<-(1/beta)*sample_gamma_2(n,alpha)$x
  }
  #If 0<alpha<1, we use the function from B.1
  else if(alpha>0 & alpha<1)
  {
    x<-(1/beta)*sample_gamma(n,alpha)
  }
  else
  {
    return(0)
  }
}
```

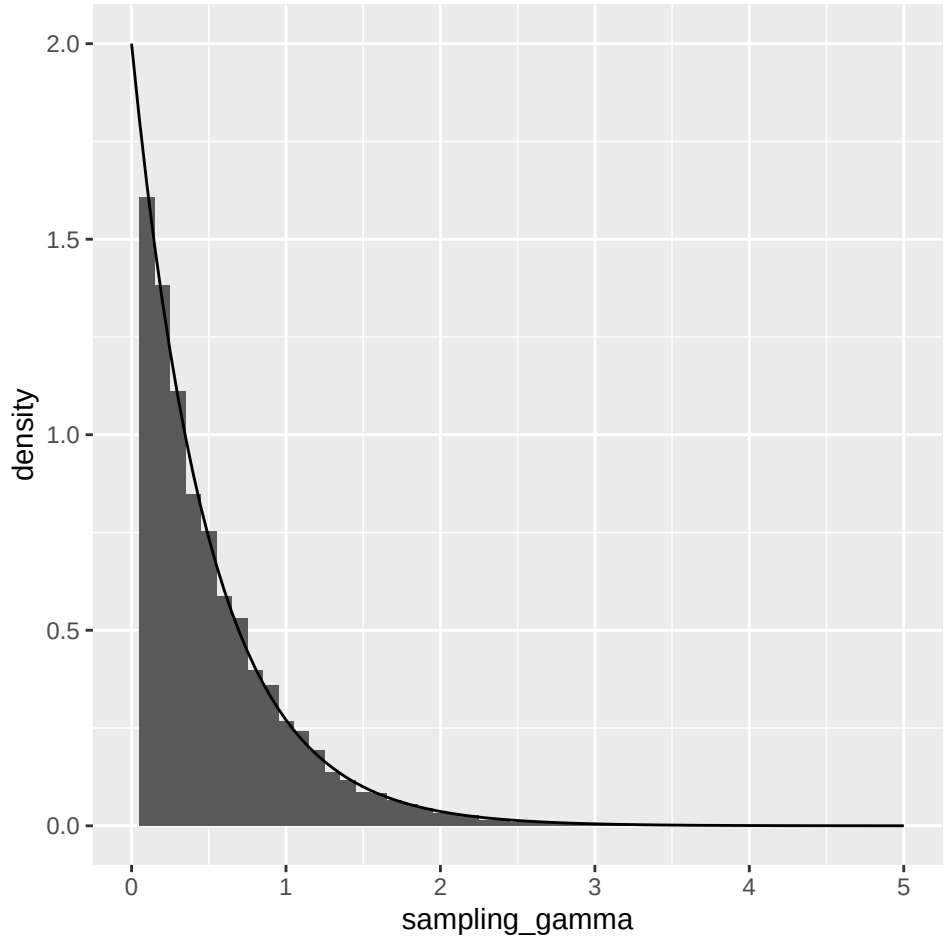
We generate realizations for the different cases. The first we do is when $\alpha = 2$ and $\beta = 3$.

```
alpha = 3
beta = 2
n = 10000
sampling_gamma <- sample_gamma_final(n, alpha, beta)

ggplot() + geom_histogram(data = as.data.frame(sampling_gamma), mapping = aes(x = sampling_gamma,
    y = ..density..), binwidth = 0.1) + xlim(0, 5) + stat_function(fun = dgamma,
    args = list(shape = alpha, rate = beta))
```



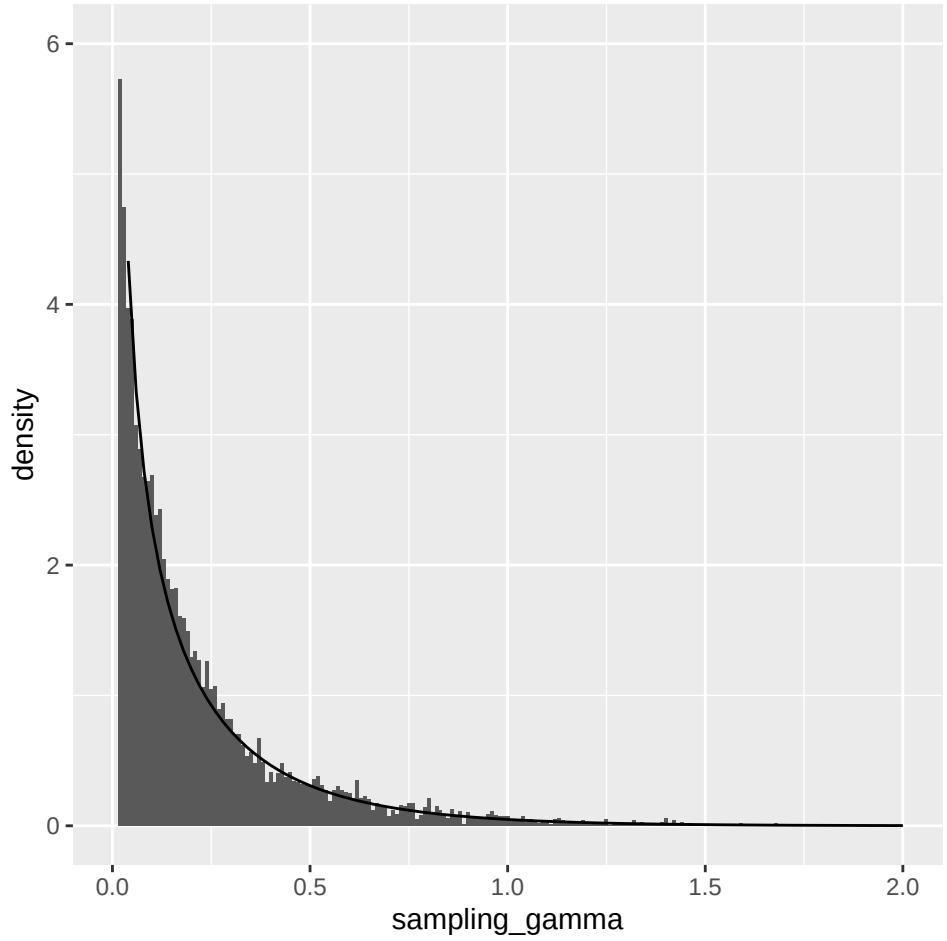
We try an example for when $\alpha = 1$ and $\beta = 2$

```
alpha = 1
beta = 2
n = 10000
sampling_gamma <- sample_gamma_final(n, alpha, beta)

ggplot() + geom_histogram(data = as.data.frame(sampling_gamma), mapping = aes(x = sampling_gamma,
    y = ..density..), binwidth = 0.1) + xlim(0, 5) + stat_function(fun = dgamma,
    args = list(shape = alpha, rate = beta))
```



At last, we try when $alpha = 0.5$ and $\beta = 3$

```
alpha = 0.5
beta = 3
n = 10000
sampling_gamma <- sample_gamma_final(n, alpha, beta)

ggplot() + geom_histogram(data = as.data.frame(sampling_gamma), mapping = aes(x = sampling_gamma,
    y = ..density..), binwidth = 0.01) + xlim(0, 5) + xlim(0, 2) + ylim(0, 6) + stat_function(fun = dgam
    args = list(shape = alpha, rate = beta))
```



4.

a)

Let $x \sim \text{Gamma}(\alpha, 1)$ and $y \sim \text{Gamma}(\beta, 1)$ be independent and let $z = x/(x+y)$. The pdfs of X and Y are

$$f_X(x) = \frac{x^{\alpha-1}e^{-x}}{\Gamma(\alpha)}$$

$$f_Y(y) = \frac{y^{\beta-1}e^{-y}}{\Gamma(\beta)}$$

The joint distribution of X and Y is given by

$$f_{X,Y}(x,y) = f_x(x) \cdot f_y(y) = \frac{x^{\alpha-1}y^{\beta-1}e^{-x-y}}{\Gamma(\beta)\Gamma(\alpha)}.$$

Let $Z = \frac{X}{X+Y}$ and $V = X + Y$, which means that we use the transformations

$$x = h_1(z,v) = z \cdot v \quad \text{and} \quad y = h_2(z,v) = v(1-z).$$

The Jacobian is

$$J = \begin{vmatrix} \frac{\partial x}{\partial z} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial z} & \frac{\partial y}{\partial v} \end{vmatrix} = \frac{\partial x}{\partial z}\frac{\partial y}{\partial v} - \frac{\partial y}{\partial z}\frac{\partial x}{\partial v}$$

$$= v(1-z) - (-v) \cdot z = v$$

The joint distribution of U and V is given by

$$f_{Z,V}(z,v) = f_{X,Y}(h_1(z,v), h_2(z,v))|J|$$

$$= f_{x,y}(z \cdot v, v(1-z)) \cdot v = \frac{(z \cdot v)^{\alpha-1}(v(1-z))^{\beta-1}e^{-zv-v+vz}}{\Gamma(\alpha)\Gamma(\beta)} \cdot v$$

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

The distribution of $V = X + Y$ will be a gamma distribution since X and Y are independent. The mgf of V is given by

$$M_V(t) = E[e^{Vt}] = E[e^{(X+Y)t}] = E[e^{Xt}] \cdot E[e^{Yt}] = M_x(t) \cdot M_y(t)$$

$$= \left(\frac{1}{1-t}\right)^{\alpha} \cdot \left(\frac{1}{1-t}\right)^{\beta} = \left(\frac{1}{1-t}\right)^{\alpha+\beta}.$$

$$\implies V \sim \text{Gamma}(\alpha+\beta, 1) \implies f_V(v) = \frac{v^{\alpha+\beta-1}e^{-v}}{\Gamma(\alpha+\beta)}$$

Since Z and V are independent, the joint distribution can be written.

$$f_{Z,V}(z,v) = f_Z(z) \cdot f_V(v) \implies f_Z(z) = \frac{f_{Z,V}(z,v)}{f_V(v)}$$

$$\implies f_Z(z) = \frac{v^{\alpha+\beta-1}z^{\alpha-1}(1-z)^{\beta-1}e^{-v}}{\Gamma(\alpha)\Gamma(\beta)} \cdot \frac{\Gamma(\alpha+\beta)}{v^{\alpha+\beta-1}e^{-v}} = \frac{\Gamma(\alpha+\beta)z^{\alpha-1}(1-z)^{1-\beta}}{\Gamma(\alpha)\Gamma(\beta)}$$

This is the density of a $beta(\alpha, \beta)-$ distribution.

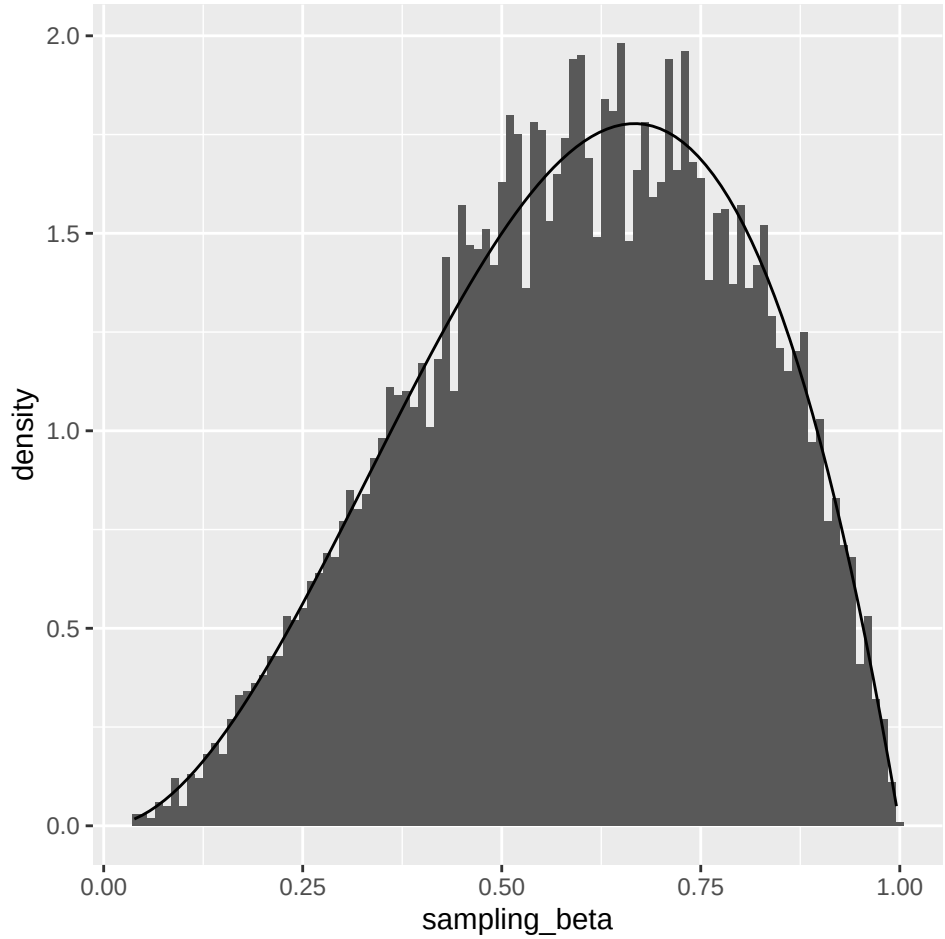
(b)

This means that we can use the function for generating samples from a Gamma(α, β) to sample from $x \sim$ Gamma$(\alpha, 1)$ and $y \sim$ Gamma$(\beta, 1)$ and return $z = x/(x+y)$ In the following code, n independent samples from a beta function is generated.

```
sample_from_beta <- function(n, alpha, beta) {
    x <- sample_gamma_final(n, alpha, 1)
    y <- sample_gamma_final(n, beta, 1)
    z <- x/(x + y)
    return(z)
}
```

We generate sample from the function and plot it together with the theoretical distribution.

```
alpha = 3
beta = 2
sampling_beta <- sample_from_beta(n, alpha, beta)

ggplot() + geom_histogram(data = as.data.frame(sampling_beta), mapping = aes(x = sampling_beta,
    y = ..density..), binwidth = 0.01) + stat_function(fun = dbeta, args = list(shape1 = alpha,
    shape2 = beta))
```



Problem C: Monte Carlo integration and variance reduction

Here we will consider Monte Carlo integration to estimate $\theta = P(X > 4)$ when $X \sim N(0, 1)$. Then we will compare the variance reduction in importance sampling and antithethic sampling.

1 Monte Carlo integration

Let $h(X) = I(X > 4)$, where I is the indicator function, such that

$$\begin{aligned} E[h(X)] &= \int_{-\infty}^{\infty} h(x)f(x)dx \\ &= \int_{-\infty}^{\infty} I(x > 4)f(x)dx \\ &= P(X > 4) \\ &= \theta. \end{aligned}$$

Then, the Monte Carlo Estimate of θ is given by

$$\hat{\theta}_{MC} = \frac{1}{n}\sum_{i=1}^{n} h(x_i).$$

Now we will find a $1-\alpha$ confidence interval for θ based on our sample set. First we need the expected value of the Monte Carlo estimator

$$\begin{aligned} E[\hat{\theta}] &= E[\frac{1}{n}\sum_{i=1}^{n} h(x_i)] \\ &= \frac{1}{n}\sum_{i=1}^{n} \theta \\ &= \theta, \end{aligned}$$

and it's variance

$$\begin{aligned} Var(\hat{\theta}) &= Var(\frac{1}{n}\sum_{i=1}^{n} h(x_i)) \\ &= \frac{1}{n^2}\sum_{i=1}^{n} Var(h(x_i)) \\ &= \frac{1}{n}\frac{1}{n-1}\sum_{i=1}^{n} (h(x_i)-\hat{\theta})^2. \end{aligned}$$

Then we get the statistic

$$T_{MC} = \frac{\hat{\theta}_{MC} - \theta}{\sqrt{\hat{Var}(\hat{\theta}_{MC})}} \sim t_{n-1}.$$

Below there is an implementation with $n = 100000$ samples of X which we use to find the Monte Carlo estimate θ.

```
set.seed(321)  # Seed for reproducibility.
n = 1e+05
x = generate_from_normal(n)  # Drawing n samples fron N(0,1)
h = x > 4
MCest = mean(h)  # Monte Carlo estimate
theta = pnorm(4, lower.tail = F)  # True theta
# Confidence interval and results
svMC = var(h)  # Sample variance
alpha = 0.05
t = qt(alpha/2, n - 1, lower.tail = F)  # (1-alpha) significance
lwrUpr = sqrt(svMC/n) * t  # lower and upper deviation from mean
ciMC = MCest + c(-lwrUpr, lwrUpr)
resultMC = c(Estimator = MCest, Confint = ciMC, Var = svMC, error = abs(theta - MCest))
resultMC
```

```
##    Estimator     Confint1     Confint2          Var        error
## 5.000000e-05 6.174219e-06 9.382578e-05 4.999800e-05 1.832876e-05
```

```
theta
```

```
## [1] 3.167124e-05
```

Here we see an error in the $1 \cdot 10^{-5}$ decimal and that the true value coincide with the 95% confidence interval.

2 Importance sampling

Here we will use importance sampling on the same problem as in C1 to try to reduce the variance of the Monte Carlo integration. The proposal distribution is

$$g(x) = \begin{cases} cxe^{-x^2/2} & , \ x > 4 \\ 0 & , \ \text{otherwise,} \end{cases}$$

where c is a normalizing constant. Now, let $x_1, ..., x_n \sim g(x)$ and let $w_i = f(x_i)/g(x_i)$ be the weights. Then, the importance sampling estimator of θ is

$$\hat{\theta}_{IS} = \frac{\sum_{i=1}^{n} h_i w_i}{n}.$$

In order to use inversion sampling on the proposal distribution g we need its cdf,

$$\begin{aligned} G(x) &= \int_4^x cye^{-y^2/2} dy \\ &= \int_8^{x^2/2} ce^{-u} du \\ &= \left[-ce^{-u}\right]_8^{x^2/2} \\ &= c(e^{-8} - e^{-x^2/2}). \end{aligned}$$

Since g is a distribution, and therefore $\int_4^\infty g(x)dx = 1$, we can find c by solving

$$\begin{aligned} c(e^{-8} - e^{-x^2/2})\Big|_{x=\infty} &= 1 \\ c &= e^8. \end{aligned}$$

Then we have $G(x) = 1 - e^{8-x^2/2}$. Now we can sample from g by solving $U = G(x) \sim Unif(0,1)$ for x, that is,

$$\begin{aligned} U &= 1 - e^{8-x^2/2} \\ -2ln(1-U) &= x^2 - 16 \\ x &= \sqrt{16 - 2ln(1-U)}. \end{aligned}$$

Thus, our samples are generated by inserting randomly selected $U \sim Unif(0,1)$ admits samples from $X \sim g$.

We also need the expected value,

$$\begin{aligned} E[\hat{\theta}_{IS}] &= E\left[\frac{\sum_{i=1}^{n} h_i w_i}{n}\right] \\ &= \frac{1}{n}\sum_{i=1}^{n} \int_0^\infty h_i \frac{f_i}{g_i} g_i dx \\ &= \frac{1}{n}\sum_{i=1}^{n} \int_0^\infty h_i f_i dx \\ &= \frac{1}{n}\sum_{i=1}^{n} E[h_i] \\ &= \frac{1}{n} n\theta \\ &= \theta, \end{aligned}$$

and the sample variance,

$$\begin{aligned} Var(\hat{\theta}_{IS}) &= Var\left(\frac{\sum_{i=1}^{n} h_i w_i}{n}\right) \\ &= \frac{1}{n(n-1)} \sum_{i=1}^{n} \left(h_i w_i - \sum_{i=1}^{n} \frac{h_i w_i}{n}\right)^2 \\ &= \frac{1}{n(n-1)} \sum_{i=1}^{n} \left(h_i w_i - \hat{\theta}_{IS}\right)^2, \end{aligned}$$

of the importance sample estimator to compute the $(1-\alpha)$ confidence interval. Below we have implemented the computation of the importance sample estimate along with the 95% confidence interval.

```
expSampler <- function(n) {
    # Samples from proposal distribution g
    u = runif(n)
    return(sqrt(16 - 2 * log(1 - u)))
}

w <- function(x) {
    # Weight function
    f = dnorm(x)
    g = ifelse(x > 4, x * exp(8 - 0.5 * x^2), 0)
    return(f/g)
}

set.seed(321)
gx = expSampler(n)  # Sample from proposal
gh = (gx > 4) * 1
ISest = mean(gh * w(gx))  # Importance sample estimate
svIS = var(gh * w(gx))
ISconfint = ISest + c(-t * sqrt(svIS/n), t * svIS/n)
resultIS = c(ISestimate = ISest, confint = ISconfint, var = svIS, error = abs(theta -
    ISest))
results = rbind(MC = resultMC, IS = resultIS)
results
```

```
##       Estimator     Confint1     Confint2          Var        error
## MC 5.000000e-05 6.174219e-06 9.382578e-05 4.999800e-05 1.832876e-05
## IS 3.167611e-05 3.166649e-05 3.167611e-05 2.410122e-12 4.866683e-09
```

```
theta
```

```
## [1] 3.167124e-05
```

Here we see that importance sampling has reduced the variance by a factor of $Var(\hat{\theta}_{MC})/Var(\hat{\theta}_{IS}) =$ `svMC/svIS` $= 2.074501 \times 10^7$. Also, the importance sample estimator is a much more precise estimate considering the error. If we let variance be a more general proxy for precision, then we can estimate the number of samples n needed in C1 to obtain the same precision as the importance sampling technique with

$m = 100000$ samples. This is done by setting

$$\begin{aligned} Var(\hat{\theta}_{MC}) &= Var(\hat{\theta}_{IS}) \\ \Leftrightarrow \frac{1}{n} Var(h(X)) &= \frac{1}{m} Var(h(X)w(X)) \\ \Leftrightarrow n &= m \frac{Var(h(X))}{Var(h(X)w(X))}. \end{aligned}$$

From the previous results we get the estimated number of samples, $n =$ `svMC / svIS` $= 2.074501 \times 10^{12}$, needed to obtain the same precision as we did using importance sampling as a variance reducing technique.

3 Antithetic Sampling

Now we will combine the importance sampling with the use of antithetic variates. We start by modifying the sample generator for g in task C2 so that it produces n pairs, $X = x_i$ and $Y = y_i$, of antithetic variates generated from G^{-1} with inputs u_i and $1 - u_i$, respectively.

```
expSamplerAnti <- function(n) {
    # Pairwise sampling from proposal distribution evaluated at u and 1-u.
    u = runif(n)
    return(data.frame(x = sqrt(16 - 2 * log(1 - u)), y = sqrt(16 - 2 * log(u))))
}
```

Then, the importance sample estimates for each of the pairs are

$$\hat{\theta}_X = \frac{1}{n} \sum_{i=1}^{n} h(x_i)w(x_i),$$

$$\hat{\theta}_Y = \frac{1}{n} \sum_{i=1}^{n} h(y_i)w(y_i),$$

and the antithetic sample estimator is

$$\hat{\theta}_A = \frac{\hat{\theta}_X + \hat{\theta}_Y}{2}.$$

We also need the expected value

$$\begin{aligned} E[\hat{\theta}_{AS}] &= E\left[\frac{\hat{\theta}_X + \hat{\theta}_Y}{2}\right] \\ &= \frac{1}{2n} \sum_{i=1}^{n} (E[h(x_i)w(x_i)] + E[h(y_i)w(y_i)]) \\ &= \frac{1}{2n} \sum_{i=1}^{n} 2\theta \\ &= \theta, \end{aligned}$$

and the variance is

$$Var(\hat{\theta}_{AS}) = \frac{1}{4}(Var(\hat{\theta}_X) + Var(\hat{\theta}_Y) + 2Cov(\hat{\theta}_X, \hat{\theta}_Y))$$

where $\rho_{XY} = Cov(\hat{\theta}_X, \hat{\theta}_Y)$ and S^2_{XY} is the sample variance of either estimator $\hat{\theta}_X$ or $\hat{\theta}_Y$. For a fair comparison to importance sampling we set the number of samples $m = n/2$ since antithetic sampling generates m samples for each of the function evaluations of $G^{-1}(\cdot)$.

```r
set.seed(321)
m = 50000
xy = expSamplerAnti(m)
hx = (xy$x > 4) * 1  # h(X)
hy = (xy$y > 4) * 1  # h(Y)
hwx = hx * w(xy$x)  # h(X)w(X)
hwy = hy * w(xy$y)  # h(Y)w(Y)
AS = (hwx + hwy)/2  # (theta_x + theta_y)/2
ASest = mean(AS)
svAS = var(AS)
lwrUprAS = c(-t, t) * sqrt(svAS/n)
confintAS = ASest + lwrUprAS
resultAS = c(ASest, confintAS, svAS, abs(theta - ASest))
rbind(results, AS = resultAS)
```

```
##       Estimator     Confint1     Confint2          Var        error
## MC 5.000000e-05 6.174219e-06 9.382578e-05 4.999800e-05 1.832876e-05
## IS 3.167611e-05 3.166649e-05 3.167611e-05 2.410122e-12 4.866683e-09
## AS 3.167262e-05 3.166931e-05 3.167593e-05 2.851883e-13 1.378996e-09
```

Problem D: Rejection sampling and importance sampling

The observed data is $y = [y_1, y_2, y_3, y_4] = [125, 18, 20, 34]$

The multinomial mass function is $f(y|\theta) \propto (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4}$. Assuming a uniform prior, the posterior will be

$$f(\theta|y) \propto f^*(\theta) = (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4} \quad \theta \in (0,1)$$

We construct a rejection sampling algorithm to simulate from $f(\theta|y)$. The proposal distribution is $g(\theta) = 1$. The constant c should satisfy

$$f(\theta|y) \leq c \cdot g(\theta|y) = c$$

We find the maximum of $f(\theta|y)$ given the observed values.

```r
f_posterior <- function(theta) {
    return((2 + theta)^125 * (1 - theta)^(18 + 20) * theta^(34))
}

c <- optimize(f_posterior, c(0, 1), maximum = TRUE)$objective
```

```r
rejection_sampling_f <- function(n, p) {
    # p is the posterior function
    x <- vector(mode = "numeric", length = n)
    num_gen = 0
    for (i in 1:n) {
        finished = 0
        while (finished == 0) {
            xi <- runif(1)
            alpha <- (1/c) * p(xi)
            u <- runif(1)
            if (u <= alpha) {
                x[i] = xi
```

```
                finished = 1
            }
            num_gen <- num_gen + 1
        }
    }
    return(list(sample_f = x, numbers = num_gen))
}
```

2.

We want to estimate the posterior mean of θ by Monte-Carlo integration using $M = 10000$ samples from $f(\theta|y)$. The Monte-carlo estimate of the mean is given by

$$\hat{\mu} = \frac{1}{M}\sum_{i=1}^{M}\Theta_i$$

where $\Theta_i, ..., \Theta_M \sim f(\theta|y)$.

We find the normalizing k for the posterior, which is found by solving the following equation for k

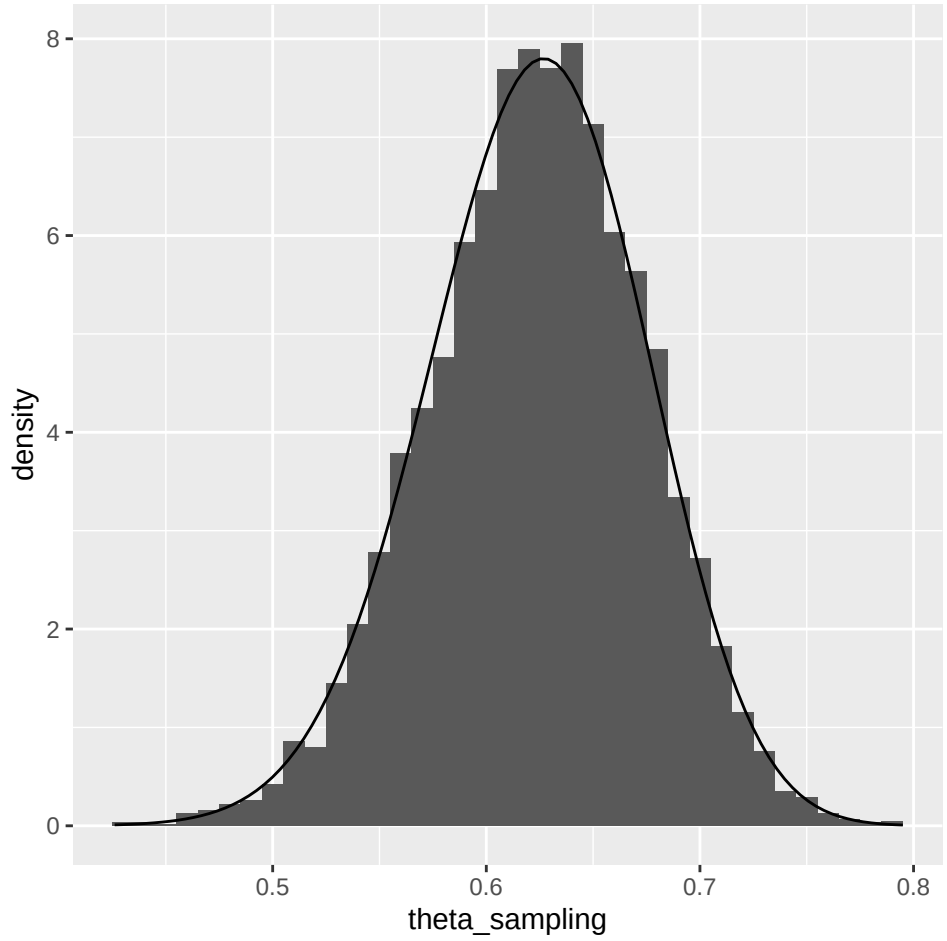
$$1 = k\int_0^1 f^*(\theta)d\theta = \int_0^1 (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4}d\theta$$

$$\implies k = \frac{1}{\int_0^1 (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4}d\theta}$$

In the code below, a histogram of the sample is drawn and is compared to the theoretical distribution. The estimated posterior mean of θ is also found and compared to the theoretical value.

```
norm_con <- integrate(f_posterior, 0, 1)$val

posterior <- function(theta) {
    return((1/norm_con) * f_posterior(theta))
}
```

```
M <- 10000
theta_sampling <- rejection_sampling_f(M, f_posterior)$sample_f

ggplot() + geom_histogram(data = as.data.frame(theta_sampling), mapping = aes(x = theta_sampling,
    y = ..density..), binwidth = 0.01) + stat_function(fun = posterior, )
```




```
mu_hat <- mean(theta_sampling)
mu_hat
```

```
## [1] 0.6234258
```

```
mu_theoretical <- integrate(function(theta) (theta * posterior(theta)), 0, 1)$val
mu_theoretical
```

```
## [1] 0.6228061
```

The estimated posterior mean of θ using Monte-Carlo integration is 0.6234258. The theoretical value of the mean using numerical integration is 0.6228061.

3.

The number of random numbers the sampling algorithm needs to generate on average is given by the total number of random numbers the the algorithm generates divided by the number of samples of $f(\theta|y)$

```
total_number <- rejection_sampling_f(M, f_posterior)$numbers
num <- total_number/10000
```

The amount of random numbers needed to generate to obtain one sample of $f(\theta|y)$ is 7.9127. The overall acceptance probability is given by

$$P(U \leq \frac{1}{c} \cdot \frac{f(X)}{g(X)}) = \int_{-\infty}^{\infty} dx = \frac{f(x)}{c \cdot g(x)} g(x) = c^{-1}$$

This means that c is the expected number of tries to obtain one sample of $f(\theta|y)$, which is 7.7993075. This number is close to the number from our implemented algorithm.

4. Beta$(1, 5)$ as prior.

The posterior distribution given a prior $\pi(\theta)$ is given by

$$f(\theta|y) = \frac{f(y|\theta)\pi(\theta)}{\int f(y|\theta)\pi(\theta)d\theta}$$

If the prior has a beta$(1, 5)$ distribution, the posterior is

$$\begin{aligned} f(\theta|y) &\propto \frac{(1-\theta)^{\beta-1}/B(1,5)(2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4}}{\int_0^1 (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4}(1-\theta)^{\beta-1}/B(1,5)d\theta} \\ &\propto (1-\theta)^{\beta-1}/B(1,5)(2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4} \end{aligned}$$

D4 chris

In section D2 we used a uniform prior for θ. Now we will assume a Beta$(1, 5)$ prior instead, that is

$$\begin{aligned} f(\theta) &= \frac{1}{B(1,5)}\theta^{1-1}(1-\theta)^{5-1} \\ &= \frac{1}{B(1,5)}(1-\theta)^4, \end{aligned}$$

where B is the beta function. Then, the posterior is

$$\begin{aligned} f(\theta|y) &= \frac{(2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4}(1-\theta)^4/B(1,5)}{\int_0^1 (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4}(1-\theta)^4/B(1,5)d\theta} \\ &\propto (2+\theta)^{y_1}(1-\theta)^{y_2+y_3+4}\theta^{y_4}, \end{aligned}$$

which is the un-scaled posterior denoted $f^*(\theta|y)$. As the proposal distribution we use the un-scaled posterior from section D2, $f^*(\theta)$, such that the importance sample weights are

$$w_i = \frac{f^*(\theta_i|y)}{f^*(\theta_i)} = (1-\theta_i)^4$$

The self-normalizing importance sample estimator is then

$$\tilde{\mu}_{IS} = \frac{\sum_{i=1}^n \theta_i w_i}{\sum_{i=1}^n w_i}.$$

```r
f_posterior15 <- function(theta) {
    # Posterior when prior is beta(1,5)
    return(f_posterior(theta) * (1 - theta)^4)
}

norm_con15 = integrate(f_posterior15, 0, 1)$val
c <- optimize(f_posterior15, c(0, 1), maximum = TRUE)$objective

posterior15 <- function(theta) {
    # Normalized posterior
    return(f_posterior15(theta)/norm_con15)
}

w <- function(x) {
    return((1 - x)^4)
}

impSamp = theta_sampling * w(theta_sampling)
# Self-normalized IS estimate
impEst = sum(impSamp)/sum(w(theta_sampling))
# Theoretical mean
mu_theoretical15 = integrate(function(theta) (theta * posterior15(theta)), 0, 1)$val
c(IS = impEst, Theoretical = mu_theoretical15)
```

```
##          IS Theoretical
##   0.5964497   0.5959316
```

```r
rbind(Estimate = c(B15 = impEst, B11 = mu_hat), NumericalInt = c(mu_theoretical15,
    mu_theoretical))
```

```
##                    B15       B11
## Estimate     0.5964497 0.6234258
## NumericalInt 0.5959316 0.6228061
```

Here we see the `B15` column corresponding to our estimated mean using $B(1,5)$ as prior with its numerically integrated value below, and the computations from section D2 in the `B11` column. We see that the former estimated mean coincide well with the corresponding numerical integration. Also we notice that these values are smaller than those in column `B11`, which can be explained by the importance sample weight function $w_i = (1-\theta_i)^4$ favouring small value of $\theta \in (0,1)$.