Excercise 1, Spring 2022

TMA4300 - Computer Intensive Statistical Methods

Sivert Laukli, Thomas Torkildsen

31 1 2022

```
# load packages here
library(ggplot2)
library(tidyverse)
```

Problem A:

A1

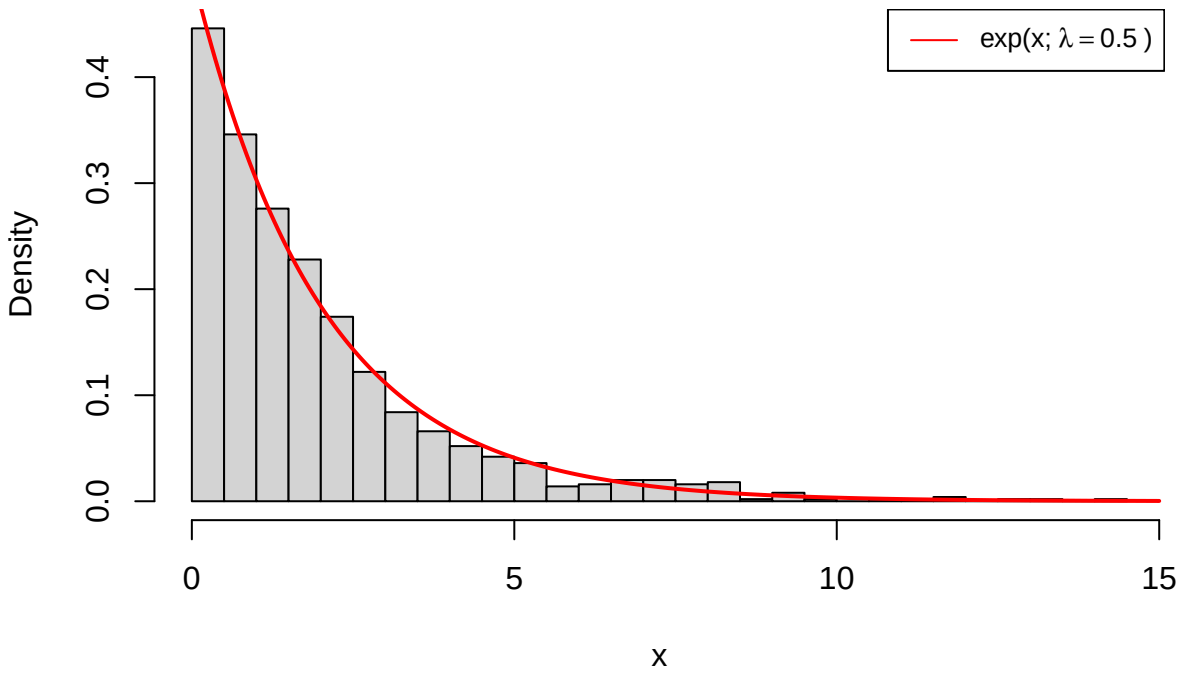
To begin with, we create a function `expsample(rate, n)` generating `n` samples from an exponential distribution with probability density function (PDF)

$$f(x;\lambda) = \frac{1}{\lambda} e^{-\lambda x}.$$

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

We ensure that the function is working properly comparing a histogram of its samples to the theoretical density function using parameter $\lambda = 0.5$ for the comparison.

```
lambda = 0.5
hist(expsample(lambda, 1000), n = 50, freq = F,
     main = bquote("Histogram of generated samples x from exponential distribution with" ~ lambda == 0.5
lines(seq(0, 15, 0.01), dexp(seq(0, 15, 0.01), rate = lambda), col = "red", lwd = 2)
legend("topright", legend=bquote("exp(x;"~lambda == 0.5 ~")"),
       col="red", lty=1:2, cex=0.8)
```



A2

We consider the PDF

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

where $\alpha \in (0, 1)$ is a parameter and c is a normalizing constant.

a) To compute the cumulative distribution function (CDF) $G(x)$ of $g(x)$, we integrate $g(x)$ over the given boundaries, such that

$$G(x) = \begin{cases} 0, & x < 0, \\ \frac{c}{\alpha}x^{\alpha}, & 0 < x < 1, \\ -ce^{-x} + \frac{c}{\alpha} + ce^{-1}, & x \geq 1 \end{cases}.$$

Inverting the CDF gives

$$G^{-1}(x) = \begin{cases} 0, & x < 0, \\ (\frac{\alpha}{c}x)^{\frac{1}{\alpha}}, & 0 < x < \frac{c}{\alpha}, \\ -\log(\frac{1}{\alpha} + e^{-1} - \frac{x}{c}), & \frac{c}{\alpha} \leq x \leq 1 \end{cases}.$$

As the total probability needs to sum to 1, we get that the normalizing constant c in this case is given as

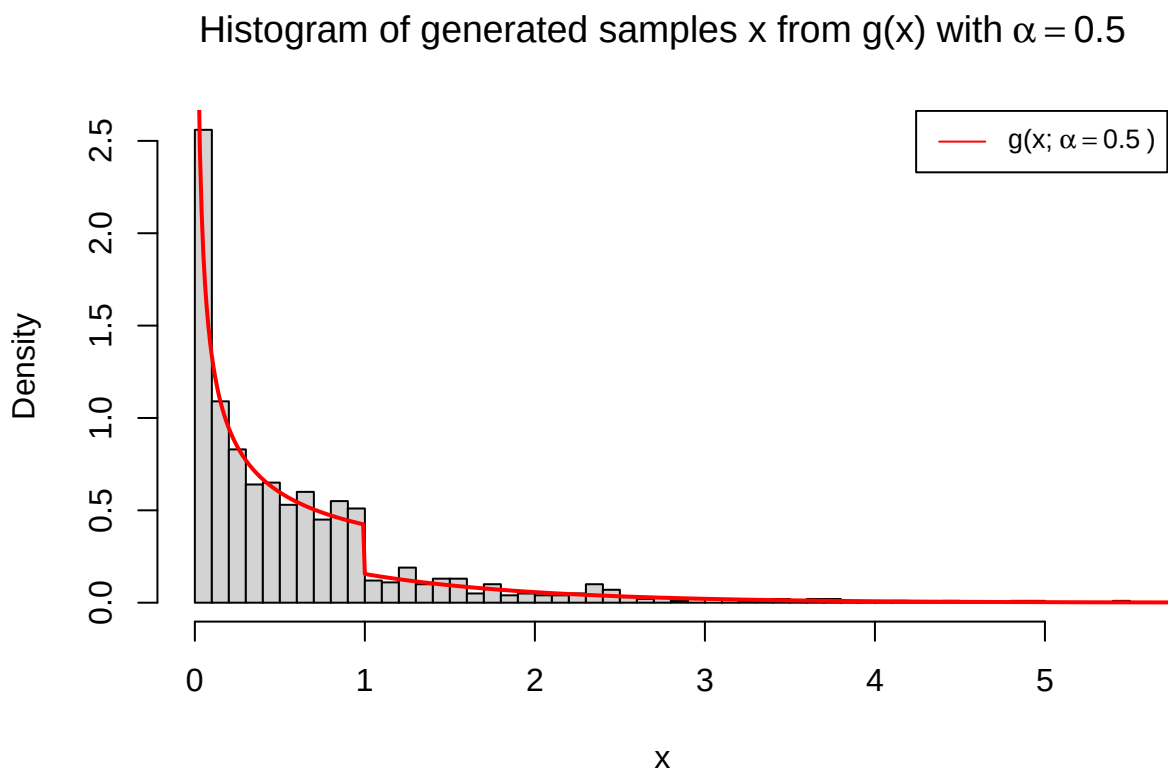
$$c = \frac{\alpha \cdot e}{\alpha + e}.$$

b) Then we create a function `gsample(alpha, n)` generating `n` samples from the distribution of $g(x)$.

```
gsample <- function(alpha, n){
  c = (alpha*exp(1))/(exp(1) + alpha)
  u = runif(n)
  threshold = c/alpha
  x = ifelse(u < threshold, (alpha*u/c)^(1/alpha), -log((1/alpha) + exp(-1) - (u/c)))
  return(x)
}
```

We ensure that the function is working properly comparing a histogram of its samples to the theoretical density function using parameter $\alpha = 0.5$ for the comparison.

```
alpha = 0.5
c = (alpha*exp(1))/(exp(1) + alpha)
xvec <- gsample(0.5, 1000)
g <- function(x){ifelse(x>=1, c*exp(-x), c*x^(alpha-1))}
hist(xvec, n = 50, freq = F,
     main = bquote("Histogram of generated samples x from g(x) with" ~ alpha == 0.5), xlab = "x")
lines(seq(0, 6, 0.01), g(seq(0, 6, 0.01)), col = "red", lwd = 2)
legend("topright", legend=bquote("g(x;"~alpha == 0.5 ~")"),
       col="red", lty=1:2, cex=0.8)
```



A3

Now we consider the distribution with PDF

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

with parameter α and normalizing constant c.

a) We will initially find the value of c. Using the fact that for any PDF

$$\int_{-\infty}^{\infty} f(x)dx = 1,$$

and integrating $f(x)$ by substitution with $u = 1 + e^{\alpha x}$ such that $\frac{du}{dx} = \alpha e^{\alpha x}$,

$$\int_{-\infty}^{\infty} f(x)dx = \int_{-\infty}^{\infty} \frac{ce^{\alpha x}}{(1+e^{\alpha x})^2}dx = \frac{c}{\alpha}\int_{1}^{\infty} u^{-2}du = \frac{c}{\alpha}\left[-\frac{1}{u}\right]_1^{\infty} = \frac{c}{\alpha} = 1,$$

such that the normalizing constant becomes

$$\underline{\underline{c = \alpha}}.$$

b) The integrated PDF, CDF, with $c = \alpha$ is found by

$$F(x) = \int_{-\infty}^{x} f(x)dx = \int_{-\infty}^{x} \frac{\alpha e^{\alpha x}}{(1+e^{\alpha x})^2}dx = \int_{1}^{1+e^{\alpha x}} u^{-2}du = \left[-\frac{1}{u}\right]_1^{1+e^{\alpha x}} = \frac{e^{\alpha x}}{1+e^{\alpha x}}.$$

The inverted CDF is then

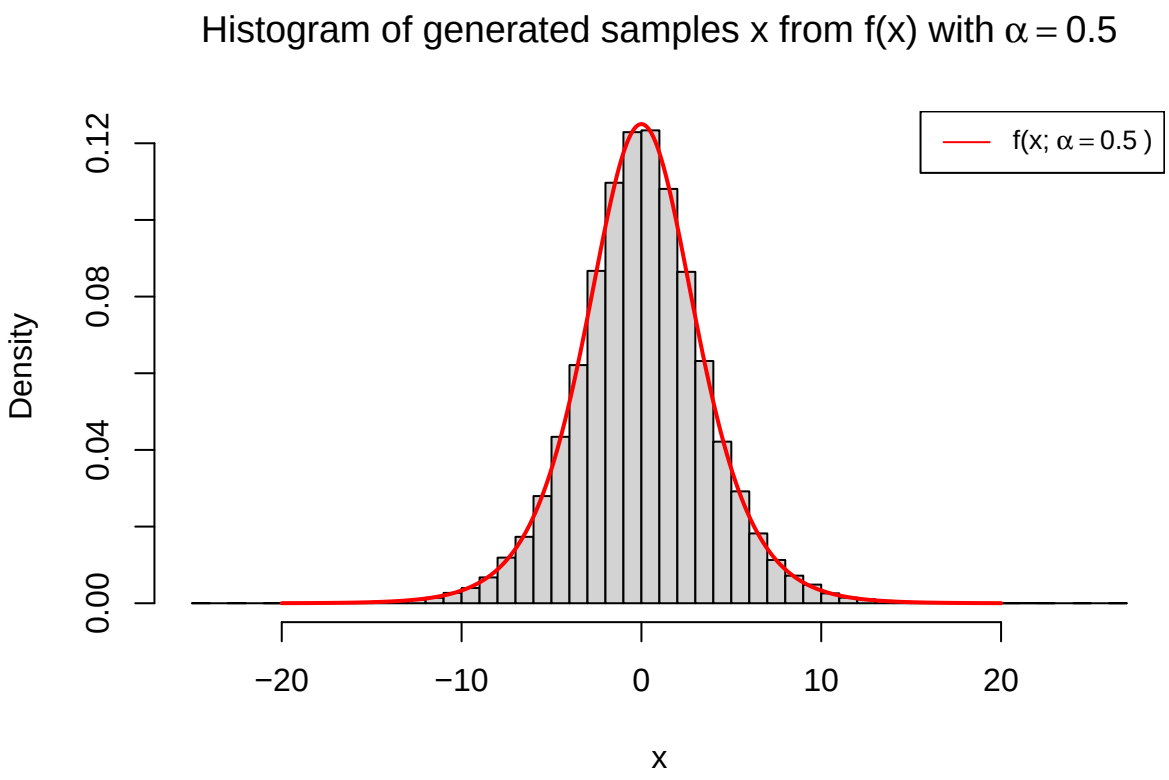
$$F^{-1}(x) = \frac{1}{\alpha}\log(\frac{x}{1-x}).$$

c) We now create a function `fsample(alpha, n)` simulating `n` samples from the distribution of $f(x)$.

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

We ensure that the function is working properly comparing a histogram of its samples to the theoretical density function using parameter $\alpha = 0.5$ for the comparison.

```
alpha = 0.5
ns = 100000
f <- function(x){alpha*exp(alpha*x)/(1 + exp(alpha*x))^2}
hist(fsample(alpha, ns), n = 50, freq = F,
     main = bquote("Histogram of generated samples x from f(x) with" ~ alpha == 0.5), xlab = "x")
lines(seq(-20,20,0.05), f(seq(-20,20,0.05)), col = "red", lwd = 2)
legend("topright", legend=bquote("f(x;"~alpha == 0.5 ~")"),
       col="red", lty=1:2, cex=0.8)
```



A4

We will now use the Box-Müller algorithm to simulate samples from a standard normal distribution.

```
n = 10000
box_muller <- function(n) {
  u1 <- runif(n, 0,2*pi)
  u2 <- expsample(0.5, n)

  x1 <- sqrt(u2)*cos(u1)
  x2 <- sqrt(u2)*sin(u1)
  return(list(x1 = x1, x2 = x2, m = matrix(c(x1,x2), nrow = 2)))
}
```

We ensure that the function is working properly comparing the mean and variance of the generated samples to the theoretical mean 0 and variance 1.

```
print(data.frame("mean_x1" = mean(box_muller(n)$x1), "variance_x1" = var(box_muller(n)$x1),
                 "mean_x2" = mean(box_muller(n)$x2), "variance_x1" = var(box_muller(n)$x2)))
```

```
##        mean_x1 variance_x1    mean_x2 variance_x1.1
## 1 -0.001693148    1.021418 0.01058752      1.009039
```

Both means and variances are close to the theoretical means and variances of 0 and 1 respectively.

A5

Now, we will create a function `multi_normal()` sampling from a multivariate normal distribution with mean vector μ and covariance matrix Σ.

```
std_normal <- function(dim, n = 1000) {
  loop = seq(1,dim-1,2)
  m = matrix(, nrow = dim, ncol = n)
  for(i in loop) {

    m[c(i,i+1),] <- box_muller(n)$m
  }
  if(i+1 < dim) {
     m[dim,] <- box_muller(n)$x1
  }
  return(m)
}

multi_normal <- function(mu, sig, n=1000) {
  X <- std_normal(length(mu), n)
  A <- chol(sig)
  return(mu + t(A) %*% X)
}
```

We assess the function evaluating the sample mean and sample covariance matrix against a predefined mean and predefined covariance matrix.

```
#testing the function
s <- matrix(c(4,1,1,4), nrow = 2)
mu <- c(4,2)
test <- multi_normal(mu, s,n=10000)

print(data.frame("Sample_mu1" = mean(test[1,]), "mu1" = mu[1], "Sample_mu2" = mean(test[2,]), "mu2" = mu
```

```
##   Sample_mu1 mu1 Sample_mu2 mu2
## 1    3.99435   4   1.963969   2
```

```
# Sample covariance matrix
print(matrix(c(cov(test[1,], test[1,]), cov(test[1,], test[2,]),
               cov(test[2,], test[1,]), cov(test[2,], test[2,])), ncol = 2))
```

```
##          [,1]     [,2]
## [1,] 4.024853 1.021617
## [2,] 1.021617 4.041583
```

```
# Covariance matrix
print(s)
```

```
##      [,1] [,2]
## [1,]    4    1
## [2,]    1    4
```

The obtained results asserts that the magnitude of the means and covariances are in the neighbourhood of the predefined ones ensuring our function to work as expected.

Problem B:

B1

Consider the gamma distribution with scale parameter $\beta = 1, \alpha \in (0, 1)$,

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

Rejection sampling can be used to generate samples from this distribution by using samples from the distribution considered in A2. Let the density be

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

a) The acceptance probability is given by

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

where

$$c = \frac{1}{\Gamma(\alpha)}$$

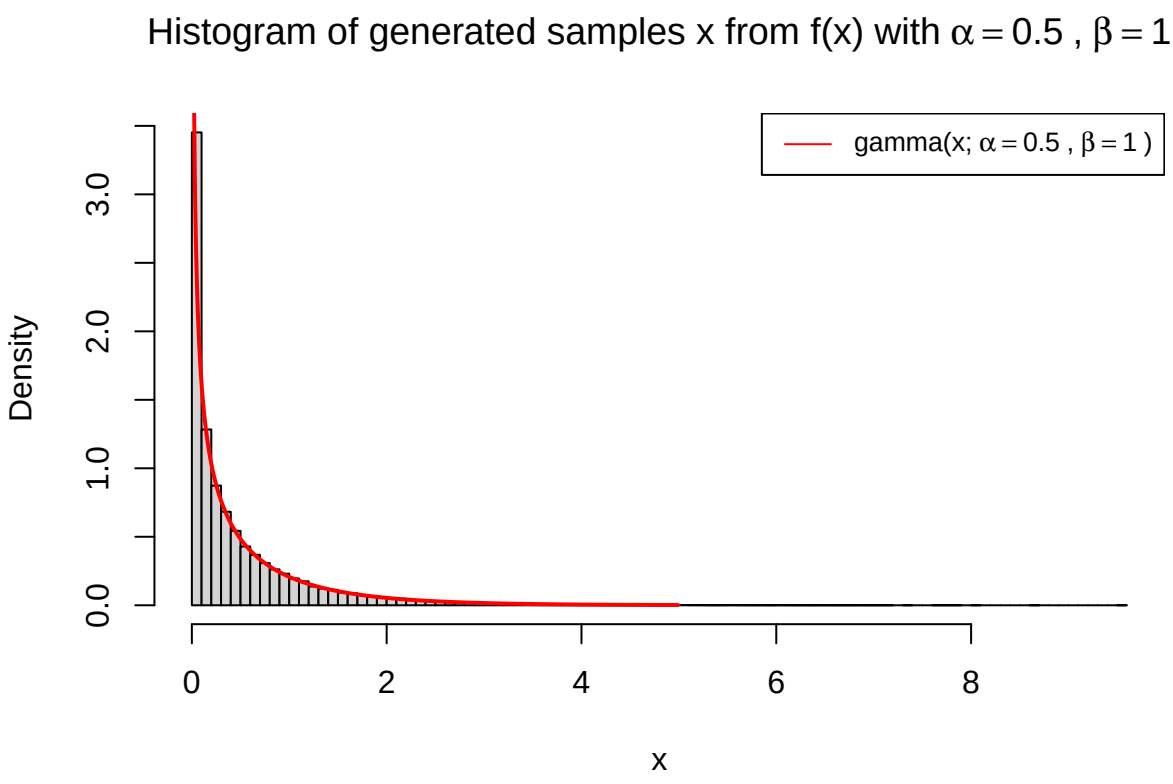
gives the highest acceptance probability for all x.

b) We now write the function `rejection_sampling1(alpha, n)` generating `n` independent samples from the distribution of $f(x)$.

```
rejection_sampling1 <- function(alpha, n){
  alpha = 0.5
  count = 0
  x = c(NA)
  while (count < n) {
    values = gsample(alpha, n)
    a = ifelse(values < 1, exp(-values), values^(alpha-1))
    u = runif(n)
    values = ifelse(u - a < 0, values, NA)
    x = c(x,values)
    x = na.omit(x)
    count = length(x)
  }
  x = x[1:n]
  return(x)
}
```

We ensure that the function is working properly comparing a histogram of its samples to the theoretical density function using parameter $\alpha = 0.5$ for the comparison.

```r
hist(rejection_sampling1(0.5, 100000),freq=F, n = 100,
     main = bquote("Histogram of generated samples x from f(x) with" ~ alpha == 0.5 ~ "," ~ beta == 1),
x <- seq(0.001, 5,.01)
lines(x,dgamma(x, 0.5,1), col = "red", lwd = 2)
legend("topright", legend=bquote("gamma(x;"~alpha == 0.5~ "," ~ beta == 1 ~")"),
       col="red", lty=1:2, cex=0.8)
```



B2

We now consider the gamma distribution with $\alpha > 1$ and $\beta = 1$. We define as in the lectures

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

and

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

such that

$$C_f \subset [0, a] \times [b_-, b_+].$$

a) We want the expressions for a, b_+ and b_-. We get the supremum of $f^*(x)$ for all x and $\alpha > 1$ by differentiating $f^*(x)$ with respect to x and setting the result equal to zero. This gives

$$\frac{df^*}{dx}(x) = x^{\alpha-1}e^{-x}\left(\frac{1}{x}(\alpha - 1) - 1\right) = 0$$

such that

$$x = \alpha - 1$$

maximizes $f^*(x)$ as $\frac{d^2 f^*}{dx^2}(\alpha - 1) < 0$ for $\alpha > 1$. By inserting this value of x into $f^*(x)$ we get its supremum, and a is then given by

$$a = \sqrt{\sup_x f^*(x)} = \sqrt{f^*(\alpha - 1)} = \sqrt{(\alpha - 1)^{\alpha - 1} e^{-(\alpha - 1)}} = \underline{\underline{(\alpha - 1)^{\frac{\alpha - 1}{2}} e^{-\frac{\alpha - 1}{2}}}}.$$

We find b_+ similarly as a, but in this case we want to maximize the function

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

Similar calculations and arguments as for a gives the maximum of $\tilde{f^*}(x)$ to be in

$$x = \alpha + 1,$$

which gives the expression for b_+

$$b_+ = \sqrt{\sup_{x \geq 0} x^2 f^*(x)} = \sqrt{\sup_{x \geq 0} \tilde{f^*}(x)} = \sqrt{(\alpha + 1)^2 f^*(\alpha + 1)} = \sqrt{(\alpha + 1)^{\alpha + 1} e^{-(\alpha + 1)}} = \underline{\underline{(\alpha + 1)^{\frac{\alpha + 1}{2}} e^{-\frac{\alpha + 1}{2}}}}.$$

Since $f^*(x) = 0$ for all $x \leq 0$ when $\alpha > 1$,

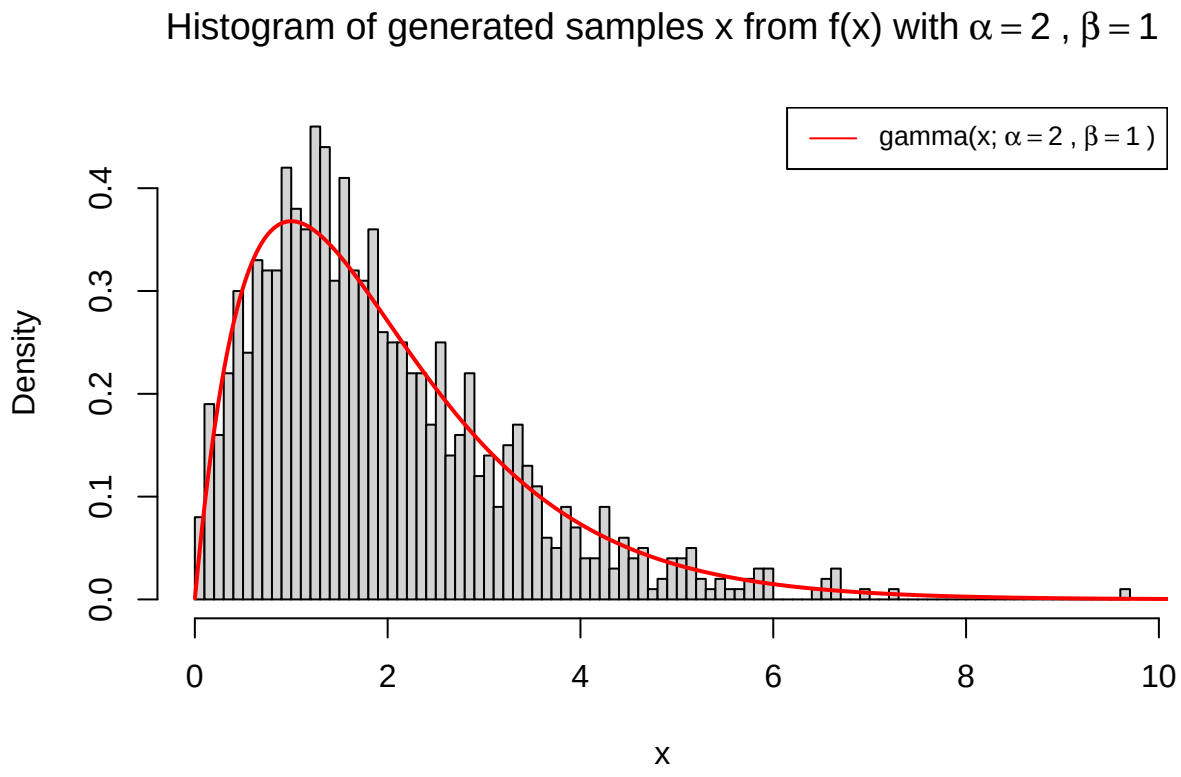
$$b_- = \sqrt{\sup_{x \leq 0} x^2 f^*(x)} = \sqrt{0} = \underline{0}.$$

b) We now use the information from a) to generate samples from the gamma distribution with $\alpha > 0, \beta = 0$ using the ratio of uniforms method. We use a log-transform during the simulation to avoid too large values of a and b_+ for large α. As well, we count the number of trials used to generate `n` samples.

```
ratio_of_uniforms_f <- function(alpha, n, beta = 1){
  loga = ((alpha - 1)/2)*log((alpha-1)/beta) + (1-alpha)/2
  bmin = 0
  logbmax = ((alpha + 1)/2)*log((alpha+1)/beta) - (alpha + 1)/2
  x = c()
  attempts = 0
  count = 0
  while(n > count) {
    logx1 = loga + log(runif(n-count,0,1))
    logx2 = logbmax + log(runif(n-count,0,1))
    y = exp(logx2 - logx1)
    z = ifelse(logx1 - (1/2)*((alpha - 1)*log(y) - beta*y)< 0, y, NA)
    z = na.omit(z)
    x = c(x,z)

    attempts = attempts + (n- length(x))

    count = length(x)
  }
  return(list(x = x, trials = n + attempts))
}
```

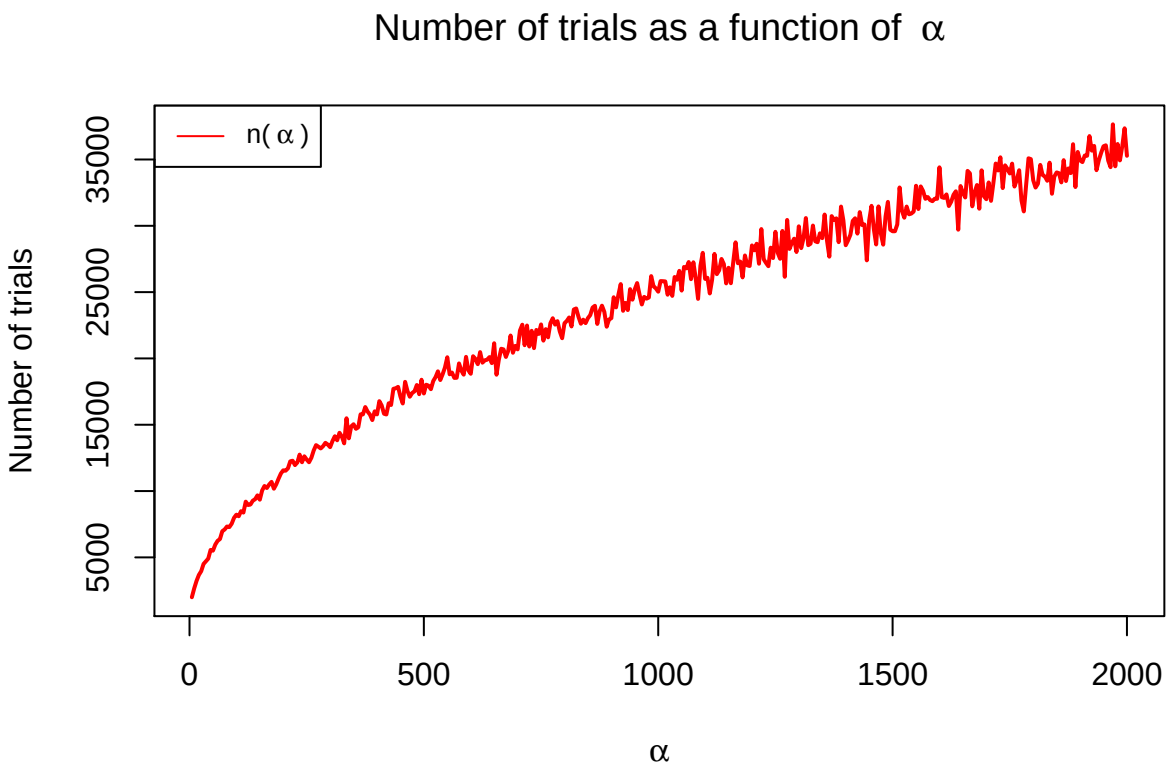
We ensure that the function is working properly comparing a histogram of its samples to the theoretical density function using parameter $\alpha = 2$ for the comparison.

```r
hist(ratio_of_uniforms_f(2, 1000)$x,freq=F, n = 100,
     main = bquote("Histogram of generated samples x from f(x) with" ~ alpha == 2 ~ "," ~ beta == 1), xl
x <- seq(0.001, 12,.01)
lines(x,dgamma(x, 2,1), col = "red", lwd = 2)
legend("topright", legend=bquote("gamma(x;"~alpha == 2~ "," ~ beta == 1 ~")"),
       col="red", lty=1:2, cex=0.8)
```



We now plot the number of trials used to generate n samples as a function of α.

```r
alphavec = seq(5,2000, 5)
trialvec = rep(NA,length(alphavec))
for(alpha_i in 1:length(alphavec)){
  trialvec[alpha_i] = ratio_of_uniforms_f(alphavec[alpha_i], 1000)$trials
}
plot(x = alphavec, y = trialvec, type = "l", col = "red", lwd = 2,
     main = bquote("Number of trials as a function of " ~ alpha),
     xlab = bquote(alpha), ylab = "Number of trials")
legend("topleft", legend=bquote("n(" ~ alpha ~ ")"),
       col="red", lty=1:2, cex=0.8)
```



We observe that the number of trials increases logarithmically as a function of increasing α. This can be interpreted as that the rectangle we make around the density function gets logarithmically larger compared to the density function itself as α increases.

B3

We will now write a function generating samples from a gamma distribution both given the shape parameter α and the scale parameter $\frac{1}{\beta}$, where

$$f(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}.$$

We approach this problem by rewriting the functions from B2 and B3 including the inverted scale parameter β, and deciding when to use the `rejection_sampling()` ($\alpha < 1$), when to use `ratio_of_uniforms()` ($\alpha > 1$), and what to do when $\alpha = 1$. This all begins in the `gsample()` function, where we need now to consider

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

where other terms with α and β independent of x are contained in c. Then

$$G(x) = \begin{cases} 0, & x < 0, \\ \frac{c}{\alpha} x^\alpha, & 0 < x < 1, \\ -\frac{c}{\beta} e^{-\beta x} + \frac{c}{\alpha} + \frac{c}{\beta} e^{-\beta}, & x \geq 1 \end{cases} \implies G^{-1}(x) = \begin{cases} 0, & x < 0, \\ (\frac{\alpha}{c} x)^{\frac{1}{\alpha}}, & 0 < x < \frac{c}{\alpha}, \\ -\frac{1}{\beta} \log(\frac{\beta}{\alpha} + e^{-\beta} - \frac{x\beta}{c}), & \frac{c}{\alpha} \leq x \leq 1 \end{cases}.$$

As the CDF sums to 1, an appropriate normalizing constant c in `gsample()` is then

$$c = \frac{1}{\frac{1}{\beta}e^{-\beta} + \frac{1}{\alpha}}.$$

For the new `rejection_sampling()`, we need a new acceptance probability, given by

$$\frac{1}{c}\frac{f(x)}{g(x)} = \begin{cases} \frac{\beta^\alpha}{c\cdot\Gamma(\alpha)}e^{-\beta x}, & 0 < x < 1, \\ \frac{\beta^\alpha}{c\cdot\Gamma(\alpha)}x^{\alpha-1}, & 1 \leq x, \\ 0, & \text{otherwise} \end{cases}$$

where the density $g(x)$ is defined as

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

, and c in this case

$$c = \frac{\beta^\alpha}{\Gamma(\alpha)}$$

yelds the highest acceptance probability in the `rejection_samping2(alpha, beta, n)` algorithm. We now update `gsample(alpha, n)` to `gsample1(alpha, beta, n)` and then `rejection_sampling1(alpha, n)` to `rejection_sampling2(alpha, beta, n)` as follows.

```
gsample1 <- function(alpha, beta, n){
  c = 1/((1/beta)*exp(-beta) + (1/alpha))
  u = runif(n)
  threshold = c/alpha
  x = ifelse(u < threshold, (alpha*u/c)^(1/alpha),
             -(1/beta)*log((beta/alpha) + exp(-beta) - (u*beta/c)))
  return(x)
}

rejection_sampling2 <- function(alpha, beta, n){
  count = 0
  x = c()
  while (count < n) {
    values = gsample1(alpha, beta, n)
    a = ifelse(values < 1, exp(-beta*values), values^(alpha-1))
    u = runif(n)
    values = ifelse(u - a < 0, values, NA)
    x = c(x,values)
    x = na.omit(x)
    count = length(x)
  }
  x = x[1:n]
  return(x)
}
```

For $\alpha > 1$, we consider the `ratio_of_uniforms_f(alpha, n)` function and adjust it to work with a predefined β. Doing similar prior calculations as above, including β gives

$$a = \left(\frac{\alpha-1}{\beta}\right)^{\frac{\alpha-1}{2}} e^{-\frac{\alpha-1}{2}}, b_+ = \left(\frac{\alpha+1}{\beta}\right)^{\frac{\alpha+1}{2}} e^{-\frac{\alpha+1}{2}}, b_- = 0,$$

such that our new `ratio_of_uniforms1(alpha, beta, n)` function becomes as follows.

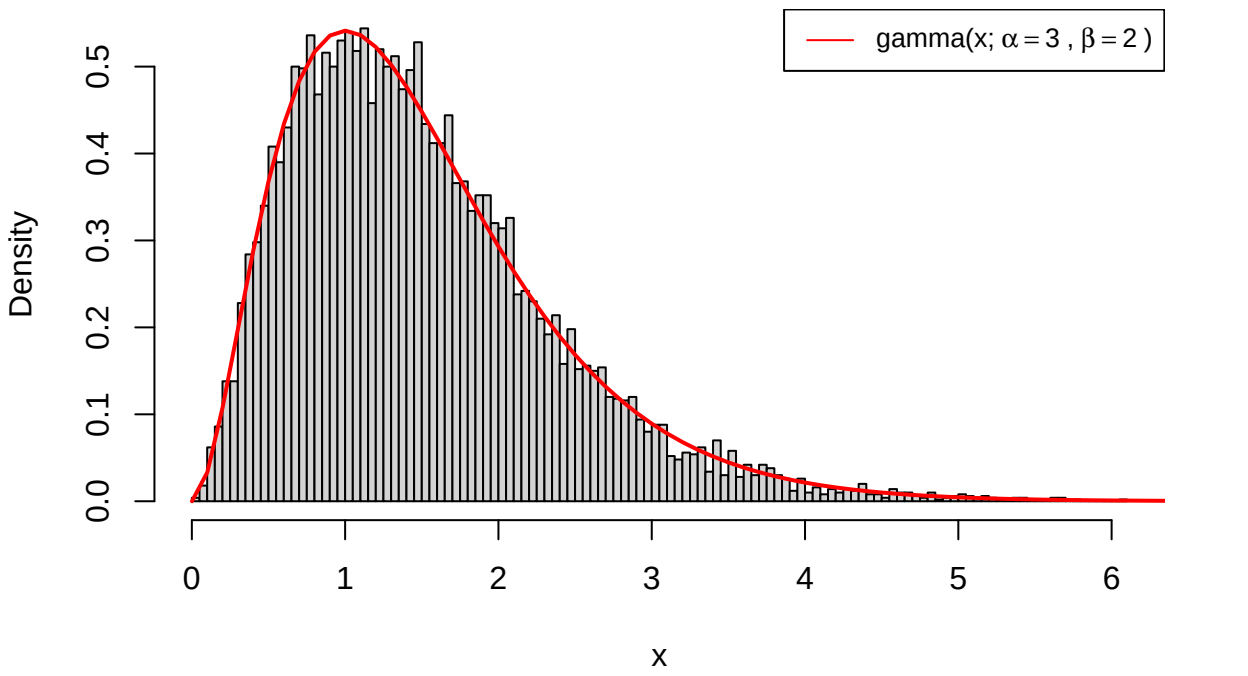
```
ratio_of_uniforms1 <- function(alpha, beta, n){
  loga = ((alpha - 1)/2)*log((alpha-1)/beta) + (1-alpha)/2
  bmin = 0
  logbmax = ((alpha + 1)/2)*log((alpha+1)/beta) - (alpha + 1)/2
  x = c()
  attempts = 0
  count = 0
  while(n > count) {
    logx1 = loga + log(runif(n-count,0,1))
    logx2 = logbmax + log(runif(n-count,0,1))
    y = exp(logx2 - logx1)
    z = ifelse(logx1 - (1/2)*((alpha - 1)*log(y) - beta*y)< 0, y, NA)
    z = na.omit(z)
    x = c(x,z)

    attempts = attempts + (n- length(x))

    count = length(x)
  }
  return(list(x = x, trials = n + attempts))
}
```

For $\alpha = 1$, we are only left with the exponential function with parameter β as the factor $x^{\alpha-1}$ in the gamma distribution becomes 1, and we simply sample from the exponential distribution. Thus, we create the general function `gamma_sample(alpha, beta, n)` as follows.

```
gamma_sample <- function(alpha, beta, n){
  if(alpha < 1){x = rejection_sampling2(alpha, beta, n)}
  if(alpha == 1){x = expsample(beta, n)}
  if(alpha > 1){x = ratio_of_uniforms1(alpha, beta,n)$x}
  return(x)
}
```

We ensure that the function is working properly comparing a histogram of its samples to the theoretical density function using parameters $\alpha = 3, \beta = 2$ for the comparison.

```
hist(gamma_sample(3,2,10000), n = 100, freq = F,
     main = bquote("Histogram of generated samples x from a gamma("~alpha==3~","~beta==2~") distributio
x <- seq(0.001, 17,.1)
lines(x,dgamma(x, 3,rate = 2), col = "red", lwd = 2)
legend("topright", legend=bquote("gamma(x;"~alpha == 3~ "," ~ beta == 2 ~")"),
       col="red", lty=1:2, cex=0.8)
```



B4

We let $x \sim Gamma(\alpha, 1)$, $y \sim Gamma(\beta, 1)$ and $z = \frac{x}{x+y}$.

a) We will now show that $z \sim Beta(\alpha, \beta)$. The joint PDF of x and y is then

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

We then introduce the transformations

$$u = \frac{x}{x+y} \text{and} v = x + y.$$

Calculating the inverse of these gives

$$x = uv \text{and} y = v - uv.$$

The determinant of the Jacobian matrix yelds $v(1-u) + uv = v = x + y$. This gives the joint PDF of u and v as

$$f(u, v) = |v| \frac{1}{\Gamma(\alpha)\Gamma(\beta)} (uv)^{\alpha-1} (v - uv)^{\beta-1} e^{-v},$$

for $0 < u < 1, 0 < v < \infty$. Setting $z = u$, we get the marginal pdf of z, $f(z)$, by integrating the joint PDF above with respect to v.

$$f(z) = \frac{1}{\Gamma(\alpha)\Gamma(\beta)} z^{\alpha-1} (1-z)^{\beta-1} \int_0^\infty v^{\alpha+\beta-1} e^{-v} dv = \frac{1}{\Gamma(\alpha)\Gamma(\beta)} z^{\alpha-1} (1-z)^{\beta-1},$$

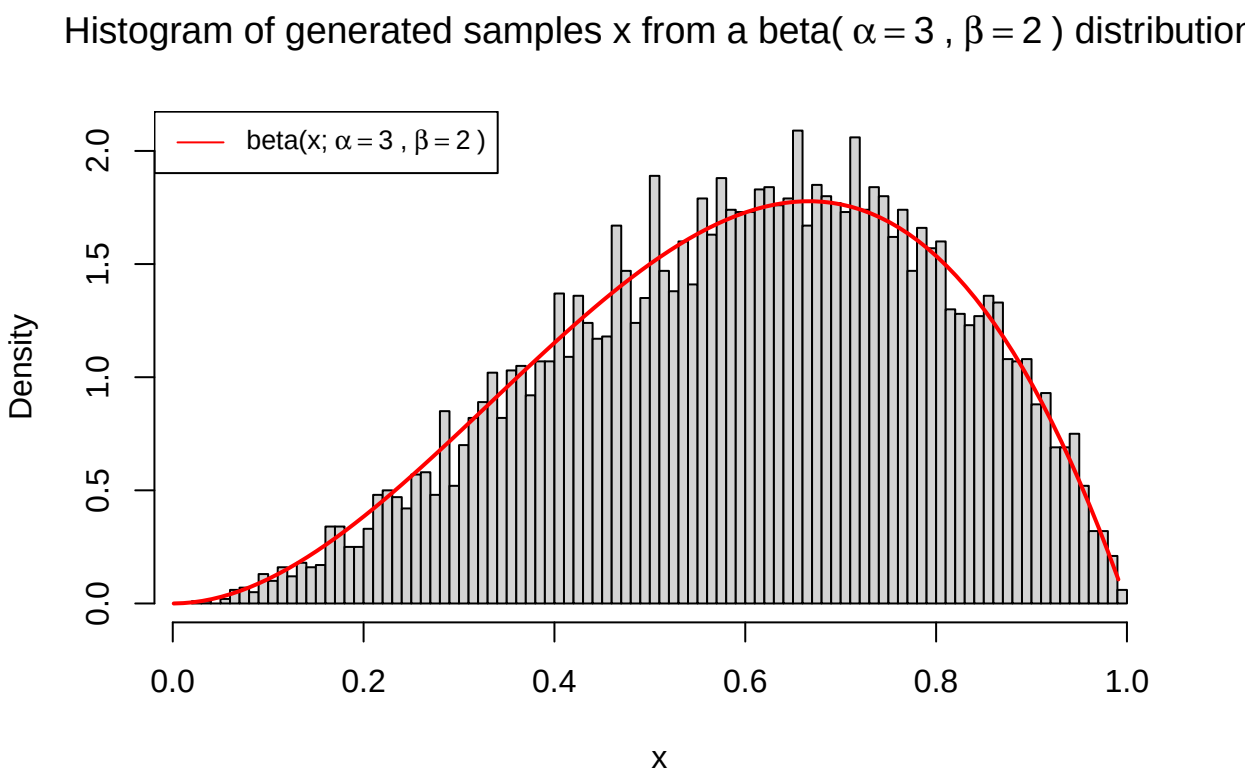
for $0 < z < 1$, which we wanted to show.

b) We now simulate from the beta distribution by the function `beta_sample(alpha, beta, n)`, first by sampling from two gamma distributions to two samples `x` and `y`, and then calculate the sample of `z` which, as shown above, is beta distributed with parameters α and β.

```
beta_sample <- function(alpha, beta, n){
  x <- gamma_sample(alpha, 1, n)
  y <- gamma_sample(beta, 1, n)
  z = x/(x+y)
  return(z)
}
```

We ensure that the function is working properly comparing a histogram of its samples to the theoretical density function using parameter $\alpha = 3, \beta = 2$ for the comparison.

```
hist(beta_sample(3,2,10000), n = 100, freq = F,
     main = bquote("Histogram of generated samples x from a beta("~alpha==3~","~beta==2~") distribution
x <- seq(0.001, 1,.01)
lines(x,dbeta(x, 3,2), col = "red", lwd = 2)
legend("topleft", legend=bquote("beta(x;"~alpha == 3~ "," ~ beta == 2 ~")"),
       col="red", lty=1:2, cex=0.8)
```



Problem C:

C1

Let $\theta = P(X > 4)$ when $X \sim N(0, 1)$. Using Monte-Carlo integration to calculate θ, we get that the estimate $\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} h(x_i)$ where

$$h(x) = \begin{cases} 1, & x > 4, \\ 0, & \text{otherwise} \end{cases}.$$

We then derive the following estimate of θ and its confidence interval, assuming central limit theorem holds in this case.

```
n <- 100000
X <- box_muller(n)$x1
H <- ifelse(X < 4, 0, 1)

#the expected value

theta_hat <- mean(H)

#the variance
var_theta_hat <- var(H)/n

#the confidence interval, assuming the central limit theorem
data <- data.frame("ThetaHat" = theta_hat, "ConfIntLower" = theta_hat-1.96*sqrt(var_theta_hat),
                   "ConfIntUpper" = theta_hat+1.96*sqrt(var_theta_hat))
row.names(data) <- "MC integration"
print(data)
```

```
##                ThetaHat ConfIntLower ConfIntUpper
## MC integration    1e-05     -9.6e-06     2.96e-05
```

```
#matrix created for a plot later.
res <- matrix(nrow = 3, ncol =2)
res[1,1] <- 1-pnorm(4)
res[2,1] <- theta_hat
res[2,2] <- var_theta_hat
```

Note that the confidence interval is big and that the lower bound can be negative, which contradicts the fact that all probabilities are within 0 and 1.

C2

We now use importance sampling to estimate θ via sampling from

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

where c is a normalizing constant.The normalizing constant is found by the integral

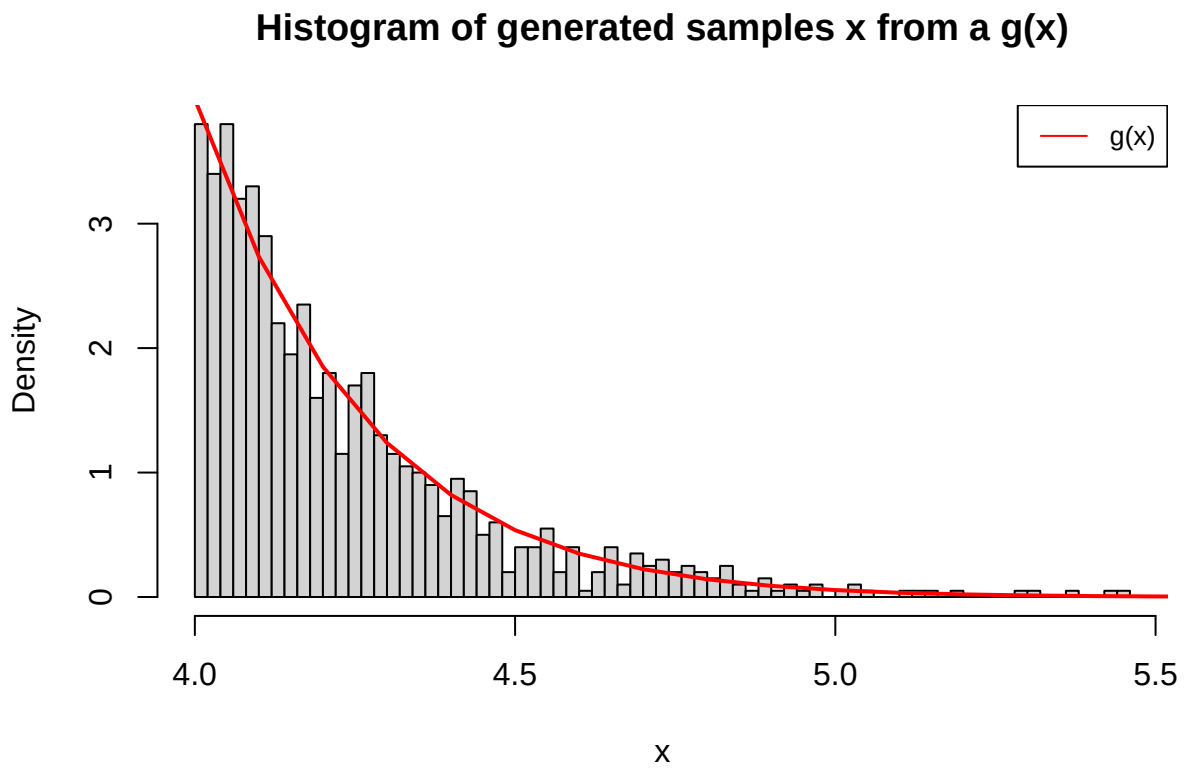
$$\int_X g(x)dx = \int_4^\infty cx\exp(-\frac{1}{2}x^2)dx = 1,$$

which yelds $c = \exp(8)$. First, a function that samples from `g(c, n)` is constructed.

```r
#sampling from g(x)
g <- function(c=exp(8), n=1000) {
  u <- runif(n, 0,c*exp(-8))
  x <- sqrt(-2*log(-u/c+exp(-8)))
  return(x)
}
```

We ensure that the sampling works properly by comparing a histogram of the samples to the theoretical PDF.

```r
hist(g(), n = 100, freq = F,
     main = bquote("Histogram of generated samples x from a g(x)"), xlab="x")
gfunc = function(x){exp(8)*x*exp(-(1/2)*x^2)}
x = seq(4,6,0.1)
lines(x, gfunc(x), type = "l", col = "red", lwd = 2)
legend("topright", legend="g(x)",
       col="red", lty=1:2, cex=0.8)
```



The weight function is given by

$$w(x) = \frac{f(x)}{g(x)} = \begin{cases} \frac{e^{-8}}{x\sqrt{2\pi}}, & x > 4, \\ 0, & \text{otherwise} \end{cases}.$$

The imporance sampling estimator is then defined by

$$\hat{\theta}_{IS} = \frac{1}{n}\sum_{i=1}^{n} h(x_i)w(x_i) = \frac{1}{n}\sum_{i=1}^{n} w(x_i), x_i > 4,$$

where the variance is given by

$$\hat{Var}[\hat{\theta}_{IS}] = \frac{1}{n(n-1)} \sum_{i=1}^{n} (w(x_i) - \hat{\theta}_{IS})^2.$$

We then implement the following function for the importance sampling.

```
importance_sampling <- function(n=100000) {
  x <- g(n)
  c <- exp(8)
  w <- 1/(c*x*sqrt(2*pi))
  theta_is_hat_exp <- mean(w)
  theta_is_hat_var <- var(w)/n
  return(list(x = x, theta_is_hat_exp = theta_is_hat_exp, theta_is_hat_var = theta_is_hat_var))
}
r <- importance_sampling()
res[3,1] <- r$theta_is_hat_exp
res[3,2] <- r$theta_is_hat_var
```

Assuming the central limit theorem holds in this case, we get the following mean and corresponding confidence interval.

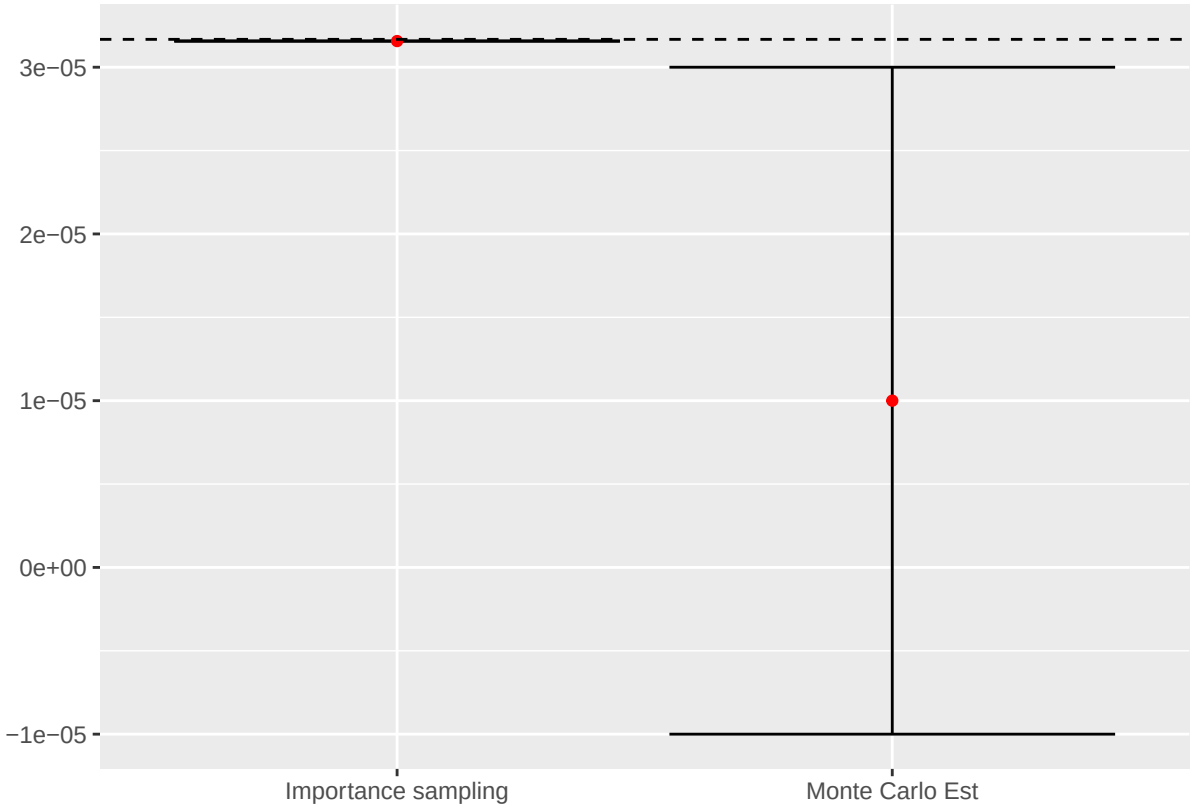
```
#the confidence interval, assuming the central limit theorem

d <- data.frame("ThetaHat" = res[3,1], "ConfIntLower" = res[3,1]-1.96*sqrt(res[3,2]),
                "ConfIntUpper" = res[3,1]+1.96*sqrt(res[3,2]))
row.names(d) <- "Importance sampling"
print(d)
```

```
##                           ThetaHat ConfIntLower ConfIntUpper
## Importance sampling 3.156763e-05 3.155782e-05 3.157744e-05
```

```
data <- rbind(data, d)
```

Above is the expected value and confidence interval from the importance sampling. Clearly importance sampling gives a more accurate result with less variance then by Monte Carlo integration. Below the differences are visualized.

```
library(tidyverse)

data.frame(model = c("Monte Carlo Est", paste("Importance sampling" )),
           mean = res[-1,1], sd =  sqrt(res[-1,2])) %>%
  mutate(lower = mean-2*sd, upper = mean+2*sd) %>%
  ggplot() + geom_errorbar(aes(x = model, ymin = lower, ymax = upper)) +
  geom_point(aes(x = model, y = mean), color = "red") +
  geom_hline(yintercept = res[1,1], lty = 2) +
  xlab("") + ylab("")
```



The relative error from importance sampling is as follows.

```
print(abs(res[1,1] - res[3,1])/res[1,1])
```

```
## [1] 0.003271536
```

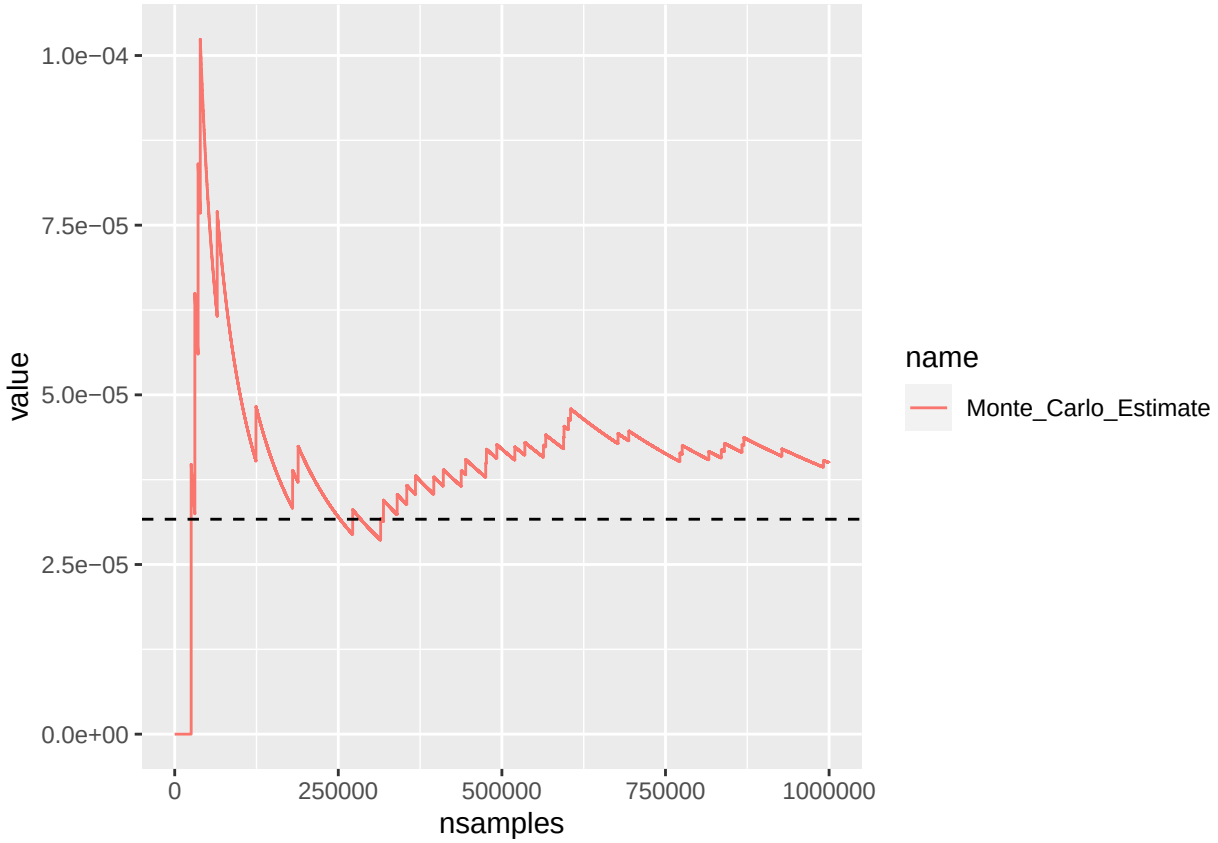
To be conservative the average relative error from importance sampling is of order 10^{-3}. To get an error of the same order for Monte-Carlo integration we need at least 10^3 samples where $x > 4$. Since

$$\theta \approx 3 \cdot 10^{-5}$$

we need about $N = 10^3 \cdot 10^5$ samples to get the same accuracy as with importance sampling.

Below is an illustration of how the Monte Carlo integration estimate converges towards the true θ as `n` increases.

```
n <- 1000000
x <- box_muller(n)$x1
H <- ifelse(x>4, 1, 0)
Monte_Carlo_Estimate <- cumsum(H)/1:n

data.frame(Monte_Carlo_Estimate = Monte_Carlo_Estimate ,
           nsamples = seq_along(Monte_Carlo_Estimate)) %>%
  pivot_longer(-nsamples) %>%
  ggplot()+geom_line(aes(x = nsamples, y = value,group = name, color = name))+
  geom_hline(yintercept = res[1,1], lty = 2)
```



It would be unneccesarily computer intensive to run 10^8 samples, but the slow convergence of the Monte-Carlo integration is however clearly illustrated by $\texttt{n} = 10^6$.

C3

a) Now the effect of using a modified version of the importance sampling function in C2 will be explored. Instead of generating n random samples from a $U \sim Uniform(0,1)$, we now generate the first half from the uniform distribution and the other half by $1-u$. These values are now used to sample from $g(x)$. The function is implemented as `g2(c, n)` below.

```
g2 <- function(c=exp(8), n=1000) {
  u <- runif(n, 0,1)
  u2 <- c(u,1-u)

  x <- sqrt(-2*log(-u2/c+exp(-8)))
  return(x)
}
```

b) The importance sampling conducted with the samples from the new function $g2()$ is implemented below. We use 50000 samples to get a fair comparison to the function in a) as that function uses pairs of 50000 samples which gives 100000 in total.

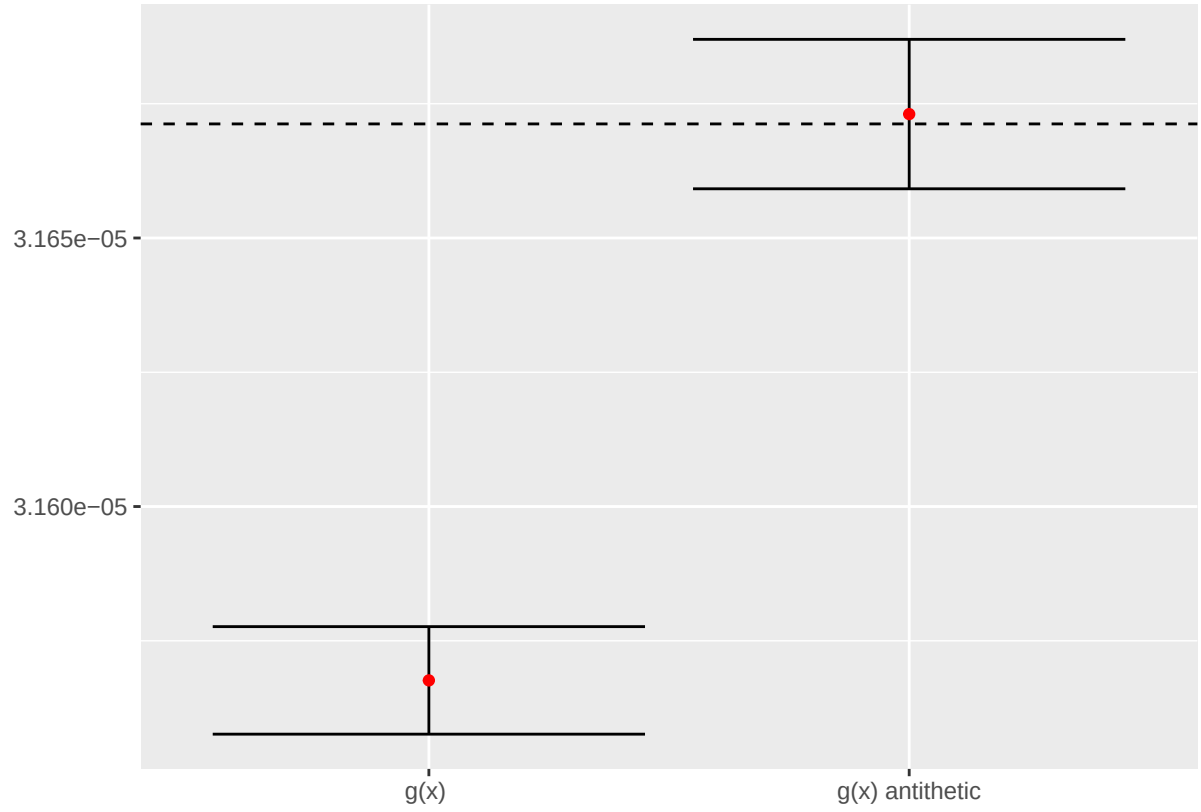
```
importance_sampling2 <- function(n = 50000) {
  x <- g2(n = n)
  c <- exp(8)
```

```
  w <- 1/(c*x*sqrt(2*pi))
  theta_is_hat_exp <- mean(w)
  theta_is_hat_var <- var(w)/n
  return(list(x = x, theta_is_hat_exp2 = theta_is_hat_exp, theta_is_hat_var2 = theta_is_hat_var))
}


ret <- importance_sampling2()

res2 <- matrix(nrow = 3, ncol = 2)
res2[2,] <- res[3,]
res2[1,1] <- res[1,1]
res2[3,1] <- ret$theta_is_hat_exp2
res2[3,2] <- ret$theta_is_hat_var2

d <- data.frame("ThetaHat" = res2[3,1], "ConfIntLower" = res2[3,1]-1.96*sqrt(res2[3,2]),
                "ConfIntUpper" = res2[3,1]+1.96*sqrt(res2[3,2]))
row.names(d) <- "Importance sampling antithetic"
print(d)
```

```
##                                    ThetaHat ConfIntLower ConfIntUpper
## Importance sampling antithetic 3.167307e-05 3.165944e-05 3.168671e-05
```

Below we see that the antithetic simulation gives the antithetic sampling is more precise than the earlier sampling, but it has a larger confidence interval. This can be a result of a positive correlation between the pair of samples.

```
data.frame(model = c("g(x) ", paste("g(x) antithetic" )),
           mean = res2[-1,1], sd =  sqrt(res2[-1,2])) %>%
  mutate(lower = mean-2*sd, upper = mean+2*sd) %>%
  ggplot() + geom_errorbar(aes(x = model, ymin = lower, ymax = upper)) +
  geom_point(aes(x = model, y = mean), color = "red") +
  geom_hline(yintercept = res[1,1], lty = 2) +
  xlab("") + ylab("")
```



Problem D:

D1

We are now going to sample from a given posterior density

$$f(\theta|\mathbf{y}) \propto (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4} \text{for} \theta \in (0,1),$$

where $\mathbf{y}$ is a predefined vector $\mathbf{y} = [125, 18, 20, 34]^T$. Using a $U(0,1)$ as a proposal distribution, we create a rejection sampling algorithm to sample from $f(\theta|\mathbf{y})$ as follows. Prior the function we define the PDF $f(\theta|\mathbf{y})$ we are sampling from and its maximum which we use as a normalizing constant.

```
f <- function(x){(2+x)^125 * (1-x)^38 * x^34}

c <- max(f(seq(0,1,0.001)))

rejection_sampling3 <- function(n){
  count = 0
  fails = 0
  x = c()
  while (count < n) {
    values = runif(n - count)
    a = (1/c)*f(values)
    u = runif(n - count)
    values = ifelse(u - a < 0, values, NA)
```

```
    x = c(x,values)
    lenx = length(x)
    x = na.omit(x)
    count = length(x)
    fails = fails + (lenx-count)
  }
  return(list(x=x, trials = n + fails))
}
```

D2

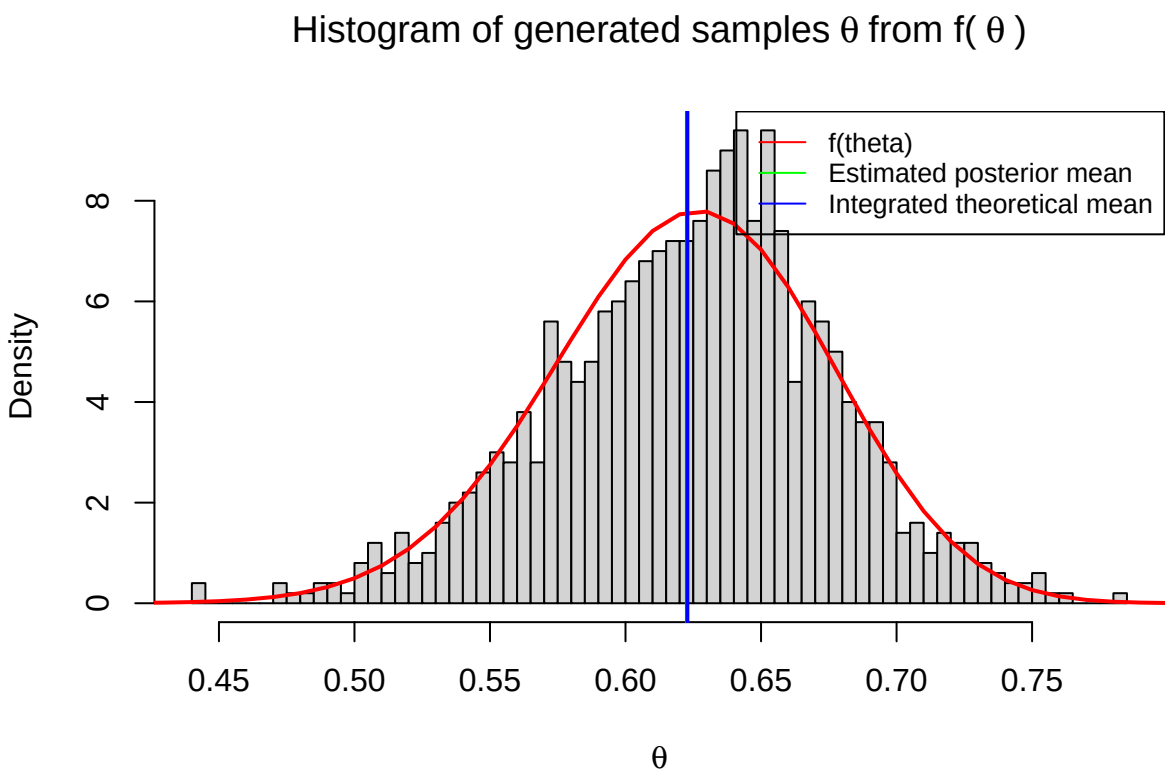
We now want to estimate a mean using Monte-Carlo integration using 10000 samples from $f(\theta|\mathbf{y})$. We compare it to the numerical mean found by integrating our function.

```
that <- (1/10000)*sum(rejection_sampling3(10000)$x) #est. mean

fint <- integrate(f, 0,1)$value
integrand = function(x){x*f(x)/fint}
numeric_mean = integrate(integrand, 0,1)$value #num. mean

print(data.frame("Estimated_mean" = that, "Numerical_mean" = numeric_mean,
                 "Absolute_difference" = abs(that - numeric_mean)))
```

```
##   Estimated_mean Numerical_mean Absolute_difference
## 1      0.6226659      0.6228061        0.0001402547
```

We now assess the sampling result comparing it to the theoretical density function. We plot as well a green line for the estimated posterior mean and a blue line for the numerical mean.

```
hist(rejection_sampling3(1000)$x, n = 50, freq = F,
     main = bquote("Histogram of generated samples" ~ theta  ~"from f(" ~ theta ~ ")"), xlab=bquote(the
x = seq(0,1,0.01)
lines(x,f(x)/fint, col = "red", lwd = 2)
abline(v = that, col = "green", lwd = 2)
abline(v = numeric_mean, col = "blue", lwd = 2)
legend("topright", legend=c(bquote(f(theta)),
                            "Estimated posterior mean", "Integrated theoretical mean"),
       col=c("red", "green", "blue"),lty = 1, cex=0.8)
```



Note that the difference in estimated and theoretical mean is barely noticeable.

D3

We have counted number of trials needed to generate `n` samples in our `rejection_sampling3()` function. Thus the number of numbers to generate one sample is the number of trials devided by `n`. The theoretical number is the integral of the proposal distribution, which is a uniform, devided by the area of our target distribution $f(x)$, as we sample uniformly over the whole proposal and only keep the samples also contained in the target distribution. Then we get as follows.

```
nTrialsPerSample = rejection_sampling3(1000)$trials/1000
theoreticalNTrials = c/fint
print(data.frame("Calculated_number_of_trials_per_sample" = nTrialsPerSample,
                 "Theoretical_number_of_trials_per_sample" = theoreticalNTrials))
```

```
##   Calculated_number_of_trials_per_sample
## 1                                  7.879
##   Theoretical_number_of_trials_per_sample
## 1                                7.799261
```

D4

We now assume that the prior distribution for θ is a Beta(1,5) distribution. We now want to use importance sample weights to estimate the posterior mean given the new prior based on samples from D2. As we don't

know the normalizing constant, we use the self-normalizing importance sapmling, where

$$\hat{\mu} = \frac{\sum h(\theta_i)w(\theta_i)}{\sum w(\theta_i)},$$

where

$$h(\theta) = \theta, \theta \in (0,1),$$

and the weight function $w(\theta)$ is the Beta(1,5) without constants, i. e.

$$w(\theta) = (1-\theta)^4.$$

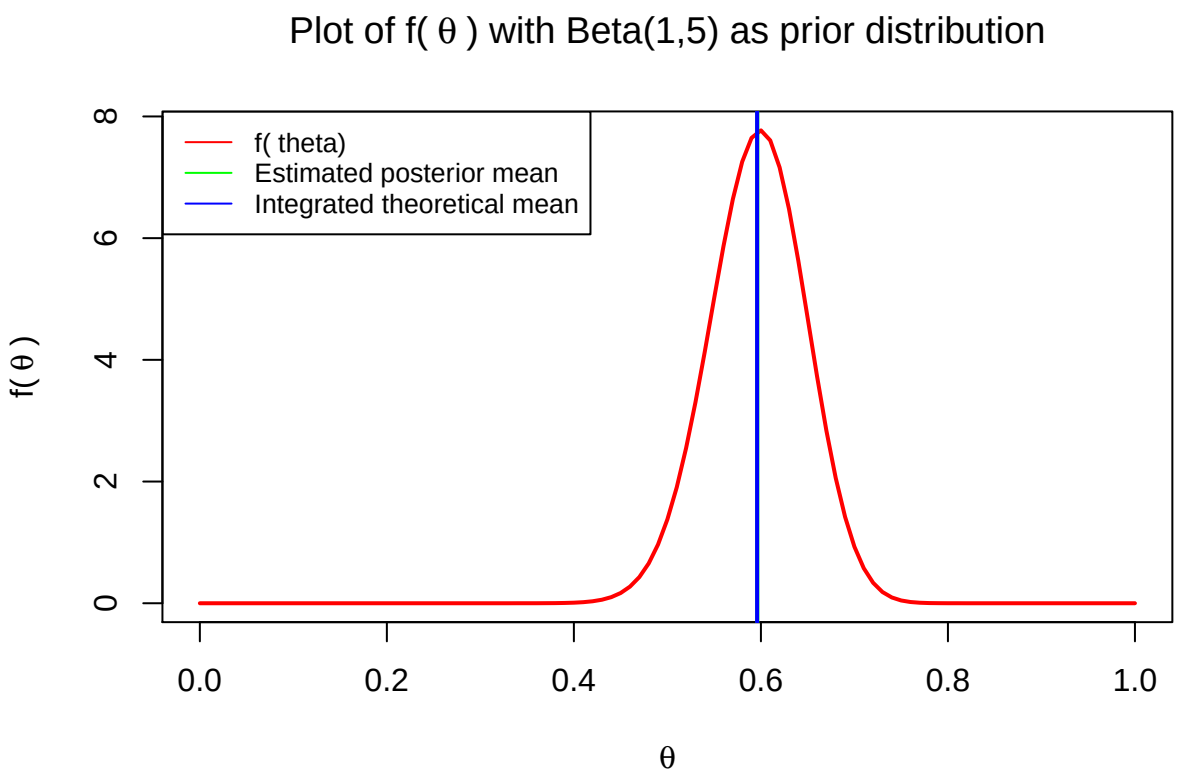
This was obtained by defining $g(\theta)$ as the posterior from D2 whith Beta(1,1) as prior distribution, and $f(\theta)$ as the posterior with Beta(1,5) as prior distribution. We found thus that

$$f(\theta) \propto (1-\theta)^4 g(\theta) \implies w(\theta) = \frac{f(\theta)}{g(\theta)} \propto (1-\theta)^4.$$

```
weight <- function(x) {(1-x)^4}

importance_sampling1 <- function(n){
  x <- rejection_sampling3(n)$x
  w <- weight(x)
  muhat <- sum(x*w)/sum(w)
  return(muhat)
}

meanest = importance_sampling1(100000)

fpost = function(x){5*(1-x)^4*f(x)}
fpostint = integrate(fpost, 0,1)$value

fpostmean = function(x){x*fpost(x)/fpostint}
meanint = integrate(fpostmean,0,1)$value

print(data.frame("Estimated_mean" = meanest, "Theoretical_mean" = meanint,
                 "Absolute difference" = abs(meanest - meanint)))
```

```
##   Estimated_mean Theoretical_mean Absolute.difference
## 1       0.596178        0.5959316        0.0002463616
```

```
plot(x, fpost(x)/fpostint, type = "l", lwd = 2, col = "red", main = bquote("Plot of f(" ~ theta  ~") wi
abline(v = meanest, col = "green", lwd = 2)
abline(v = meanint, lwd = 2, col = "blue")
legend("topleft", legend=c(bquote("f( theta)"), "Estimated posterior mean", "Integrated theoretical mea
       col=c("red", "green", "blue"),lty = 1, cex=0.8)
```



We observe that the theoretical mean and posterior mean are different, but not any more that we cant see any difference plotting them. We expect some difference as the self-normalized importance sampling is biased for finite n, which we got.