TMA4300 Computer Intensive Statistical Methods: Exercise 1

Group 6: Jim Totland, Martin Tufte

1/29/2022

Problem A

A.1

The exponential distribution has a cumulative density function given by

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

where λ is the rate parameter. By defining $u := F(x)$, x can be expressed as

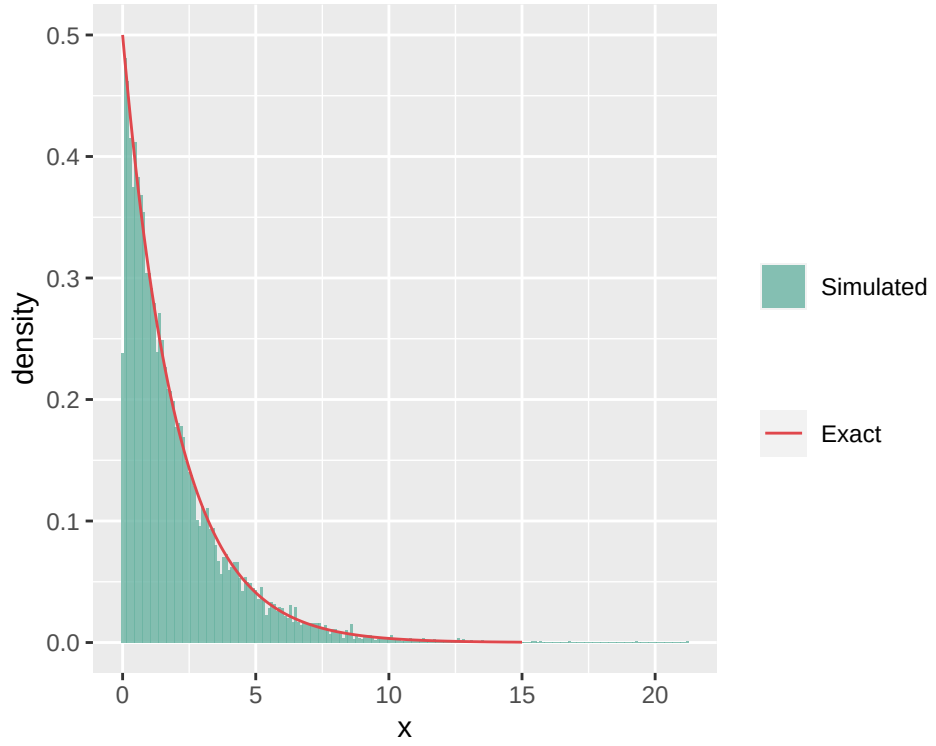
$$x = -\frac{1}{\lambda}\ln(1-u) =: F^{-1}(u).$$

This means that we can use the *inversion method* to simulate from the exponential distribution. Let $U \sim \mathcal{U}_{[0,1]}$ and calculate $X = F^{-1}(U)$, then $X \sim \text{Exp}(\lambda)$. A function which simulates from the exponential distribution is given below.

```
sim.exp <- function(rate, n){
  # Vector of n uniform values
  u <- runif(n, 0, 1)
  # Note that (1-U) ~ U, so we can use u instead of 1-u in the returned value.
  -1/rate * log(u)
}
```

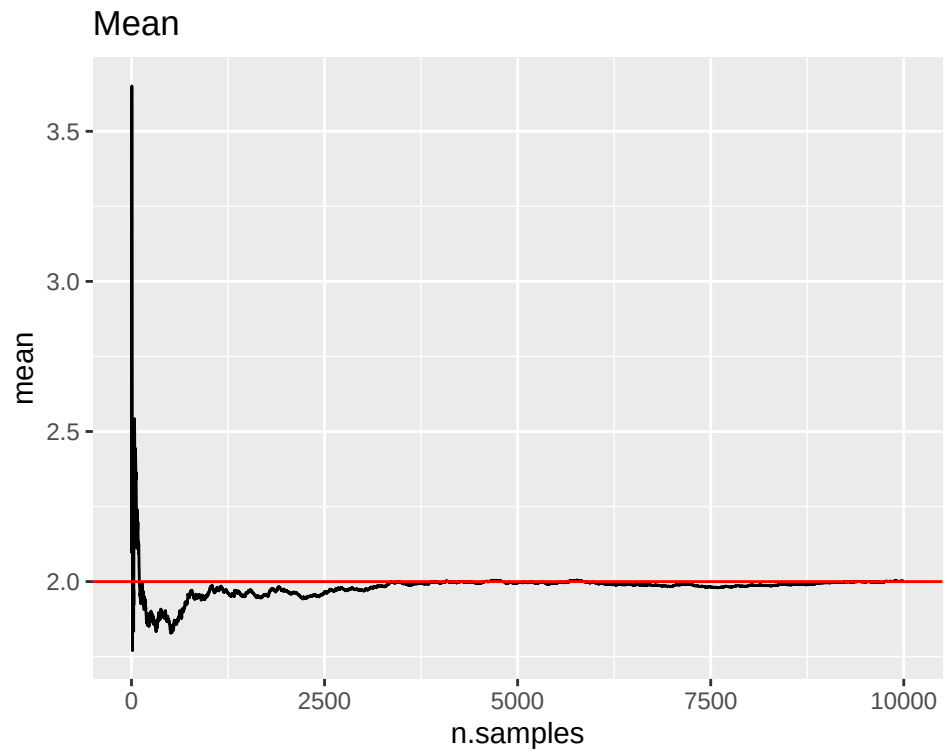
Next, we need to check if this gives reasonable results by comparing our simulated values to the theoretical knowledge using $\lambda = 0.5$ and $n = 1000$.

```
rate <- 0.5
n <- 10000
sim <- data.frame(x = sim.exp(rate, n))
x = seq(from = 0, to = 15, by = 0.1)
exact <- data.frame(x = x, y = rate*exp(-rate*x))

ggplot(sim) +
  geom_histogram(aes(x = x, y = ..density.., fill = "Simulated"),
                 alpha = 0.8, binwidth = 0.1) +
  geom_line(data = exact , aes(x = x, y = y, color = 'Exact'))  +
  scale_color_manual(name = "", values = c("Exact" = "#e0474c")) +
  scale_fill_manual(name = "", values = c("Simulated" = "#69b3a2"))
```

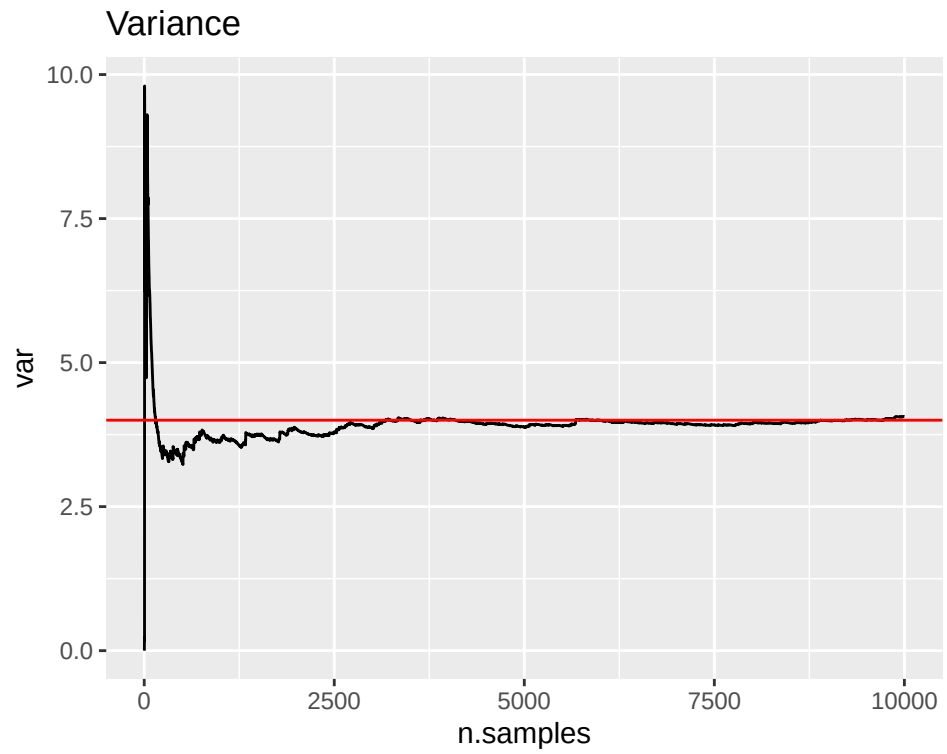


We also compare the estimated mean and variance to first and second central moments of the exponential distribution:

```
# Calculate the cumulative mean
data.frame(n.samples = 1:n,
           mean = cumsum(sim$x)/(1:n)) %>%
  ggplot() + geom_line(aes(n.samples, mean)) +
  geom_hline(yintercept = 1/rate, color = "red") +
  ggtitle("Mean")
```




```
# Calculate the cumulative variance
data.frame(n.samples = 1:n,
           mean = cumsum(sim$x)/(1:n),
           mean2 = cumsum(sim$x^2)/(1:n)) %>%
  mutate(var = mean2-mean^2) %>%
  ggplot() + geom_line(aes(n.samples, var)) +
  geom_hline(yintercept = 1/rate^2, color = "red") +
  ggtitle("Variance")
```



We observe that the the sampled mean and variance approach the theoretical values as the number of samples grows larger.

A.2

We consider the probability distribution given by

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

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

(a)

Normalizing the density using $\int_{-\infty}^{\infty} g(x)\ \mathrm{d}x = c\int_0^1 x^{\alpha-1}\ \mathrm{d}x + c\int_1^{\infty} e^{-x}\ \mathrm{d}x = c(\frac{1}{\alpha} + \frac{1}{e}) = 1$, it follows that the normalizing constant is $c = (\frac{1}{\alpha} + \frac{1}{e})^{-1}$. The cumulative distribution function is given by

$$G(x) = \int_{-\infty}^{x} g(y)\ \mathrm{d}y = \begin{cases} c\int_0^x y^{\alpha-1}\ \mathrm{d}y, & 0 < x < 1 \\ \frac{c}{\alpha} + c\int_1^x e^{-y}\ \mathrm{d}y, & 1 \leq x \\ 0, & \text{otherwise} \end{cases},$$

which evaluates to

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

The inverse is the unique function G^{-1} such that $x = G^{-1}(u)$. Since $G(x)$ is piecewise continuous and monotonic, we can calculate the inverses for its constituencies and find appropriate intervals. The intervals

are simply found from calculating the corresponding boundary points. In $G(x)$ one of these points is located at $x = 1$, meaning $G(1) = \frac{c}{\alpha}$ will separate two continuous parts in G^{-1}. After solving for u, we end up with the inverse given by

$$G^{-1}(u) = \begin{cases} \left(\frac{\alpha u}{c}\right)^{-\alpha}, & 0 \leq u < \frac{c}{\alpha} \\ -\ln\left(\frac{1-u}{c}\right), & \frac{c}{\alpha} \leq u \end{cases}.$$

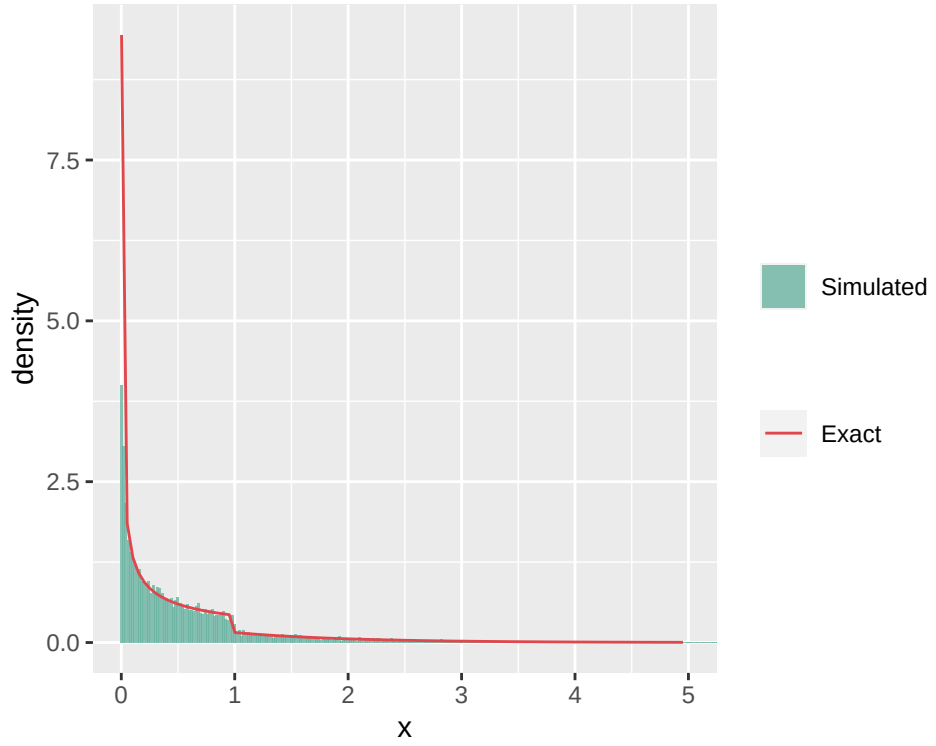
(b)

Firstly, we implement g and an algorithm to generate samples from g:

```
g <- function(alpha, x){
  # Normalizing constant
  c <- (1/alpha + exp(-1))^-1
  c*x^(alpha-1) * (0<x)*(x<1) + c*exp(-x) *(x>=1)
}

sim.g <- function(alpha, n){
  # Normalizing constant
  c <- (1/alpha + exp(-1))^-1
  # Vector of n uniform values
  u <- runif(n, 0, 1)

  (alpha/c * u)^(1/alpha) * (u < c/alpha) - log((1-u)/c) * (c/alpha <= u)
}
```

To test our implementation we use $\alpha = 0.5$ and simulate $n = 10000$ samples.

```
alpha <- 0.5
n <- 10000
sim <- data.frame(x = sim.g(alpha, n))
x = seq(from = 0.002, to = 5, by = 0.05)
exact <- data.frame(x = x, y = g(alpha, x))

ggplot(sim) +
  geom_histogram(aes(x = x, y = ..density.., fill = "Simulated"),
                 alpha = 0.8, binwidth = 0.02) +
  geom_line(data = exact , aes(x = x, y = y, color = 'Exact'))  +
  scale_color_manual(name = "", values = c("Exact" = "#e0474c")) +
  scale_fill_manual(name = "", values = c("Simulated" = "#69b3a2")) +
  coord_cartesian(xlim=c(0, 5))
```



From the histogram, we see that we are able to generate samples from g corresponding to the true density. Next we find the theoretical mean and variance and compare with the estimated values. The theoretical mean is given by

$$\mu_g := \int_{-\infty}^{\infty} x\, g(x)\, \mathrm{d}x = c\int_0^1 x^\alpha\, \mathrm{d}x + c\int_1^\infty xe^{-x}\, \mathrm{d}x = \frac{c}{\alpha+1} + \frac{2c}{e}.$$

Similarly, we find the variance

$$\begin{aligned}\mathrm{Var}_g &:= \int_{-\infty}^{\infty} x^2 f(x)\, \mathrm{d}x - \mu_g^2 \\ &= c\int_0^1 x^{\alpha+1}\, \mathrm{d}x + c\int_1^\infty x^2 e^{-x}\, \mathrm{d}x - \mu_g^2 \\ &= \frac{c}{\alpha+2} + \frac{5c}{e} - \left(\frac{c}{\alpha+1} + \frac{2c}{e}\right)^2.\end{aligned}$$

Comparing the true mean and variance with estimated values: ikke plott her?

```
alpha <- 0.5
c <- (1/alpha + exp(-1))^-1

# 10000 realizations from simulation
sim <- sim.g(alpha, 10000)

# Theoretical mean and variance
mean.g <- c/(alpha+1) + 2*c/exp(1)
var.g <- c/(alpha+2) + 5*c/exp(1) - mean.g^2

# Estimated mean and variance
mean.est <- mean(sim)
```

```r
var.est <- var(sim)

# Comparing means
c(mean.g, mean.est)
```

```
## [1] 0.5922707 0.5886037
```

```r
# Comparing variance
c(var.g, var.est)
```

```
## [1] 0.5949550 0.6088946
```

We see that our simulation gives approximate correct values for the theoretical mean and variance. Legge til kumulativ mean og varians og se at de konvergerer for økende n. Akkurat som i oppgavene før.

A.3

(a)

We consider the probability density function

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

To find the value of c, we integrate the density over the entire domain and equate the result to 1:

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

To progress from here, we introduce the substitution, $v = e^{\alpha x}$, which gives

$$1 = \frac{c}{\alpha}\int_0^{\infty} \frac{dv}{(1+v)^2} = \frac{c}{\alpha}\left(-\frac{1}{1+v}\right)\Bigg|_0^{\infty} = \frac{c}{\alpha}.$$

Consequently, $c = \alpha$.

(b)

The cumulative distribution function is

$$\begin{aligned} F(x) &= \int_{-\infty}^{x} f(z)\ dz = \int_0^{e^{\alpha x}} \frac{dv}{(1+v)^2} \\ &= 1 - \frac{1}{1+e^{\alpha x}} = \frac{e^{\alpha x}}{1+e^{\alpha x}}, \end{aligned}$$

where we have used the same substitution as earlier, namely $v = e^{\alpha z}$. We notice that $F(x)$ is the Sigmoid function, which has the logit-function as its inverse. That is,

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

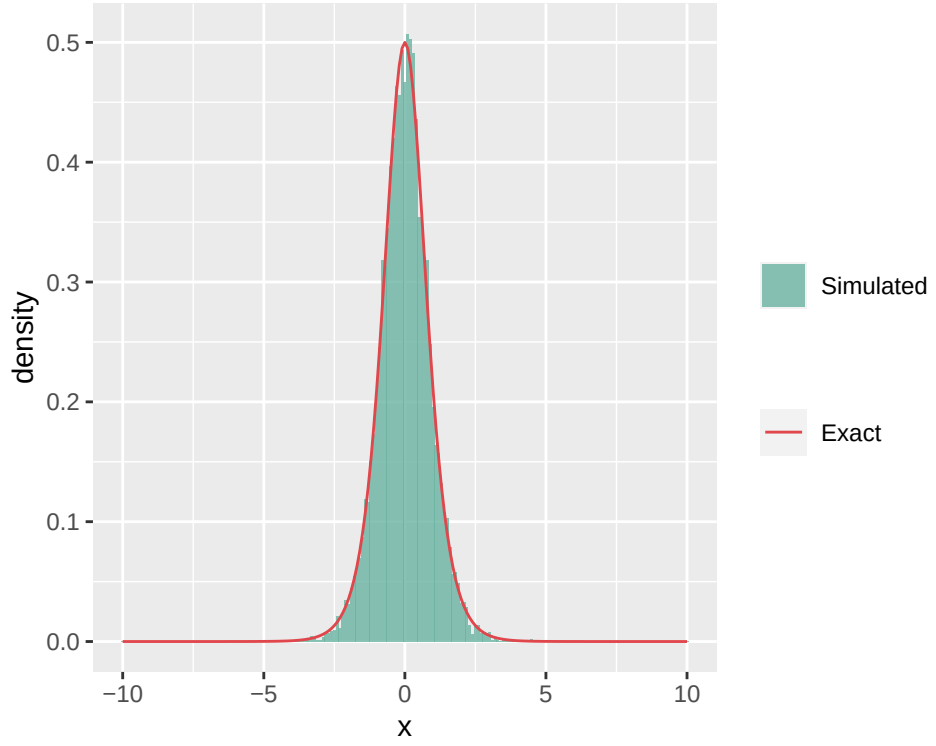
(c)

Since we have an analytic expression for the inverse, we can again use the *inversion method* to sample from f. The sampling-function is given as follows.

```r
sim.sigm <- function(alpha, n){
  u <- runif(n,0,1)
  1/alpha * log(u/(1-u))
}
```

We compare the simulated values with the theoretical distribution:

```r
alpha <- 2
n <- 10000
sim <- data.frame(x = sim.sigm(alpha, n))
x = seq(from = -10, to = 10, by = 0.1)
exact <- data.frame(x = x, y = alpha*exp(alpha*x)/(1 + exp(alpha*x))^2)

ggplot(sim) +
  geom_histogram(aes(x = x, y = ..density.., fill = "Simulated"),
                 alpha = 0.8, binwidth = 0.1) +
  geom_line(data = exact , aes(x = x, y = y, color = 'Exact'))  +
  scale_color_manual(name = "", values = c("Exact" = "#e0474c")) +
  scale_fill_manual(name = "", values = c("Simulated" = "#69b3a2"))
```



To compare the simulated values with the theoretical mean and variance, we first need to compute the expression of these moments. Think maybe the mean is undefined??Den er vel åpenbart null?

Jeg tror vi skal legge til en sammenligning til teoretisk mean (=0?) og varians.

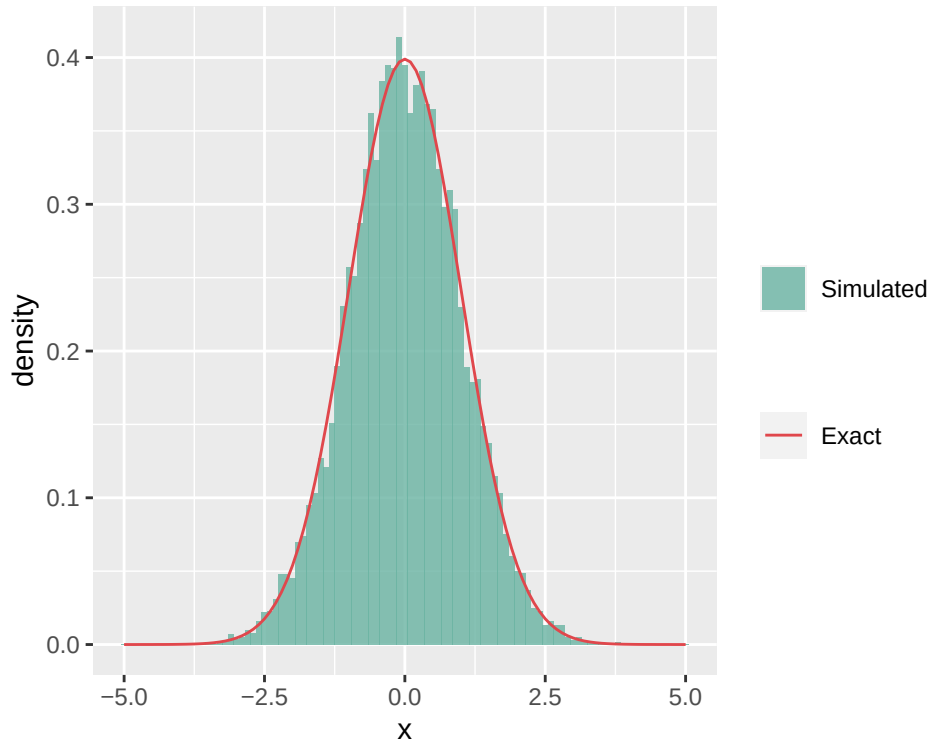
A.4

To generate samples from the standard normal distribution, we make use of the Box-Muller algorithm. Our implementation of the algorithm is given below:

```
box.mul <- function(n){
  # Number of sample pairs to generate
  m <- ceil(n/2)

  x1 <- 2*pi * runif(m, 0, 1)
  x2 <- sim.exp(0.5, m)

  y1 <- sqrt(x2)*cos(x1)
  y2 <- sqrt(x2)*sin(x1)

  # Concatenate y1 and y2
  concat <- c(y1, y2)

  if(n %% 2){ # Drop last element if n is odd.
    head(concat, -1)
  }
  else{
    concat
  }
}
```

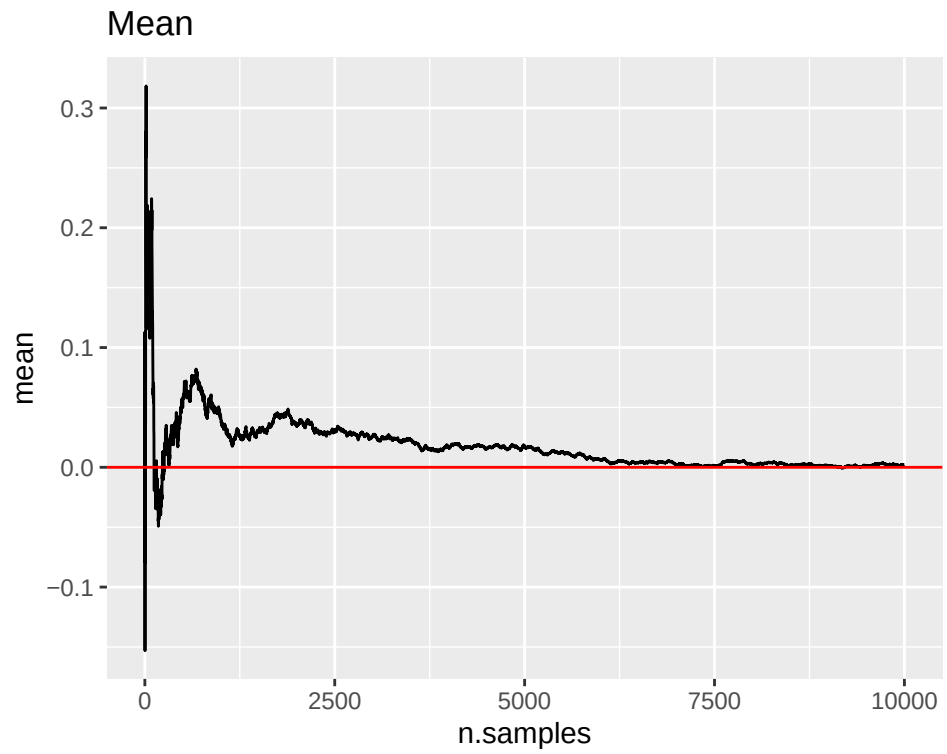
As usual, we compare the results of a simulation against the theoretical distribution.

```
n <- 10000
sim.box.mul <- data.frame(x = box.mul(n))
x = seq(from = -5, to = 5, by = 0.1)
exact <- data.frame(x = x, y = 1/sqrt(2*pi)*exp(-1/2*x^2))

ggplot(sim.box.mul) +
  geom_histogram(aes(x = x, y = ..density.., fill = "Simulated"),
                 alpha = 0.8, binwidth = 0.1) +
  geom_line(data = exact , aes(x = x, y = y, color = 'Exact'))  +
  scale_color_manual(name = "", values = c("Exact" = "#e0474c")) +
  scale_fill_manual(name = "", values = c("Simulated" = "#69b3a2"))
```

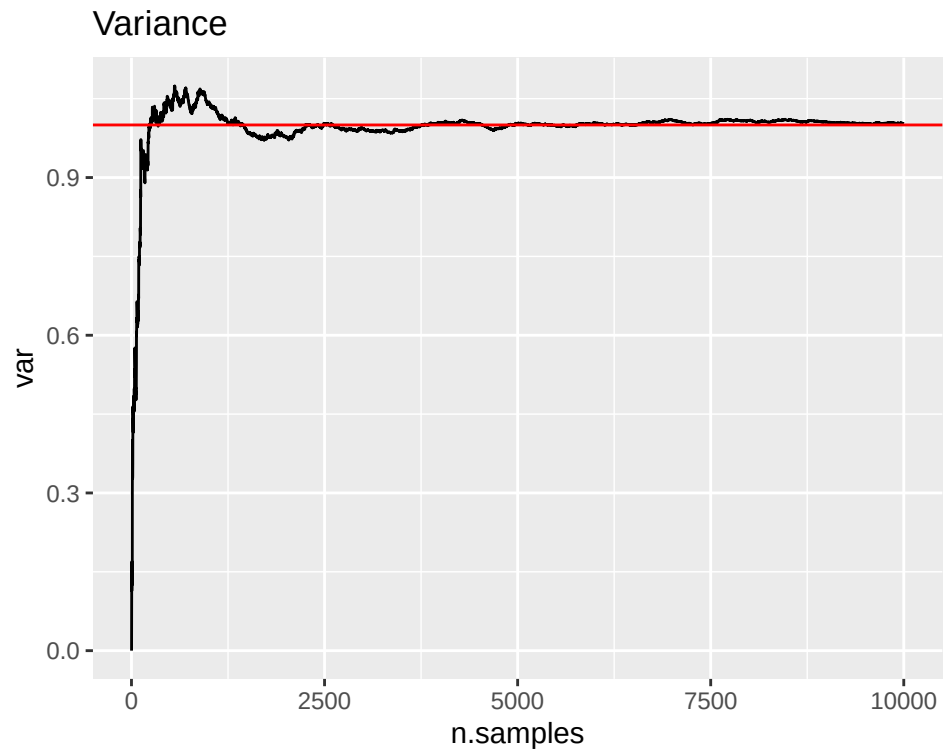


Finally, we compare the mean and variance resulting from the simulation with the theoretical values.

```
# Mean
data.frame(n.samples = 1:n,
           mean = cumsum(sim.box.mul$x)/(1:n)) %>%
  ggplot() + geom_line(aes(n.samples, mean)) +
  geom_hline(yintercept = 0, color = "red") +
  ggtitle("Mean")
```




```r
# Variance
data.frame(n.samples = 1:n,
           mean = cumsum(sim.box.mul$x)/(1:n),
           mean2 = cumsum(sim.box.mul$x^2)/(1:n)) %>%
  mutate(var = mean2-mean^2) %>%
  ggplot() + geom_line(aes(n.samples, var)) +
  geom_hline(yintercept = 1, color = "red") +
  ggtitle("Variance")
```



A.5

Let $\mathcal{N}_d(\mu, \Sigma)$ be a d-variate normal distribution with mean vector μ and covariance matrix Σ. This is our target distribution. To simulate from it, let $x \sim \mathcal{N}_d(0, I_d)$ be a standard d-variate normal distribution. Then, it follows that

$$y = \mu + Ax \quad \Longrightarrow \quad y \sim \mathcal{N}(\mu, AA^T),$$

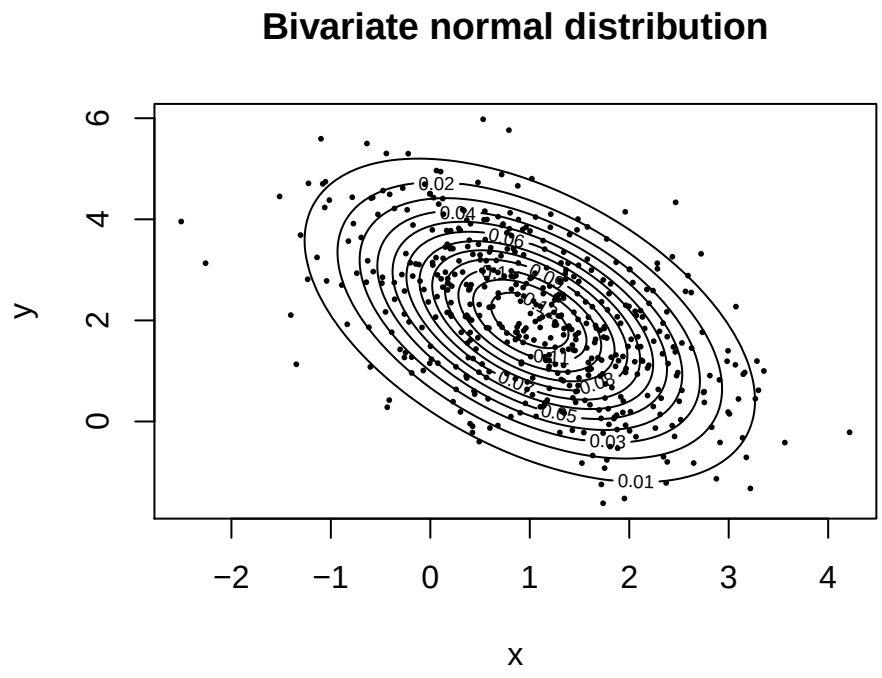
where A is a $d \times d$ matrix. Since the covariance matrix Σ is both symmetric and positive-definite, we can write $\Sigma = AA^T$ using the Cholesky decomposition. Thus, for a given μ and Σ, we are able to simulate from this distribution after simulating from $\mathcal{N}_d(0, I_d)$.

Reusing our implementation of the Box-Muller algorithm from A.4, a function for generating samples from the multivariate normal distribution is shown below:

```
sim.mvnorm <- function(mu, Sigma, n){
  # Returns n realizations from a d-variate normal distribution
  d <- length(mu)

  # Get n d-variate standard normal realizations using the Box-Muller algorithm
  x <- array(box.mul(n*d), dim=c(d,n))

  # Cholesky decomposition of Sigma
  A <- t(chol(Sigma)) # Transposes to get a lower triangular matrix

  mu + A %*% x
}
```

To test our implementation, we generate a bivariate normal distribution and plot some of the samples with the theoretical level curves:

```r
# Bivariate example
mu <- c(1,2)
Sigma <- matrix(c(1, -.7, -.7, 2), 2)

# Simulate 10000 realizations
y <- sim.mvnorm(mu, Sigma, 10000)

# Create theoretical level curves
x.points <- seq(-3,4,length.out=100)
y.points <- seq(-2, 6, length.out=100)
z <- matrix(0,nrow=100,ncol=100)
for (i in 1:100) {
   for (j in 1:100) {
    z[i,j] <- dmvnorm(c(x.points[i],y.points[j]),
    mean=mu,sigma=Sigma)
   }
}

# Plot first 500 points
plot(y[1,1:500], y[2,1:500], main="Bivariate normal distribution",
     xlab="x", ylab="y", pch=16, cex = 0.4)
contour(x.points, y.points, z, add = TRUE)
```



Comparing the true mean and covariance with estimated values:

```r
# Estimate mean
mu.est <- rowMeans(y)
mu.est
```

```
## [1] 1.015967 1.986247
```

```r
# Estimate covariance matrix
Sigma.est <- cov(t(y))
Sigma.est
```

```
##            [,1]       [,2]
## [1,]  1.0059498 -0.6910678
## [2,] -0.6910678  1.9524734
```

From the printout in R, we see that the estimated values are close to the true values $\mu = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ and $\Sigma = \begin{pmatrix} 1 & -0.7 \\ -0.7 & 2 \end{pmatrix}$. We conclude that our function works properly.

Problem B

B.1

(a)

We consider the gamma distribution with $\alpha \in (0,1)$ and $\beta = 1$, i.e.

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

Let g denote the proposal distribution an be given by $g(x)$ considered in A.2. Then, we want to find $k > 1$ such that $f(x) \leq kg(x)\ \forall x \in \mathbb{R}$. The inequality is trivially satisfied for any k when $x \leq 0$. For $x \in (0,1)$,

$$\frac{f(x)}{g(x)} = \frac{\left(\frac{1}{\alpha} + \frac{1}{e}\right) x^{\alpha-1} e^{-x}}{\Gamma(\alpha) x^{\alpha-1}} = \frac{\left(\frac{1}{\alpha} + \frac{1}{e}\right)}{\Gamma(\alpha)} e^{-x} \leq \frac{\left(\frac{1}{\alpha} + \frac{1}{e}\right)}{\Gamma(\alpha)}.$$

For $x \geq 1$, we have

$$\frac{f(x)}{g(x)} = \frac{\left(\frac{1}{\alpha} + \frac{1}{e}\right) x^{\alpha-1} e^{-x}}{\Gamma(\alpha) e^{-x}} = \frac{\left(\frac{1}{\alpha} + \frac{1}{e}\right) x^{\alpha-1}}{\Gamma(\alpha)} \leq \frac{\left(\frac{1}{\alpha} + \frac{1}{e}\right)}{\Gamma(\alpha)}.$$

Consequently, to maximize the acceptance probability given by $p = \frac{1}{k}$, we choose $k := \left(\frac{1}{\alpha} + \frac{1}{e}\right) / \Gamma(\alpha)$. That is, the acceptance probability becomes

$$p = \frac{\Gamma(\alpha)}{\left(\frac{1}{\alpha} + \frac{1}{e}\right)}.$$

(b)

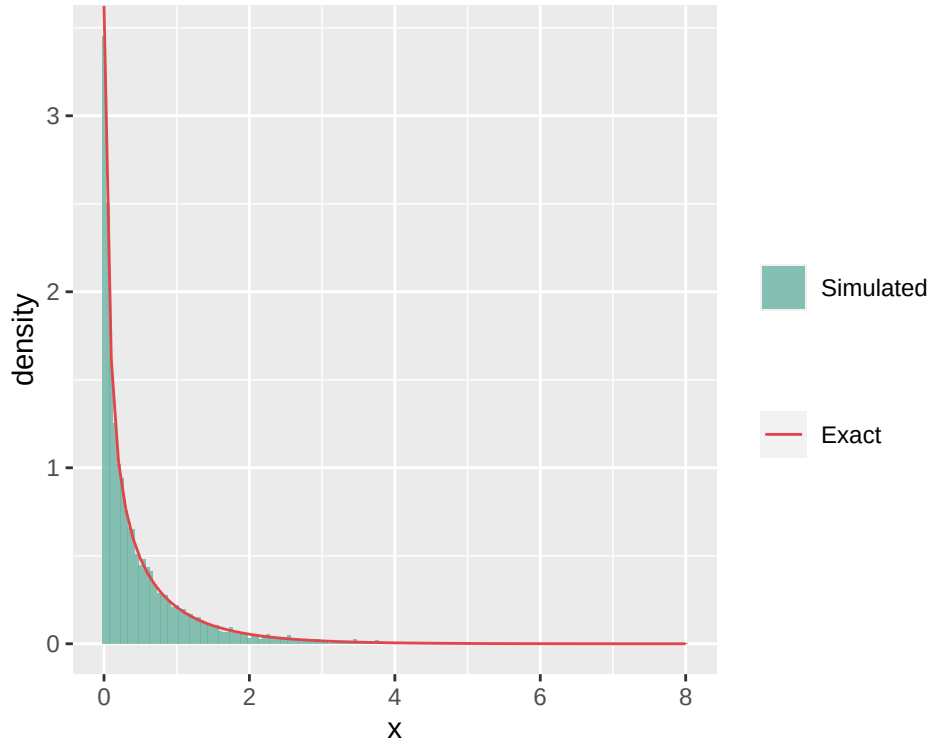
The corresponding rejection sampling algorithm is implemented below.

```
gamma.rej.samp <- function(n, alpha){
  count <- 0
  result <- rep(NA, n)
  p <- gamma(alpha)/(1/alpha + 1/exp(1)) # Acceptance probability
  while(count < n){
    x <- sim.g(alpha, 1)
    g.val <- g(alpha, x)
    f.val <- 1/gamma(alpha) * x^(alpha-1)*exp(-x)
    u <- runif(1,0,1)
    if(u <= p*f.val/g.val){
      result[count + 1] = x
      count = count + 1
     }
  }
  result
}
```

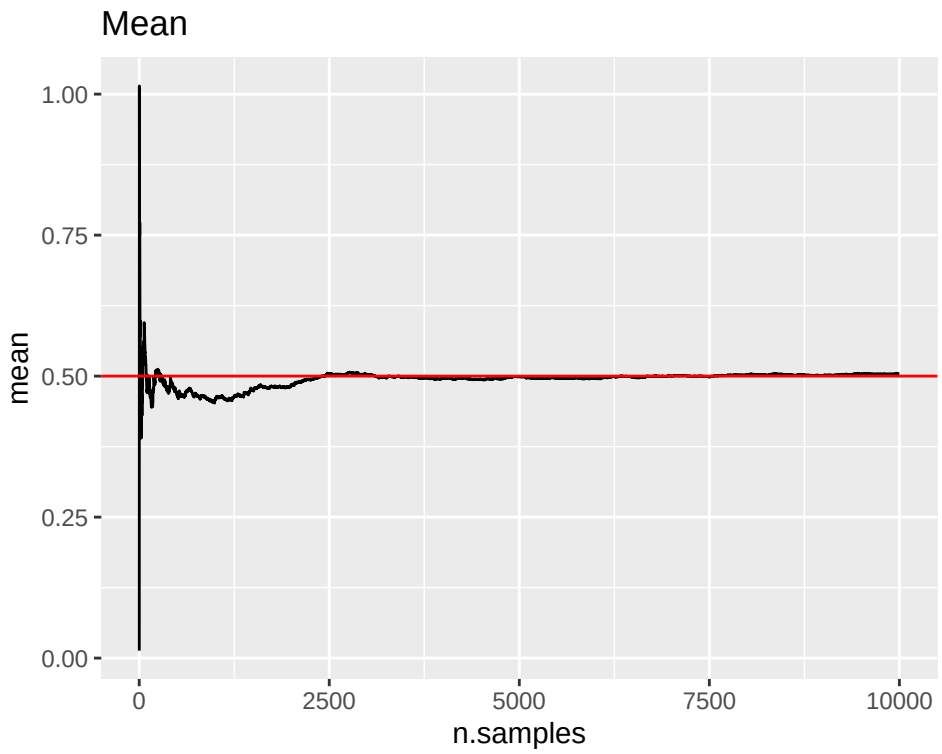
We conduct our usual sanity check.

```
n <- 10000
alpha <- 0.5

sim.gamma.rej <- data.frame(x = gamma.rej.samp(n, alpha))
x = seq(from = 0, to = 8, by = 0.1)
exact <- data.frame(x = x, y = 1/gamma(alpha) * x^(alpha-1)*exp(-x))
```

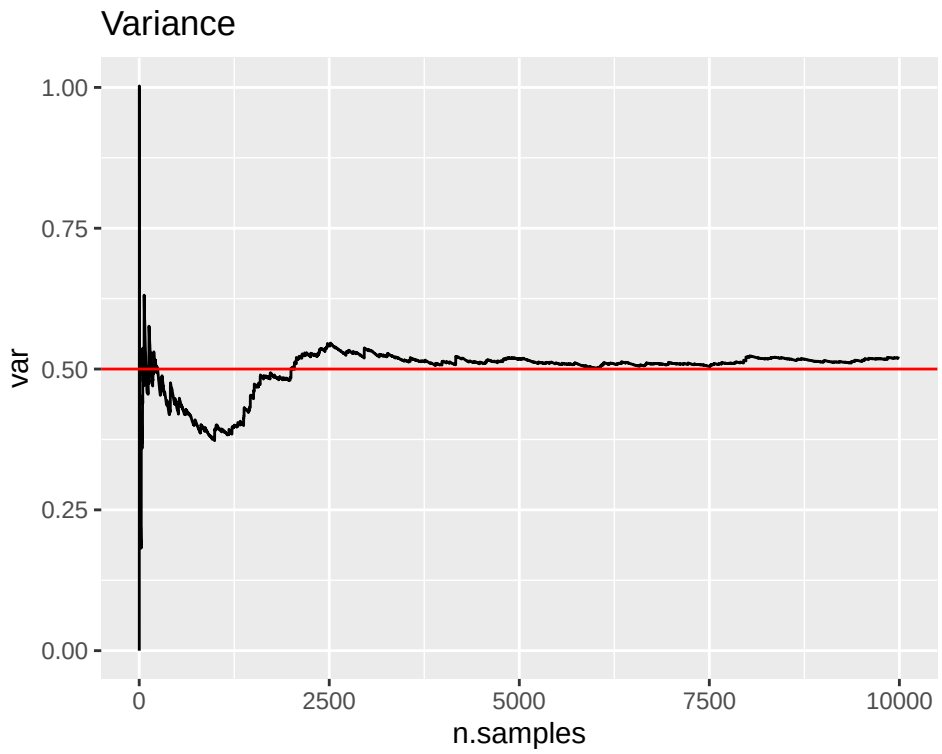
```r
ggplot(sim.gamma.rej) +
  geom_histogram(aes(x = x, y = ..density.., fill = "Simulated"),
                 alpha = 0.8, binwidth = 0.05) +
  geom_line(data = exact , aes(x = x, y = y, color = 'Exact'))  +
  scale_color_manual(name = "", values = c("Exact" = "#e0474c")) +
  scale_fill_manual(name = "", values = c("Simulated" = "#69b3a2"))
```




```r
# Mean
data.frame(n.samples = 1:n,
           mean = cumsum(sim.gamma.rej$x)/(1:n)) %>%
  ggplot() + geom_line(aes(n.samples, mean)) +
  geom_hline(yintercept = alpha, color = "red") +
  ggtitle("Mean")
```




```
# Variance
data.frame(n.samples = 1:n,
           mean = cumsum(sim.gamma.rej$x)/(1:n),
           mean2 = cumsum(sim.gamma.rej$x^2)/(1:n)) %>%
  mutate(var = mean2-mean^2) %>%
  ggplot() + geom_line(aes(n.samples, var)) +
  geom_hline(yintercept = alpha, color = "red") +
  ggtitle("Variance")
```



We conclude that the rejection sampling function works as intended.

B.2

(a)

We repeat that

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

To find $a = \sqrt{\sup_{x\geq 0} f^*(x)}$, we first note that $f^*(x=0) = 0$ and only consider

$$\sup_{x>0} f^*(x),$$

which amounts to solving

$$\begin{aligned} \frac{d}{dx} f^*(x) = 0 & \\ \iff & (\alpha-1)x^{\alpha-2}e^{-x} - x^{\alpha-1}e^{-x} = 0 \\ \iff & x = \alpha - 1 > 0, \end{aligned}$$

which clearly is a global maximum. Consequently,

$$a = \sqrt{f^*(\alpha-1)} = (\alpha-1)^{(\alpha-1)/2}e^{-(\alpha-1)/2} = \left[\frac{\alpha-1}{e}\right]^{\frac{1-\alpha}{2}}.$$

Analogously, to find $b_+ = \sqrt{\sup_{x\geq 0} x^2 f^*(x)} = \sqrt{\sup_{x\geq 0} g(x)}$, we note that $g(z=0) = 0$ and only consider $\sup_{x>0} g(x)$, which amounts to solving

$$\begin{aligned}\frac{d}{dx}g(x) = 0 & \\ &\iff (\alpha+1)x^{\alpha}e^{-x} - x^{\alpha+1}e^{-x} = 0, \\ &\iff x = \alpha + 1 > 0,\end{aligned}$$

which clearly is a global maximum for $x \geq 0$. Consequently,

$$b_+ = \sqrt{(\alpha+1)^2 f^*(\alpha+1)} = (\alpha+1)^{(\alpha+1)/2}e^{-(\alpha+1)/2} = \left[\frac{\alpha+1}{e}\right]^{\frac{\alpha+1}{2}}.$$

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

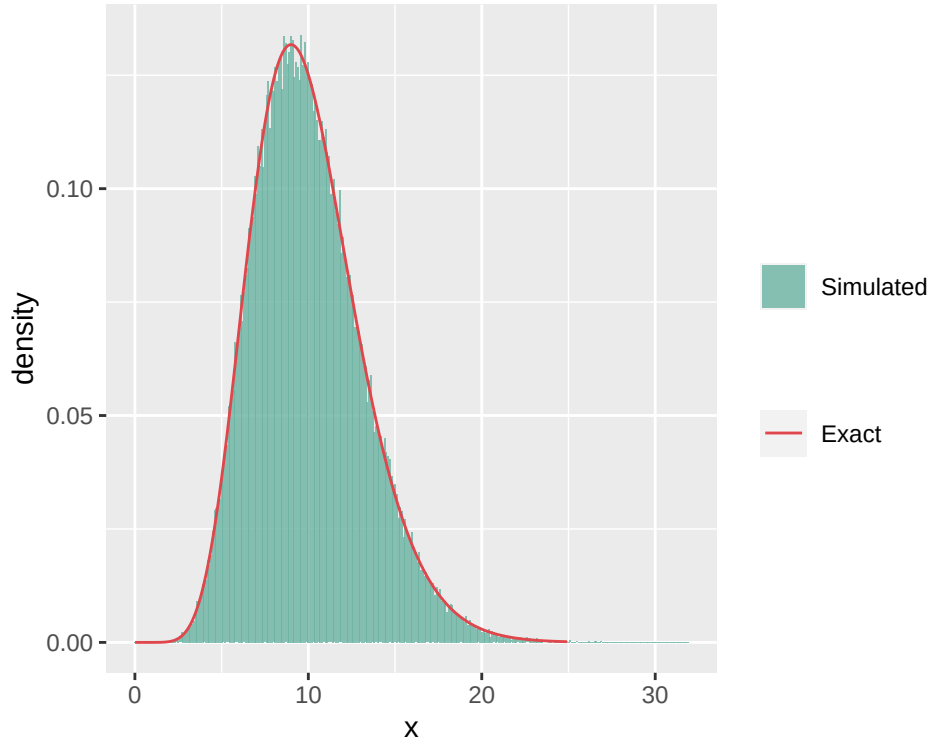
(b)

The algorithm which uses the ratio of uniforms-method to sample from the gamma distribution with $\alpha > 0$ and $\beta = 1$ is given below. Note that the algorithm is implemented on the log scale.

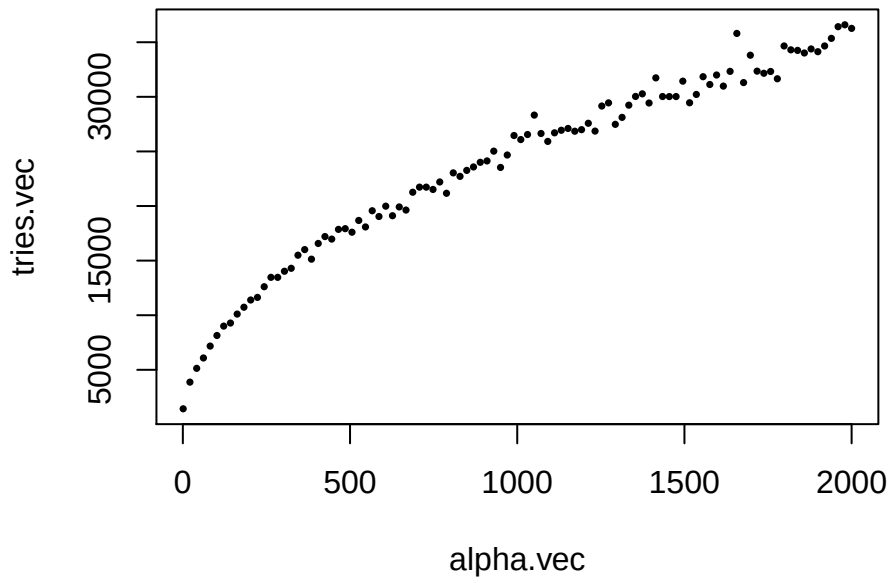
```
samp.log.unif <- function(n, log.b){
  u <- runif(n)
  x <- log.b + log(u) # log((b-a)*u + a) = log(b) + log(u) when a = 0.
  return(x)
}

log.core.gamma <- function(x, alpha){
  if(x <= 0){return(-Inf)}
  else{return((alpha - 1)*log(x) - x)}
}

gamma.rou <- function(n, alpha){
  log.a <-  (alpha - 1)/2 * (log(alpha - 1) - 1)
  log.b.plus <- (alpha + 1)/2 * (log(alpha + 1) - 1)
  a <- ((alpha - 1)/exp(1))^((alpha - 1)/2)
  b.plus <- ((alpha + 1)/exp(1))^((alpha + 1)/2)
  b.minus <- 0

  count <- 0
  tries <- 0
  result <- rep(0,n)

  while(count < n){
    log.x1 <- samp.log.unif(1, log.a)
    log.x2 <- samp.log.unif(1, log.b.plus)

    if(log.x1 <= 0.5*log.core.gamma(exp(log.x2 - log.x1), alpha)){
      result[count + 1] = exp(log.x2 - log.x1)
      count = count + 1
    }
    tries = tries + 1
  }
  list("sim" = result, "tries" = tries)
}
```

First, we check that the simulation algorithm gives reasonable results:

```r
n <- 100000
alpha <- 10

sim.gamma <- gamma.rou(n, alpha)
x = seq(from = 0.01, to = 25, by = 0.1)
exact <- data.frame(x = x, y = 1/gamma(alpha) * x^(alpha - 1) * exp(-x))

ggplot(data.frame(x = sim.gamma$sim)) +
geom_histogram(aes(x = x, y = ..density.., fill = "Simulated"),
                 alpha = 0.8, binwidth = 0.1) +
  geom_line(data = exact , aes(x = x, y = y, color = 'Exact'))  +
  scale_color_manual(name = "", values = c("Exact" = "#e0474c")) +
  scale_fill_manual(name = "", values = c("Simulated" = "#69b3a2"))
```




```r
n <- 1000 # Number of desired samples
m <- 100 # Length of alpha vector
alpha.vec <- seq(from = 1.01, to = 2000, length.out = m)
tries.vec <- rep(0, m)
for(i in 1:m){
  alpha <- alpha.vec[i]
  sim.gamma <- gamma.rou(n, alpha)
  tries.vec[i] = sim.gamma$tries
}

plot(alpha.vec, tries.vec, pch = 16, cex=0.5)
```



The number of necessary tries increases with α. This indicates that the proportion of $[0, a] \times [b_-, b_+]$ which C_f occupies is inversely proportional to α.

B.3

Recall that the gamma distribution has probability density function given as

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

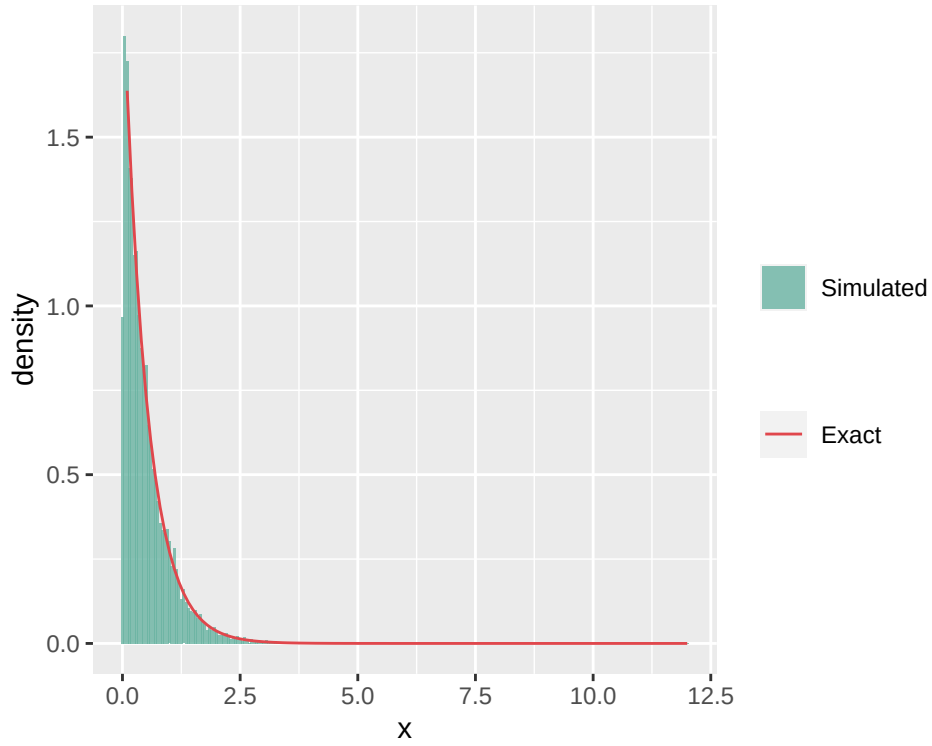
We use Marsaglia and Tsang's method to generate from the gamma distribution:

```
marsa.tsang <- function(n, alpha, beta){
  d <- alpha - 1/3
  c <- 1/sqrt(9*d)
  count <- 0
  result <- rep(0, n)
  while(count < n){
    z <- box.mul(1)
    u <- runif(1, 0, 1)
    v <- (1 + c*z)^3
    if(z > -1/c && log(u) < 0.5*z^2 + d - d*v + d*log(v)){
      result[count + 1] = d*v
      count <- count + 1
    }
  }
  return(1/beta * result)
}
```

```r
sample.gamma <- function(n, alpha, beta){
  if(alpha == 1){
    return(sim.exp(beta, n))
  } else if( alpha < 1){
    return(gamma.rej.samp(n, alpha)/beta)
  } else if( alpha > 1) {
    return(gamma.rou(n, alpha)/beta)
  } else{
    stop("Invalid value of alpha")
  }
}
```

Next, we check if our simulation algorithm gives sensible results.

```r
n <- 10000
alpha <-  1
beta <- 2

sim.gamma <- sample.gamma(n, alpha, beta)
x = seq(from = 0.1, to = 12, by = 0.1)
exact <- data.frame(x = x, y = beta^alpha/gamma(alpha) * x^(alpha - 1) * exp(-beta*x))

ggplot(data.frame(x = sim.gamma)) +
geom_histogram(aes(x = x, y = ..density.., fill = "Simulated"),
                alpha = 0.8, binwidth = 0.05) +
  geom_line(data = exact , aes(x = x, y = y, color = 'Exact'))  +
  scale_color_manual(name = "", values = c("Exact" = "#e0474c")) +
  scale_fill_manual(name = "", values = c("Simulated" = "#69b3a2"))
```



B.4

(a)

We repeat the problem description for completeness. Let $X \sim \text{Gamma}(\alpha, 1)$ and $Y \sim \text{Gamma}(\beta, 1)$. Since X and Y are independent, their joint density is $f_{X,Y}(x, y) = f_X(x)f_Y(y)$. Next, define $Z = \frac{X}{X+Y}$ and $V = X + Y$. Then, $X = ZV$ and $Y = V(1 - Z)$, and, by the transformation formula, the joint density of Z and V is

$$\begin{aligned} f_{Z,V}(z, v) &= f_{X,Y}(zv, v(1 - z)) \cdot \begin{vmatrix} v & z \\ -v & 1 - z \end{vmatrix} \\ &= (\Gamma(\alpha)\Gamma(\beta))^{-1} \cdot (zv)^{\alpha-1} e^{-zv} \cdot (v(1 - z))^{\beta-1} e^{-v(1-z)} \cdot [v(1 - z) + vz] \\ &= (\Gamma(\alpha)\Gamma(\beta))^{-1} \cdot z^{\alpha-1}(1 - z)^{\beta-1} \cdot v^{(\alpha+\beta)-1} \cdot e^{-v}. \end{aligned}$$

The marginal distribution of Z is found from integrating over the whole domain for V. Thus it follows that

$$\begin{aligned} f_Z(z) = \int_0^\infty f_{Z,V}(z, v)\, dv &= \Gamma(\alpha)\Gamma(\beta))^{-1} \cdot z^{\alpha-1}(1 - z)^{\beta-1} \int_0^\infty v^{(\alpha+\beta)-1} \cdot e^{-v} dv \\ &= \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} z^{\alpha-1}(1 - z)^{\beta-1}. \end{aligned}$$

We recognize that this density is identical to $\text{Beta}(\alpha, \beta)$, so it follows that $Z \sim \text{Beta}(\alpha, \beta)$.

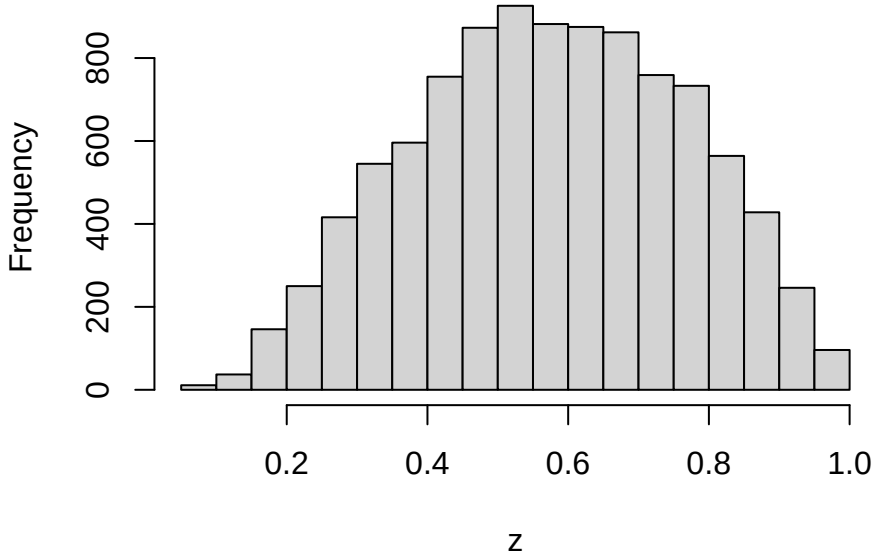
(b)

To generate a vector of n independent samples from a beta distribution with parameters α and β, i.e. to sample from Z, we will reuse our function from B.1 to sample from $X \sim \text{Gamma}(\alpha, 1)$ and $Y \sim \text{Gamma}(\beta, 1)$, then calculate $Z = \frac{X}{X+Y}$. Our implementation for sampling n independent samples from $\text{Beta}(\alpha, \beta)$ is below:

```
sim.beta <- function(n, alpha, beta){
  # Generate n samples from Gamma(alpha, 1) and Gamma(beta, 1)
  x <- gamma.rej.samp(n, alpha)
  y <- gamma.rej.samp(n, beta)

  # Return z=x/(x+y), n independent values from the Beta(alpha, beta)
  x/(x+y)
}
```

Vi må sjekke at simuleringsfunksjonen fungerer som den skal med et eksempel.

```
z <- sim.beta(10000, 5, 2)
hist(z)
```


Histogram of z
Frequency
0
200
400
600
800
0.2
0.4
0.6
0.8
1.0
z

Problem C

C.1

We observe that

$$\theta = \Pr\{X > 4\} = \int_4^\infty f(x)\ \mathrm{d}x = \int_{-\infty}^\infty \mathbb{I}(x > 4) f(x)\ \mathrm{d}x = \mathrm{E}_f[\mathbb{I}(X > 4)].$$

Then, using Monte Carlo integration, an estimate of θ is

$$\hat{\theta} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(x_i > 4).$$

Here, n is the number of samples, $\mathbb{I}$ is the indicator function and $X \sim \mathcal{N}(0, 1)$. The estimate is computed below with $n = 100000$.

```
# Simulate 100000 samples from the standard normal distribution
n <- 100000
x <- box.mul(n)
# Boolean array, indicates which values of x has value >4
h <- (x > 4)
# Monte Carlo estimate
theta.hat <- 1/n * sum(h)
theta.hat
```

```
## [1] 2e-05
```

```
# Theoretical value
pnorm(4, lower.tail = FALSE)
```

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

To compute a confidence interval, we need the variance of $\hat{\theta}$. We define $h(X) = \mathbb{I}(X > 4)$. Then

$$\begin{aligned}\mathrm{Var}[\hat{\theta}] &= \frac{1}{n^2} \sum_{i=1}^{n} \mathrm{Var}_f[h(X_i)] = \frac{1}{n} \mathrm{Var}_f[h(X)] \\ &\approx \frac{1}{n} \left(\frac{1}{n-1} \sum_{i=1}^{n} (h(x_i) - \hat{\theta})^2 \right) =: \hat{\sigma}^2.\end{aligned}$$

Dobbeltsjekk at dette riktig utregning ved bruk av indikatorfunksjonen?

We also utilize the central limit theorem, such that $\hat{\theta} \sim \mathcal{N}\left(\theta, \mathrm{Var}[\hat{\theta}]\right)$. Then we know that

$$\frac{\hat{\theta} - \theta}{\hat{\sigma}} \sim t_{n-1}.$$

Consequently, a 95% confidence interval is given by

$$\left[\hat{\theta} - t_{n-1,0.025} \cdot \hat{\sigma},\ \hat{\theta} + t_{n-1,0.025} \cdot \hat{\sigma}\right].$$

It is computed and displayed below.

```
sigma.hat <- sqrt(1/(n*(n-1)) * sum((h - theta.hat)^2))
1/sigma.hat^2
```

```
## [1] 5000050001
```

```r
lower <- theta.hat - qt(0.975, n - 1)*sigma.hat
upper <- theta.hat + qt(0.975, n - 1)*sigma.hat

c(lower, upper)
```

```
## [1] -7.718273e-06  4.771827e-05
```

C.2

To simulate from g, we need to find the normalization constant:

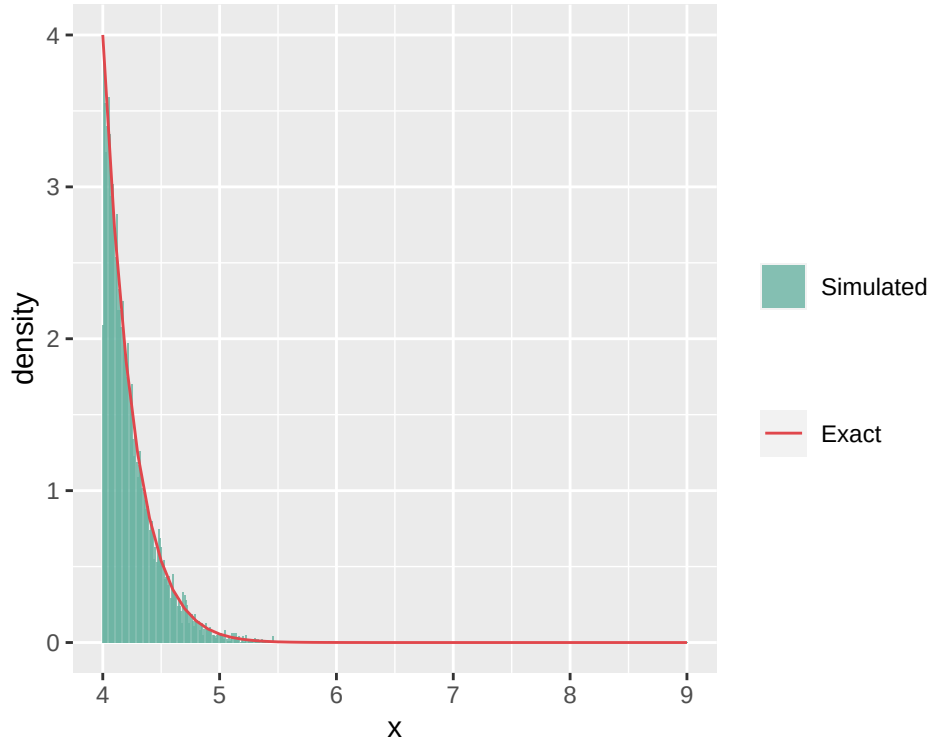
$$1 = \int_4^\infty g(x)dx = -c\exp\left(-\frac{x^2}{2}\right)\Big|_4^\infty \iff c = \exp(8).$$

The CDF, $G(x)$, is thus given as

$$G(x) = \int_4^x g(t)dt = 1 - \exp\left(-\frac{x^2}{2} - 8\right).$$

It can easily be shown that $G^{-1}(x) = \sqrt{16 - 2\ln(1-x)}$, which we utilize in the inversion sampling algorithm below. Sanity check:

```r
inv.samp <- function(n){
  u <- runif(n, 0, 1)
  x <- sqrt(16 - 2*log(1 - u))
  return(x)
}

sim <- inv.samp(10000)
x = seq(from = 4, to = 9, by = 0.1)
exact <- data.frame(x = x, y = exp(8)*x*exp(-x^2/2))

ggplot(data.frame(x = sim)) +
 geom_histogram(aes(x = x, y = ..density.., fill = "Simulated"),
                alpha = 0.8, binwidth = 0.01) +
  geom_line(data = exact , aes(x = x, y = y, color = 'Exact'))  +
  scale_color_manual(name = "", values = c("Exact" = "#e0474c")) +
  scale_fill_manual(name = "", values = c("Simulated" = "#69b3a2"))
```



We note that $g(x) > 0$ where $h(x)f(x) > 0$ show this?? Sara did say it is not necessary, but it is nice. Then, in the importance sampling scheme, the estimator is given as

$$\hat{\theta}_{IS} = \frac{1}{n}\sum_{i=1}^{n} \frac{h(X_i)f(X_i)}{g(X_i)} = \frac{1}{n}\sum_{i=1}^{n} h(X_i)w(X_i)$$

We also note that $w(x) = \left(\sqrt{2\pi}cx\right)^{-1}$ Below we perform the importance sampling

```
n <- 100000
c <- exp(8)

x <- inv.samp(n)
w <- 1/(sqrt(2*pi)*c*x)
h <- (x > 4)

theta.is <- 1/n * sum(h*w)
theta.is
```

```
## [1] 3.166351e-05
```

To create a confidence interval, we need to estimate the variance.

$$\begin{aligned}\mathrm{Var}[\hat{\theta}_{IS}] &= \frac{1}{n^2}\sum_{i=1}^{n} \mathrm{Var}_g[h(X_i)w(X_i)] = \frac{1}{n}\mathrm{Var}_g[h(X)w(X)] \\ &\approx \frac{1}{n}\left(\frac{1}{n-1}\sum_{i=1}^{n}(h(x_i)w(x_i) - \hat{\theta}_{IS})^2\right) =: \hat{\sigma}^2_{IS}.\end{aligned}$$

We use the same procedure as above, which leads to the confidence interval

$$\left[\hat{\theta}_{IS} - t_{n-1,0.025} \cdot \hat{\sigma}_{IS},\ \hat{\theta}_{IS} + t_{n-1,0.025} \cdot \hat{\sigma}_{IS}\right].$$

It is computed numerically below.

```
sigma.is <- sqrt(1/(n*(n-1)) * sum((h*w - theta.is)^2))
lower <- theta.is - qt(0.975, n - 1)*sigma.is
upper <- theta.is + qt(0.975, n - 1)*sigma.is

c(lower, upper)
```

```
## [1] 3.165384e-05 3.167319e-05
```

```
1/sigma.is^2
```

```
## [1] 4.106615e+16
```

```
n*sigma.hat^2/sigma.is^2
```

```
## [1] 821314726570
```

We define the precision of the estimators as the reciprocal of the variance of the estimators. Hence we have $\text{Precision}(\hat{\theta}) = 5.00005 \times 10^9$ and $\text{Precision}(\hat{\theta}_{IS}) = 4.1066147 \times 10^{16}$. We observe that the importance sampling yields a much more precise estimator. How many many samples would we have needed to achieve the same precision in C.1? Let m denote the number of samples used to compute $\hat{\theta}$ and $n = 100000$ the number of samples used to compute $\hat{\theta}_{IS}$. Then we want to find m such that

$$\begin{aligned} & \text{Var}[\hat{\theta}] = \text{Var}[\hat{\theta}_{IS}] \\ \iff & \frac{1}{m}\text{Var}_f[h(X)] = \frac{1}{n}\text{Var}_g[h(X)w(X)] \\ \iff & m = n\frac{\text{Var}_f[h(X)]}{\text{Var}_g[h(X)w(X)]} \approx 8.2131473 \times 10^{11}, \end{aligned}$$

where we have used the sample variances computed earlier to create the estimate of m.

C.3

(a)

We define the antithetic estimator, or sampler, as

$$\hat{\theta}_{AS} = \frac{1}{2}\left(\hat{\theta}_{IS}^{(1)} + \hat{\theta}_{IS}^{(2)}\right),$$

where

$$\hat{\theta}_{IS}^{(1)} = \sum_{i=1}^{n} h\left(X_i^{(1)}\right) w\left(X_i^{(1)}\right) \quad \text{and} \quad \hat{\theta}_{IS}^{(2)} = \sum_{i=1}^{n} h\left(X_i^{(2)}\right) w\left(X_i^{(2)}\right)$$

are simulated by the same importance sampling procedure as in C.2, but the $X_j^{(1)}$'s are based on u while the $X_j^{(2)}$'s are based on $u - 1$ in the inversion sampling. The algorithm which samples n antithetic pairs from g is given below.

```
anti <- function(n){
  u <- runif(n, 0, 1)
  x.1 <- sqrt(16 - 2*log(u))
```

```
  x.2 <- sqrt(16 - 2*log(1 - u))
  list(x.1 = x.1, x.2 = x.2)
}
```

(b)

Below, we generate $n = 50000$ pairs of antithetic variates which we use in the importance sampling. We choose n as such because we generate pairs and $2n = 100000$, which is what we used in C.2.

```
n <- 50000
c <- exp(8)

x <- anti(n)
x.1 = x$x.1
x.2 <- x$x.2

w.1 <- 1/(sqrt(2*pi)*c*x.1)
h.1 <- (x.1 > 4)
theta.1 <- 1/n * sum(h.1*w.1)

w.2 <- 1/(sqrt(2*pi)*c*x.2)
h.2 <- (x.2 > 4)
theta.2 <- 1/n * sum(h.2*w.2)

theta.as <- 1/2 * (theta.1 + theta.2)
theta.as
```

```
## [1] 3.167508e-05
```

To compute a confidence interval, we first note the central limit theorem is still applicable, since we can write the antithetic sampler as

$$\hat{\theta}_{AS} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\left(h\left(X_i^{(1)}\right)w\left(X_i^{(1)}\right) + h\left(X_i^{(2)}\right)w\left(X_i^{(2)}\right)\right) =: \frac{1}{n}\sum_{i=1}^{n} Y_i.$$

We assume these Y_i's to be independent, so that we can utilize the t-distribution when estimating the variance. We observe that the variance of the antithetic sampler is:

$$\begin{aligned}\mathrm{Var}\left[\hat{\theta}_{AS}\right] &= \frac{1}{4}\left(\mathrm{Var}\left[\hat{\theta}_{IS}^{(1)}\right] + \mathrm{Var}\left[\hat{\theta}_{IS}^{(2)}\right] + 2\mathrm{Cov}\left[\hat{\theta}_{IS}^{(1)}, \hat{\theta}_{IS}^{(2)}\right]\right) \\ &= \frac{\sigma^2(1+\rho)}{2n},\end{aligned}$$

where $\sigma^2 = \mathrm{Var}[h(X)w(X)]$ and $\rho = \mathrm{Corr}\left[h(X^{(1)})w(X^{(1)}), h(X^{(2)})w(X^{(2)})\right]$. Finally, we compute an approximate 95% confidence interval:

```
Y <- 1/2 * (h.1*w.1 + h.2*w.2)
S.Y <- var(Y) # Sample variance of Y's
sigma.as <- sqrt(S.Y/n) # Estimated sd of theta.as

lower <- theta.as - qt(0.975, n - 1)*sigma.as
upper <- theta.as + qt(0.975, n - 1)*sigma.as

c("conf.int.lower" = lower, "conf.int.upper" = upper, "precision" = 1/sigma.as^2)
```

```
## conf.int.lower conf.int.upper      precision
##   3.167043e-05   3.167974e-05   1.771870e+17
```

Hvordan er ρ implementert her?

We observe that the precision is higher than for regular importance sampling. This can be explained be a negative correlation coefficient, ρ. To check this, we calculate the sample correlation:

```
c("sample.correlation" = cor(h.1*w.1, h.2*w.2))
```

```
## sample.correlation
##         -0.7650111
```

As expected, the sample correlation is negative, which explains the increased precision compared to regular importance sampling.

Problem D

1.

We consider a multinomial distribution with mass function given by

$$f(y \mid \theta) \propto (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4},$$

where $\theta \in (0,1)$. Using a prior $g(\theta)$ on $(0,1)$, the observed posterior density is $f(\theta \mid y) \propto f(y \mid \theta)\ g(\theta)$. First, let $g(\theta) = \mathcal{U}_{[0,1]}$, i.e. the uniform distribution, which is equivalent to Beta$(1,1)$. Then the observed posterior density becomes

$$f(\theta \mid y) \propto f(y \mid \theta)\ \mathcal{U}_{[0,1]} \propto \underbrace{(2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4}}_{\tilde{f}(\theta|y)} \qquad \text{for} \quad \theta \in (0,1).$$

We can now use rejection sampling to sample from $f(\theta \mid y)$ using the improper density $\tilde{f}(\theta \mid y)$ defined in the eauation above. For the rejection sampling to work optimally, meaning we want highest possible acceptance rate, we want to scale the density such that $\max_\theta \frac{\tilde{f}(\theta|y)}{\tilde{c}\ g(\theta)} = 1$, where $\tilde{c}$ is a constant. When $g(\theta) = \mathcal{U}_{[0,1]}$, the maximum of $\tilde{f}(\theta \mid y)$ can be found from maximizing its logarithm. Deriving $\ln \tilde{f}(\theta \mid y)$ with respect to θ and setting equal to zero, we obtain

$$\frac{\mathrm{d}\ \ln \tilde{f}(\theta \mid y)}{\mathrm{d}\theta} = \frac{y_1}{2+\theta} - \frac{y_2+y_3}{1-\theta} + \frac{y_4}{\theta} = 0,$$

which on the interval $(0,1)$ has a maximum at

$$\theta^* = \frac{15+\sqrt{53809}}{394} \approx 0.6268,$$

where we used the values for $\{y_i\}_{i=1}^4$ given in the problem description. Next we use this to find the constant $\tilde{c} = \tilde{f}(\theta^* \mid y) \approx \exp(67.38)$. In effect, this means that $\max_{\theta\in(0,1)} \frac{1}{\tilde{c}}\tilde{f}(\theta \mid y) = 1$ and $\tilde{f}(\theta \mid y) \geq 0$ for $\theta \in (0,1)$. Our implementation of the rejection algorithm is summarized below:

Rejection sampling algorithm

1. Sample θ from the prior distribution $g(\theta)$.
2. Sample u from the uniform distribution $\mathcal{U}_{[0,1]}$.
3. Compute $\alpha = \frac{1}{\tilde{c}}\frac{\tilde{f}(\theta)}{g(\theta)}$.
4. If $u \leq \alpha$: keep θ. Otherwise do nothing.

```
# Calculating the constant c
y <- c(125, 18, 20, 34) # From project description
theta.star <- (15 + sqrt(53809))/394 # Theoretical argmax
c <- exp(y[1]*log(2+theta.star) + (y[2]+y[3])*log(1-theta.star) + y[4]*log(theta.star))

sim.rej.D <- function(n, y = c(125, 18, 20, 34)){
  samples <- rep(0,n)
  n_simulated <- 0

  i = 1
  while(i<=n){
    n_simulated <- n_simulated + 1
    # Sample from prior distribution
    theta <- runif(1,0,1)
    # Calculate acceptance probability
    f.value <- (2+theta)^y[1] * (1-theta)^(y[2]+y[3]) * theta^y[4]
```

```
    alpha <- 1/c * f.value
    # Check if sample is accepted
    if(runif(1,0,1) <= alpha){
      samples[i] <- theta
      i <- i+1
    }
  }
  # Return samples and acceptance rate
  list(samples = samples, rate = n/n_simulated)
}
```

2.

We denote the posterior mean of θ by $\mu_\theta = \mathrm{E}(f(\theta \mid y))$. Using Monte-Carlo integration, the mean can be estimated as

$$\hat{\mu}_\theta = \frac{1}{M}\sum_{i=1}^{M}\theta_i,$$

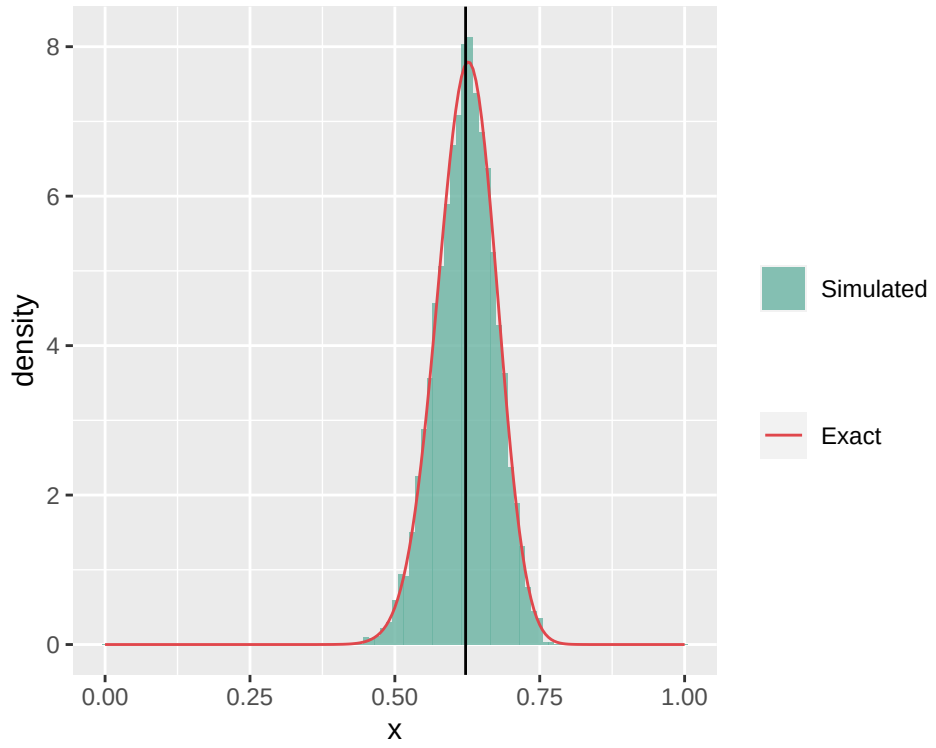
where $\{\theta_i\}_{i=1}^{M}$ are sampled from the posterior density $f(\theta \mid y)$. The posterior mean is estimated with $M = 10000$.

```
# Monte-Carlo estimate for mean
M <- 10000
samples <- sim.rej.D(M)$samples
mu.theta <- mean(samples)
mu.theta
```

```
## [1] 0.6221356
```

Using $M = 10000$ samples, Monte-Carlo integration gives the estimate $\hat{\mu}_\theta = 0.6221356$. Next we perform our usual sanity check and draws a vertical line indicating the mean.

```
sim.rej <- data.frame(x = samples)

# Posterior density (up to a multiplicative constant)
f <- function(theta, y=c(125, 18, 20, 34)){
  (2+theta)^y[1] * (1-theta)^(y[2]+y[3]) * theta^y[4]
}

# Calculating the theoretical density and normalizing
theta = seq(from = 0, to = 1, by = 0.005)
norm.const <- integrate(f,0,1)$value
y <- f(theta) / norm.const

exact <- data.frame(x = theta, y = y)


# Histogram + theoretical density + vertical line at estimated mean
ggplot(sim.rej) +
  geom_histogram(aes(x = x, y = ..density.., fill = "Simulated"),
                 alpha = 0.8, binwidth = 0.01) +
  geom_line(data = exact , aes(x = x, y = y, color = 'Exact'))  +
  scale_color_manual(name = "", values = c("Exact" = "#e0474c")) +
  scale_fill_manual(name = "", values = c("Simulated" = "#69b3a2")) +
  geom_vline(xintercept=mu.theta, color="#000000")
```



Below is an estimate of the mean using numerical integration:

```
# Calculating the mean numerically
mu.num <- integrate(function(x) x*f(x)/norm.const, 0, 1)
mu.num$value
```

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

We see that the estimate of the mean obtained using Monte Carlo integration, $\hat{\mu}_\theta = 0.6221356$, is close to the numerical value $\hat{\mu}_\theta^{(\text{num})} = 0.6228061$.

3.

To get an estimate for how many generations our algorithm needs on average to produce one sample, we can first find the acceptance rate using Monte-Carlo simulation. Then the average number of samples needed is equal to the inverse of the acceptance rate. To find this practically, we simply count how many generations is needed for a large number of samples and divide by the total number of samples generated.

In addition to the practical approach, a numerical estimate can be carried out by calculating the expected acceptance rate. Let the acceptance rate be $\alpha(\theta) = \frac{\tilde{f}(\theta|y)}{\tilde{c}\ g(\theta)}$. Then the expected value is

$$\mathrm{E}(\alpha) := \int_0^1 \alpha(\theta)\ g(\theta)\ \mathrm{d}\theta = \frac{1}{\tilde{c}} \int_0^1 \tilde{f}(\theta)\ \mathrm{d}\theta,$$

which is computed numerically. Thus the theoretical expected number of generations needed for one sample is $\frac{1}{\mathrm{E}(\alpha)}$.

```
# Practically calculated acceptance rate
M <- 10000
acc.rate.practical <- sim.rej.D(M)$rate
1/acc.rate.practical
```

```
## [1] 7.7907
```

```
# Numerically calculated acceptance rate
acc.rate.numerical <- integrate(f,0,1)$value / c
1/acc.rate.numerical
```

```
## [1] 7.799308
```

We see that the average number of generations needed to produce one sample are calculated practically as 7.7907 and numerically as 7.7993077. The practical estimate is close to the numerical value, as expected.

4.

Now we assume that the prior density is $g(\theta) = \text{Beta}(1, 5) = 5(1-\theta)^4$. The new posterior density then becomes

$$f_2(\theta \mid y) \propto f(y \mid \theta)\ 5(1-\theta)^4 \propto 5(2+\theta)^{y_1}(1-\theta)^{4+y_2+y_3}\theta^{y_4} \qquad \text{for} \quad \theta \in (0,1),$$

where we have used the subscript (2) to indicate that this posterior density is different than the one using the uniform distribution. We want to estimate the posterior mean $\mu = \text{E}(\theta \mid y)$ under the new prior density based on the samples obtained in part (2). To do so, we will use importance sampling, where $f_2(\theta \mid y)$ is our target distribution and $f(\theta \mid y)$ is our proposal distribution. An estimate for the posterior mean is thus given as

$$\hat{\mu}_{\theta,2} = \text{E}(\theta \mid y) = \frac{\sum_{i=1}^N \theta_i \frac{f_2(\theta_i|y)}{f(\theta_i|y)}}{\sum_{i=1}^N \frac{f_2(\theta_i|y)}{f(\theta_i|y)}} = \frac{\sum_{i=1}^N \theta_i w(\theta_i)}{\sum_{i=1}^N w(\theta_i)},$$

where the importance weights are $w(\theta_i) = \frac{f_2(\theta_i|y)}{f(\theta_i|y)} = 5(1-\theta_i)^4$. The new posterior mean is calculated below:

```
# Weight function (equals to the new prior divided by the old prior)
w <- function(theta) 5*(1-theta)^4

# Calculate new posterior mean
mu.theta2 <- sum(samples*w(samples)) / sum(w(samples))
mu.theta2
```

```
## [1] 0.5949495
```

We see that when using the $\text{Beta}(1, 5)$ prior, our estimate for the posterior mean $\hat{\mu}_{\theta,2} = 0.5949495$ is lower than the estimated posterior mean using the uniform prior. The reason for this is that the densities in $\text{Beta}(1, 5)$ is higher in the lower end of the interval $(0, 1)$, such that samples with lower values of θ has higher importance weights and thus contributes more relative to samples with higher values of θ. Intuitively, this in turn means that the estimate of the mean must be lower using this prior.

In this particular example we see that the uniform prior got an estimate closer to the numerical value than the $\text{Beta}(1, 5)$ prior. This highlights the importance of choosing good priors to get reasonable posterior estimates.