TMA4300: Exercise 1

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

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

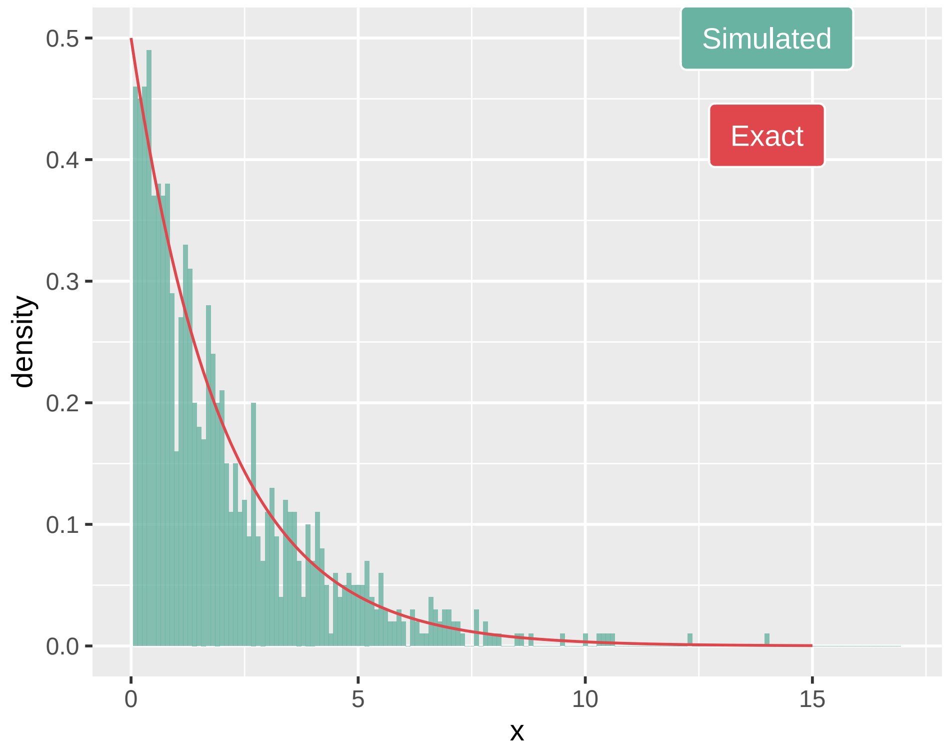
This means that we can use the *inversion method* to simulate from the exponential distribution. Let $U \sim \mathcal{U}_{[0,1]}$ and calculate $X = F^{-1}(U)$, then $X \sim \text{Exp}(\lambda)$. A function which simulates from the exponential distribution is given below.

```r
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 $n = 1000$.

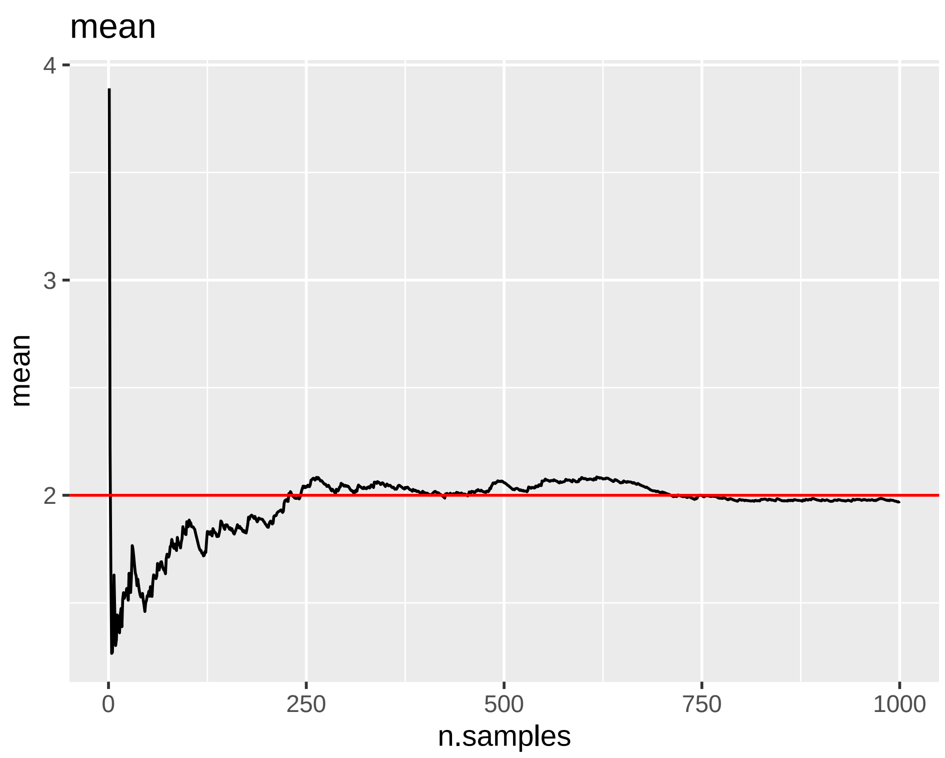
```r
rate <- 0.5
n <- 1000
sim <- data.frame(x = sim.exp(rate, n))
x = seq(from = 0, to = 15, by = 0.1)
exact <- data.frame(x = x, y = rate*exp(-rate*x))

ggplot(sim) +
  geom_histogram(aes(x = x, y = ..density..),
                 alpha = 0.8, fill = "#69b3a2", binwidth = 0.1) +
  geom_line(data = exact , aes(x = x, y = y), color = "#e0474c")  +
  geom_label(
    label="Simulated",
    x=14,
    y=0.5,
    label.padding = unit(0.55, "lines"), # Rectangle size around label
    label.size = 0.35,
    color = "white",
    fill= "#69b3a2"
  ) +
  geom_label(
    label="Exact",
    x=14,
    y=0.42,
    label.padding = unit(0.55, "lines"), # Rectangle size around label
    label.size = 0.35,
    color = "white",
    fill = "#e0474c"
  ) +
  xlim(0,17)
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```

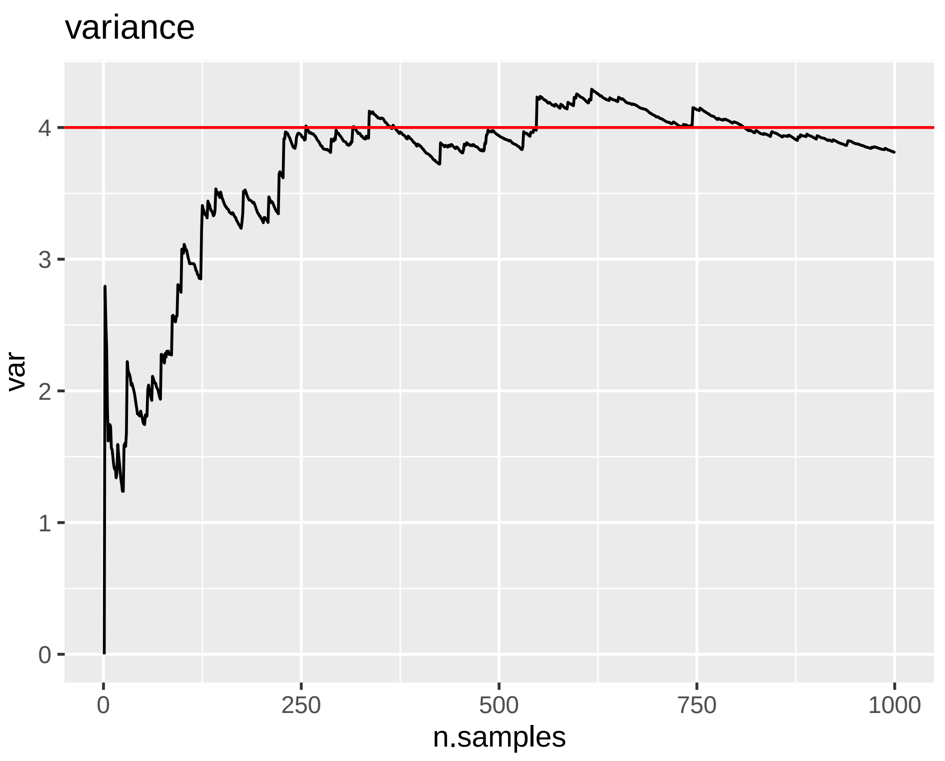


We also compare the estimated mean and variance to first and second central moments of the exponential distribution:

```
# 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

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

which evaluates to

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

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

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

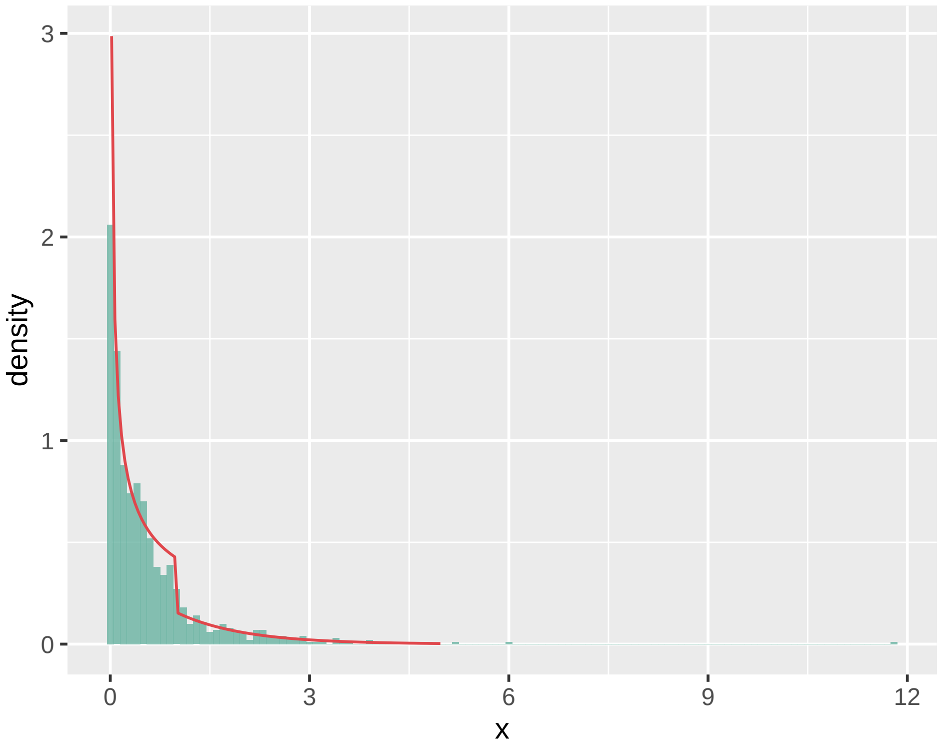
b)

Function for generating samples from g:

```r
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 for $n = 1000$.

```r
alpha <- 0.5
n <- 1000
sim <- data.frame(x = sim.g(alpha, n))
x = seq(from = 0.02, to = 5, by = 0.05)
exact <- data.frame(x = x, y = g(alpha, x))

ggplot(sim) +
  geom_histogram(aes(x = x, y = ..density..),
                 alpha = 0.8, fill = "#69b3a2", binwidth = 0.1) +
  geom_line(data = exact , aes(x = x, y = y), color = "#e0474c")
```



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

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

Comparing the true mean and variance with estimated values:

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

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

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

```
## [1] 0.5922707 0.5881271
```

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

```
## [1] 0.5949550 0.5760642
```

We see that our simulation gives approximate correct values for the theoretical mean and variance.

A.3

a)

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

$$1 = \frac{c}{\alpha}\int_{0}^{\infty} \frac{dv}{(1+v)^2} = \frac{c}{\alpha}(-\frac{1}{1+v})\Big|_0^{\infty} = \frac{c}{\alpha}.$$

Consequently, $c = \alpha$.

b)

The CDF is defined as follows,

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

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 well known logit-function as its inverse. I.e.,

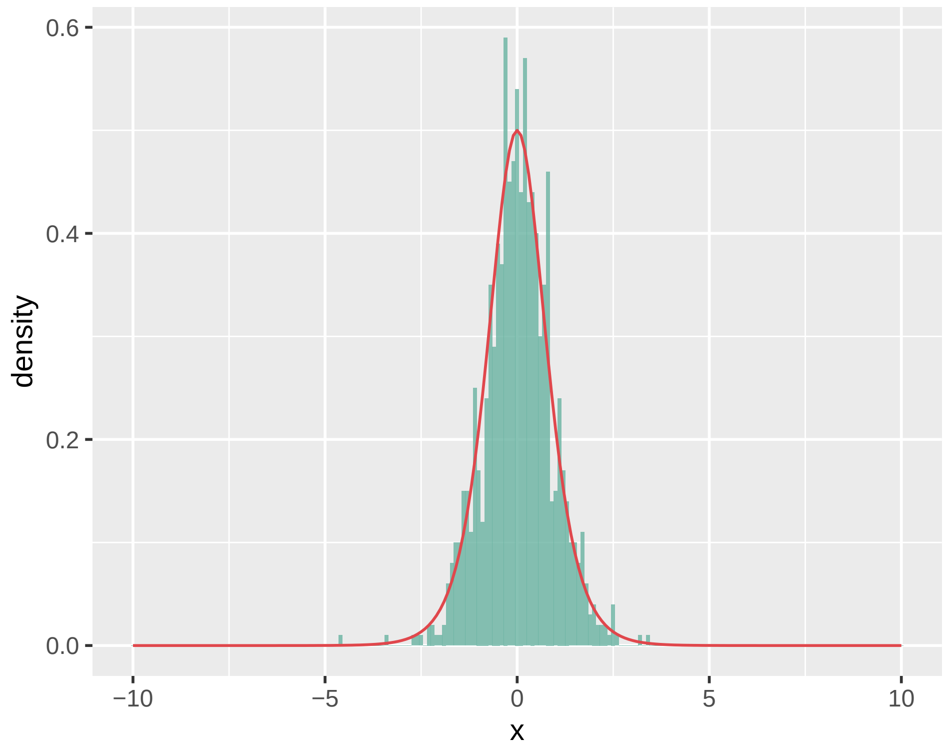
$$F^{-1}(x) = \frac{1}{\alpha}\ln\left(\frac{x}{1-x}\right).$$

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 below.

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

```r
alpha <- 2
n <- 1000
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..),
                 alpha = 0.8, fill = "#69b3a2", binwidth = 0.1) +
  geom_line(data = exact , aes(x = x, y = y), color = "#e0474c")
```



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??

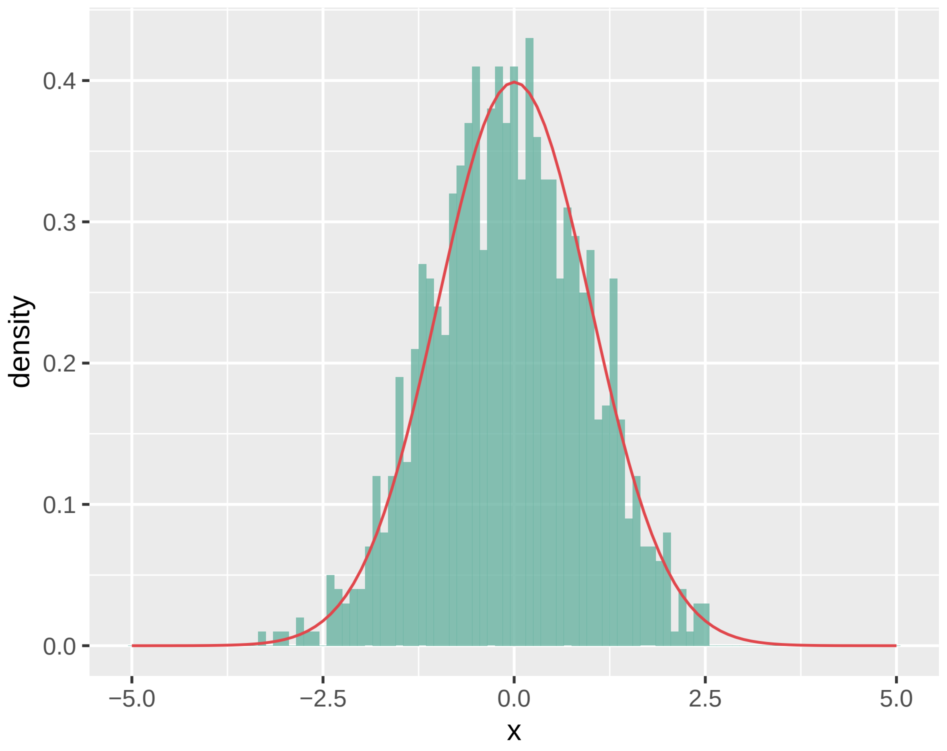
A.4

Below is our implementation of the Box-Muller algorithm.

```r
box.mul <- function(n){
  odd <-  FALSE
  if(n %% 2 != 0){
    odd <-  TRUE
    n <- n + 1
  }

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

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

  concat <- c(y1,y2)
  if(odd){
    return(head(concat, -1))
  }
  else{
    return(concat)
  }
}
```

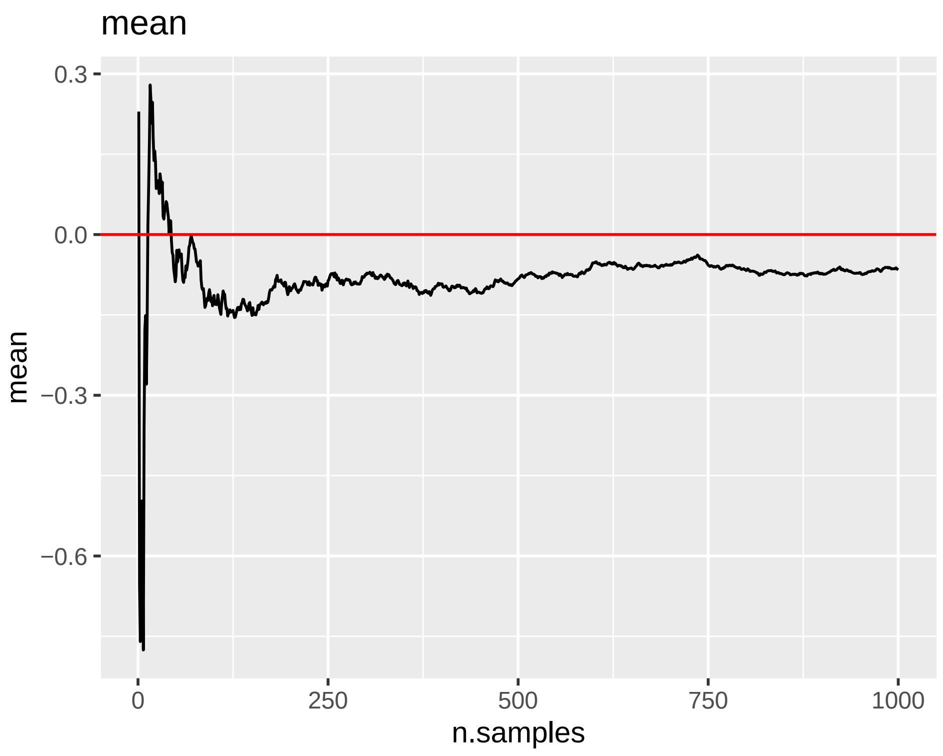
As usual, we compare the results of a simulation against the theoretical distribution.

```
n <- 1000
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..),
                 alpha = 0.8, fill = "#69b3a2", binwidth = 0.1) +
  geom_line(data = exact , aes(x = x, y = y), color = "#e0474c")
```

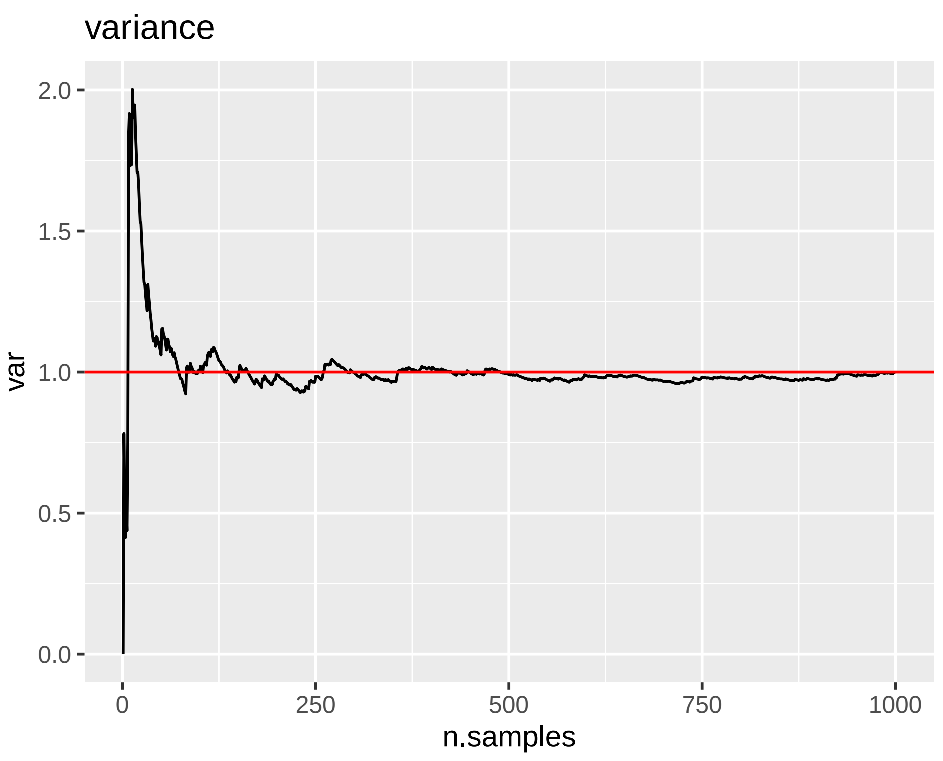


Finally, we compare the mean and variance resulting from the simulation with the theoretical values.

```
# 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")
```



```
# 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 $N_d(\mu, \Sigma)$ be a d-variate normal distribution with mean μ and covariance Σ. This is our target distribution. To simulate from it, let $x \sim N_d(0, I_d)$ be a standard d-variate normal distribution. Then it follows that

$$y = \mu + Ax \quad \implies \quad y \sim N(\mu, AA^T),$$

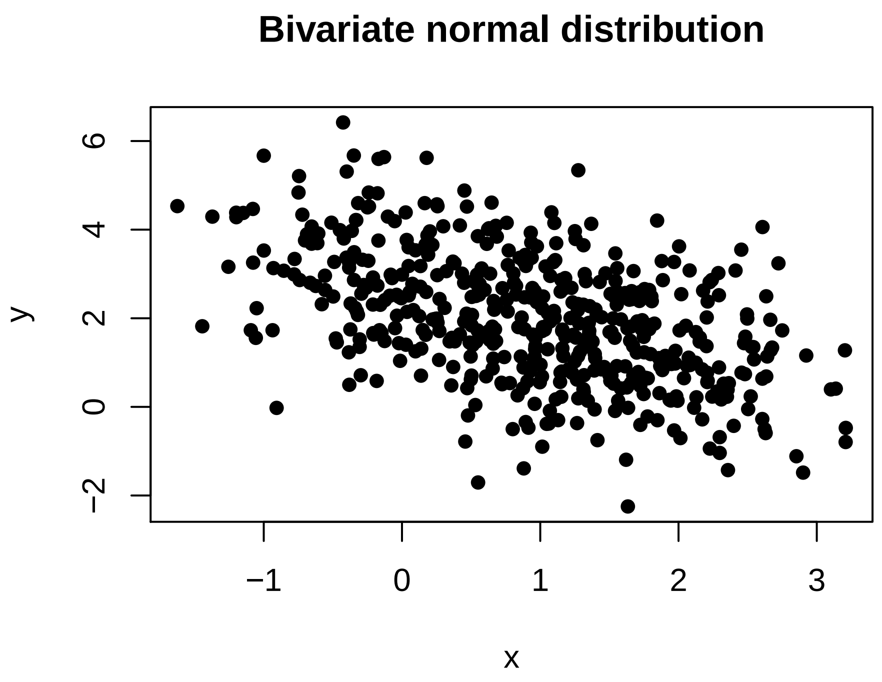
where A is a $d \times d$ matrix. Since the covariance matrix Σ is both symmetric and positive-definite, we can write $\Sigma = AA^T$ using the Cholesky decomposition. Thus, for a given μ and Σ, we are able to simulate from this distribution after simulating from $N_d(0, I_d)$.

Reusing our implementation of the Box-Muller algorithm from A.4, a function for generating realizations from the target is shown below:

```r
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:

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

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

# Plot first 500 points
plot(y[1,1:500], y[2,1:500], main="Bivariate normal distribution", xlab="x", ylab="y", pch=16)
```



Comparing the true mean and covariance with estimated values:

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

```
## [1] 0.9843779 2.0141289
```

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

```
##            [,1]       [,2]
## [1,]  1.0307664 -0.7399571
## [2,] -0.7399571  2.0673293
```

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 $\Sigma = \begin{pmatrix} 1 & -0.7 \\ -0.7 & 2 \end{pmatrix}$. This means that our function works properly.

Problem B

B.1

(a)

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,

$$a = \sqrt{\alpha - 1}.$$

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,

$$b_+ = \sqrt{\alpha + 1}.$$

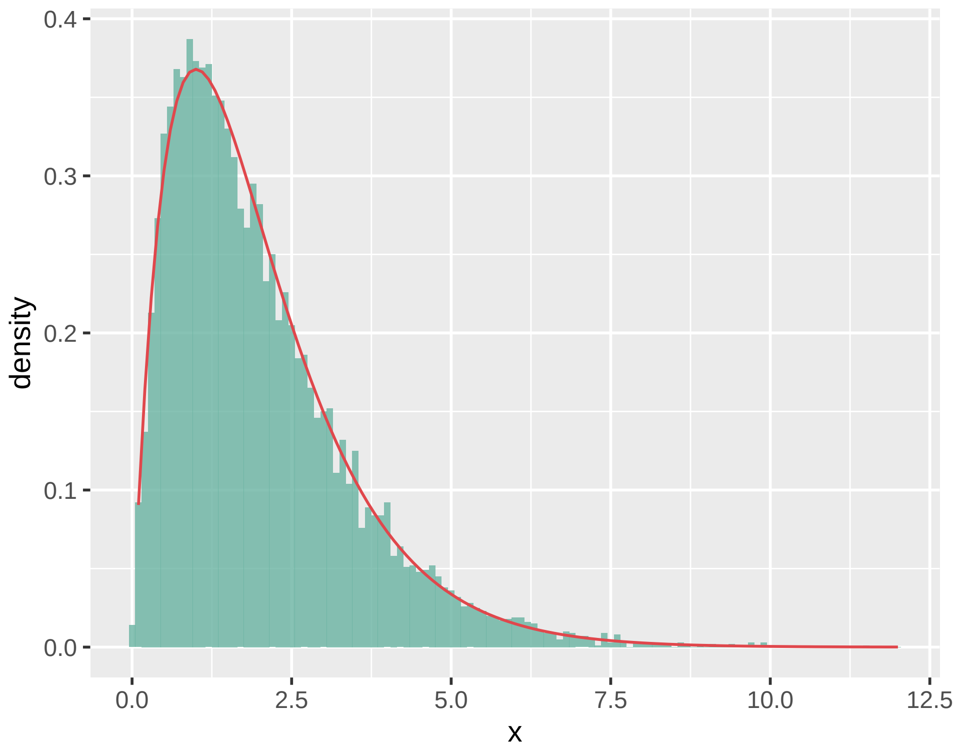
Finally, $b_- = -\sqrt{\sup_{x\leq 0} x^2 f^*(x)} = -\sqrt{0} = 0.$

(b)

```r
rou.gamma <- function(n, alpha){
  a <-  sqrt(alpha - 1)
  b.minus <- 0
  b.plus <- sqrt(alpha + 1)

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

  while(count < n){
    x1 <- a*runif(1, 0, 1)
    x2 <- b.minus + b.plus*runif(1, 0, 1)

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

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

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

sim.gamma <- rou.gamma(n, alpha)
x = seq(from = 0.1, to = 12, 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..),
                 alpha = 0.8, fill = "#69b3a2", binwidth = 0.1) +
  geom_line(data = exact , aes(x = x, y = y), color = "#e0474c")
```



```
sim.gamma$tries
```

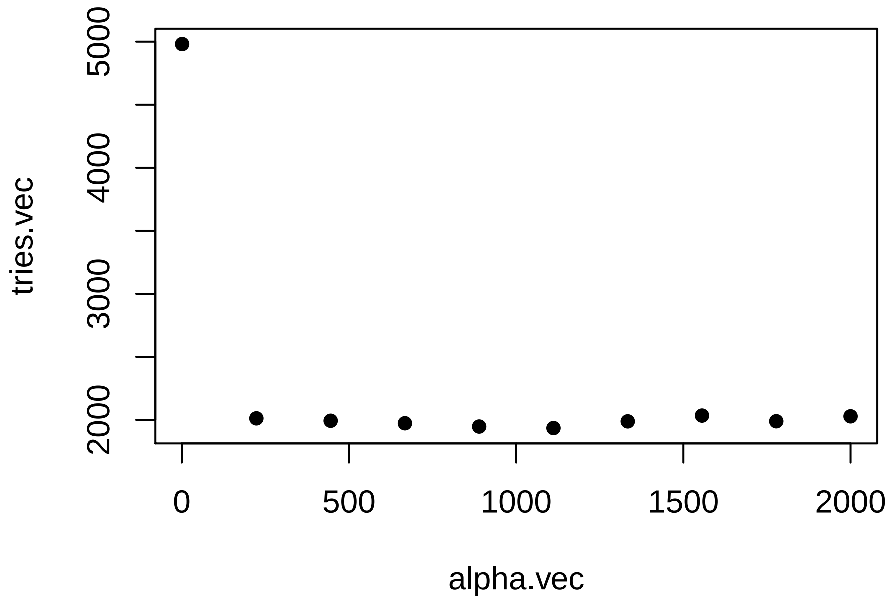
```
## [1] 34615
```

```
library(pracma)
```

```
##
## Attaching package: 'pracma'
```

```
## The following object is masked from 'package:purrr':
##
##     cross
```

```
n <- 1000
alpha.vec <- seq(1.01, 2000, length.out = 10)
tries.vec <- rep(0, 10)
for(i in 1:length(alpha.vec)){
  alpha <- alpha.vec[i]
  sim.gamma <- rou.gamma(n, alpha)
  tries.vec[i] = sim.gamma$tries
}

plot(alpha.vec, tries.vec, pch = 16)
abline(h = 3, col="blue")
```



The number of necessary tries is highest for low values of α, i.e. $\ln \alpha < 2$, while it stabilizes after that. This indicates that C_f constitutes a smaller proportion of $[0, a] \times [b_-, b_+]$. mer tolkning?? Tror det er feil. Tror antall tries skal øke med alpha.

B.3

We recall that the gamma distribution has probability density function (PDF)

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

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

```
sim.gamma <- 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)
}
```

Check that it works god:...

B.4

We repeat that $X \sim \text{Gamma}(\alpha, 1)$ and $Y \sim \text{Gamma}(\beta, 1)$, and note that their joint density is $f_{X,Y}(x,y) = f_X(x)f_Y(y)$, since they are independent. Next, we define $Z = \frac{X}{X+Y}$ and also introduce $V = X + Y$. Then, $X = ZV$ and $Y = V(1-Z)$, and, by the transformation formula, the joint density of Z and V is

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

Thus, the marginal distribution of Z is given as

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

which is what we wanted show.

Problem C

C.1

We observe that $\theta = \Pr\{X > 4\} = \int_4^\infty f(x) = \int_{-\infty}^\infty \mathbb{I}(x > 4)f(x) = \text{E}_f[\mathbb{I}(X > 4)]$. Then, by the method of Monte Carlo (MC) 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.

```
n <- 100000
x <- box.mul(n)
h <- (x > 4)
theta.hat <- 1/n * sum(h)
theta.hat
```

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

```
pnorm(4, lower.tail = FALSE)
```

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

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

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

We also utilize the central limit theorem, such that $\hat{\theta} \sim \mathcal{N}\left(\theta, \text{Var}[\hat{\theta}]\right)$. 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] 1e+10
```

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

c(lower, upper)
```

```
## [1] -9.599877e-06  2.959988e-05
```

C.2

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

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

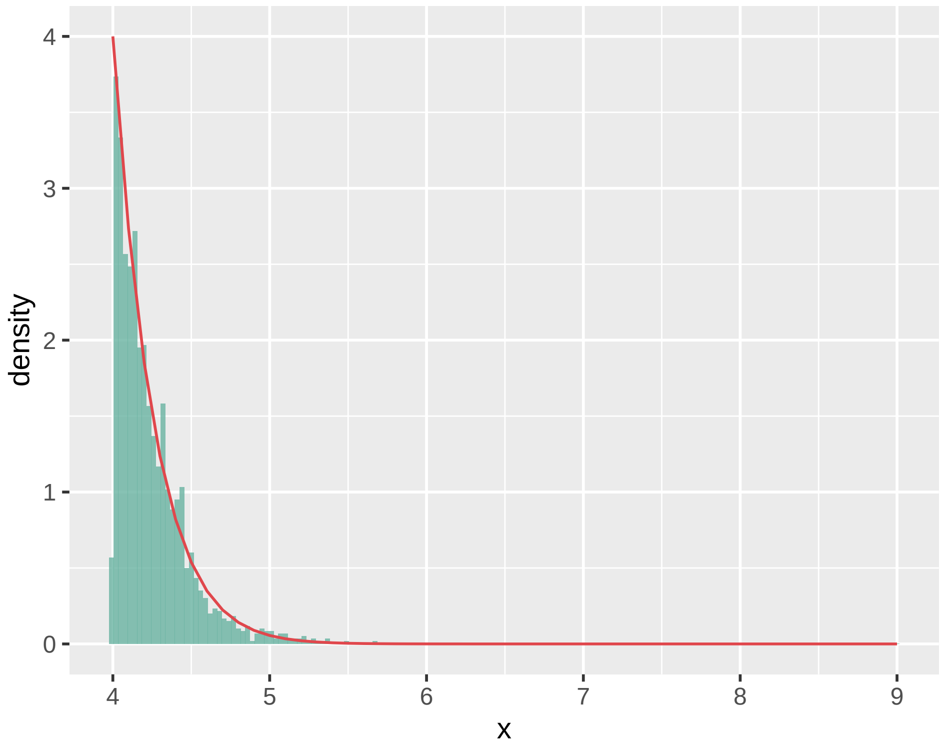
The CDF, $G(x)$, is thus given as

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

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:

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

```
sim <- inv.samp(2000)
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..),
                 alpha = 0.8, fill = "#69b3a2", binwidth = 0.03) +
  geom_line(data = exact , aes(x = x, y = y), color = "#e0474c")
```



We note that $g(x) > 0$ where $h(x)f(x) > 0$ show this??. 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] 3.167368e-05
```

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

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

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] 3.166407e-05 3.168330e-05
```

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

```
## [1] 4.156408e+16
```

We define the precision of the estimators as the reciprocal of the variance of the estimators. Hence we have Precision$(\hat{\theta}) = 10^{10}$ and Precision$(\hat{\theta}_{IS}) = 4.1564081 \times 10^{16}$. unsure how to compare them with respect to n...

C.3

(a)

The algorithm which samples n antithetic from g is given below.

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

(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 <- 100000
c <- exp(8)

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

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

```
## [1] 3.166607e-05
```

Finally, we compute a 95% confidence interval:

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

c(lower, upper)
```

```
## [1] 3.165636e-05 3.167579e-05
```

```
1/sigma.a^2
```

```
## [1] 4.069904e+16
```

We observe that the precision is lower than for regular importance sampling.