TMA4300: Exercise 1

Jim Totland, Martin Tufte

1/29/2022

Problem A

A.1

The exponential distribution has cumulative density function (CDF)

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

with rate parameter λ. By defining $u := F(x)$, we can express x as

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

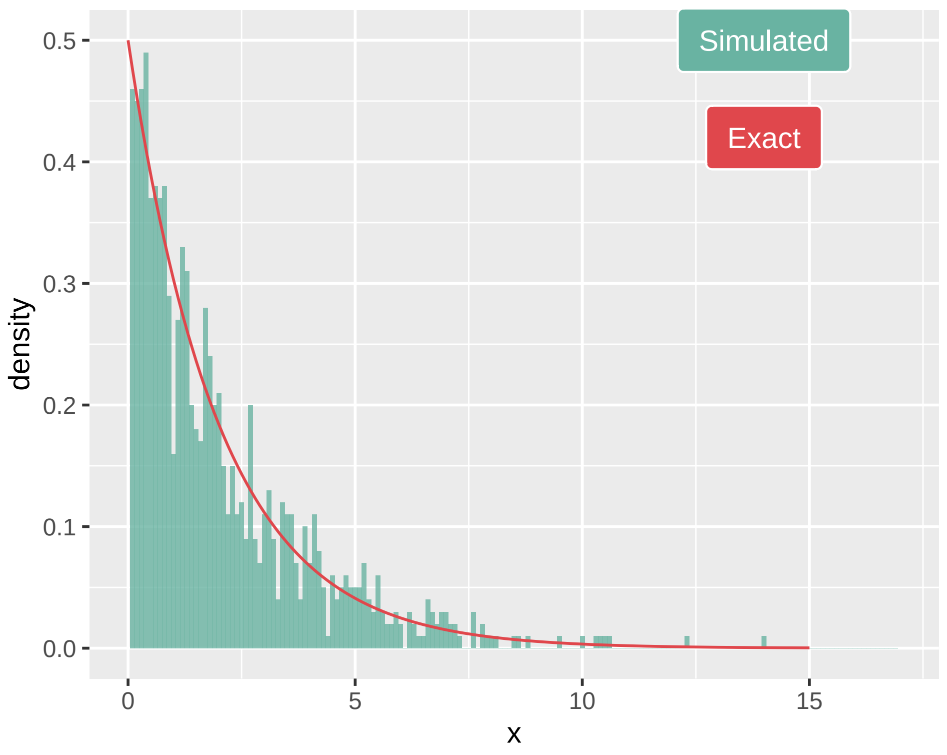
This means that we can use the *inversion method* to simulate from the exponential distribution. I.e., we let $U \sim \mathcal{U}_{[0,1]}$ and calculate $X = F^{-1}(U)$. Then, $X \sim \text{Exp}(\lambda)$. The function which simulates the exponential distribution is given below.

```
sim.exp <- function(rate, n){
  u <- runif(n,0,1)
  return(-1/rate * log(1 - u))
}
```

Next, we need to check if this gives reasonable results by comparing our simulated values to the theoretical knowledge.

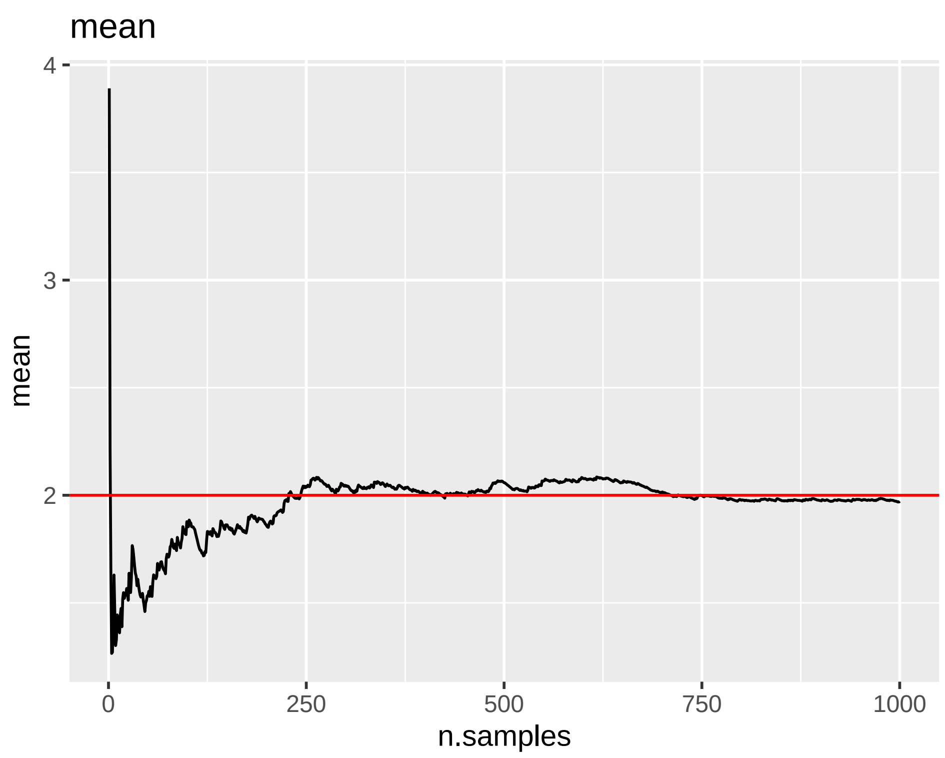
```
rate <- 0.5
n <- 1000
sim <- data.frame(x = sim.exp(rate, n))
x = seq(from = 0, to = 15, by = 0.1)
exact <- data.frame(x = x, y = rate*exp(-rate*x))

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).
```

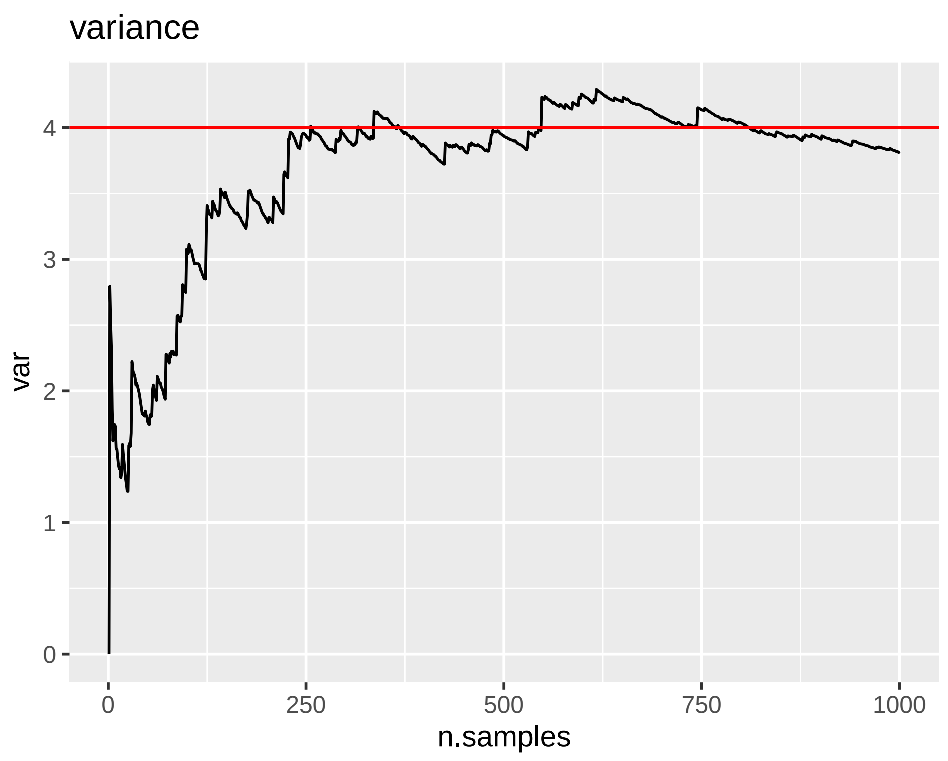


We also compare the estimated mean and variance to first and second central moments of the exponential distribution

```
# 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")
```



```
# 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.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.,

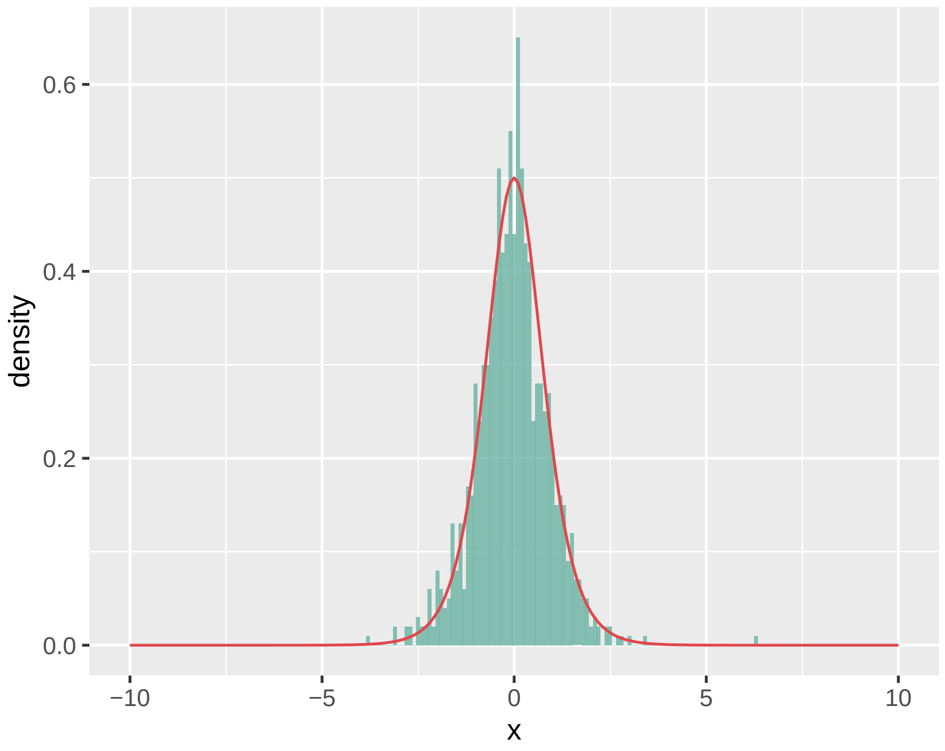
$$F^{-1}(x) = \frac{1}{\alpha} \ln\left(\frac{x}{1-x}\right).$$

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)
  return(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??

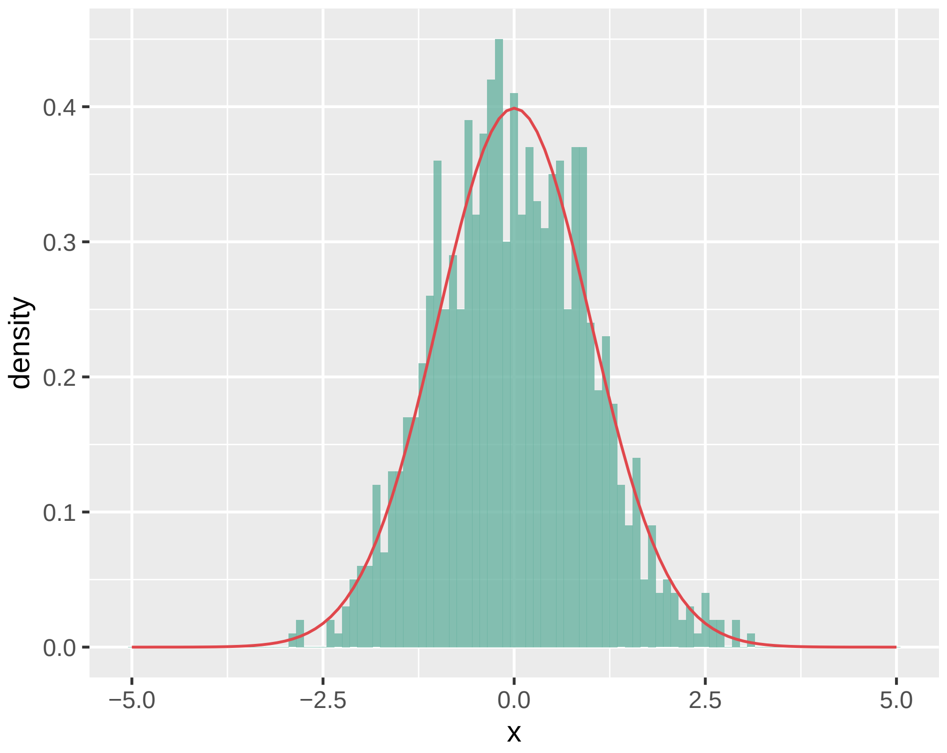
A.4

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)
  }
}
```

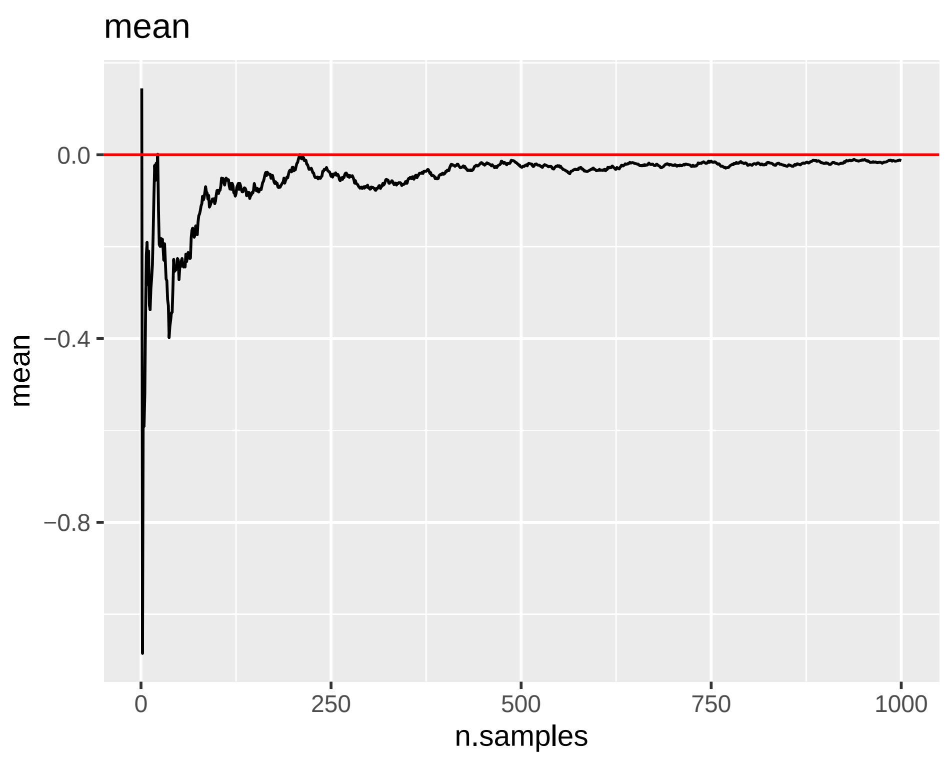
As usual, we compare the results of a simulation against the theoretical distribution.

```r
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")
```

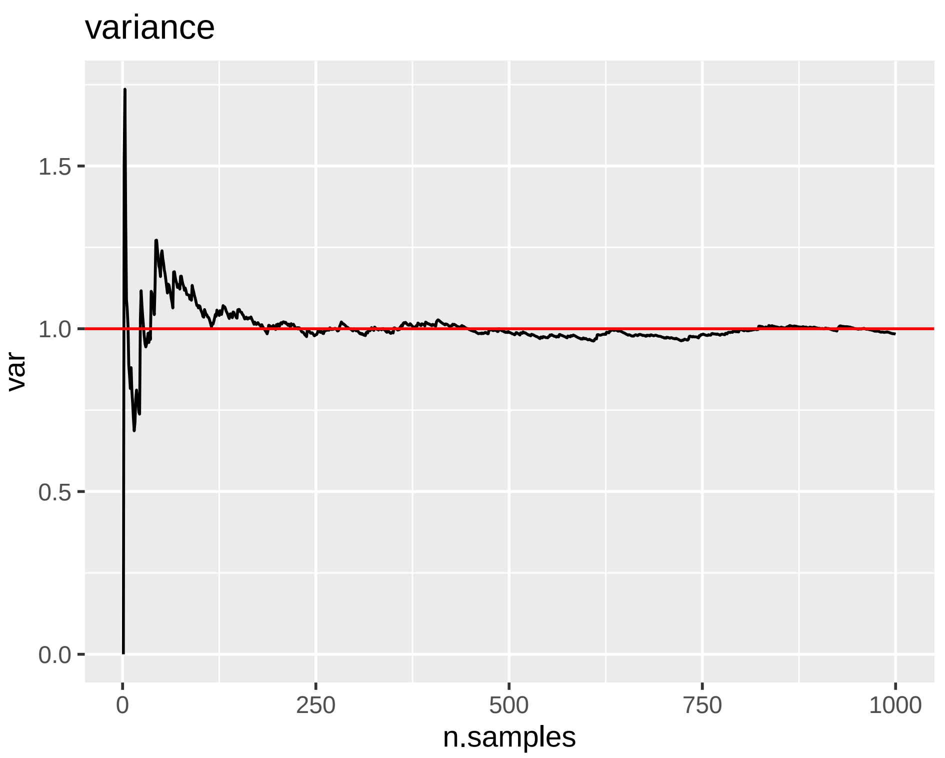


Finally, we compare the mean and variance resulting from the simulation with the theoretical values.

```
# 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

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

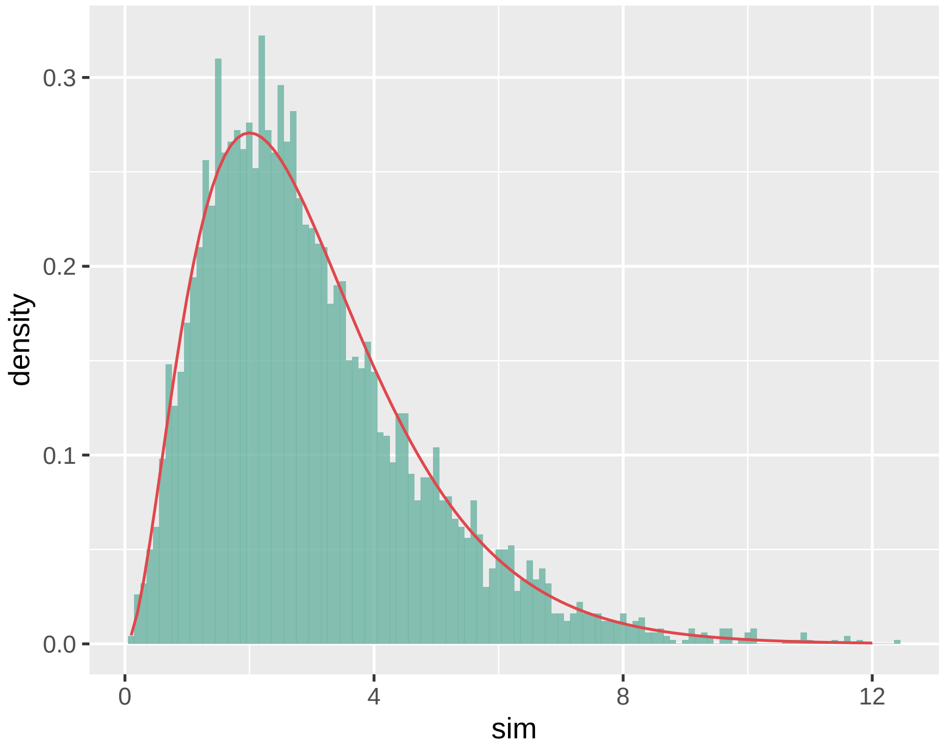
(b)

```r
alpha <-  1
a <-  sqrt(1-alpha)
b.minus <- 0
b.plus <- sqrt(alpha + 1)

rou.gamma <- function(n, a, b.minus, b.plus, alpha){
  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(data.frame(sim = result, tries = tries))
}
```

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

```r
n <- 5000
alpha <-  3
a <-  sqrt(alpha - 1)
b.minus <- 0
b.plus <- sqrt(alpha + 1)
```

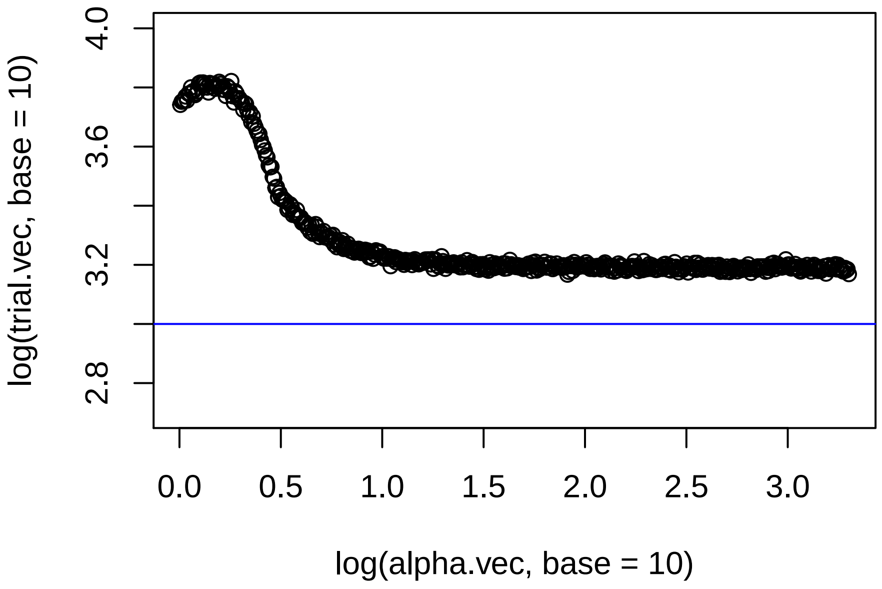
```
sim.gamma <- rou.gamma(n, a, b.minus, b.plus, alpha)
x = seq(from = 0.1, to = 12, by = 0.1)
exact <- data.frame(x = x, y = 1/gamma(alpha) * x^(alpha - 1) * exp(-x))

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



```
library(pracma)
```

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

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

```
n <- 1000
alpha.vec <- logseq(1.01, 2000, n = 500)
trial.vec <- rep(0, 100)
for(i in 1:length(alpha.vec)){
  alpha = alpha.vec[i]
  sim.gamma <- rou.gamma(n, a, b.minus, b.plus, alpha)
  trial.vec[i] = sim.gamma$tries[1]
}

plot(log(alpha.vec, base = 10), log(trial.vec, base = 10), ylim = c(2.7,4))
abline(h = 3, col="blue")
```



The number of necessary tries is highest for low values of α, and decreases fast, before it stabilizes around $10^{3.2}$. mer tolkning??

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:

```r
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