Mandatory Assigment 1

Jone Damås - 256140

2022-09-14

Exercise 1:

a.)

The function is a probability density function if the integral over the whole distribution equals 1:

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

$$\int_0^2 \frac{x}{2}\,dx = \frac{x^2}{4}|_0^2 = \frac{2^2}{4} - \frac{0^2}{4} = 1$$

Expected value is calculated by:

$$E(X) = \int_{-\infty}^{\infty} x \cdot f(x)\,dx$$

$$\int_0^2 \frac{x^2}{2}\,dx = \frac{x^3}{6}|_0^2 = \frac{2^3}{6} - \frac{0^3}{6} = \frac{4}{3} = 1.333$$

Variance is calculated by:

$$Var(X) = E(X^2) - E(X)^2$$

$$E(X^2) = \int_0^2 \frac{x^3}{2}\,dx = \frac{x^4}{8}|_0^2 = \frac{2^4}{8} - \frac{0^4}{8} = 2$$

$$Var(X) = 2 - (\frac{4}{3})^2 = \frac{2}{9} = 0.222$$

$$E\left(\frac{1}{2}(X+1)\right) = \int_{-\infty}^{\infty} \frac{1}{2}(x+1) \cdot f(x)\,dx$$

$$\frac{1}{2}\int_0^2 \frac{x^2}{2}\,dx + \frac{1}{2}\int_0^2 \frac{x}{2}\,dx = \frac{x^3}{12}|_0^2 + \frac{x^2}{8}|_0^2 = \frac{2^3}{12} + \frac{2^2}{8} = \frac{7}{6} = 1.166$$

$$E(X^3) = \int_{-\infty}^{\infty} x^3 \cdot f(x)\,dx$$

$$\int_0^2 \frac{x^4}{2}\,dx = \frac{x^5}{10}|_0^2 = \frac{2^5}{10} - \frac{0^5}{10} = \frac{16}{5} = 3.200$$

b.)

The cumulative distribution function is found by:

$$F(x) = \int_{-\infty}^{x} f\,(t)\,dt = \int_0^x \frac{t}{2}\,dx = \frac{t^2}{4}|_0^x = \frac{x^2}{4}$$

Using the cumulative distribution function:

$$P(X < 1) = F(1) = \frac{1^2}{4} = \frac{1}{4} = 0.250$$

$$P(0.5 < X < 1.5) = F(1.5) - F(0.5) = \frac{1.5^2}{4} - \frac{0.5^2}{4} = \frac{1}{2} = 0.500$$

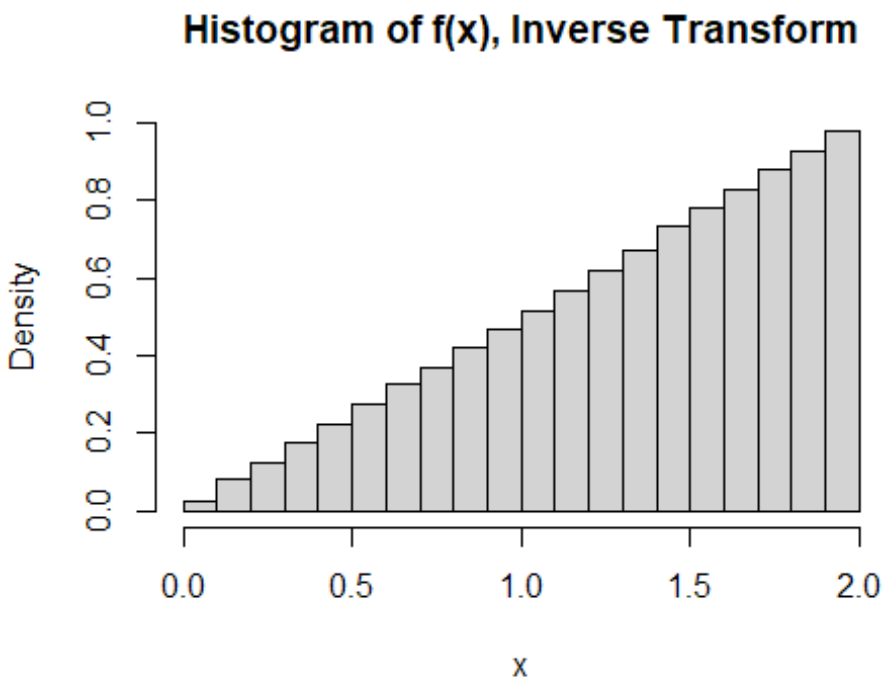
Finding the inverse cumulative distribution function:

$$u = \frac{x^2}{4} \Rightarrow x = \pm\sqrt{4u} \Rightarrow x = \sqrt{4u}$$

Only positive values for x gives positive probabilities, therefore, only the square root of 4 times u makes sense to use.

c.)

```
rm(list=ls())

# Function returning n random numbers with density f(x) by the inverse
transform method:
f_pdf_IT <- function(n){

  u <- runif(n, 0, 1)
  return(sqrt(4 * u))
}
# Start time measuring:
start_time <- Sys.time()

# Histogram of the 100000 random numbers with density f(x):
hist(f_pdf_IT(100000), probability=TRUE, ylim=c(0, 1), xlab='x',
main='Histogram of f(x), Inverse Transform')
```




```
# Stop time measuring:
end_time <- Sys.time()

cat('Duration:', end_time - start_time)

## Duration: 0.02047992
```

d.)

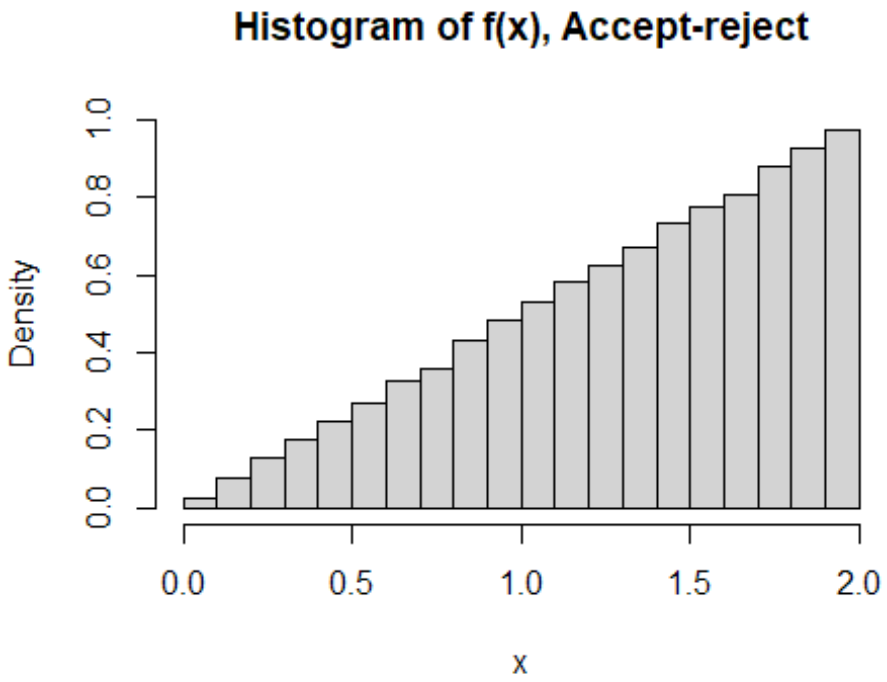
Setting g(x)=1/2. As g(x) is constant, f/g is then proportional to f, and the function is maximized at the higher endpoint at x=2. Therefore:

$$c = \frac{f(2)}{g(2)} = \frac{\frac{2}{2}}{\frac{1}{2}} = 2$$

e.)

```
# Function returning n random numbers with density f(x) by the accept-reject
method:

# c value obtained from calculations:
c <- 2

# Draw dunction
AR_draw <- function(){
  while(TRUE){
    x <- runif(1, min=0, max=2)
```

```
    a <- (x/2)/(c * 1/2)
    if(runif(1) < a) return(x)
  }
}

# Function which generates a pdf with the draw function:
f_pdf_AR <- function(n){
  v <- vector(length=n)
  for(i in 1:n){
    v[i] <- AR_draw()
  }
  return(v)
}

# Start measurement of time
start_time <- Sys.time()

# Histogram of the pdf:
hist(f_pdf_AR(100000), probability=TRUE, ylim=c(0,1), xlab='x',
main='Histogram of f(x), Accept-reject')
```




```
# Stopmeasurement of time
end_time <- Sys.time()

cat('Duration:', end_time - start_time)

## Duration: 1.08509
```

The inverse transform algorithm used 0.0391 seconds and the accept-reject algorithm used 1.243 seconds to execute the calculations and plot. This clearly demonstrates that the inverse transform algorithm is superior in terms of calculation speed.

f.)

Checking expectations and variance for the distributions created by the inverse transform and accept-reject methods:

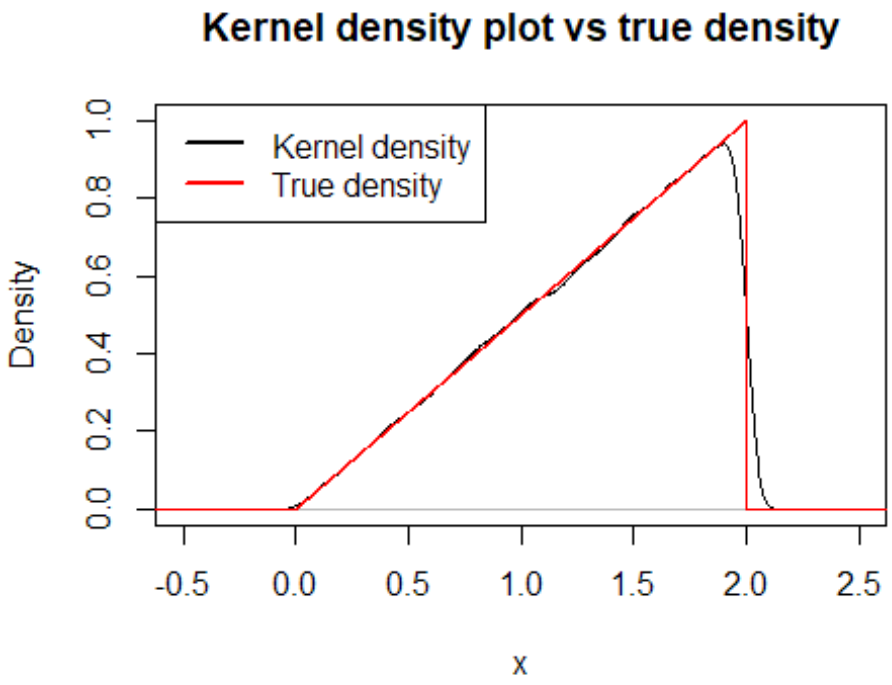
```
# Create matrix for result storage
data <- matrix(ncol=2, nrow=2)
# Expected value of the random numbers created from the inverse transform
method:
data[1, 1] <- mean(f_pdf_IT(100000))
# Expected value of the random numbers created from the accept-reject method:
data[2, 1] <- mean(f_pdf_AR(100000))

# Variance of the random numbers created from the inverse transform method:
data[1, 2] <- var(f_pdf_IT(100000))

# Variance of the random numbers created from the accept-reject method:
data[2, 2] <- var(f_pdf_AR(100000))

# Setting row and column names for result table:
colnames(data) = c('E(X)','Var(X)')
rownames(data) <- c('Inverse Transform','Accept-reject')

# Creating result table:
tbl <- as.table(data)
print(tbl)
```

```
##                          E(X)    Var(X)
## Inverse Transform 1.3356319 0.2217052
## Accept-reject     1.3337029 0.2222394
```

g.)

```
# Kernel density estimation:
plot(density(f_pdf_IT(100000)), xlim=c(-0.5, 2.5), ylim=c(0, 1), xlab='x',
main='Kernel density plot vs true density')

# pdf function:
f <- function(x){
  return(x/2)
}

# Creating true probability density from function:
x0 <- seq(-1, 3, 0.001)
f_pdf <- rep(0, times=length(x0))
f_pdf[x0>=0 & x0<=2] <- f(x0[x0>=0 & x0<=2])
lines(x0, f_pdf, col='red')
```

```
legend("topleft", legend = c("Kernel density", "True density"), lwd = 2, col
= c("black", "red"))
```



Exercise 2:

a.)

$$E(A) = E\left(\sum_{i=1}^{n} X_i\right) = n \cdot E(X_i) = n\mu_i = n$$

$$Var(A) = Var\left(\sum_{i=1}^{n} X_i\right) = n \cdot Var(X_i) = n \cdot \sigma^2 = n$$

$$E(B) = E\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{1}{n}(n \cdot \mu_i) = \mu_i = 1$$

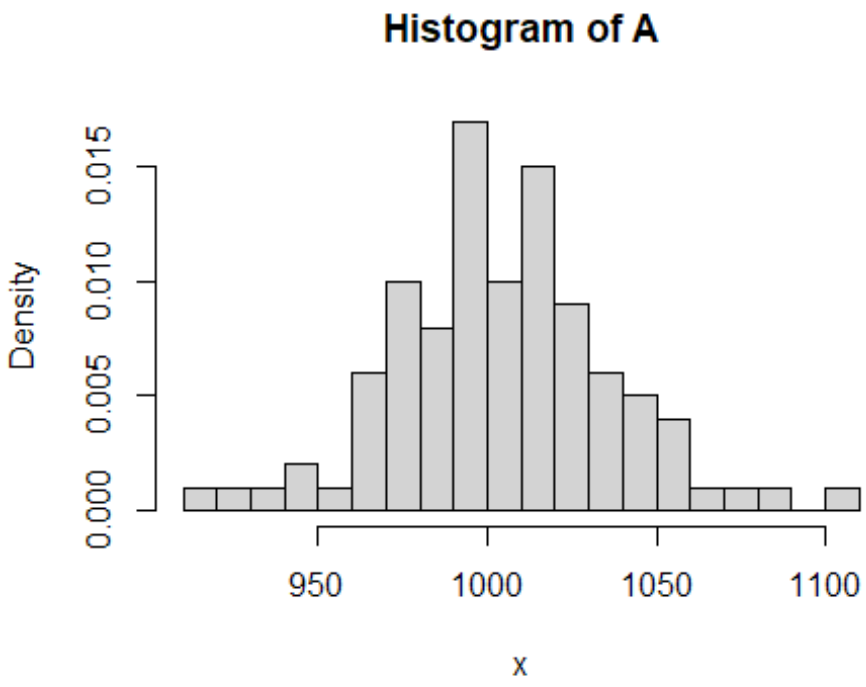
$$Var(B) = Var\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{1}{n^2} \cdot n \cdot Var(X_i) = \frac{\sigma^2}{n} = \frac{1}{n}$$

C has a normal distribution with ~N(0,1) when n is large, and is explained by the central limit theorem.

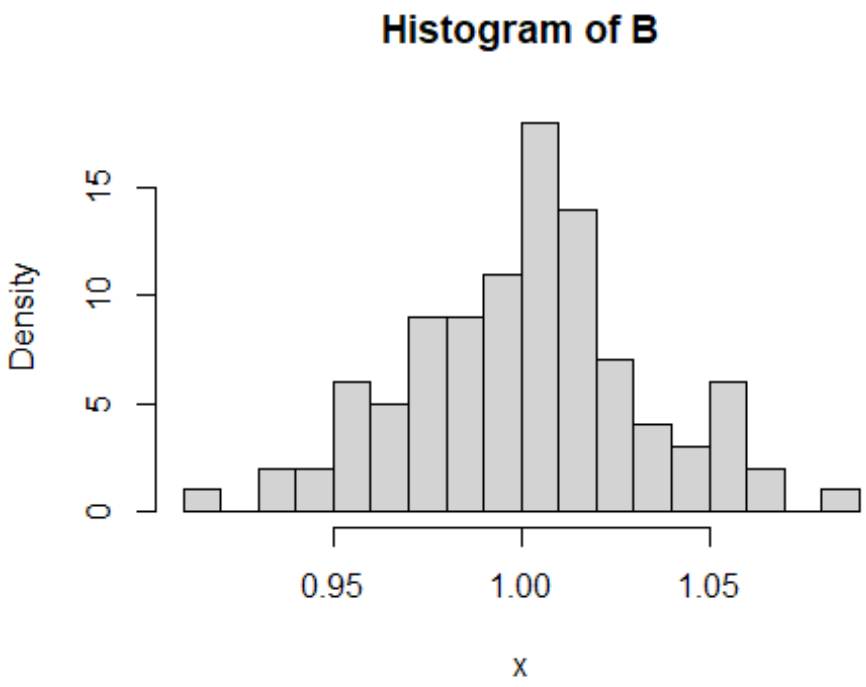
$$\sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n} X_i - 1\right)$$

b.)

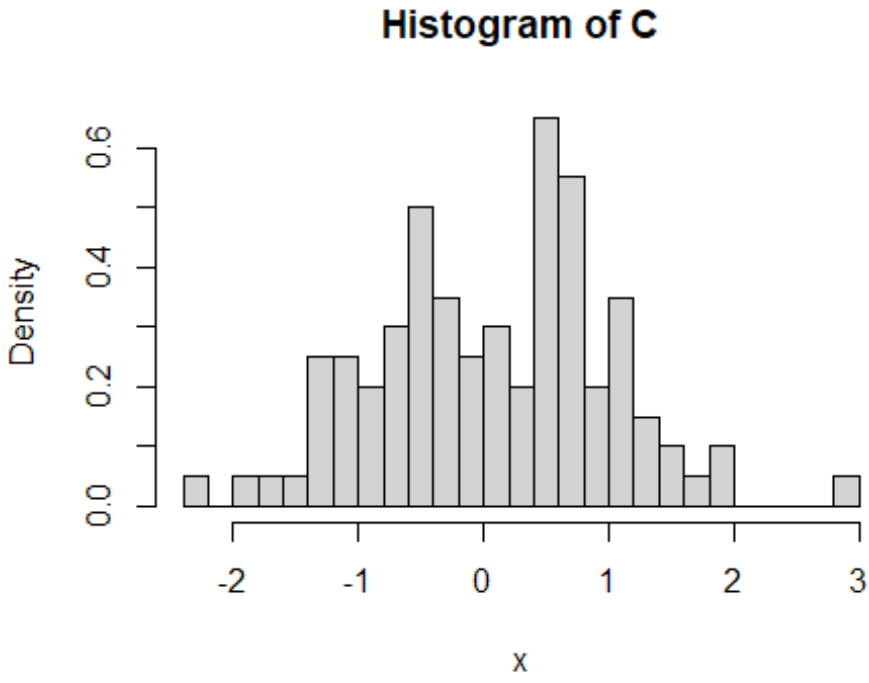
```
# Number of values:
n <- 1000
# Number of simulation:
Nsim <- 100


A_vec <- vector(length=Nsim)
B_vec <- vector(length=Nsim)
C_vec <- vector(length=Nsim)

# Data generation for distribution A, B and C:
for(i in 1:100){
  A_vec[i] <- sum(rexp(n))
  B_vec[i] <- mean(rexp(n))
  C_vec[i] <- sqrt(n)*(mean(rexp(n))-1)
}

# Histogram of distributions for A,B and C:
hist(A_vec, prob=T, nclass=20, main='Histogram of A', xlab='x')
```




```
hist(B_vec, prob=T, nclass=20, main='Histogram of B', xlab='x')
```




```
hist(C_vec, prob=T, nclass=20, main='Histogram of C', xlab='x')
```



c.)

```
n_list <- c(10, 100, 1000)

data <- matrix(nrow=length(n_list), ncol=2)

# Confidece interval generation:
for(j in 1:length(n_list)){
  B_vec <- vector(length=length(n_list))
  n <- n_list[j]

  for(i in 1:n){
    B_vec[i] <- mean(rexp(n_list))
  }
  B <- mean(B_vec)
  quant <- 1.96 * sd(B_vec)/sqrt(length(B_vec))
  data[j ,1] <- B - quant
  data[j ,2] <- B + quant
}

colnames(data) = c('lower', 'upper')
rownames(data) <- n_list

# Creating result table:
tbl <- as.table(data)
print(tbl)
```

```
##          lower     upper
## 10   0.4301195 1.2014134
## 100  0.8495418 1.0699240
## 1000 0.9432162 1.0162681
```

Exercise 3:

a.)

Considering the complexity of finding the inverse of the cumulative distribution function of f(x), the best approach would be to use the accept-reject method.

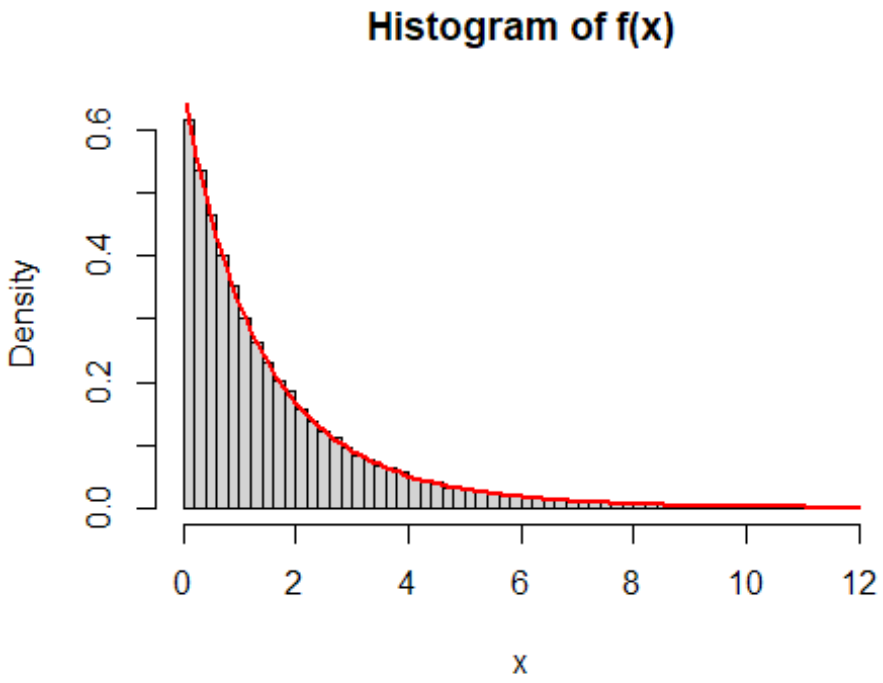
Setting g(x)=1/2. As g(x) is constant, f/g is then proportional to f, and the function is maximized at the lower endpoint at x=0. Therefore:

$$c = \frac{f(0)}{g(0)} = \frac{\frac{1}{3}\left(e^{-0} + e^{-\frac{0}{2}}\right)}{\frac{1}{2}} = \frac{4}{3}$$

b.)

```
fx <- function(x){
  return(1/3*(exp(-x)+exp(-x/2)))
}
```

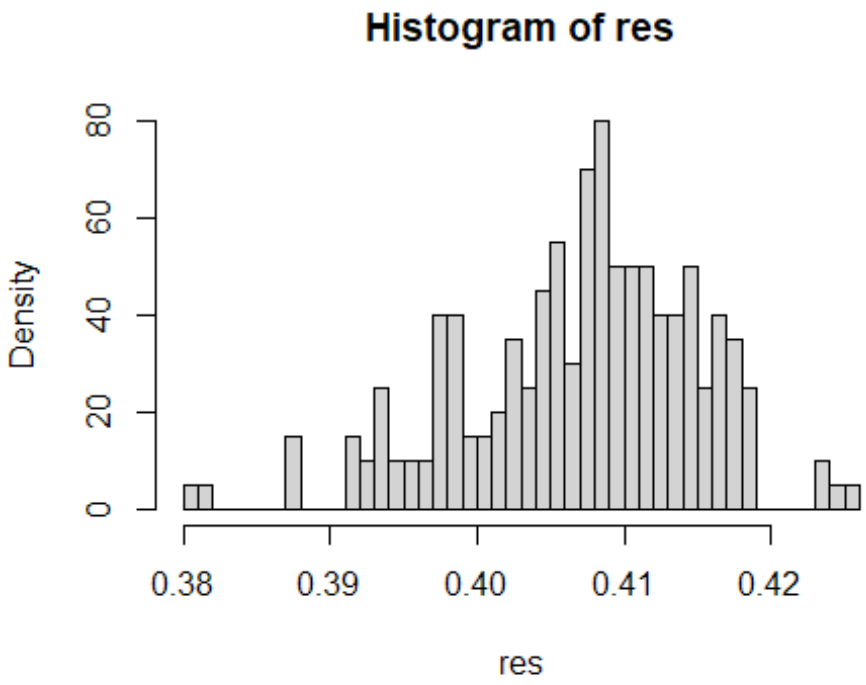
```r
# The constant c:
c <- 4/3

fx_AR_draw <- function(){
  while(TRUE){
    x <- runif(1, min=0, max=12)
    a <- fx(x)/(c * 1/2)
    if(runif(1) < a) return(x)
  }
}

fx_pdf_AR <- function(n){
  v <- vector(length=n)
  for(i in 1:n){
    v[i] <- fx_AR_draw()
  }
  return(v)
}



# Adding true density plot:
hist(fx_pdf_AR(100000), prob=T, main='Histogram of f(x)', xlab= 'x',
nclass=50)
curve(fx, col='red', type='l', lwd=2.0, xlim=c(0, 12), add=T)
```



Exercise 4:

a.)

```
# Load the lawn.R script:
source('lawn.R')

sim_Etime <- function(e, n){
  result <- vector(length = n)
  for(i in 1:n){
    result[i] <- sim.lawn(e)
  }
  return(result)
}

res <- sim_Etime(5, 200)
hist(res, prob=T, nclass=50)
```



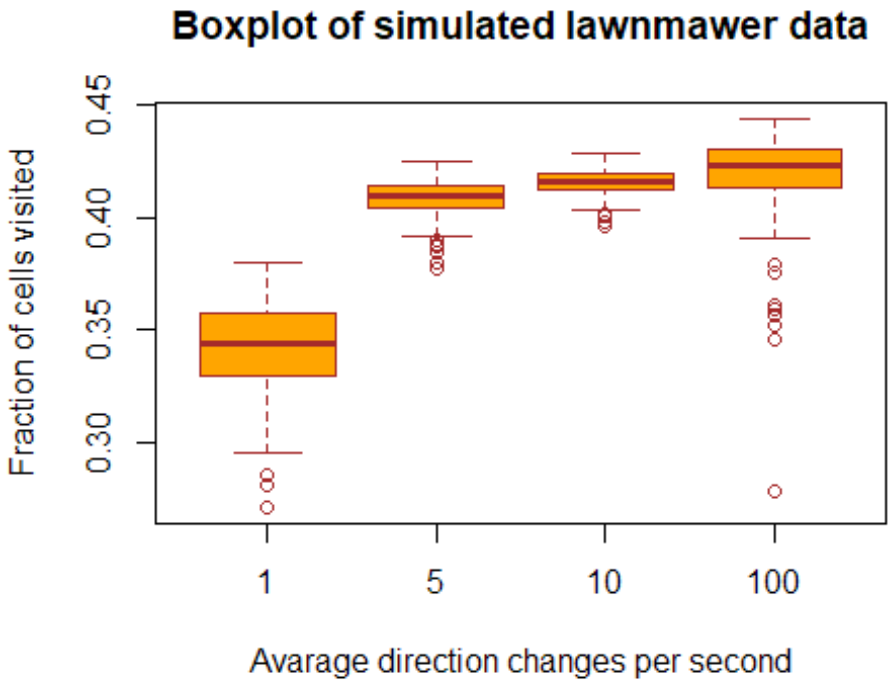

```
cat('Mean:', mean(res), '\n')

## Mean: 0.407066

cat('Median:', median(res), '\n')

## Median: 0.4081

cat('Standard deviation:', sd(res), '\n')

## Standard deviation: 0.007982602
```

b.)

```
# sequence of Etime values
Etime <- c(1.0, 5.0, 10.0, 100.0)
# Etime dataframe and table column names
names <- c('1', '5', '10', '100')

data <- matrix(nrow=length(Etime), ncol = 3)

for(i in 1:length(Etime)){

  result <- sim_Etime(Etime[i], 200)

  data[i, 1] <- mean(result)
  data[i, 2] <- median(result)
  data[i, 3] <- quantile(result, probs=0.05)

  if(i == 1){
    df <- data.frame('1'=result, check.names = FALSE)
  }
  else{
    df[names[i]] <- result
  }
}

colnames(data) = c('Mean','Median', '0.05-Quantile')
rownames(data) <- names

# Create table
tbl <- as.table(data)
print(tbl)
```

```
##          Mean    Median 0.05-Quantile
## 1   0.3424963 0.3438667     0.3025933
## 5   0.4084213 0.4095000     0.3935867
## 10  0.4156483 0.4161667     0.4047167
## 100 0.4194097 0.4234667     0.3916167
```

```
boxplot(df, col="orange", border="brown", main='Boxplot of simulated
lawnmawer data', xlab='Avarage direction changes per second', ylab='Fraction
of cells visited')
```



Based on the Boxplot results, the best choice is arguably the Etime=10 setting because of the smaller standard deviation, even though it has a lower fraction of the cells visited compared to Etime=100. The also Etime=100 has some lone points with a considerably lower fraction than that of Etime=10.