Mandatory Assigment 3

Jone Damås - 256140

2022-10-19

Exercise 1:

a.)

$$A = \int_0^{\infty} \exp(-x)\sqrt{x} \cdot dx = \int_0^{\infty} \sqrt{x} \cdot f(x) \cdot dx$$

$$\text{where} \quad f(x) = exp(-1) \quad \Rightarrow \lambda = 1$$

$$\hat{\theta_A} = \frac{1}{n}\sum_{i=1}^{n} \sqrt{X_i} \quad \text{where} \quad X \sim \exp(1)$$

b.)

$$B = \int_0^{10} \exp(-x)\sqrt{x} \cdot dx = \int_0^{\infty} I(x < 10)\sqrt{x} \cdot f(x) \cdot dx$$

$$\text{where} \quad f(x) = exp(-1) \quad \Rightarrow \lambda = 1$$

$$\hat{\theta}_B = \frac{1}{n}\sum_{i=1}^{n} I(X_i < 10)\sqrt{X_i} \quad \text{where} \quad X \sim \exp(1)$$

c.)

$$\hat{\theta}_C = \frac{2}{n}\sum_{i=1}^{n} \sqrt{1 + \cos(X_i)} \quad \text{where} \quad X \sim \text{Uniform}(-1, 1)$$

$$\hat{\theta_D} = \frac{2}{n}\sum_{i=1}^{n} \sqrt{1 + \sin(X_i)} \quad \text{where} \quad X \sim \text{Uniform}(-1, 1)$$

d.)

$$f(x) = \frac{1}{1^{\frac{3}{2}}\Gamma(\frac{3}{2})} x^{\frac{3}{2}-1} e^{-\frac{x}{1}} = \frac{2}{\sqrt{\pi}}\sqrt{x} \cdot e^{-x}$$

$$\hat{\theta}_{IS,A} = \frac{1}{n}\sum_{i=1}^{n} \frac{g(x)}{f(x)} = \frac{1}{n}\sum_{i=1}^{n} \frac{\sqrt{X_i} \cdot e^{-X_i}}{\frac{2}{\sqrt{\pi}}\sqrt{X_i} \cdot e^{-X_i}} = \frac{1}{n}\sum_{i=1}^{n} \frac{\sqrt{\pi}}{2}$$

$$\hat{\theta}_{IS,B} = \frac{1}{n}\sum_{i=1}^{n} \frac{I(x < 10)g(x)}{f(x)} = \frac{1}{n}\sum_{i=1}^{n} \frac{I(X_i < 10)\sqrt{X_i} \cdot e^{-X_i}}{\frac{2}{\sqrt{\pi}}\sqrt{X_i} \cdot e^{-X_i}} = \frac{1}{n}\sum_{i=1}^{n} I(X_i < 10)\frac{\sqrt{\pi}}{2}$$

The estimator ends up as a constant and gives the exact solution of the integral in the case of integral A, and a very good approximation for integral B. Therefore, this method is a good method for estimating A and B.

e.)

$$f(x) = \frac{1}{(\frac{1}{4})^4 \Gamma(4)} x^{4-1} e^{-4x} = \frac{128}{3} x^3 \cdot e^{-4x}$$

$$\hat{\theta}_{IS,A} = \frac{1}{n} \sum_{i=1}^{n} \frac{g(x)}{f(x)} = \frac{1}{n} \sum_{i=1}^{n} \frac{\sqrt{X_i} \cdot e^{-X_i}}{\frac{128}{3} X_i^3 \cdot e^{-4X_i}} = \frac{1}{n} \sum_{i=1}^{n} \frac{3}{128} X_i^{-\frac{5}{2}} \cdot e^{3X_i}$$

$$\hat{\theta}_{IS,B} = \frac{1}{n} \sum_{i=1}^{n} \frac{I(x < 10) g(x)}{f(x)} = \frac{1}{n} \sum_{i=1}^{n} \frac{I(X_i < 10) \sqrt{X_i} \cdot e^{-X_i}}{\frac{128}{3} X_i^3 \cdot e^{-4X_i}} = \frac{1}{n} \sum_{i=1}^{n} I(X_i < 10) \frac{3}{128} X_i^{-\frac{5}{2}} \cdot e^{3X_i}$$

We see that f(x) is not close to g(x) in shape, which does not make for a flat g(x)/f(x). This is not great for reducing variance.

f.)

$$V_i = a + b - U_i = -1 + 1 - U_i = -U_i$$

$$\hat{\theta}_{AT,C} = \frac{b-a}{n} \left(\sum_{i=1}^{n/2} g(U_i) + \sum_{i=1}^{n/2} g(V_i) \right) = \frac{2}{n} \left(\sum_{i=1}^{n/2} \sqrt{1 + \cos(U_i)} + \sum_{i=1}^{n/2} \sqrt{1 + \cos(-U_i)} \right)$$

$$V_i = a + b - U_i = -1 + 1 - U_i = -U_i$$

$$\hat{\theta}_{AT,D} = \frac{b-a}{n} \left(\sum_{i=1}^{n/2} g(U_i) + \sum_{i=1}^{n/2} g(V_i) \right) = \frac{2}{n} \left(\sum_{i=1}^{n/2} \sqrt{1 + \sin(U_i)} + \sum_{i=1}^{n/2} \sqrt{1 + \sin(-U_i)} \right)$$

```r
# estimators for d)
est_A_d <- mean(rep(sqrt(pi)/2, 100000))

est_B_d <- mean((rgamma(100000, shape=3/2, rate=1) <= 10)*sqrt(pi)/2)

# estimators for e)
x <- rgamma(100000, shape=4, rate=1/4)
est_A_e <- mean(3/128*x^(-5/2)*exp(3*x))

x <- rgamma(100000, shape=4, rate=1/4)
est_B_e <- mean((x <= 10)*3/128*x^(-5/2)*exp(3*x))

# estimators for f)

# antithetic Monte Carlo:
u <- runif(100000/2, -1, 1)
v <- -u

est_C <- 2*mean(sqrt(1+cos(c(u, v))))

est_D <- 2*mean(sqrt(1+sin(c(u, v))))

print(paste('Estimaton of integral A with gamma(3/2,1):', round(est_A_d, 8)), quote=F)
```

g.)

```
## [1] Estimaton of integral A with gamma(3/2,1): 0.88622693
```

```
print(paste('Estimaton of integral B with gamma(3/2,1):', round(est_B_d, 8)), quote=F)
```

```
## [1] Estimaton of integral B with gamma(3/2,1): 0.88612058
```

```
print(paste('Estimaton of integral A with gamma(4,1/4):', round(est_A_e, 8)), quote=F)
```

```
## [1] Estimaton of integral A with gamma(4,1/4): 6.60515628308458e+86
```

```
print(paste('Estimaton of integral B with gamma(4,1/4):', round(est_B_e, 8)), quote=F)
```

```
## [1] Estimaton of integral B with gamma(4,1/4): 15180630.0295914
```

```
print(paste('Estimaton of integral C:', round(est_C, 8)), quote=F)
```

```
## [1] Estimaton of integral C: 2.71179674
```

```
print(paste('Estimaton of integral D:', round(est_D, 8)), quote=F)
```

```
## [1] Estimaton of integral D: 1.91752986
```

As expected, the estimations of integral A and B using the gamma(3/2, 1) importance sampling method gave very accurate estimations. Its however apparent that the estimates for integral A and B using the gamma(4, 1/4) importance sampling method is not good at all. The antithetic Monte Carlo estimators work well for the estimations of integral C and D as expected by theory.

Exercise 2:

a.)

$$X \sim binomial(n, V_c(2x)^{-d})$$

Because:

1.) We have a fixed number of random variables generated, n.

2.) Each outcome of $\mathbf{1}(u_j)$ is independent.

3.) $\mathbf{1}(u_j)$ has only two outcomes, either 0 or 1.

4.) The probability of the outcome is the same for each value.

The probability of success in a binomial distribution is defined by:

$$p = \frac{E(X)}{n}$$

We see that $E(X)$ is X in our case, since $X = \sum_{i=1}^{n} \mathbf{1}(u_j)$.

p is therefore:

$$\hat{V}_c = \frac{(2c)^d}{n} X \Rightarrow \frac{X}{n} = \hat{V}_c(2c)^{-d} \Rightarrow p = \hat{V}_c(2c)^{-d}$$

b.)

$$\frac{SD(\hat{V}_1)}{V_1} < r \Rightarrow \frac{\sqrt{Var(\hat{V}_1)}}{V_1} < r \Rightarrow \frac{\sqrt{Var((2)^d X/n)}}{V_1} < r \Rightarrow \frac{Var((2)^d X)}{V_1^2 n} < r^2$$

$$\Rightarrow \frac{2^{2d} Var(X)}{V_1^2 n} < r^2 \Rightarrow n > \left(\frac{2^d \sigma_X}{V_1 r}\right)^2$$

Inserting expression for V_1:

$$n > \left(\frac{2^d \sigma_X}{(2^d/d!)r}\right)^2 \Rightarrow n > \left(\frac{d! \cdot \sigma_X}{r}\right)^2$$

As d is in a factorial expression and squared for the minimum number of simulations. There would be needed a rapidly increasing number of simulations as the dimension (d) increases.

```r
# Estimation function:
est_V <- function(d, c, n){
  X <- 0
  for(j in 1:n){
    u <- runif(d, min=-c, max=c)
    if(sum(abs(u))<=c){
      X = X + 1
    }
  }
  V <- X*(2*c)^d/n
  return(V)
}

# True value function:
tv <- function(d){
  return(2^d/(factorial(d)))
}

data <- matrix(nrow=3, ncol=2)

data[1,1] <- est_V(2, 1, 100000)
data[2,1] <- est_V(4, 1, 100000)
data[3,1] <- est_V(8, 1, 100000)

data[1,2] <- tv(2)
data[2,2] <- tv(4)
data[3,2] <- tv(8)

colnames(data) <- c('Estimated values', 'True values:')
rownames(data) <- c('d=2:', 'd=4:', 'd=8:')

table <- as.table(data)
table
```

c.)

```
##      Estimated values True values:
## d=2:      1.996480000  2.000000000
## d=4:      0.663680000  0.666666667
## d=8:      0.005120000  0.006349206
```

```r
V_c <- function(d, c, n, s){
  sum_vec <- vector(length=n)
  for(j in 1:n){
    z <- rnorm(d, mean = 0, sd=s)
    I <- 0
    # Indicator function on the vector z:
    if(sum(abs(z))<=c){
      I <- 1
    }
    product <- prod(dnorm(z, sd=s))
    sum_vec[j] <- I / product
  }
  V <- mean(sum_vec)
  SE <- sd(sum_vec)/tv(d)
  return(c(V, SE))
}

data <- matrix(nrow=3, ncol=2)

V_d2 <- V_c(2, 1, 100000, 1)
V_d4 <- V_c(4, 1, 100000, 1)
V_d8 <- V_c(8, 1, 100000, 0.2)

data[1,1] <- V_d2[1]
data[2,1] <- V_d4[1]
data[3,1] <- V_d8[1]

# Relative standard error
data[1,2] <- V_d2[2]
data[2,2] <- V_d4[2]
data[3,2] <- V_d8[2]

colnames(data) <- c('Estimated values', 'Relative standard error:')
rownames(data) <- c('d=2:', 'd=4:', 'd=8:')

table <- as.table(data)
table
```

d.)

```
##      Estimated values Relative standard error:
## d=2:      2.008805118              1.655825441
## d=4:      0.653039395              8.081144623
## d=8:      0.006234713              2.609870156
```

```r
library(mvtnorm)

mu <- c(1/4, 1/4, 1/4, 1/4)

sig <- matrix(c(1, 1/4, 1/4, 1/4,
                1/4, 1, 1/4, 1/4,
                1/4, 1/4, 1, 1/4,
                1/4, 1/4, 1/4, 1), nrow=4)

a_sim <- function(n, c, mu, sigma){
  d <- length(mu)
  sum_vec <- vector(length=n)
    for(j in 1:n){

    x <- runif(4, -c, c)

    u <- dmvnorm(x, mean=mu, sigma=sigma)

    I <- 0
    if(sum(abs(x))<=c){
      I <- 1
    }
    sum_vec[j] <- u * I
  }
  return(c((2*c)^4*sum(sum_vec)/n, sd(sum_vec)))
}

print(paste('a=', round(a_sim(10000, 1, mu, sig)[1], 5)), quote=F)
```

e.)

```
## [1] a= 0.0157
```

```r
print(paste('SD=', round(a_sim(10000, 1, mu, sig)[2], 5)), quote=F)
```

```
## [1] SD= 0.00483
```

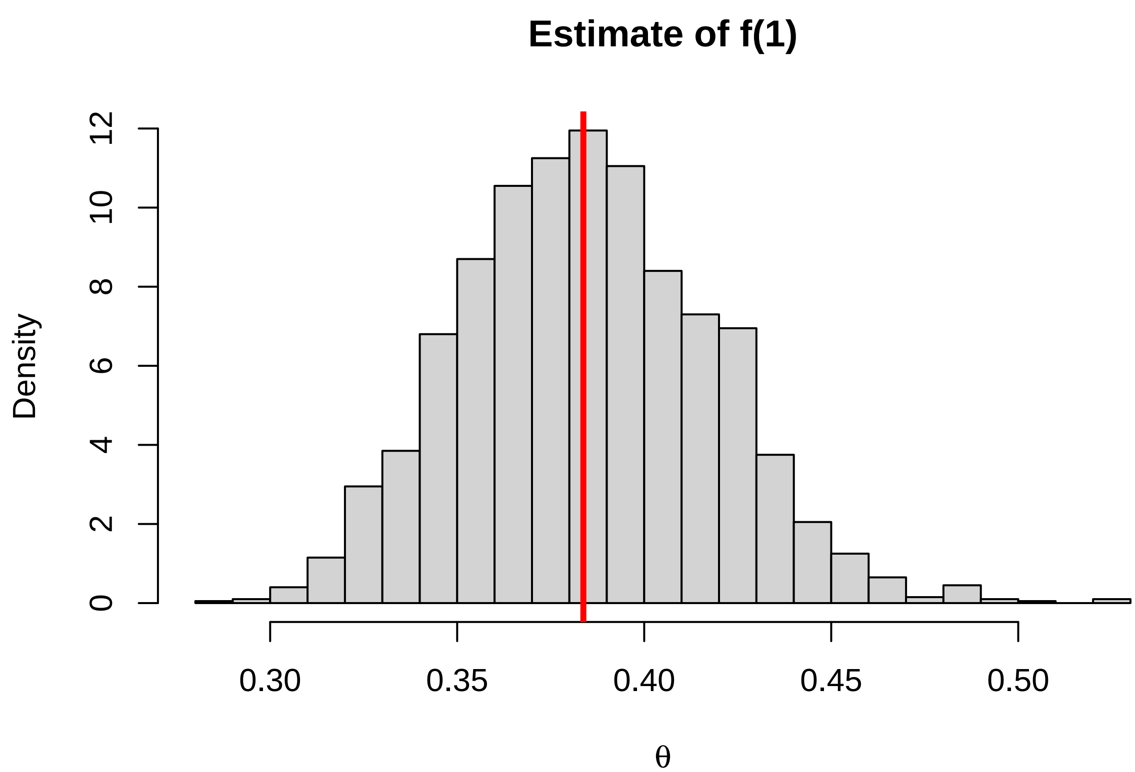
Exercise 3:

```r
x <- read.table("M3data.txt")$V1

# KDE estimator
f1.hat <- function(x){
  return(density(x,from=1,to=1,n=1)$y)
}

f1.o <- f1.hat(x)
print(paste0("Estimate of f(1) : ",f1.o), quote=F)
```

a.)

```
## [1] Estimate of f(1) : 0.383732864871009
```

```r
B <- 2000

n <- length(x)
f1.estimate <- numeric(B)

for(i in 1:B){
  f1.estimate[i] <- f1.hat(sample(x,size=n,replace=TRUE))
}

hist(f1.estimate,prob=TRUE,nclass=sqrt(B)/2, main='Estimate of f(1)', xlab=expression(theta))
abline(v=f1.hat(x),col="red",lwd=3)
```




```r
# Standard error
print(paste('Standard error:', round(sd(f1.estimate), 6)), quote=F)
```

```
## [1] Standard error: 0.033415
```

```r
# Bias
print(paste('Bias:', round(mean(f1.estimate) - f1.o, 6)), quote=F)
```

```
## [1] Bias: 0.00077
```

```r
# Function for estimating f(1) using a histogram:
snr <- function(x){
  n <- length(x)
  w <- 3.49/2 * sd(x) * n^(-1/3)

  # Number of observation in interval:
  A <- sum(x > (1-w) & x < (1+w))

  # Scott's Normal Reference rule:
  estimator <- A / (2*n*w)
  return(estimator)
}

# Import data from file:
data <- read.table("M3data.txt")$V1
# Apply function to data:
print(paste('Estimator:', round(snr(data), 5)), quote=F)
```

b.)

```
## [1] Estimator: 0.37401
```

```r
B <- 2000

n <- length(x)
f1.estimate <- numeric(B)

for(i in 1:B){
  f1.estimate[i] <- f1.hat(sample(x,size=n,replace=TRUE))
}

# Standard error
print(paste('Standard error:', round(sd(f1.estimate), 6)), quote=F)
```

c.)

```
## [1] Standard error: 0.032099
```

```r
# Bias
print(paste('Bias:', round(mean(f1.estimate) - snr(data), 6)), quote=F)
```

```
## [1] Bias: 0.009663
```

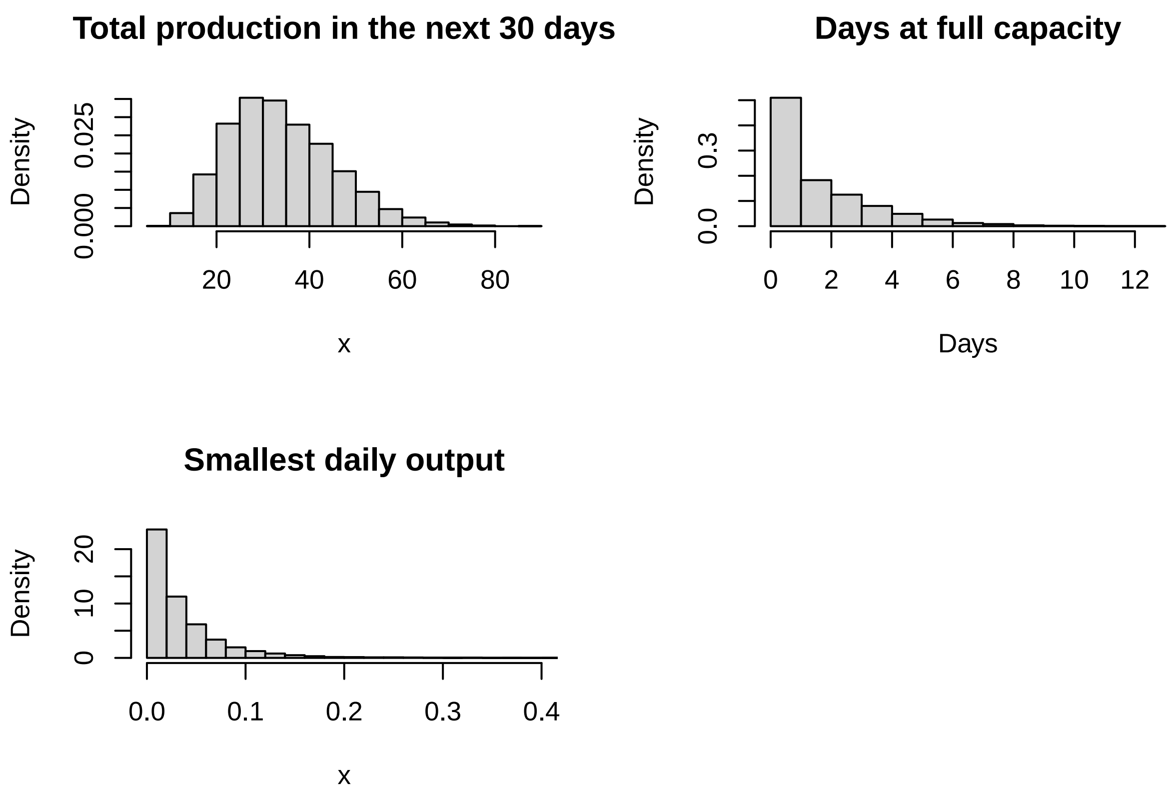
The alternative method using a histogram has a larger Bias than the KDE-based estimate of f(1), and would therefore be the preferred method for estimation.

Exercise 4.)

```r
Nsim <- 10000

res_matrix <- matrix(nrow=Nsim, ncol=3)

for(i in 1:Nsim){
  n <- 32
  x <- numeric(n)
  x[1] <- 1.0
  x[2] <- 0.8

  for(j in 3:n){
    x[j] <- min(4.0,rexp(1)*(0.4+0.7*x[j-1]+0.1*x[j-2]))
    }

  # Finding the total production the next 30 days:
  res_matrix[i,1] <- sum(x[3:n])
  # Finding number of days at full capacity:
  res_matrix[i,2] <- length(x[x == 4.0])
  # Finding the minimum value of the simulation:
  res_matrix[i,3] <- min(x)
}

par(mfrow=c(2,2))

hist(res_matrix[,1], prob=T, main='Total production in the next 30 days', breaks=20, xlab='x')
hist(res_matrix[,2], prob=T, main='Days at full capacity', xlab='Days')
hist(res_matrix[,3], prob=T, main='Smallest daily output', breaks=30, xlab='x', xlim=c(0,0.4))

tp <- c(round(mean(res_matrix[,1]), 3), round(quantile(res_matrix[,1], c(.25, .50, .75), names = F), 2)
fc <- c(round(mean(res_matrix[,2]), 4), round(quantile(res_matrix[,2], c(.25, .50, .75), names = F), 2)
mn <- c(round(mean(res_matrix[,3]), 4), round(quantile(res_matrix[,3], c(.25, .50, .75), names = F), 4)

res_data <- rbind(tp, fc, mn)

colnames(res_data) <- c('Mean', 0.25, 0.50, 0.75)
rownames(res_data) <- c('Total production:', 'Days at full cap.:', 'Smallest daily output:')

table <- as.table(res_data)
table
```

a.)

```
##                            Mean    0.25     0.5    0.75
## Total production:      33.9510 25.5600 32.6800 41.1800
## Days at full cap.:      1.9031  0.0000  1.0000  3.0000
## Smallest daily output:  0.0359  0.0089  0.0218  0.0471
```




```r
n <- length(res_matrix[,1])

S_vector <- vector(length=n)
for(i in 1:n){
  S_vector[i] <- max(c(0, 30 - res_matrix[i,1]))
}

price <- rgamma(n, shape=10, scale=1)
res <- S_vector * price

money <- quantile(res, 0.95, names=F)
print(paste('Money needed for 95% certanty:', round(money, 2)), quote=F)
```

c.)

```
## [1] Money needed for 95% certanty: 126.53
```