Computational Statistics Assignment 1

CID: 01389741

February 20, 2022

Question 1

For this question we will use the last two digits of my CID, $a_1 = 4$ and $a_2 = 1$. The pdf is defined as:

$$f(x) = k\left(1 + \cos(x)^3\right)\exp\left(-\frac{1}{2}x^2\right) \quad \text{for } -\frac{3}{2} \leq x \leq \frac{3}{2} \tag{1}$$

a)

For this pdf to be well defined, its integral over all its range should equal 1. Namely,

$$\int_{-3/2}^{3/2} f(x)\ dx = 1 \tag{2}$$

Performing such an integration using the function `integrate` in `R`, we see that the pdf is well defined when $k = 0.2972959$.

b)

Now we use an acceptance-rejection algorithm to generate samples from f using a Normal distribution of mean 0 and variance 1.

The acceptance-rejection algorithm consists of sampling from a uniform distribution with bounds 0 and 1, and accepting a new sample if this value is less or equal to $\frac{f(x)}{Cg(x)}$, where $f(x)$ is the function we want to sample from, $g(x)$ the density of a distribution we are approximating is similar to $f(x)$ (in this case $N(0,1)$), and C a constant of proportionality. To calculate C we require that $\frac{f(x)}{g(x)} \leq C$ for any x, in our case, we note that:

$$\frac{k\left(1 + \cos(x)^3\right)\exp\left(-\frac{1}{2}x^2\right)}{\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}x^2\right)} \leq \frac{2k}{\frac{1}{\sqrt{2\pi}}} = \frac{2\sqrt{2}k}{\pi} \tag{3}$$

which means that $C = \frac{2\sqrt{2}k}{\pi}$ and $k = 0.2972959$.

We can plot the theoretical cumulative distribution function and our estimate using the acceptance-rejection method on the same graph to see the performance. The theoretical cumulative distribution function is assumed to be the numerical integration of the function $f(x)$. The plot can be seen in figure 1.

c)

Now we will use the samples created in part b) to compute $E(X^2)$ using Monte Carlo integration. We know that the Monte Carlo estimate is given by:

$$\hat{I}_{MC} = \frac{1}{n}\sum_{i=1}^{n} X_i^2 \tag{4}$$

where X_i are independently and identically distributed according to the CDF of $f(x)$ given in 1. Note that these are going to be generated as samples using the Acceptance-Rejection algorithm used in part b).

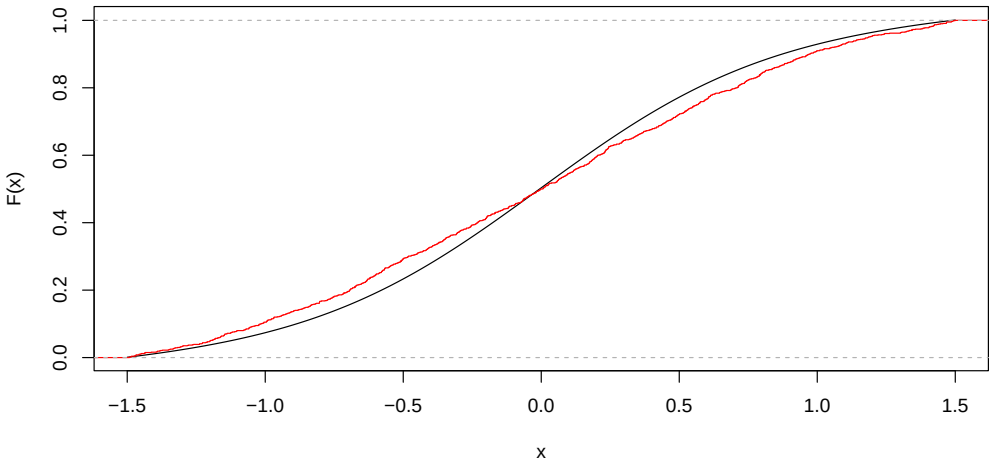

Figure 1: Plot of theoretical (black line) an approximate (red line) CDF of function defined in equation 3 using acceptance-rejection algorithm.

The Monte Carlo estimate was found to be 0.55296. The $1-\alpha$ confidence interval for I is given by $\left[\hat{I}_{MC} - \frac{sz_{\alpha/2}}{\sqrt{n}}, \hat{I}_{MC} + \frac{sz_{1-\alpha/2}}{\sqrt{n}}\right]$ where n is the sample size, s is the sample standard deviation and z_γ is defined by $P(Z \leq z_\gamma) = \gamma$ for $Z \sim N(0,1)$. The 90% confidence interval for the Monte Carlo estimate is [0.54331, 0.56260].

d)

In order to obtain a 90% confidence interval with length at most equal to 10^{-4} we would need 371,848,633 samples. This can be seen from the proportionality of the error of a Monte Carlo estimate, namely:

$$l\sqrt{(10^4)}/\sqrt{N} = 10^{-4} \tag{5}$$

where l is the length of the confidence interval. This is actually a very large number of samples and the computational cost would be immense.

Question 2

In this question we consider a function of the form:

$$I(\sigma) = \frac{1}{2}\int_{-\infty}^{\infty} x\left[\frac{1}{\pi\left(1+x^2\right)} + \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-5)^2}{2\sigma^2}\right)\right]dx \quad \text{for } \sigma \in \mathcal{S} := \{0.01, 0.1, 0.5, 1\}$$

Note that by plotting the function we obtained some intuition towards how this function behaves. As seen in figure 2 the function has a peak at the value of 5, and for different values of σ the peak is sharper. We chose not to plot the value of $\sigma = 0.01$ since the peak was extremely sharp and the rest of the lines would not be visible. Note also that at the x value close to 0 all functions have a small oscillation which is not very visible in figure 2.

When using the function `integrate` we can see that for $\sigma = 0.01, 0.1$ the result is 0. This makes sense because the bound of the integral are infinity, and the function is so sharply peaked that the integration approximates its area as 0. For $\sigma = 0.5, 1$, integration yields the values 2.5 for both σ values, and this shows that the peak was wide enough this time for some of the cuts i approximating the function to include the peak.

To implement a numerical method that gives roughly the correct answers, we will split the integral into 5 parts, each part corresponding to a range of integration. The intervals will be: $[-\infty$,-1], [-1,1], [1,3], [3,6] and [6, $\infty]$. These intervals correspond to places in the graph where we intuitively decided that need to be careful in case the function has a suspicious

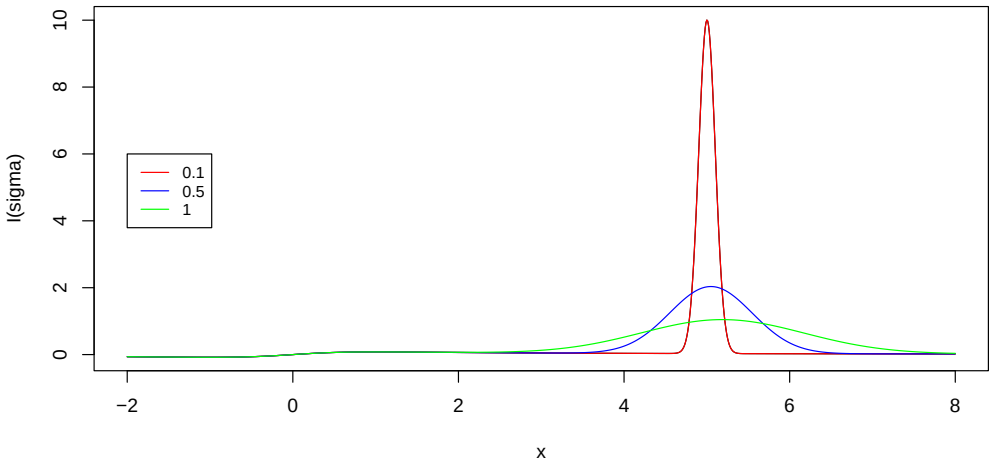

Figure 2: Plot of the function $I(\sigma)$ against x for the values of sigma 0.1, 0.5, 1.

behaviour. This split was done for $\sigma = 0.1, 0.5, 1$ and for the $\sigma = 0.01$ we split the integral to 3 parts $[-\infty, 4.8]$, $[4.8, 5.1]$ and $[5.1, \infty]$ (since it is more sharply peaked). With this split, all integrals give a value of 2.5, irrespective with the value of σ. The absolute error for all estimates is of the order of 10^{-5}.

b)

For importance sampling we will approximate $E(\phi(x)) \approx E(\phi(Y)\frac{f(Y)}{g(Y)})$ where in our case $\phi(x) = X$ (we just want to find the expected value), $f(x)$ is given by:

$$f(x) = \frac{1}{2}\left[\frac{1}{\pi\left(1+x^2\right)} + \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-5)^2}{2\sigma^2}\right)\right] \tag{6}$$

, $g(x)$ is the density of a Normal distribution with mean 0 and variance 4, namely:

$$g(x) = \frac{1}{4\sqrt{2\pi}}\exp\left[-\frac{1}{2}\left(\frac{x}{4}\right)^2\right]$$

and $Y_1, \ldots, Y_k \sim_{i.i.d} \mathcal{N}(0, 4)$.

Note that the condition of $\phi(x)f(x) > 0 \rightarrow g(x) > 0$ is satisfied in this example so we can indeed use importance sampling. Performing the calculation we summarise results in table 1.

σ	0.01	0.1	0.5	1
$I(\sigma)$	2.154(606)	2.679(216)	2.572(113)	2.539(75)

Table 1: Summary of Importance sampling estimations of $E(X)$ for $I(\sigma)$

part c)

The estimate of Monte Carlo integration is given by:

$$\hat{I}_{MC} = \frac{1}{n}\sum_{i=1}^{n} X_i \tag{7}$$

where X_is are drawn from the CDF of 6. Since the CDF of this function is not known, an inversion formula or acceptance-rejection method should be used to sample from f. Due to time constraints this was not done.

part d)

The sampling error is defined as $\frac{S}{\sqrt{n}}$, where n the size of the sample and S is the Sample variance defined as:

$$S^2 = \frac{\sum (x_i - \bar{x})^2}{n-1} \tag{8}$$

where x_i is the value of one observation and $\bar{x}$ the mean of all observations. The performance of the sampler in part c) could not be computed since in part c) we were not able to find the estimate. However, if we had a numerical value, we would compute the standard error of the Monte-Carlo estimate and adjust it by the acceptance probability and compare it with the standard error for importance sampling for the different values of σ. The lowest standard error would elude the most efficient sampler.

Question 3

In this question we are considering the two-dimentional density:

$$f(x,y) = k\left(1 + \frac{(x-2y)^2}{4}\right)^{-5/2}\left(1 + \frac{(x+2y)^2}{3}\right)^{-2} \tag{9}$$

In order to compute $E[(X-Y)^2]$ we will use importance sampling. The proposal distribution will be $X, Y \sim N(0,1)$ where X and Y are *independent*. This means that the joint density of X and Y can be expressed as $g(x,y) = g_1(x)g_2(y)$ where $g_1(x)$ and $g_2(y)$ are the density functions of standard normal distributions. This means that we can compute:

$$E[(X-Y)^2] = E\left[(X-Y)^2 \frac{f(x_i,y_i)}{g(x_i,y_i)}\right] \quad i = 1, 2, \ldots \tag{10}$$

Note that since this is a density we can integrate $f(x,y)$ and equate it to 1 in order to get the value of constant k. Integrating with the `integrate` function we get a value $k = 0.55$. We can proceed with importance sampling and we obtain an estimate of 1.108 with a sampling error of 0.031. We know that the true value should be equal to 1 since we normalised the function, therefore we perform more iterations to check the behaviour of our estimate. These are summarised in table 2.

Estimate	Standard Error	Max $\left[\frac{f(x_i,y_i)}{g(x_i,y_i)}\right]$
1.11	0.03	31.50
1.15	0.06	15.04
1.12	0.04	25.41
1.10	0.03	15.35
1.49	0.29	450.06
1.24	0.10	77.11
1.18	0.05	118.89
1.20	0.05	50.47
1.15	0.03	22.20
1.13	0.04	57.21
1.07	0.02	8.43
1.18	0.04	40.76
1.08	0.02	14.48
1.14	0.05	15.27
1.24	0.07	17.85

Table 2: 15 Iterations using importance sampling with a Standard Normal distribution.

We can clearly observe that the maximum weight vector which is the ratio $\frac{f(x_i,y_i)}{g(x_i,y_i)}$ overshoots at some particular values. This is not ideal when we are using importance sampling and it suggests we have to look at the tail behaviour of the distributions.

Indeed, as seen in figure 3 we can clearly see that there is a problem towards large n values.

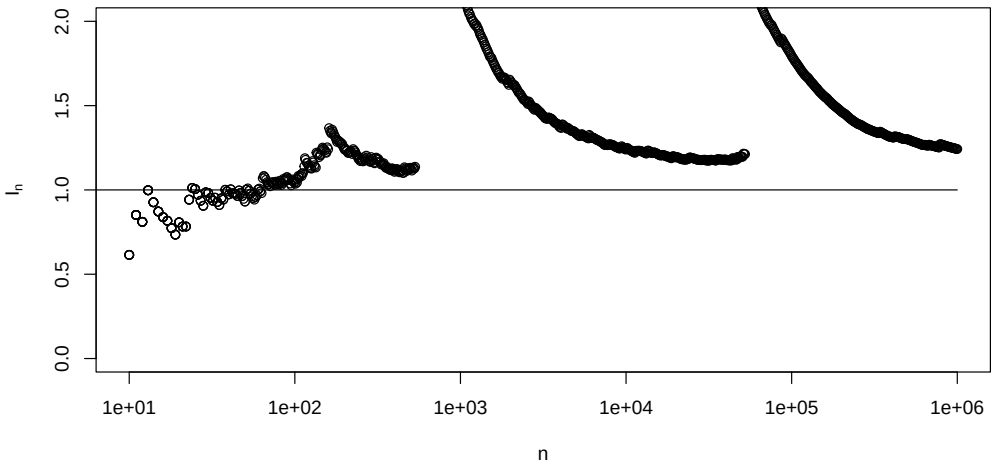

Figure 3: Plot of I_n for different n values with a Normal distribution. Horizontal line is the exact value of this integral.

We could potentially use a different distribution instead of a Normal one since we know that the Normal in its tails tends to 0 very quickly. We suggest a Cauchy distribution which is not tending to 0 very quickly in its tail. Table 3 summarises the results and we see a stabilization in the Maximum weight, although the estimate now is further away from 1. A small change in the location and the scale of the distribution could potentially resolve this issue. This is definetly an improvement though since now we have a stable algorithm with lower standard error and we can increase the number of iterations to obtain an estimate closer to the real value without worrying about tail behaviour.

Estimate	Standard Error	Max $\left[\frac{f(x_i,y_i)}{g(x_i,y_i)}\right]$
1.28	0.02	5.44
1.32	0.03	5.44
1.30	0.02	5.44
1.33	0.02	5.44
1.33	0.02	5.44
1.29	0.02	5.44
1.31	0.02	5.44
1.35	0.03	5.44
1.29	0.03	5.44
1.30	0.02	5.44
1.28	0.02	5.44
1.26	0.02	5.44
1.33	0.03	5.44
1.29	0.02	5.44
1.34	0.02	5.44

Table 3: Table of 15 Iterations using importance sampling with a Cauchy Distribution.

Question 4

In this question we will be estimating

$$I = E[X\mathbb{1}\{X \geq 1\}] \text{ where } X \sim \mathcal{N}(0,1) \tag{11}$$

a)

The Monte Carlo estimate can be approximated as:

$$E[X\mathbb{1}\{X \geq 1\}] \approx \frac{1}{n}\sum_{i=1}^{n} X_i\mathbb{1}\{X_i \geq 1\} \tag{12}$$

Computing this in `R` we obtain an estimate of 0.231 with a sampling error of 0.006.

b)

Using importance sampling with a shifted exponential distribution with density $g(y) = \exp -(y-1)$ we can approximate the integral $E(\phi(x)) \approx E(\phi(Y)\frac{f(Y)}{g(Y)})$ where $\phi(Y) = E[Y\mathbb{1}\{Y \geq 1\}$, $f(Y)$ is the density function of the Standard Normal distribution, $g(Y)$ is the density function of the Shifted Exponential distribution and Y are the random samples from the shifted exponential distribution.

Implementing this is in `R` we find an estimate of 0.242 with sampling error 0.001.

c)

The method of control variate is used when we are interested in $E(X)$ and we know $E(Y)$ for any X,Y. We can define a new variable T:

$$T = X + a(Y - E(Y)) \tag{13}$$

Then have $E(T) = E(X)$. The reason we might want to do this is instead of using a different sampling distribution, is that we can use a known quantity that we do not need to estimate and we get the required result, with a lower variance, see lecture notes [1]. In our example, $X = X\mathbb{1}\{X \geq 1\}$ and $Y = \mathbb{1}\{X \geq 1\}$. We can approximate a by $a = -\frac{Cov(X,Y)}{Var(Y)}$ where X and Y are as defined above, and we obtain an estimate for T . Using `R` this estimate was 0.220 and the sampling error was 0.011.

d)

The sampling error of importance sampling is lower than the Monte Carlo estimate and this makes sense because the variance of the estimators is reduced. It is odd that the sampling error of the control variate method is larger from both Monte Carlo and Importance sampling, and this suggests that we might have done something wrong in the calculations (see code in the Appendix), since we would expect this error to be at least smaller or the same as Monte Carlo. In principle, it should be subtracting off the singularity of the integrand if it is present and it should perform better than the other two methods.

References

[1] Dr Sarah Filippi. Lecture notes. *Imperial College London*, page 32, October 2021.

A Code for Question 1

```
rm(list = ls())
set.seed(222)
setwd("~/Documents/UNI_Imperial/CompStats/Assignments/A1")
library(nor1mix)

f <- function(x) ifelse(x>= -3/2 & x<= 3/2,
                        (1+cos(x)^3)*exp(-0.5*x^2), 0)
x <- seq(-5,5,by=0.001)
plot(x,f(x), type = 'l')
lines(x, dnorm(x), col ='red')
# abline(v =3/2)
# abline(v=-3/2)

int <- integrate(f,-3/2,3/2)

k <- 1/int$value

new_f <- function(x) ifelse(x>= -3/2 & x<= 3/2,
                            k*(1+cos(x)^3)*exp(-0.5*x^2), 0)
g <- function(x) dnorm(x)

x <- seq(-5,5,by=0.001)
plot(x,new_f(x), type = 'l')
lines(x, dnorm(x), col ='red')

C <- 2*sqrt(2)*k/pi


rf <- function(){

  while(1){
    x <- rnorm(1)
    if (runif(1) < new_f(x)/(C*g(x))){
      return(x)
    }
  }
}

x_samples <- replicate(1e3, rf())
x <- seq (-3/2 ,3/2 ,0.01)


fx <- Vectorize(new_f)
dx <- 0.01

# pdf('cdf.pdf', width=9, heigh=5)
plot(x, cumsum(fx(x) * dx), type = "l", ylab='F(x)')
lines(ecdf(x_samples), col ='red')
# dev.off()
```

```
#part c

x_samples <- replicate(1e4, rf())

mc <- mean(x_samples^2)

mc_sd <- sd(x_samples^2)/sqrt(1e4)
CI <- mc + mc_sd* qnorm(c(0.05, 0.95))

N =round(diff(CI)^2*10^12,0)
```

B Code for Question 2

```
rm(list = ls())
set.seed(222)

sigma <- 1
sigmas <- c(0.01,0.1,0.5,1)

I_sigma <- function(x, sigma=0.01){
  1/2*x*(1/(pi*(1+x^2)) + 1/(sigma*sqrt(2*pi)) * exp(-(x-5)^2/(2*sigma^2)))
}

x <- seq(-2,8,0.01)

# pdf('Isigma.pdf', width=9, heigh=5)
plot(x, I_sigma(x,0.1), type='l', ylab='I(sigma)')
lines(x, I_sigma(x,0.1), type='l', col ='red')
lines(x, I_sigma(x,0.5), type='l', col ='blue')
lines(x, I_sigma(x,1), type='l', col ='green')
legend(-2, 6, legend=c(0.1, 0.5, 1),
       col=c("red", "blue", "green"), lty=1:1, cex=0.8)
# dev.off()

for(sigma in sigmas){
  print(integrate(I_sigma, -Inf,Inf, sigma))
  }

sigma=1
integrate(I_sigma,-Inf,-1,sigma, subdivisions=2000)$value+
  integrate(I_sigma,-1,1,sigma)$value+
  integrate(I_sigma,1,3,sigma)$value+
  integrate(I_sigma, 3,6,sigma)$value+
  integrate(I_sigma, 6,Inf,sigma, subdivisions=2000)$value

x <- seq(4.8,5.1,0.01)
plot(x, I_sigma(x,0.01), type='l', ylab='I(sigma)')

integrate(I_sigma, -Inf, 4.8, sigma=0.01 ,subdivisions=2000)$value+
  integrate(I_sigma, 4.8, 5.1, sigma=0.01)$value+
  integrate(I_sigma, 5.1, Inf, sigma=0.01, subdivisions=2000)$value


#part b
```

```
f_i <- function(x, sigma=1){
  1/2*(1/(pi*(1+x^2)) + 1/(sigma*sqrt(2*pi)) * exp(-(x-5)^2/(2*sigma^2)))
}
g_i <- function(x) dnorm(x,0,4)

phi <- function(x) x

y <- rnorm(1e4,0,4)
w <- y*f_i(y)/g_i(y)
mean(w)

my_ISamp <- function(phi,f,g,y){
  IS <- mean(phi(y)*f(y)/g(y))
  IS_sd <- sd(phi(y)*f(y)/g(y))/sqrt(length(y))
  res <- c(IS, IS_sd)
  res

}

my_ISamp(phi,f_i,g_i,y)

n <- 1e4
X <- rcauchy(n) + rnorm(n,0,5)

mean(X)
x <- seq(-20,20,0.01)
plot(x,dcauchy(x), type ='l', ylim=c(0,0.5))
lines(x, dnorm(x,0,5), col='red')
lines(x, dcauchy(x) + dnorm(x,0,5), col='green')
lines(x, dcauchy(x, 0,0.5), col='blue')


#monte carlo
#need to sample first
#this was not done.

rf <- function(){

  while(1){
    x <- rnorm(1)
    if (runif(1) < new_f(x)/(C*g(x))){
      return(x)
    }
  }
}

x_samples <- replicate(1e4, rf())
```

C Code for Question 3

```
rm(list = ls())
set.seed(222)


f <- function(x,y){(1+(x-2*y)^2/4)^(-5/2) * (1+((x+2*y)^2)/3)^-2
}
```

```
I <- function(x) sapply(x,
      function(t_i) integrate(function(y) f(t_i,y),-Inf,Inf)$value)

k = 1 / integrate(I, -Inf, Inf)$value
k

f <- function(x,y){k *(1+(x-2*y)^2/4)^(-5/2) * (1+((x+2*y)^2)/3)^-2
}

set.seed(222)
n <- 10^4

x <- rnorm(n)
y <- rnorm(n)
w <- mapply(f, x, y)/(dnorm(x)*dnorm(y))

is <- mean((x-y)^2*w)

is_sd <- sd((x-y)^2*w)/sqrt(n)
is
is_sd


set.seed(222)
N <- 10^4
it <- 15
est <- rep(0,it); mw <- rep(0, it); se <- rep(0,it)
c=1
for (n in rep(N,it)){
  x <- rnorm(n)
  y <- rnorm(n)
  w <- mapply(f, x, y)/(dnorm(x)*dnorm(y))
  est[c] <- mean((x-y)^2*w)
  mw[c] <- max(w)
  se[c] <- sd((x-y)^2*w)/sqrt(n)
  c = c +1
}

df <- data.frame(Estimate = est, SE = se, Max_Weight = mw)
library(xtable)
print(xtable(df),include.rownames=FALSE) #Remove rownames
#################################################################


set.seed(124321)
N <- 10^6
x <- rnorm(N)
y <- rnorm(N)
w <- mapply(f, x, y)/(dnorm(x)*dnorm(y))

In <- cumsum((x-y)^2*w)/1:N

alln <- round(10^seq(1,6,length.out=1000))
# exact value for reference line
I2 <- function(x) sapply(x,
          function(t_i) integrate(function(y) f(t_i,y),-Inf,Inf)$value)
```

```
Iexact <- integrate(I, -Inf, Inf)$value
Iexact
plot(alln, In[alln], xlab='n', ylab=expression(I[n]),
     ylim=Iexact+c(-1,1), log='x')
lines(c(1,N), c(Iexact,Iexact))


################################################################
set.seed(222)
N <- 10^4
it <- 15
est <- rep(0,it) ; mw <- rep(0, it) ; se <- rep(0,it)
c=1
for (n in rep(N,it)){
  x <- rcauchy(n)
  y <- rcauchy(n)
  w <- mapply(f, x, y)/(dcauchy(x)*dcauchy(y))
  est[c] <- mean((x-y)^2*w)
  mw[c] <- max(w)
  se[c] <- sd((x-y)^2*w)/sqrt(n)
  c = c +1
}

df <- data.frame(Estimate = est, SE = se, Max_Weight = mw)
library(xtable)
print(xtable(df),include.rownames=FALSE) #Remove rownames
```

D Code for Question 4

```
rm(list = ls())
set.seed(222)
n <- 1e4
X <- rnorm(n)
mc <- mean(X*(X>=1))
mc_sd <- sd(X*(X>=1)) / sqrt(n)
mc
mc_sd

#b)
Y <- 1+ rexp(n)
IC <- mean(Y*(Y>=1) *dnorm(Y)/dexp(Y-1))
IC_sd <- sd(Y*(Y>=1) *dnorm(Y)/dexp(Y-1)) / sqrt(n)
IC
IC_sd

a <- -cov(X*(X>=1), (X>=1)) / var(X>=1)
E_t <- mean(X*(X>=1) -a*((X>=1)- (1-pnorm(1))))
Et_sd <- sd(X*(X>=1) -a*((X>=1)- (1-pnorm(1)))) / sqrt(n)
E_t
Et_sd
```