ST303 Coursework 2

Candidate Number: 26617

Problem 1

Methodology: Testing

So far in this course we have only dealt with single integrals when applying Monte-Carlo methods. As an extension of what I learnt, I have come up with a potential solution to estimating double integrals using Monte-Carlo techniques. To test whether this works, I will first apply it to a similar double integral which I know the analytical solution to so that I can test whether my extended method works. The double integral I will be using is

$$\begin{aligned}\int_0^1 \int_0^1 sin(x)sin(y)dxdy &= \int_0^1 sin(y)\Big[-cos(x)\Big]_{x=0}^{1} dy \\ &= \Big(cos(0) - cos(1)\Big)\Big[-cos(x)\Big]_{y=0}^{1} \\ &= \Big(1 - cos(1)\Big)^2 \\ &\approx 0.211322.\end{aligned}$$

To estimate the integral, I use Monte-Carlo combined with Control Variables. Since we have

$$\int_0^1 \int_0^1 sin(x)sin(y)dxdy = \mathbb{E}(sin(X)sin(Y))$$

where $\boldsymbol{X, Y \sim U(0, 1)}$, the Monte-Carlo estimator for this double-integral is

$$\overline{Z}_n = \frac{1}{n}\sum_{i=1}^{n} sin(X_i)sin(Y_i)$$

where $\boldsymbol{X_i, Y_i \sim U(0, 1)}$ for i = 1, 2, 3…

The Monte-Carlo estimator relies on the **Strong Law of Large Numbers** which implies that our estimator is strongly consistent, i.e.

$$\lim_{n\to\infty} \overline{Z}_n = \mathbb{E}(sin(X)sin(Y)).$$

Control Variable

I will also apply a control variable as my variance reduction method. This is to test how well control variables perform for double integrals. There are many different control variates to choose from however the ones I will be using are

$$V_i = X_i - \mathbb{E}(X_i) = X_i - 0.5$$

and

$$W_i = Y_i - \mathbb{E}(Y_i) = Y_i - 0.5.$$

My MC estimator is therefore given by

$$\overline{Z}_n(b, c) = \frac{1}{n}\sum_{i=1}^{n} (Z_i - bV_i - cW_i)$$

where $\boldsymbol{Z_i = sin(X_i)\ sin(Yi)}$ **for** i = 1, 2, 3…

Assuming control variables work in a similar way for double integrals, I will take b and c to be

$$b^* = \frac{\text{Cov}(Z, X)}{\text{Var}(X)}$$

and

$$c^* = \frac{\text{Cov}(Z, Y)}{\text{Var}(Y)}$$

respectively to minimize the variance of the estimator.

Simple proof of why my control variable works
Firstly, we have

$$Z_i(b) = Z_i - bV_i - cW_i$$

hence

$$\begin{aligned}\mathbb{E}(Z_i(b)) &= \mathbb{E}(Z_i - bV_i - cW_i) \\ &= \mathbb{E}(Z_1) - b\mathbb{E}(V_1) - c\mathbb{E}(W_1) \\ &= \mathbb{E}(Z_1).\end{aligned}$$

The control variate reduces the variance of the MC estimator the most when the covariance between the control variate and the estimator is largest in magnitude. At least that was the case for standard MC estimators we have used so far in the course. **Note**: the value of b^* is proven by minimizing the variance with respect to b.

Application and Results

```
# Clearing Workspace before each question to keep it tidy
rm(list = ls())
set.seed(56473)
nSamples = 10^6

# Generating 2 samples of standard uniform random variables
U1 <- runif(nSamples)
U2 <- runif(nSamples)

# Standard MC Estimator
X = sin(U1)*sin(U2)

# Creating my control variables
Y1 <- 0.5-U1
Y2 <- 0.5-U2

# Calculating the constant c that minimizes the variance of my estimate
c1 = -12*cov(X,Y1)
c2 = -12*cov(X,Y2)

# Calculating my control estimator
control_Variate = X + c1*Y1 + c2*Y2

# This is the true closed-form value of the double integral
true_value = (1-cos(1))^2

MC_soln = mean(X)
Control_soln = mean(control_Variate)

MC_var = var(X)/nSamples
Control_var = var(control_Variate)/nSamples

cbind(true_value, MC_soln, Control_soln)

##      true_value   MC_soln Control_soln
## [1,]   0.211322 0.2112155    0.2113498

cbind(MC_var, Control_var)
```

```
##           MC_var Control_var
## [1,] 2.96719e-08 3.90107e-09
```

	Solution	Variance
True Value	0.211322	N/A
MC Estimate	0.211216	2.96719e-08
Control Estimate	0.211350	3.90107e-09

As you can see from the code above, the MC estimator as well as the control variables have performed very well as expected to. This confirms that my extended method to solve double integrals works hence I will now apply it to the unknown integral given by

$$\int_0^1 \int_0^1 sin(\sqrt{(xy)}dxdy.$$

Methodology: Applying to Problem

Monte-Carlo

Since we have

$$\int_0^1 \int_0^1 sin(\sqrt{xy}dxdy = \mathbb{E}\left(sin(\sqrt{XY})\right)$$

where ***X, Y ~ U(0, 1),*** the Monte-Carlo estimator for this double-integral is

$$\overline{Z}_n = \frac{1}{n}\sum_{i=1}^{n} sin(\sqrt{X_i Y_i}$$

where $X_i, Y_i \sim U(0,1)$ for i = 1, 2, 3 …

Control Variable

The controls I will be using are the exact same as those used above for the testing of my method. These are

$$V_i = X_i - \mathbb{E}(X_i) = X_i - 0.5$$

and

$$W_i = Y_i - \mathbb{E}(Y_i) = Y_i - 0.5 \text{ ,}$$

therefore, my MC estimator with a control variate is given by

$$\overline{Z}_n(b,c) = \frac{1}{n}\sum_{i=1}^{n}(Z_i - bV_i - cW_i)$$

where

$$Z_i = sin(\sqrt{X_i Y_i})$$

for i = 1, 2, 3…

The variance of the MC estimator after applying a control variable is minimized when we take the value of b and c as

$$b^* = \frac{\mathrm{Cov}(\mathrm{X_i}, \mathrm{V_i})}{\mathrm{Var}(\mathrm{V_i})} = 12\mathrm{Cov}(\mathrm{X_i}, \mathrm{V_i})$$
$$c^* = \frac{\mathrm{Cov}(\mathrm{X_i}, \mathrm{W_i})}{\mathrm{Var}(\mathrm{W_i})} = 12\mathrm{Cov}(\mathrm{X_i}, \mathrm{W_i}).$$

Application and Results

```
rm(list = ls())
set.seed(56473)
```

```
nSamples = 10^6

U1 <- runif(nSamples)
U2 <- runif(nSamples)

Y1 <- 0.5-U1
Y2 <- 0.5-U2

X = sin(sqrt(U1*U2))

c1 = -12*cov(X,Y1)
c2 = -12*cov(X,Y2)

control_estimator = X + c1*Y1 + c2*Y2

MC_soln = mean(X)
Control_soln = mean(control_estimator)

MC_var = var(X)/nSamples
Control_var = var(control_estimator)/nSamples

# Calculating the 95% Confidence Interval for my MC estimation
lower = Control_soln - 1.96*sqrt(Control_var)
upper = Control_soln + 1.96*sqrt(Control_var)

cbind(MC_soln, MC_var)

##        MC_soln       MC_var
## [1,] 0.4183413 4.050721e-08

cbind(Control_soln, Control_var, lower, upper)

##      Control_soln  Control_var     lower     upper
## [1,]    0.4185008 4.161054e-09 0.4183744 0.4186272
```

	Solution	**Variance**
MC Estimate	0.4183413	4.050721e-08
Control Estimate	0.4185008	4.161054e-09

We can see above that the variance of the control estimate is significantly lower. I have also included the 95% confidence interval for the control estimation. The interval is ***(0.4183744, 0.4186272)*.**

Problem 2

Methodology

The solution to problem 2 is to create a function that returns the score of the selected candidate then taking a sample mean of the score after many realizations. **Extra:** I will also investigate the expected score for different number of candidates as well as different control group size. Therefore, my function should have 3 arguments. The arguments are **k** (size of control), **N** (number of candidates), and a dummy variable for generating many samples.

In the function, I will first generate 2 samples from the standard uniform distribution which will be the scores of the candidates. First sample being of size **k** which is the control group and second sample of size **N-k** which are the potential candidates. From here, there are 2 paths

1. **Max of Sample 1 >= Max of Sample 2:** the selected candidate is the very last person, and their score is given by the last value from the sample of size **N-k**
2. **Max of Sample 1 < Max of Sample 2:** the selected candidate is the first person to exceed the maximum of sample - Their score is identified by the index of the selected candidate.

By the **Strong Law of Large Numbers,** we have that the sample mean of the scores converges to the **expected score of the successful candidate**. Therefore, we take the average score for each value of the parameters k and N over a large sample size to identify the optimal value.

Application and Results

```
rm(list = ls())
set.seed(56473)
nSamples = 10^6

# Defining function that returns the score of the selected candidate for given para
meters k and N
f_uniform <- function(k, N, a) {
  X <- runif(k)  # Generating Scores of the first k candidates that will be rejecte
d
  Y <- runif(N-k)  #  Generating scores of the potential successful candidates

  # The following IF statement determines who is the selected candidate
  if (max(Y)>max(X)){
    winner = min(which(Y>max(X)))
  } else  {
    winner = N-k
  }

  # Retrieving the score of the selected candidate
  score = Y[winner]
  return(score)
}
```

The next block finds the optimal k which maximises the expectation of the score.

```
k = 1:11
X = c()

for (i in k) {
  # Calculating the average score for each value of k for a sample size of nSamples
(and fixing N = 12)
  X[i] = mean(sapply(1:nSamples, f_uniform, k=i, N=12))
}

X

##  [1] 0.7079273 0.7500625 0.7502433 0.7337422 0.7083751 0.6779733 0.6456744
##  [8] 0.6112047 0.5750271 0.5379369 0.4994801

max_expectation = max(X)
optimal_k = which.max(X)
```

The following block investigates the effects of changing the number of candidates.

```
# Lets now relax the value of N and fix k = optimal_k
N = 10:18
Y = c()

for (i in N) {
  # Calculating the average score for each value of N (and fixing k = optimal_k)
  Y[i-9] = mean(sapply(1:nSamples, f_uniform, k=optimal_k, N=i))
}

Y

## [1] 0.7250046 0.7388871 0.7502237 0.7592180 0.7672850 0.7749706 0.7815429
## [8] 0.7863514 0.7919875

max_expectation_part2 = max(Y)
optimal_N = which.max(Y) + 9
```

```
# For optimal k and optimal N, the average score is the following
Optimal_expectation = Y[optimal_N-9]

cbind(optimal_k, optimal_N, Optimal_expectation)

##      optimal_k optimal_N Optimal_expectation
## [1,]         3        18           0.7919875
```

Assuming that the optimal value of **N** is independent of the value of **k**, we have that the optimal combination is **k=3** and **N=18** which yields an average score of **0.7920**.

Different Distribution

In this case, the choice of distribution matters because we are interested in the actual score that the selected candidate achieved not just whether they achieved the maximum or not. In other words, we are now interested in the expectation which means we are interested in the entirety of the distribution rather than just the probability of a certain outcome. To investigate this, I sampled the scores from a ***Beta(5, 1)*** distribution. I chose this distribution because it is tilted more towards 1 hence, we should expect a larger expected score of the successful candidate, I.e., we are more likely to get larger values.

```
rm(list = ls())
set.seed(56473)
nSamples = 10^6

f_beta <- function(k, N, a) {
  # This time I have used the beta(5,1) distribution
  X <- rbeta(k,5,1)
  Y <- rbeta(N-k,5,1)

  # The following IF statement determines who is the selected candidate
  if (max(Y)>max(X)){
    winner = min(which(Y>max(X)))
  } else  {
    winner = N-k
  }

  score = Y[winner]
  return(score)
}

k = 1:11
X = c()

for (i in k) {
  X[i] = mean(sapply(1:nSamples, f_beta, k=i, N=12))
}

X

##  [1] 0.9202491 0.9312741 0.9282439 0.9201840 0.9098930 0.8984269 0.8861738
##  [8] 0.8734154 0.8603659 0.8466425 0.8333290

max_expectation = max(X)
optimal_k = which.max(X)

# Lets now relax the value of N and fix k = optimal_k
N = 10:18
Y = c()

for (i in N) {
  Y[i-9] = mean(sapply(1:nSamples, f_beta, k=optimal_k, N=i))
}

Y
```

```
## [1] 0.9252345 0.9286659 0.9313493 0.9336631 0.9355431 0.9373518 0.9387465
## [8] 0.9399178 0.9412108

max_expectation_part2 = max(Y)
optimal_N = which.max(Y) + 9

# For optimal k and optimal N, the probability of choosing the correct candidate is
...
Optimal_expectation = Y[optimal_N-9]
cbind(optimal_k, optimal_N, Optimal_expectation)

##      optimal_k optimal_N Optimal_expectation
## [1,]         2        18           0.9412108
```

The optimal expectation is now **0.9412** compared to **0.7920** for the standard uniform which is what we expected since the ***Beta (5, 1)*** distribution generates more larger values compared to the standard uniform.

Problem 3

Issue in the Market

To calculate the prices of the Asian call option (A_0), and Quantile call option (Q_0), we take the discounted risk-neutral expectation of the payoffs of the options. Given that $r = 0$ and $S(0) = 100$ in the market, we face an issue since the dynamic of the stock price is not risk-neutral. Using the MGF of normal distribution, it can be shown that

$$\begin{aligned} S(0) &= \mathbb{E}_0^{\mathbb{Q}}\left(e^{-rt}S(t)\right) \\ &= \mathbb{E}_0^{\mathbb{Q}}\left(100exp\left(0.6B(t)\right)\right) \\ &= 100\mathbb{E}_0^{\mathbb{Q}}\left(exp\left(0.6\sqrt{t}Z\right)\right) \\ &= 100exp\left(\frac{1}{2}(0.6\sqrt{t})^2\right) \\ &= 100exp\left(0.18t\right), \end{aligned}$$

where $Z \sim N(0, 1)$, does not hold. E.g., for $T=1$, we have

$$\begin{aligned} \mathbb{E}_0^{\mathbb{Q}}\left(e^{-rt}S(1)\right) &= 100exp\left(0.18\right) \\ &\neq S(0). \end{aligned}$$

A suggestion is that there are continuously paid dividends with dividend yield q. If that is the case, then we should have

$$\begin{aligned} S(0) &= \mathbb{E}_0^{\mathbb{Q}}\left(e^{qT}S(T)\right) \\ &= \mathbb{E}_0^{\mathbb{Q}}\left((e^{qT}-1)S(T)\right) + \mathbb{E}_0^{\mathbb{Q}}(S(T)). \end{aligned}$$

However, we have shown that

$$\mathbb{E}_0^{\mathbb{Q}}(S(T)) > S(0)$$

Therefore

$$\mathbb{E}_0^{\mathbb{Q}}\left((e^{qT}-1)S(T)\right) < 0$$

which only holds when $q < 0$. It logically makes no sense for the dividend yield to be negative therefore this does not fix the problem. In fact, all we need to make this problem solvable is to make $r > 0$. In fact, we need r to satisfy

$$\begin{aligned} S(0) &= \mathbb{E}_0^{\mathbb{Q}}\left(e^{-rT}S(T)\right) \\ &= e^{-rT} \cdot 100exp\left(0.18T\right) \\ &= 100 \end{aligned}$$

giving us $r = 0.18$. Using this, we can now calculate the prices of the call options since the dynamic is now under the risk-neutral measure. I.e., the assumption of zero interest rates must be neglected to make this problem solvable.

Methodology

Since the stock price at time **t** is given by

$$S(t) = 100e^{0.6B(t)}$$

where ***B(t) ~ N(0, t),*** we must first generate random variables from the standard normal distribution. To generate a sample of the path of the stock price with time increments of $\frac{1}{50}$, I must generate 50 random variables from a normal distribution with mean 0 and variance $\frac{1}{50}$ then take a cumulative sum of the sample to obtain the Brownian motion, ***B(ti)*** for i = 0, 1, 2, … I.e.,

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{pmatrix} \begin{pmatrix} \hat{B}(t_1) - \hat{B}(0) \\ \hat{B}(t_2) - \hat{B}(t_1) \\ \vdots \\ \hat{B}(t_m) - \hat{B}(t_{m-1}) \end{pmatrix} = \begin{pmatrix} \hat{B}(t_1) \\ \hat{B}(t_2) \\ \vdots \\ \hat{B}(t_m) \end{pmatrix}.$$

where $\hat{B}(t_0) = \hat{B}(0) = B(0) = 0.$
Now we take the simulated stock price to be

$$\hat{S}(t_i) = 100e^{0.6\hat{B}(t_i)}$$

for i = 1, 2, 3, … I repeat this n times to obtain n independent sample paths of the stock price. After this, I take the sample mean (***M***) and sample median (***M****) for each sample path allowing me to compute the payoff of the option at maturity.

$$M = \frac{1}{50}\sum_{i=1}^{50} S\left(\frac{i}{50}\right)$$
$$M^* = \frac{1}{2}\left(S\left(\frac{25}{50}\right) + S\left(\frac{26}{50}\right)\right)$$

Since the price of the options are giving by the discounted risk-neutral expectations of the future payoffs, we have

$$A_0 = \mathbb{E}_0^{\mathbb{Q}}(e^{-rT}(M - b)^+),$$
$$Q_O = \mathbb{E}_0^{\mathbb{Q}}(e^{-rT}(M^* - b)^+).$$

Therefore, if we take

$$\hat{M}_j = \frac{1}{50}\sum_{i=1}^{50} \hat{S}_j\left(\frac{i}{50}\right)$$
$$\hat{M}_j^* = \frac{1}{2}\left(\hat{S}_j\left(\frac{25}{50}\right) + \hat{S}_j\left(\frac{26}{50}\right)\right),$$

the Monte-Carlo estimators of the option prices are given by

$$A_0^{MC} = \frac{1}{n}\sum_{j=1}^{n} e^{-rT}(\hat{M}_j - k)^+$$
$$Q_0^{MC} = \frac{1}{n}\sum_{j=1}^{n} e^{-rT}(\hat{M}_j^* - k)^+.$$

Variance Reduction

Control 1

I will now apply a control variate to reduce the variance of my MC estimation. Since we now have

$$S(0) = \mathbb{E}_0^{\mathbb{Q}}\left(e^{-rT}S(T)\right),$$

we can apply the control variable

$$\begin{aligned} Y_i &= X_i - b\left(e^{-rT}S_i(T) - \mathbb{E}_0^{\mathbb{Q}}\left(e^{-rT}S_i(T)\right)\right) \\ &= X_i - b\left(e^{-rT}S_i(T) - S(0)\right), \end{aligned}$$

giving us the estimator for the price of an Asian call

$$A_0^{Control1}(b) = \frac{1}{n}\sum_{i=1}^{n} X_i - b_A\left(e^{-rT}S_i(T) - S(0)\right)$$

where

$$X_i = e^{-rT}(\hat{M}_j - k)^+.$$

Applying the exact same control for the Quantile call yields the estimator

$$Q_0^{Control1}(b) = \frac{1}{n}\sum_{i=1}^{n} V_i - b_Q\left(e^{-rT}S_i(T) - S(0)\right)$$

where

$$V_i = e^{-rT}(\hat{M}_j^* - k)^+.$$

To minimize the variance for both estimations, we take

$$\begin{aligned} b_A &= \frac{\mathrm{Cov}(X, e^{-rT}S(T)}{\mathrm{Var}(e^{-rT}S(T))} \\ b_Q &= \frac{\mathrm{Cov}(V, e^{-rT}S(T)}{\mathrm{Var}(e^{-rT}S(T))} \end{aligned}$$

which is shown by minimizing the variance with respect to **b**.

Control Variable 2

Another control would be the sum of Brownian Motions. That is

$$\sum_{i=1}^{n}(\hat{B}_{t_i} - \hat{B}_{t_{i-1}}) = \hat{B}_T - \hat{B}_{t_0} = \hat{B}_T \sim \mathcal{N}(0, T).$$

therefore, the control is given by

$$Y_i = \hat{B}_{T,i} - \mathbb{E}(\hat{B}_T) = \hat{B}_{T,i} - 0 = \hat{B}_{T,i}$$

yielding the MC estimator

$$\begin{aligned} A_0^{Control1} &= \frac{1}{n}\sum_{j=1}^{n}\left((\hat{M}_j - k)^+ - b\left(\hat{B}_{T,i}\right)\right) \\ Q_0^{Control1} &= \frac{1}{n}\sum_{j=1}^{n}\left((\hat{M}_j^* - k)^+ - c\left(\hat{B}_{T,i}\right)\right). \end{aligned}$$

Application and Results

```
rm(list = ls())
set.seed(56473)
nSamples = 10^6

r=0.18
S0 = 100
```

```
# This function returns the series of stock prices for all 50 periods in a year and
the sum of the brownian motions
stock_price <- function(a) {
  Z = rnorm(50, 0, sqrt(1/50))
  B = cumsum(Z)
  S = 100*exp(0.6*B)
  W = B[50]

  S = append(S, W)
  return(S)
}

Results= sapply(1:nSamples, stock_price)

S_prices = Results[-51,]
ST = S_prices[50,]
W = Results[51,]

S_mean = colMeans(S_prices) # Calculating the mean stock price over the entire peri
od for each sample
S_median = apply(S_prices,2,median) # Calculating the median stock price over the e
ntire period for each sample
```

Now repeating for different strike prices b

```
# Defining a function that returns the price of an asian option for different strik
e prices
f_asian <- function(x) {
  payoffs_asian = pmax(S_mean - x, 0)
  asian_price = exp(-r*1)*mean(payoffs_asian)
  asian_variance = var(payoffs_asian)/nSamples

  Y = c(asian_price,asian_variance)
  return(Y)
}

# Defining a function that returns the price of a quantile option for different str
ike prices
f_quantile <- function(x) {
  payoffs_quantile = pmax(S_median - x, 0)
  quantile_price = exp(-r*1)*mean(payoffs_quantile)
  quantile_variance = var(payoffs_quantile)/nSamples

  Y = c(quantile_price,quantile_variance)
  return(Y)
}

# This is the range of strike prices I am testing for
b = c(70,80,90,100,110,120,130)

asian <- sapply(b, f_asian)
quantile <- sapply(b, f_quantile)

Asian_price = asian[1,]
Asian_var = asian[2,]
Quantile_price = quantile[1,]
Quantile_var = quantile[2,]


data.frame(Asian_price, Asian_var, Quantile_price, Quantile_var)

##      Asian_price      Asian_var     Quantile_price     Quantile_var
## 1   34.299823    0.0015905130      32.498848     0.0015245814
## 2   27.505300    0.0014542658      25.856895     0.0013831636
## 3   21.672337    0.0012786626      20.131308     0.0012119702
## 4   16.841508    0.0010862072      15.435579     0.0010276385
## 5   12.951619    0.0008974085      11.754601     0.0008474791
```

```
## 6    9.888176     0.0007257608       8.934550      0.0006849368
## 7    7.513327     0.0005779011       6.790382      0.0005461829
```

The next block applies control variable 1

```
f_asian_control <- function(x) {
  payoffs_asian = pmax(S_mean - x, 0)
  b = cov(payoffs_asian, exp(-r*1)*ST)/var(exp(-r*1)*ST)

  control = exp(-r*1)*payoffs_asian - b*(exp(-r*1)*ST-100)

  asian_price = mean(control)
  asian_var = var(control)/nSamples

  Y = c(asian_price, asian_var)
  return(Y)
}

f_quantile_control <- function(x) {
  payoffs_quantile = pmax(S_median - x, 0)
  b = cov(payoffs_quantile, exp(-r*1)*ST)/var(exp(-r*1)*ST)

  control = exp(-r*1)*payoffs_quantile- b*(exp(-r*1)*ST-100)

  quantile_price = mean(control)
  quantile_var = var(control)/nSamples

  Y = c(quantile_price, quantile_var)
  return(Y)
}

asian_control1 <- sapply(b, f_asian_control)
quantile_control1 <- sapply(b, f_quantile_control)

Asian_price_control1 = asian_control1[1,]
Asian_var_control1 = asian_control1[2,]
Quantile_price_control1 = quantile_control1[1,]
Quantile_var_control1 = quantile_control1[2,]


data.frame(Asian_price_control1, Asian_var_control1, Quantile_price_control1, Quant
ile_var_control1)
```

```
##   Asian_price_control1 Asian_var_control1 Quantile_price_control1  Quantile_var_control1
## 1        34.325043       0.0003104345             32.521760         0.0004040506
## 2        27.529345       0.0002880710             25.878563         0.0003750452
## 3        21.694706       0.0002633635             20.151297         0.0003435038
## 4        16.861827       0.0002390475             15.453560         0.0003107127
## 5        12.969688       0.0002158448             11.770418         0.0002769112
## 6        9.903957        0.0001933992             8.948221          0.0002430116
## 7        7.526907        0.0001714585             6.802042          0.0002102389
```

The next block carries out control variable 2

```
f_asian_control2 <- function(x) {
  payoffs_asian = pmax(S_mean - x, 0)
  b = cov(payoffs_asian, W)/var(W)

  control = exp(-r*1)*payoffs_asian - b*(W)

  asian_price = mean(control)
  asian_var = var(control)/nSamples

  Y = c(asian_price, asian_var)
  return(Y)
}

f_quantile_control2 <- function(x) {
  payoffs_quantile = pmax(S_median - x, 0)
```

```
  b = cov(payoffs_quantile, W)/var(W)

  control = exp(-r*1)*payoffs_quantile- b*(W)

  quantile_price = mean(control)
  quantile_var = var(control)/nSamples

  Y = c(quantile_price, quantile_var)
  return(Y)
}

asian_control2 <- sapply(b, f_asian_control2)
quantile_control2 <- sapply(b, f_quantile_control2)

Asian_price_control2 = asian_control2[1,]
Asian_var_control2 = asian_control2[2,]
Quantile_price_control2 = quantile_control2[1,]
Quantile_var_control2 = quantile_control2[2,]


data.frame(Asian_price_control2, Asian_var_control2, Quantile_price_control2, Quant
ile_var_control2)

##   Asian_price_control2  Asian_var_control2  Quantile_price_control2  Quantile_var_control2
## 1        34.353240         0.0004022480             32.548183             0.0004602332
## 2        27.554971         0.0004029113             25.902350              0.0004527494
## 3        21.717096         0.0003953984             20.171915             0.0004367426
## 4        16.880711         0.0003767849             15.470811             0.0004092091
## 5        12.985144         0.0003474398             11.784445             0.0003704383
## 6        9.916298          0.0003102655              8.959391             0.0003248681
## 7        7.536562          0.0002693388              6.810801             0.0002776894
```

We can see clearly that the control variables have significantly reduced the variance of the MC estimator of the option prices.

For all the strike prices, we can see that the Asian call options are slightly more expensive than the quantile call options. This is as expected since the dynamic of the stock price is exponential therefore it grows exponentially overtime. As a result, it is more likely that the mean is greater than the median on average hence the payoff of the Asian option is more likely to be greater than the payoff of a Quantile call. Since the price of the options are given by the discounted risk-neutral expectation of the payoff, it makes sense that the Asian option is more expensive.

Problem 4

Approximating C

For

$$f(y) = C\frac{exp(\frac{y}{2})}{\sqrt{\Gamma(y+1)}}$$

to be a pdf for $y > 0$, we need

$$\begin{aligned}\int_0^\infty f(y)dy &= \int_0^\infty C\frac{exp(\frac{y}{2})}{\sqrt{\Gamma(y+1)}}dy \\ &= C\int_0^\infty \frac{exp(\frac{y}{2})}{\sqrt{\Gamma(y+1)}}dy \\ &= 1\end{aligned}$$

therefore, we take C to be

$$C = \frac{1}{\int_0^\infty \frac{exp(\frac{y}{2})}{\sqrt{\Gamma(y+1)}}dy}.$$

To estimate ***C***, I will use MC alongside variance reduction techniques to estimate

$$\int_0^\infty \frac{exp(\frac{y}{2})}{\sqrt{\Gamma(y+1)}}dy$$

Firstly, I must transform the integral such that the bounds are finite. Applying the transformation

$$x = e^{\frac{-y}{2}}$$

gives me

$$\begin{aligned} I &= \int_0^\infty \frac{exp(\frac{y}{2})}{\sqrt{\Gamma(y+1)}}dy \\ &= \int_1^0 \frac{1}{x\sqrt{\Gamma(-2\ln(x)+1)}} \cdot \left(-2\frac{1}{x}\right) dx \\ &= \int_0^1 \frac{2}{x^2\sqrt{\Gamma(-2\ln(x)+1)}}dx \\ &= \mathbb{E}\left(\frac{2}{X^2\sqrt{\Gamma(-2\ln(X)+1)}}\right) \end{aligned}$$

where ***X ~ U (0, 1).***

Therefore, my MC estimator of ***I*** is given by

$$I^{MC} = \frac{1}{n}\sum_{i=1}^{n} \frac{2}{U_i^2\sqrt{\Gamma(-2\ln(U_i)+1)}}$$

where U_i ***~ U (0, 1)*** for i = 1, 2, 3…

Stratified Sampling + Antithetic Variable

To obtain the best estimate of ***I***, I have applied stratified sampling with antithetic variables giving us the MC estimator

$$I^{Stratified} = \frac{1}{n}\sum_{i=1}^{n}\left(h(\frac{U_i+j-1}{n}) + h(\frac{j-U_i}{n})\right).$$

Finally, my estimation of ***C*** is given by

$$\hat{C} = \frac{1}{I^{Stratified}}$$

```
rm(list = ls())
nSamples = 10^6
set.seed(56473)

# Applying stratified sampling to estimate the integral to therefore estimate the c
onstant C
j = 1:nSamples
U = runif(nSamples)

I_MC = mean(2/(((U+j-1)/nSamples)^2 * sqrt(gamma(log(((U+j-1)/nSamples)^-2)+1))))
c = 1/I_MC
c

## [1] 0.09640242
```

Generating Sample

After approximating ***C***, I used acceptance-rejection to generate a sample from ***f***. I decided to use the ***Gamma (1, 8)*** distribution, which is the same as ***Exponential ($\frac{1}{8}$)*** distribution, as an envelope for my A-R algorithm. I.e.,

$$g(y) = \frac{1}{\Gamma(1)8^1} x^0 e^{-\frac{x}{8}} = \frac{1}{8} e^{-\frac{x}{8}}$$

We must now find ***b*** such that

$$f(y) \leq bg(y)$$
$$C\frac{exp(\frac{y}{2})}{\sqrt{\Gamma(y+1)}} \leq b \cdot \frac{1}{8} e^{-\frac{x}{8}}$$
$$C\frac{exp(\frac{y}{2})}{\sqrt{\Gamma(y+1)}} \cdot 8e^{\frac{x}{8}} \leq b$$

I will use the optimize function built in R to maximize the LHS and let b take the value of the maximum.

```
A_R_function <- function(x){
  y =(c*exp(x/2)/sqrt(gamma(x+1)) )* (gamma(1)*8^1*exp(x/8)/(x^(1-1)))
  return(y)
}

# Finding the maximum of the function
opt = optimize(A_R_function, lower = 0, upper = 430, maximum=TRUE)

b = as.numeric(opt[1])
b

## [1] 2.978514
```

After obtaining **b**, I will apply the following algorithm.

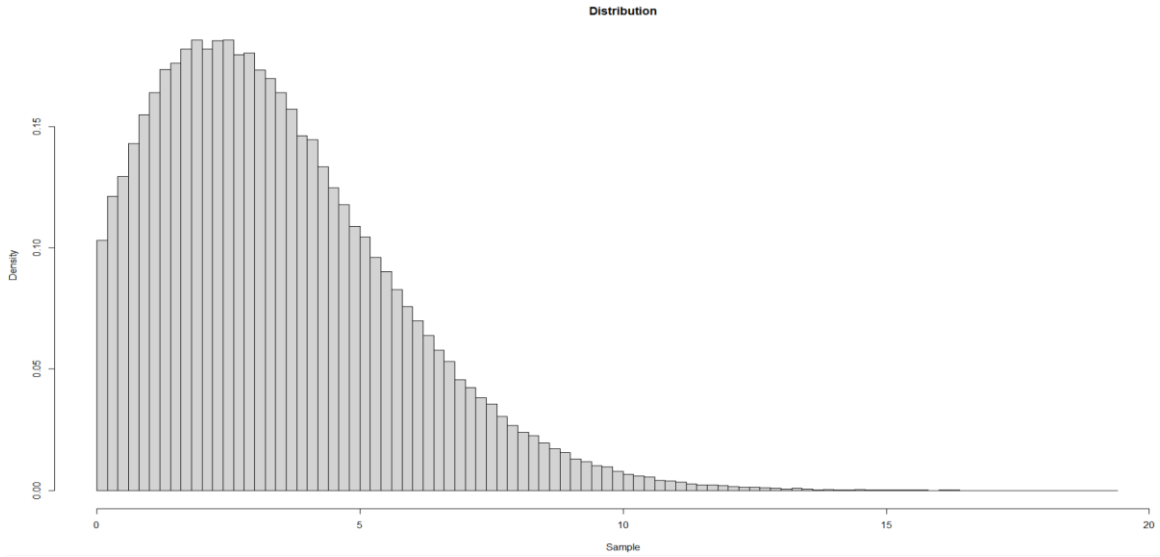
Algorithm

1. Generate $Y \sim Gamma(1, 8)$
2. Generate $U \sim \mathcal{U}(0, 1)$
3. Accept Y if $U \leq \frac{f(Y)}{bg(Y)}$, otherwise return to step 1

```
# Generating a sample of candidates from the gamma(1,8) distribution
X <- rgamma(nSamples, shape=1, scale =8)
U <- runif(nSamples)

# Carrying out acceptance_rejection criterion
Y = (U <=  (c*exp(X/2)/sqrt(gamma(X+1)))/(b*exp(-X/8)/8)) * X
Y= Y[!Y %in% c(0)]  # Removing the rejected values

hist(Y, main = paste("Distribution"), xlab = "Sample", xlim = range(0:20), freq= FA
LSE, breaks = 100)
```




```
Actual_acceptance_rate = length(Y)/nSamples
Theoretical_acceptance_rate = 1/b

cbind(Actual_acceptance_rate, Theoretical_acceptance_rate)

##      Actual_acceptance_rate Theoretical_acceptance_rate
## [1,]               0.335454                   0.3357379
```

Comparing Moments

To test whether the sample comes from ***f***, I have used MC to estimate the moments of the distribution then compared this to the sample moments. For ***X ~ U(0, 1)***, we have

First Moments

$$\begin{aligned}\mathbb{E}(Y) &= \int_0^\infty y \cdot \frac{\hat{C}exp(\frac{y}{2})}{\sqrt{\Gamma(y+1)}}dy \\ &= \int_0^1 \frac{2\hat{C}(-2\ln x)}{x^2\sqrt{\Gamma(-2\ln x+1)}}dx \\ &= \mathbb{E}\left(\frac{2\hat{C}(-2\ln X)}{X^2\sqrt{\Gamma(-2\ln X+1)}}\right)\end{aligned}$$

Second Moments

$$\begin{aligned}\mathbb{E}(Y^2) &= \int_0^\infty y^2 \cdot \frac{\hat{C}exp(\frac{y}{2})}{\sqrt{\Gamma(y+1)}}dy \\ &= \int_0^1 \frac{2\hat{C}(-2\ln x)^2}{x^2\sqrt{\Gamma(-2\ln x+1)}}dx \\ &= \mathbb{E}\left(\frac{2\hat{C}(-2\ln X)^2}{X^2\sqrt{\Gamma(-2\ln X+1)}}\right)\end{aligned}$$

WLOG, we can see that higher moments are simply given by

$$\mathbb{E}(Y^m) = \mathbb{E}\left(\frac{2\hat{C}(-2\ln X)^m}{X^2\sqrt{\Gamma(-2\ln X+1)}}\right)$$

```
U_moments <- runif(nSamples)

# Calculating Monte-Carlo moments
MC_Moment1 = mean((-4*c*log(U_moments))/(U_moments^2 * sqrt(gamma(-2*log(U_moments)+1))))

MC_Moment2 = mean((8*c*log(U_moments)^2)/(U_moments^2 * sqrt(gamma(-2*log(U_moments)+1))))

MC_Moment3= mean((-16*c*log(U_moments)^3)/(U_moments^2 * sqrt(gamma(-2*log(U_moments)+1))))

MC_Moment4= mean((32*c*log(U_moments)^4)/(U_moments^2 * sqrt(gamma(-2*log(U_moments)+1))))


MC_Moments = c(MC_Moment1,MC_Moment2,MC_Moment3,MC_Moment4)
Sample_Moments = c(mean(Y), mean(Y^2), mean(Y^3), mean(Y^4))

data.frame(MC_Moments,Sample_Moments)

##   MC_Moments Sample_Moments
## 1   3.439044       3.423972
## 2  16.935255      16.845217
## 3 103.786923     103.209960
## 4 747.311026     742.600958
```

You can see immediately that the MC Moments and Sample Moments are very similar therefore it is likely that the sample is generated from ***f*** and the method I used works well.

Problem 5

Generating Sample

In this question, we are dealing with the random sum

$$Y = \sum_{i=1}^{N} X_i$$

where ***N ~ Poisson (7)*** and X_i follows the distribution

$$f(x) = \begin{cases} 0.1, & \text{if } x = 1, \\ 0.3, & \text{if } x = 2, \\ 0.2, & \text{if } x = 3, \\ 0.3, & \text{if } x = 4, \\ 0.1, & \text{if } x = 5. \end{cases}$$

To generate a sample of ***Y***, I must create a function which first generates ***N*** then generates ***N*** samples of ***X***. The function returns the sum of ***X***.

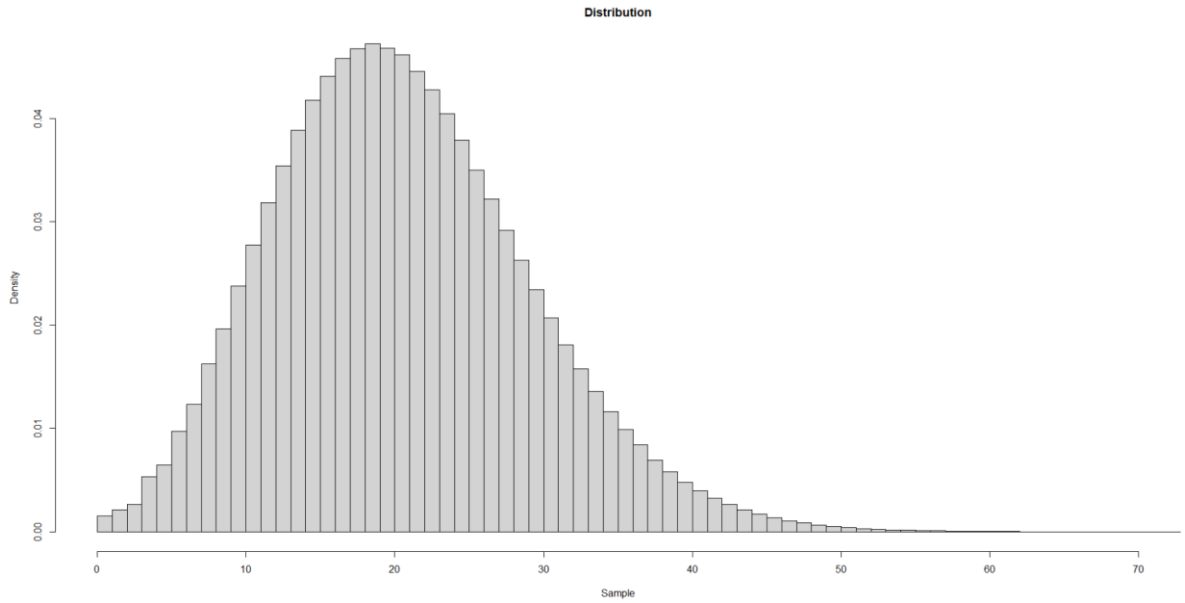
```
rm(list = ls())
nSamples = 10^7
set.seed(56473)

# Setting the probabilities for X
p = c(0.1,0.3,0.2,0.3,0.1)

# Defining a function to genreate a sample from the Random Sum
sample_generator <- function(a) {
N <- rpois(1,7)
X <- sample(1:5, N, replace=TRUE, prob = p)  # Sampling from X
Y = sum(X)

return(Y)
}
```

```
Y = sapply(1:nSamples, sample_generator)
hist(Y, main = paste("Distribution"), xlab = "Sample", xlim = range(0:70), freq= FA
LSE, breaks = 100)
```



Theoretical Moments

We know immediately that ***E(N) = 7*** and ***Var(N) = 7***. I will now calculate the moments for ***X***.

$$\begin{aligned}\mathbb{E}(X) &= \sum_{x=1}^{5} x \cdot f(x) \\ &= 1(0.1) + 2(0.3) + 3(0.2) + 4(0.3) + 5(0.1) \\ &= 3\end{aligned}$$

and

$$\begin{aligned}\mathbb{E}(X^2) &= \sum_{x=1}^{5} x^2 \cdot f(x) \\ &= 1^2(0.1) + 2^2(0.3) + 3^2(0.2) + 4^2(0.3) + 5^2(0.1) \\ &= 10.4\end{aligned}$$

therefore

$$\begin{aligned}\mathrm{Var}(X) &= \mathbb{E}(X^2) - \mathbb{E}(X) \\ &= 10.4 - 3^2 \\ &= 1.4.\end{aligned}$$

Using this, I can now compute

$$\begin{aligned}\mathbb{E}(Y) &= \mathbb{E}(\mathbb{E}(Y|N)) \\ &= \mathbb{E}\left(\mathbb{E}\left(\sum_{i=1}^{N} X_i \middle| N\right)\right) \\ &= \mathbb{E}\left(N \cdot \mathbb{E}(X)\right) \\ &= \mathbb{E}(N) \cdot \mathbb{E}(X) \\ &= 7 \cdot 3 \\ &= 21\end{aligned}$$

and

$$
\begin{aligned}
\mathrm{Var}(Y) &= \mathbb{E}\left(\mathrm{Var}(Y|N)\right) + \mathrm{Var}\left(\mathbb{E}(Y|N)\right) \\
&= \mathbb{E}\left(\mathrm{Var}\left(\sum_{i=1}^{N} X_i \middle| N\right)\right) + \mathrm{Var}\left(\mathbb{E}\left(\sum_{i=1}^{N} X_i \middle| N\right)\right) \\
&= \mathbb{E}\left(N \cdot \mathrm{Var}(X)\right) + \mathrm{Var}(N \cdot \mathbb{E}(X)) \\
&= 1.4 \cdot \mathbb{E}(N) + 9 \cdot \mathrm{Var}(N) \\
&= (1.4 \cdot 7) + (9 \cdot 7) \\
&= 9.8 + 63 \\
&= 72.8.
\end{aligned}
$$

```
Sample_Moments = c(mean(Y), var(Y))
Theoretical_Moments = c(21, 72.8)

cbind(Sample_Moments,Theoretical_Moments)

##      Sample_Moments Theoretical_Moments
## [1,]       21.00052                21.0
## [2,]       72.77250                72.8
```

It is clear above that the sample mean and sample variance is very close to the theoretical values. This suggests that our sample of ***Y*** is indeed representative.

To estimate **P(Y > 38)**, we make use of the fact that

$$\mathbb{P}(Y > 38) = \mathbb{E}(1_{[Y>38]}(Y)).$$

Therefore the estimator of **P(Y > 38)** is

$$\frac{1}{n}\sum_{i=1}^{n} 1_{[Y_i>38]}(Y_i)$$

where $\mathbf{Y_i}$ are independent realizations of **Y**.

Importance Sampling

Since the theoretical probability of **Y** being greater 38 is extremely low, we need a very high sample size to get a good estimation of the probability. Even then, the estimate tends to have a very high variance. A solution to this problem is importance sampling which involves transforming the distribution so that it shifts the distribution to larger numbers. As a result, more values exceed 38. Of course, the estimator compensates for the shifted distribution.

Firstly, we establish the MGFs

$$
\begin{aligned}
M_X(t) &= \mathbb{E}(e^{tX}) = 0.1e^{t} + 0.3e^{2t} + 0.2e^{3t} + 0.3e^{4t} + 0.1e^{5t} \\
M_Y(t) &= M_N\left(\log(M_X(t))\right) \\
&= exp\left(\lambda \cdot (M_X(t) - 1)\right) \\
&= exp\left(\lambda \cdot \left(0.1e^{t} + 0.3e^{2t} + 0.2e^{3t} + 0.3e^{4t} + 0.1e^{5t} - 1\right)\right)
\end{aligned}
$$

Since we are dealing with a random sum, we apply the transformation

$$
\begin{aligned}
M_{Y^*}(t) &= \frac{M_Y(t+r)}{M_Y(r)} \\
&= \frac{exp\big(\lambda \cdot \big(0.1e^{t+r} + 0.3e^{2(t+r)} + 0.2e^{3(t+r)} + 0.3e^{4(t+r)} + 0.1e^{5(t+r)} - 1\big)\big)}{exp\big(\lambda \cdot \big(0.1e^{r} + 0.3e^{2r} + 0.2e^{3r} + 0.3e^{4r} + 0.1e^{5r} - 1\big)\big)} \\
&= exp\big(\lambda\big(0.1e^{r}e^{t} + 0.3e^{2r}e^{2t} + 0.2e^{3r}e^{3t} + 0.3e^{4r}e^{4t} + 0.1e^{5r}e^{5t} - (0.1e^{r} + 0.3e^{2r} + 0.2e^{3r} + 0.3e^{4r} + 0.1e^{5r})\big)\big) \\
&= exp\big(\lambda\big(0.1e^{r}e^{t} + 0.1e^{2r}e^{2t} + 0.1e^{3r}e^{3t} + 0.1e^{4r}e^{4t} + 0.1e^{5r}e^{5t} - \theta\big)\big) \\
&= exp\big(\theta\lambda\big(\frac{0.1e^{r}e^{t}}{\theta} + \frac{0.3e^{2r}e^{2t}}{\theta} + \frac{0.2e^{3r}e^{3t}}{\theta} + \frac{0.3e^{4r}e^{4t}}{\theta} + \frac{0.1e^{5r}e^{5t}}{\theta} - 1\big)\big)
\end{aligned}
$$

where

$$\theta = 0.1e^{r} + 0.3e^{2r} + 0.2e^{3r} + 0.3e^{4r} + 0.1e^{5r}$$

giving us the new MGF for ***Y**** . We now have

$$N^* \sim Poisson(7\theta)$$

and ***X**** following

$$
f^*(x^*) = \begin{cases}
\frac{0.1e^{r}}{\theta}, & \text{if } x^* = 1, \\
\frac{0.3e^{r}}{\theta}, & \text{if } x^* = 2, \\
\frac{0.2e^{r}}{\theta}, & \text{if } x^* = 3, \\
\frac{0.3e^{r}}{\theta}, & \text{if } x^* = 4, \\
\frac{0.1e^{r}}{\theta}, & \text{if } x^* = 5.
\end{cases}
$$

Ideally, we want ***r*** to the be the value such that the expectation ***Y**** is **38**. I.e.,

$$\mathbb{E}(Y^*) = \mathbb{E}(N^*)\mathbb{E}(X^*) = 38.$$

Since we have

$$\mathbb{E}(N^*) = 7\theta$$

and

$$\mathbb{E}(X^*) = \frac{1}{\theta} \cdot \left(0.1e^{r} + 0.6e^{2r} + 0.6e^{3r} + 1.2e^{4r} + 0.5e^{5r}\right),$$

we want

$$
\begin{aligned}
\mathbb{E}(Y^*) &= \mathbb{E}(X^*)\mathbb{E}(N^*) \\
&= 7\theta \cdot \frac{1}{\theta} \cdot \left(0.1e^{r} + 0.6e^{2r} + 0.6e^{3r} + 1.2e^{4r} + 0.5e^{5r}\right) \\
&= 7 \cdot \left(0.1e^{r} + 0.6e^{2r} + 0.6e^{3r} + 1.2e^{4r} + 0.5e^{5r}\right) \\
&= 38
\end{aligned}
$$

To solve for ***r,*** we rearrange the equation into

$$0.1e^{r} + 0.6e^{2r} + 0.6e^{3r} + 1.2e^{4r} + 0.5e^{5r} - \frac{38}{7} = 0$$

then we use R to solve this using the optimize function. I will find the minimum of

$$\left|0.1e^{r} + 0.6e^{2r} + 0.6e^{3r} + 1.2e^{4r} + 0.5e^{5r} - \frac{38}{7}\right|$$

using the optimize function.

Applying importance sampling gives the estimator of **P(Y > 38)** to be

$$\frac{1}{n}\sum_{j=1}^{n} exp(\lambda(\theta-1)) \cdot e^{-X_j^*} \cdot 1_{[X_j^*>38]}(X_j^*)$$

```
probability_sample = mean((Y>38))
probability_sample

## [1] 0.0303403
```

Now applying importance sampling to attempt to obtain better estimation of the probability.

```
# The following is a function for calculating the optimal value of 'r'
function_r <- function(r){
  y = abs( 0.1*exp(1*r) + 0.6*exp(2*r) + 0.6*exp(3*r) + 1.2*exp(4*r) + 0.5*exp(5*r)
-38/7)
  return(y)
}

# Finding the minimum of the function on (0,5)
opt = optimize(function_r, lower = 0, upper = 5, maximum=FALSE)
r = as.numeric(opt[1])

opt

## $minimum
## [1] 0.166435
##
## $objective
## [1] 0.0006658194

theta = 0.1*exp(1*r) + 0.3*exp(2*r) + 0.2*exp(3*r) + 0.3*exp(4*r) + 0.1*exp(5*r)

# Defining the new distribution for X*
p_star = c(0.1*exp(1*r) , 0.3*exp(2*r) , 0.2*exp(3*r) , 0.3*exp(4*r) , 0.1*exp(5*r)
) / theta

# Generating sample of Y*
sample_generator_star <- function(a) {
N_Star <- rpois(1,theta*7)
X_Star <- sample(1:5, N_Star, replace=TRUE, prob = p_star)
Y_Star = sum(X_Star)

return(Y_Star)
}

Y_star = sapply(1:nSamples, sample_generator_star)

probability_estimator = exp(7*(theta-1)) * exp(-Y_star*r) *(Y_star>38)

probability_importance = mean(probability_estimator)

cbind(probability_sample,probability_importance)

##      probability_sample probability_importance
## [1,]          0.0303403             0.03034233
```

It is clear that the importance sampling has worked well since the sample probability is similar.

Challenge 5

We first generate 2 sets of coordinates in the ring by sampling from a ***Uniform (-2, 2)*** distribution. We then calculate the equation of the line joining the 2 points together

$$y - y_1 = b(x - x_1)$$

where **b** is the gradient of the line.

We now need to check if the **segment** crosses through the inner circle. To do this, we first check if the **line** crosses the inner circle. If it doesn't, then we end there otherwise we must check if the intersection is within the segment or outside the segment. We must check if there are solutions for the set of equations

$$x^2 + y^2 = 1$$
$$y = b(x - x_1) + y_1$$

The line intersects the inner circle if there are solutions to the quadratic

$$x^2 + y^2 = 1$$
$$x^2 + (b(x - x_1) + y_1)^2 = 1$$
$$\dots$$
$$(1 + b^2)x^2 + (-2b^2x_1 + 2by_1)x + (b^2x_1^2 - 2by_1x_1 + y_1^2 - 1) = 0$$

We can check if there are solutions by checking if the **discriminant < 0**.

Case 1: (**discriminant < 0**) the line never crosses the inner circle.

Case 2: (**discriminant >= 0**) the line crosses the inner circle at some point. We must now check if the intersection is within the segment or not which we do by solving the quadratic and checking if the intersections lay within the 2 original points.

```
# Function to generate a set of coordinates within the ring using acceptance-reject
ion
Acceptance <- function(v) {
  while(1) {
    X1 <- runif(1,-2,2)
    Y1 <- runif(1,-2,2)
    value = X1^2 + Y1^2
    if ((value >= 1)*(value <= 4)) {
      return(c(X1,Y1))
    }
  }
}

Generator <- function(v) {
  M <- Acceptance(1)   # Generates a set of coordinates
  N <- Acceptance(1)

  x1 = M[1] # Extracting x and y components of each coordinate
  y1 = M[2]

  x2 = N[1]
  y2 = N[2]

  b = (y2 - y1)/(x2 - x1) # Calculating gradient of the line joining the 2 points

  # Here we use the quadratic equation for ax^2 + bx + c = 0
  aa = b^2 + 1
  bb = -2*b^2*x1 + 2*y1*b
  cc = b^2*x1^2 + y1^2 - 2*y1*b*x1 - 1


  discriminant = bb^2 - 4*aa*cc # Calculating the determinant of the quadratic
```

```
  if (discriminant < 0) {
    return(1) # No solutions i.e. line never intercepts inner circle

  } else {  # Line intercepts inner circle but we need to check if the interception
is within the segment joining the two points

    soln_1 = (-bb + sqrt(discriminant))/(2*aa)  # Calculating both solutions to the
quadratic equation
    soln_2 = (-bb - sqrt(discriminant))/(2*aa)

    # Checking if the solutions lay on the segment by seeing if the x coordinate of
the solution is between x1 and x2
    No_intercepts = (soln_1 >= min(x1,x2))*(soln_1 <= max(x1,x2)) + (soln_2 >= min(
x1,x2))*(soln_2 <= max(x1,x2))

    # If No_intercepts = 0, that means the intersection with the inner circle is no
t within the segment
    if (No_intercepts == 0) {
      return(1)
    } else {
      return(0)
    }
  }
}

Sample <- sapply(1:10^5, Generator)

probability = mean(Sample)
probability

## [1] 0.52174
```