Financial Engineering - HA1

Burzler R., Krejci T., Wittmann M., Zaborskikh E.

March 9, 2019

Problem 1

Part (a)

$$\Theta = \int_2^4 5x^4 dx = \left[x^5\right]_2^4 = 1024 - 32 = 992$$

Simple check by integrate(f,2,5) where $f(x) = 5x^4$

Part (b)

We obtain the Monte Carlo estimate of integral value by generating 2000 uniformly distributed values and taking the mean of evaluation of $f(U)$, $U \sim U(0,1)$:

$$\hat{\Theta}_n = \frac{4-2}{n} \sum_{i=1}^{n} f(2+2U), U \sim U(0,1)$$

$$\hat{\Theta}_n = 983.884$$

```
set.seed(123)

f<- function(x){
  return(5*x^4)
}

ndraws<-2000

draws<-runif(n = ndraws,min = 0,max = 1)

mean(f(2+2*draws))*2
```

```
## [1] 983.884
```

Part (c) MCE is obtained simply as a difference of results in (a) and (b) :

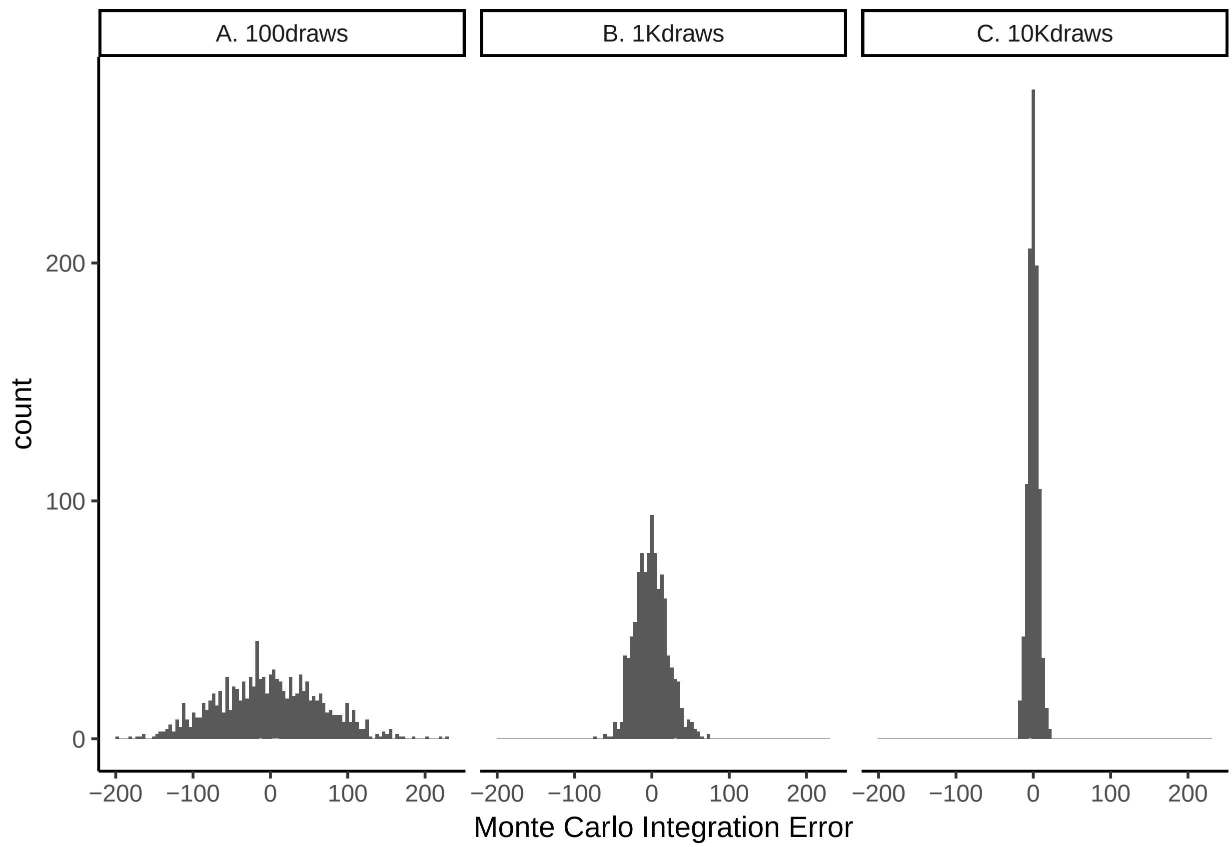
$$MCE_n = \hat{\Theta}_n - \Theta = 983.88 - 992 = -8.12$$

With growing number of independent draws, we can clearly observe a decline in variance of the resulting MCE_n distribution.

```
rslt<-data.frame(matrix(rep(NA,3000),nrow = 1000))
names(rslt)<-c('A. 100draws','B. 1Kdraws','C. 10Kdraws')
k<-0
for(i in c(100,1000,10000)){
  k<-k+1
  for(j in 1:1000){
  draws<-runif(n = i,min = 0,max = 1)
  mce<-mean(f(2+2*draws))*2-992
  rslt[j,k]<-mce
  }
```

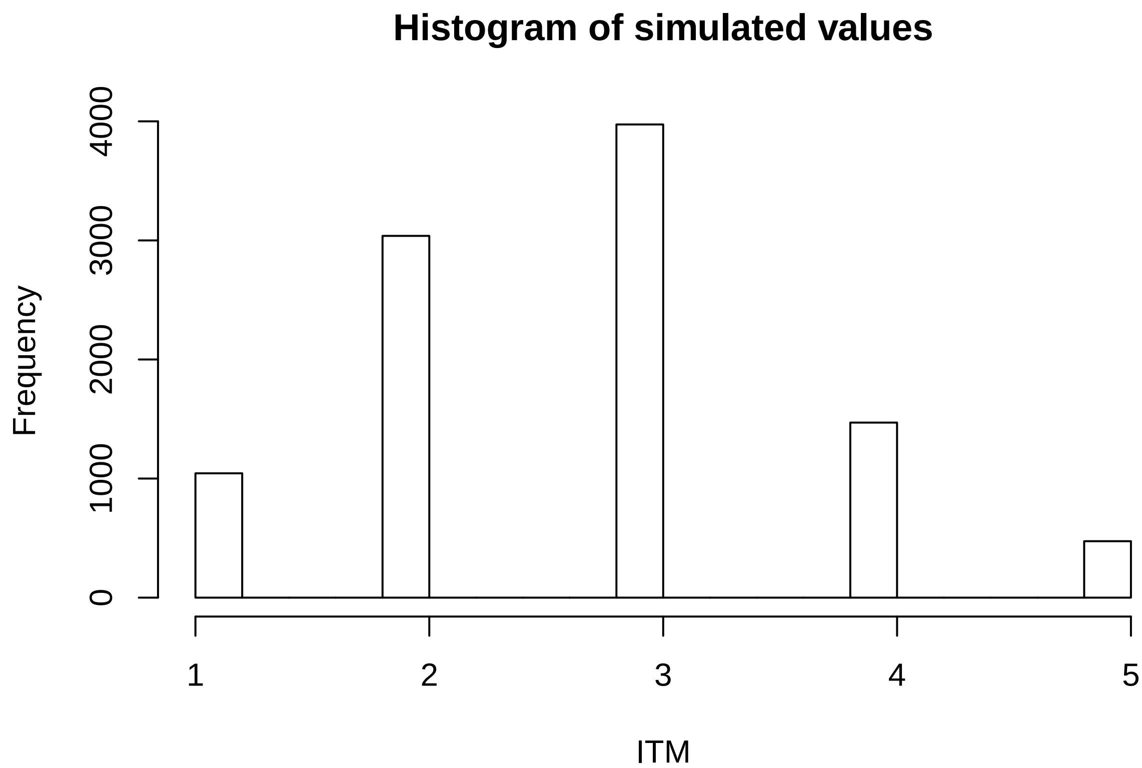
```
}
rslt$iter_num<-c(1:1000)
```

```
tdy_rslt<-gather(data = rslt,key = iter_num,value = MCE)
names(tdy_rslt)<-c('draws','MCE')

ggplot(tdy_rslt,aes(tdy_rslt$MCE))+
  geom_histogram(bins = 100)+
  facet_grid(~draws)+
  xlab('Monte Carlo Integration Error')+
  theme_classic()
```



Problem 2 - Inverse Transform Method (discrete random variable)

```r
set.seed(2019)

# a) inverse transform function
inv.trans <- function(n, values, probs) {
  # find cumulative probs
  cum <- c()
  cum[1] <- probs[1]
  for (i in 2:length(probs)) {
    cum[i] <- probs[i] + cum[i-1]
  }

  # draw uniform random number
  u <- runif(n, 0, 1)
  draws <- c()
  vec <- rep(0, length(probs))
  for(i in 1:length(u)) {
    urand <- u[i]
    for(j in 1:length(probs)) {
      if(urand > cum[j]) {
        vec[j] <- 1
        } else {
        vec[j] <- 0
        }
    }
    draws[i] <- values[sum(vec)+1]
  }
  draws
}

# b)
# inputs
n <- 10000
values <- c(1, 2, 3, 4, 5)
probs <- c(0.1, 0.3, 0.4, 0.15, 0.05)

# simulation & plotting
ITM <- inv.trans(n, values, probs)
hist(ITM, main = "Histogram of simulated values")
```




```r
# check
Check_probs <- c(sum(ITM==values[1])/n, sum(ITM==values[2])/n, sum(ITM==values[3])/n,
                 sum(ITM==values[4])/n, sum(ITM==values[5])/n)
Prob_Ovw <- rbind(probs, Check_probs)
rownames(Prob_Ovw) <- c("Input", "ITM")
Prob_Ovw
```

```
##         [,1]   [,2]   [,3]  [,4]   [,5]
## Input 0.1000 0.3000 0.4000 0.150 0.0500
## ITM   0.1044 0.3038 0.3974 0.147 0.0474
```

Problem 3 - Composition Method

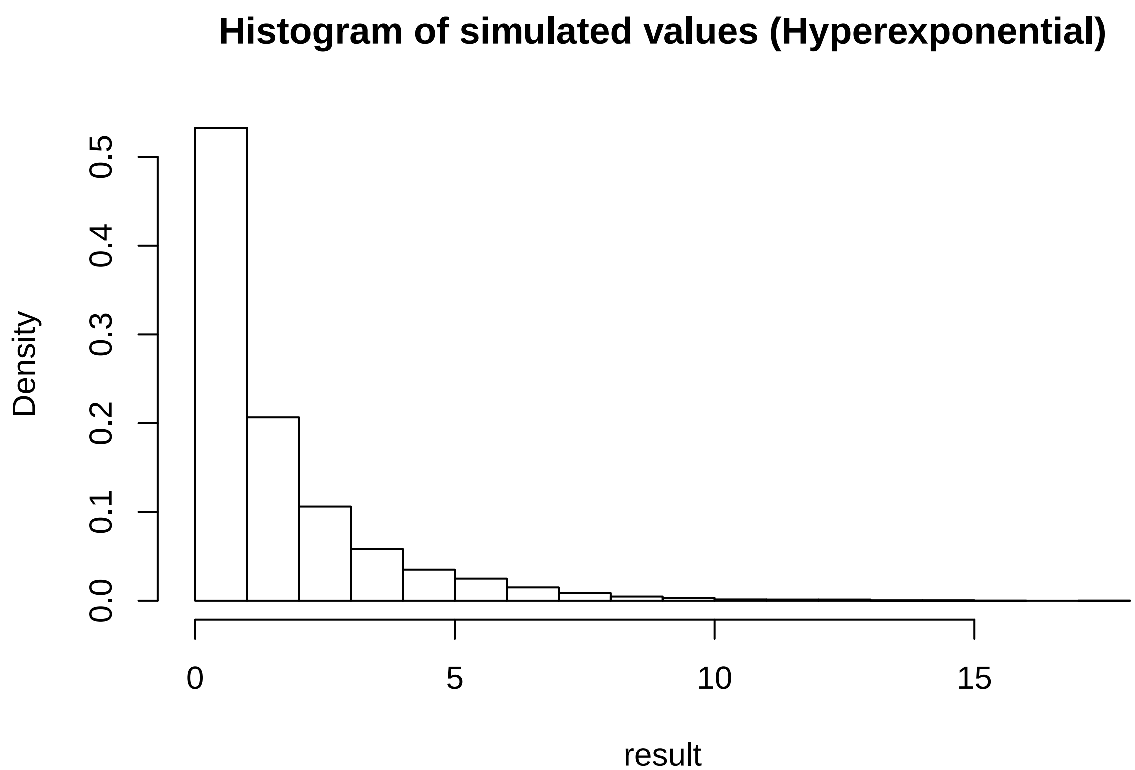
- In general, we want to sample from a CDF F_X with the composition method.
- More specific, we want to generate m random values from a random variable X with *Hyperexp*$(\lambda_1, \alpha_1, \lambda_2, \alpha_2)$ distribution, i.e. X with density

$$f(x) = \sum_{i=1}^{2} \alpha_i \lambda_i e^{-\lambda_i x}$$

- Meaning α_1, α_2 are our probabilities (weights) and $F_i \sim \lambda_i e^{-\lambda_i x}; i = 1, 2$ are exponential distributions, thus we have a mixture of two exponentials with different rate parameter λ.
  In our case we have given $m = 10000, \lambda_1 = 0.5, \alpha_1 = 0.7, \lambda_2 = 2, \alpha_2 = 0.3$
- The composition method consists of two steps. First, we generate a discrete random variable K, such that $P(K = i) = \alpha_i$. Next, given $K = i$, we generate a random variable Z with distribution function F_i. The random variable Z has now desired distribution function F. Thus we implement following algorithm:
  - Generate K that is distributed on the positive integers s.t $P(K = i) = \alpha_i$, by discrete inverse transform.
  - If $K = i$, then generate Z_i from the cdf F_i.
  - Set $X = Z_i$.

```r
set.seed(2019)

# input
m <- 10000
lambda.1 <- 0.5
lambda.2 <- 2
alpha.1 <- 0.7
alpha.2 <- 0.3

# composition function
composition <- function(m, lambda.1, lambda.2, alpha.1, alpha.2) {
  # Generate discrete rv K with P[K=k] = p_k, via discrete inverse transform
  U.1 <- runif(m, 0, 1)
  K <- c()

  for (i in 1:length(U.1)) {
    if (U.1[i] <= alpha.1) {
      K[i] <- 1
      } else {
      K[i] <- 2
      }
  }

  # Generate U ~ U[0,1] independent of K and generate X as exponential rv
  U.2 <- runif(m, 0, 1)
  Z.1 <- log(1-U.2)/(-lambda.1)
  Z.2 <- log(1-U.2)/(-lambda.2)
  X <- c()

  for (i in 1:length(U.2)) {
    if (K[i] == 1) {
```

```
      X[i] <- Z.1[i]
      } else {
      X[i] <- Z.2[i]
      }
  }
  X
}

result <- composition(m, lambda.1, lambda.2, alpha.1, alpha.2)

# plotting
hist(result, freq = FALSE, main = "Histogram of simulated values (Hyperexponential)")
```



Problem 4 - Conditional Distribution with Acceptance-Rejection

part (a)

For f,g as the density functions of distributions F and G, respectively, and for some $c \geq 0$, F is a conditional distribution of $\mathbf{X}$ given that:

$$U \leq \frac{f(X)}{cg(X)} \tag{1}$$

(following from our choice c.)

Proof

We have to show that the conditional distribution of X given that $U \leq \frac{f(X)}{cg(X)}$ ("we accept") is the cdf F, i.e.

$$P\left(X \leq x \mid U \leq \frac{f(X)}{cg(X)}\right) = F(x)$$

By definition of conditional probability we can rewrite this as

$$P\left(X \leq x \mid U \leq \frac{f(X)}{cg(X)}\right) = \frac{P\left(X \leq x, U \leq \frac{f(X)}{cg(X)}\right)}{P\left(U \leq \frac{f(X)}{cg(X)}\right)} = F(x)$$

Thus, it remains to calculate the probabilities in the numerator and denominator

- For $P\left(U \leq \frac{f(X)}{cg(X)}\right)$:

  First note that, as U is uniform $P\left(U \leq \frac{f(X)}{cg(X)} \mid X = x\right) = \frac{f(x)}{cg(x)}$.
  Then unconditioning and knowing that X has density $g(x)$ gives:
  $P\left(U \leq \frac{f(X)}{cg(X)}\right) = \int_{-\infty}^{\infty} \frac{f(x)}{cg(x)} g(x)dx = \frac{1}{c} \int_{-\infty}^{\infty} f(x)dx = \frac{1}{c}$

- For $P\left(X \leq x, U \leq \frac{f(X)}{cg(X)}\right)$:

$$\begin{aligned} P\left(X \leq x, U \leq \frac{f(X)}{cg(X)}\right) &= \int_{-\infty}^{x} P\left(U \leq \frac{f(X)}{cg(X)} \mid X = y \leq x\right) g(y)dy \\ &= \int_{-\infty}^{x} \frac{f(y)}{cg(y)} g(y)dy \\ &= \frac{1}{c} \int_{-\infty}^{x} f(y)dy \\ &= \frac{F(x)}{c} \end{aligned}$$

Hence, we get as desired:

$$P\left(X \leq x \mid U \leq \frac{f(X)}{cg(X)}\right) = \frac{\frac{F(x)}{c}}{\frac{1}{c}} = F(x)$$

part (b)

Because $F(x) = \frac{G(x)-G(a)}{G(b)-G(a)}$, the density of the desired distribution is obtained as:

$$\frac{\partial F(x)}{\partial x} = \frac{\partial}{\partial x}\left(\frac{G(x)-G(a)}{G(b)-G(a)}\right) = \frac{g(x)}{G(b)-G(a)} \tag{2}$$

This means that the condition $f(x) \leq cg(x)$ holds for $c = \frac{1}{G(b)-G(a)}$:

$$\begin{aligned} f(x) &= cg(x) && (3)\\ \frac{g(x)}{G(b)-G(a)} &= cg(x) && (4)\\ \frac{1}{G(b)-G(a)} &= c && (5) \end{aligned}$$

Then the acceptance condition in our algorithm is automatically fullfiled:

$$\begin{aligned} \mathbf{U} &\leq \frac{\frac{g(x)}{G(b)-G(a)}}{cg(x)} && (6)\\ \mathbf{U} &\leq \frac{\frac{1}{G(b)-G(a)}}{c} && (7)\\ \mathbf{U} &\leq \frac{\frac{1}{G(b)-G(a)}}{\frac{1}{G(b)-G(a)}} && (8)\\ \mathbf{U} &\leq 1 && (9) \end{aligned}$$

Thus we can accept any realized (drawn and transformed) $\mathbf{X}$'s that lie between a and b since the remaining condition in (1) is fullfiled for all such $\mathbf{X}$'s.

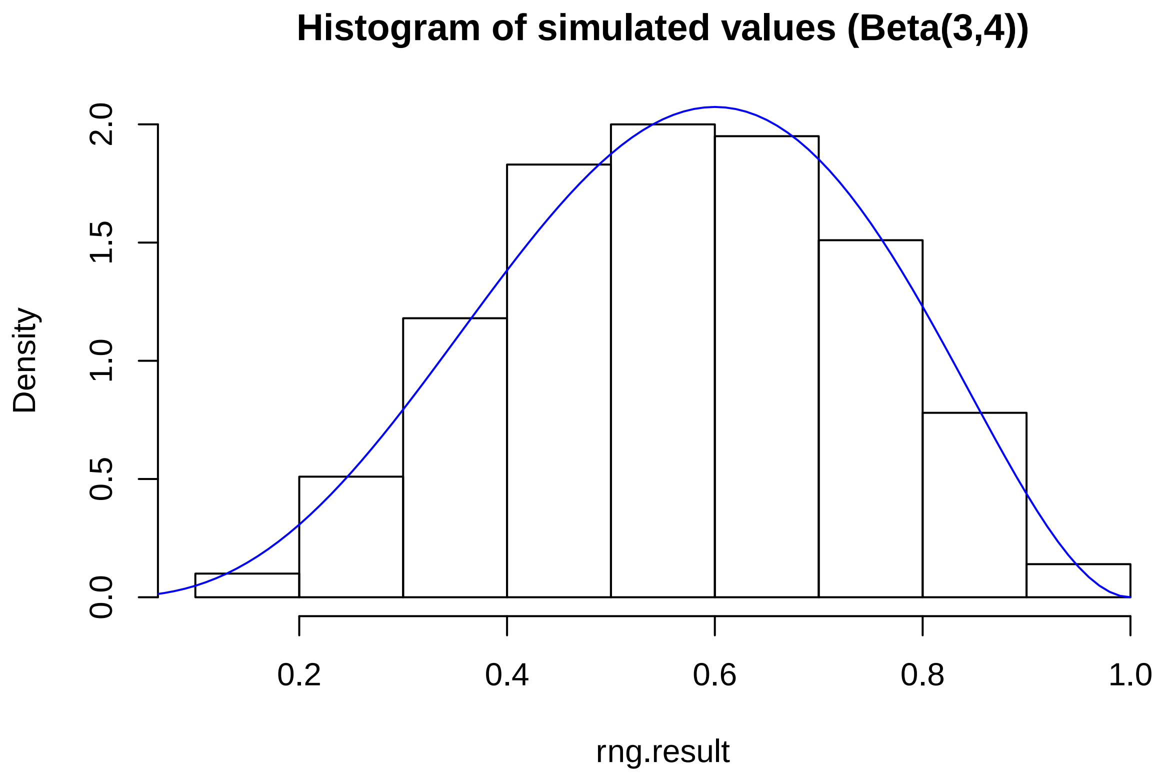
Prolem 5 - Acceptance-Rejection Sampling

- We want to simulate from $X \sim Beta(4, 3)$, that is X has the density

$$f(x) = 60x^3(1 - x^2), \quad 0 \le x \le 1$$

- We simulate 1000 values by using the acceptance-rejection method, where $Y \sim U(0, 1)$.
- In order to choose a reasonable constant, we check the maximum value of $Beta(4, 3)$ over $(0, 1)$.
- Thus, we implement following algorithm:
  - Generate Y from the uniform distribution g.
  - Generate $U \sim U(0, 1)$.
  - If $U \le f(Y)/cg(Y)$, then return Y. Otherwise, start form the beginning again.

```
set.seed(2019)

# input
f <- function(x) 60*(x^3)*(1-x)^2
g <- function(x) 1

beta <- function (x) dbeta(x,4,3)
opt <- optimize(beta, c(0,1), maximum = TRUE)
v <- opt$objective
n <- 1000

# acceptance-rejection function
accep.reject <- function(f, g, v, n) {
  n.accept <- 0
  draws <- c()

  while (n.accept < n) {
    y <- runif(1)
    u <- runif(1)

    if ( u <= f(y)/(v*g(y)) ) {
      n.accept <- n.accept + 1
      draws[n.accept] = y
    }
  }
  draws
}

rng.result <- accep.reject(f, g, v, n)

# plotting
hist(rng.result, freq = FALSE, main = "Histogram of simulated values (Beta(3,4))")
lines(seq(0, 1, 0.01), dbeta(seq(0, 1, 0.01), 4, 3), type = "l", col = "blue")
```


Histogram of simulated values (Beta(3,4))
Density
0.0
0.5
1.0
1.5
2.0
0.2
0.4
0.6
0.8
1.0
rng.result

Problem 6 - Covariance Estimation by MC

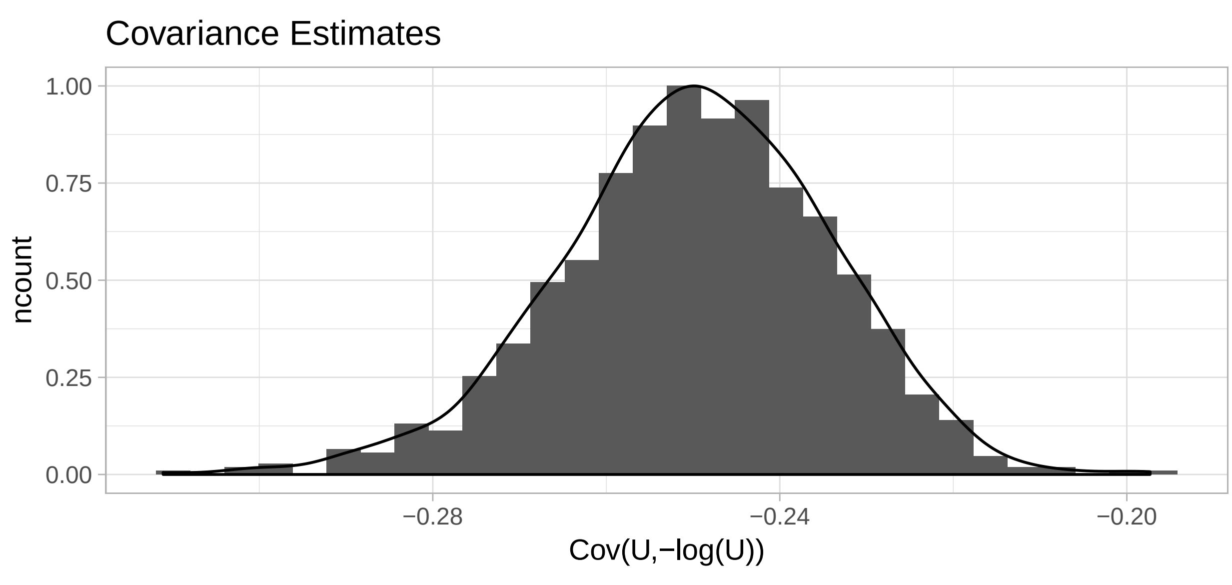
Part (a)

```
set.seed(123)
N<-500
nreps<-1000
COV_UlU<-rep(NA,nreps)

for(i in 1:nreps){

U<-runif(n = N,min = 0,max = 1)
lU<- -log(U)
COV_UlU[i]<- mean((U-mean(U))*(lU-mean(lU)))

}


mean(COV_UlU)
```

```
## [1] -0.2497711
```

```
dt<-as.data.frame(COV_UlU)
ggplot(data = dt,aes(dt$COV_UlU))+
  geom_histogram(aes(y=..ncount..))+
  geom_density(aes(y=..ndensity..))+
  theme_light()+
  ggtitle("Covariance Estimates")+
  xlab("Cov(U,-log(U))")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Part (b)

$$
\begin{aligned}
\mathbb{C}ov(U,Y) &= \mathbb{E}((U-\mathbb{E}(U))(Y-\mathbb{E}(Y))) & (1)\\
&= \mathbb{E}(UY)-\mathbb{E}(Y)\mathbb{E}(U)-\mathbb{E}(U)\mathbb{E}(Y)+\mathbb{E}(U)\mathbb{E}(Y) & (2)\\
&= \mathbb{E}(UY)-\mathbb{E}(Y)\mathbb{E}(U) & (3)
\end{aligned}
$$

From the distributions of X and Y, we know the $\mu_y = \frac{1}{2}$ and $\mu_y = 1$. The individual components can thus be computed as:

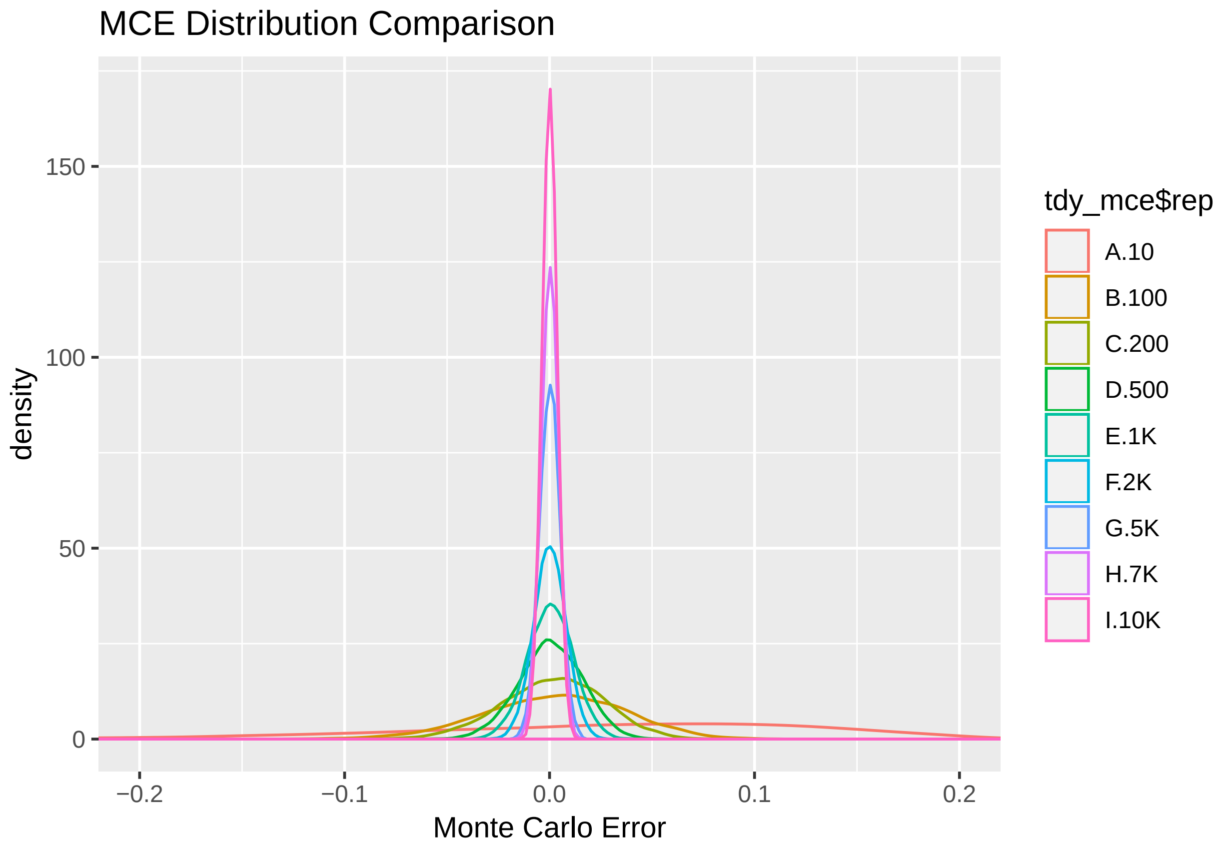
$$
\mathbb{E}(UY) = \mathbb{E}(Y|U)\mathbb{E}(Y) = \int_0^1 -log(u)e^{log(u)}du \int_0^\infty ye^{-y}dy = \left(\frac{1}{4}-0\right)\cdot(0-(-1)) = \frac{1}{4} \tag{4}
$$

$$
\mathbb{E}(Y)\mathbb{E}(U) = \int_0^\infty yf_Y(y)dy \int_0^1 uf_U(x)du = \int_0^\infty ye^{-y}dy \int_0^1 u\cdot 1du = (0-(-1))\cdot\left(\frac{1}{2}-0\right) = \frac{1}{2} \tag{5}
$$

Thus we have

$$
\begin{aligned}
\mathbb{C}ov(U,Y) &= \mathbb{E}(UY)-\mathbb{E}(U)\mathbb{E}(Y) & (6)\\
&= \frac{1}{4}-\frac{1}{2} & (7)\\
&= -\frac{1}{4} & (8)
\end{aligned}
$$

```r
set.seed(123)
nreps<-10000
N<-c(10,100,200,500,1000,2000,5000,7000,10000)
mce<-as.data.frame(matrix(rep(NA,length(N)*nreps),nrow = nreps))
mce<-cbind(c(1:nreps),mce)
names<-c('rep','A.10','B.100','C.200','D.500','E.1K','F.2K','G.5K','H.7K','I.10K')
names(mce)<-names
rslt<-rbind(t(names[2:length(names)]),t(rep(NA,length(N))))
row.names(rslt)<-c('Draws','Average Absolute Error')
for(j in 1:length(N)){

  for(i in 1:nreps){

  U<-runif(n = N[j],min = 0,max = 1)
  lU<- -log(U)
  mce[i,j+1]<- mean((U-mean(U))*(lU-mean(lU)))-(-0.25)

  }
  rslt[2,j]<-mean(mce[,j+1])
}
rslt<-rbind(rslt[1,],abs(round(as.numeric(rslt[2,]),digits = 7)))
tdy_mce<-gather(mce,key = rep,value = MCE)
ggplot(data = tdy_mce,aes(tdy_mce$MCE,colour=tdy_mce$rep,group=tdy_mce$rep))+
  geom_density()+
  ggtitle('MCE Distribution Comparison')+
  xlab('Monte Carlo Error')+
  coord_cartesian(xlim = c(-0.2, 0.2))
```



The following table shows the mean absolute errors depending on number of random draws from U(0,1):

	A.10	B.100	C.200	D.500	E.1K	F.2K	G.5K	H.7K	I.10K
MAE	0.0261	0.0021	0.0015	4.7e-04	3.9e-04	9.8e-05	6e-05	7.8e-05	1.26e-05