Homework 9

Mei Jiao

11/14/2020

7.5.1

1.Implement the important sampling method, with g(x) being the standard normal density. Report your estimates using 1000, 10000 and 50000 samples. Also estimate the variances of the estimates.

As we have the function of F(x), we also need to have calculate the $E(x^2)$. Then set g(x) as standard normal density $N(0,1)$.

```
fx <- function(x){
  x^2/(5*sqrt(2*pi))*exp(-(x-2)^2/2)
}
Hx <- function(x){
  x^2*fx(x)
}
integrate(Hx,-Inf,Inf)
```

```
## 8.6 with absolute error < 8.3e-05
```

```
sd_density <- function(N){
  data <- matrix(nrow = N)
  for(i in 1:N){
    result <- matrix(nrow = 100)
    for(j in 1:100){
      x <- rnorm(1,0,1)
      result[j] <- x^2*fx(x)/dnorm(x,0,1)
    }
    data[i] <- mean(result)
  }
  data
}
x1 <- sd_density(1000)
m1 <- mean(x1)
sd1 <- sd(x1)

x2 <- sd_density(10000)
m2 <- mean(x2)
sd2 <- sd(x2)
```

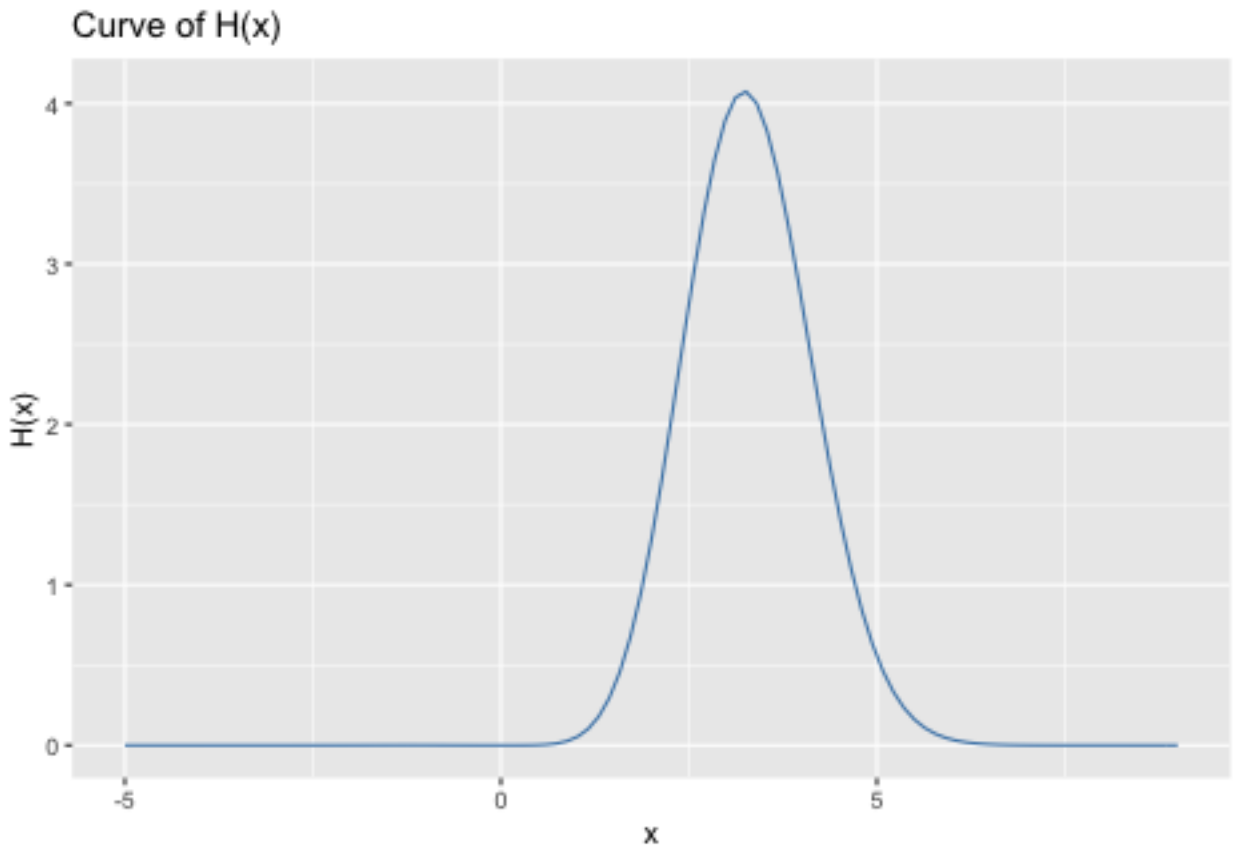
```
x3 <- sd_density(50000)
m3 <- mean(x3)
sd3 <- sd(x3)

data <- matrix(c(m1,m2,m3,sd1,sd2,sd3),ncol=3,byrow=TRUE)
colnames(data) <- c("1000","10000","50000")
rownames(data) <- c("mean","sd")
data <- as.table(data)
data
```

```
##            1000      10000      50000
## mean   7.866642   8.299355   9.478423
## sd    28.069055  33.136762 231.829989
```

2. Design a better importance sampling method to estimate E(X2) using a different g(x). Justify your choice of g(x).Implement your method and estimate E(X2) using 1000, 10000 and 50000 samples. Also estimate the variances of the importance sampling estimates.

First, draw the graph of $H(X) = X^2F(X)$, Then as graph showed we can improve $g(x) = N(4, 1)$

```
library(ggplot2)
p <- ggplot(data = data.frame(x = 0), mapping = aes(x = x))+
  labs(x = 'x', y = 'H(x)', title='Curve of H(x)')
p + stat_function(fun = Hx,col="steelblue") + xlim(-5,9)
```




```
sd_density1 <- function(N){
  data <- matrix(nrow = N)
  for(i in 1:N){
    result <- matrix(nrow = 100)
    for(j in 1:100){
      x <- rnorm(1,4,1)
      result[j] <- x^2*fx(x)/dnorm(x,4,1)
    }
    data[i] <- mean(result)
  }
  data
}
x1 <- sd_density1(1000)
m1 <- mean(x1)
sd1 <- sd(x1)

x2 <- sd_density1(10000)
m2 <- mean(x2)
sd2 <- sd(x2)

x3 <- sd_density1(50000)
m3 <- mean(x3)
sd3 <- sd(x3)

data <- matrix(c(m1,m2,m3,sd1,sd2,sd3),ncol=3,byrow=TRUE)
colnames(data) <- c("1000","10000","50000")
```

```
rownames(data) <- c("mean","sd")
data <- as.table(data)
data
```

```
##           1000     10000     50000
## mean 8.5839580 8.6041402 8.6032677
## sd   0.6536706 0.6446377 0.6416569
```

4. compare the result.

After modified g(x), the result is more closed to 8.6 and SD is smaller than before.

7.5.2

1. Write down and implement an algorithm to sample the path of S(t).

As what we have in the question,

$$\frac{dS(t)}{S(t)} = r\,dt + \sigma dW(t)$$

let

$$V = \log S(t)$$

, we have

$$\frac{\partial V}{\partial t} = 0$$
$$\frac{\partial V}{\partial S} = \frac{1}{S}$$
$$\frac{\partial^2 V}{\partial S^2} = -\frac{1}{S^2}$$

So that

$$dv = \frac{1}{S}dS + \frac{1}{2}(-\frac{1}{S^2})dS^2$$

It equals to

$$dv = (r - \frac{1}{2}\sigma^2)dt + \sigma dW$$

Integral both sides,

$$\log\frac{S(T)}{S(t)} = (r - \frac{1}{2}\sigma^2)(T-t) + \sigma\sqrt{T-t}N(0,1)$$

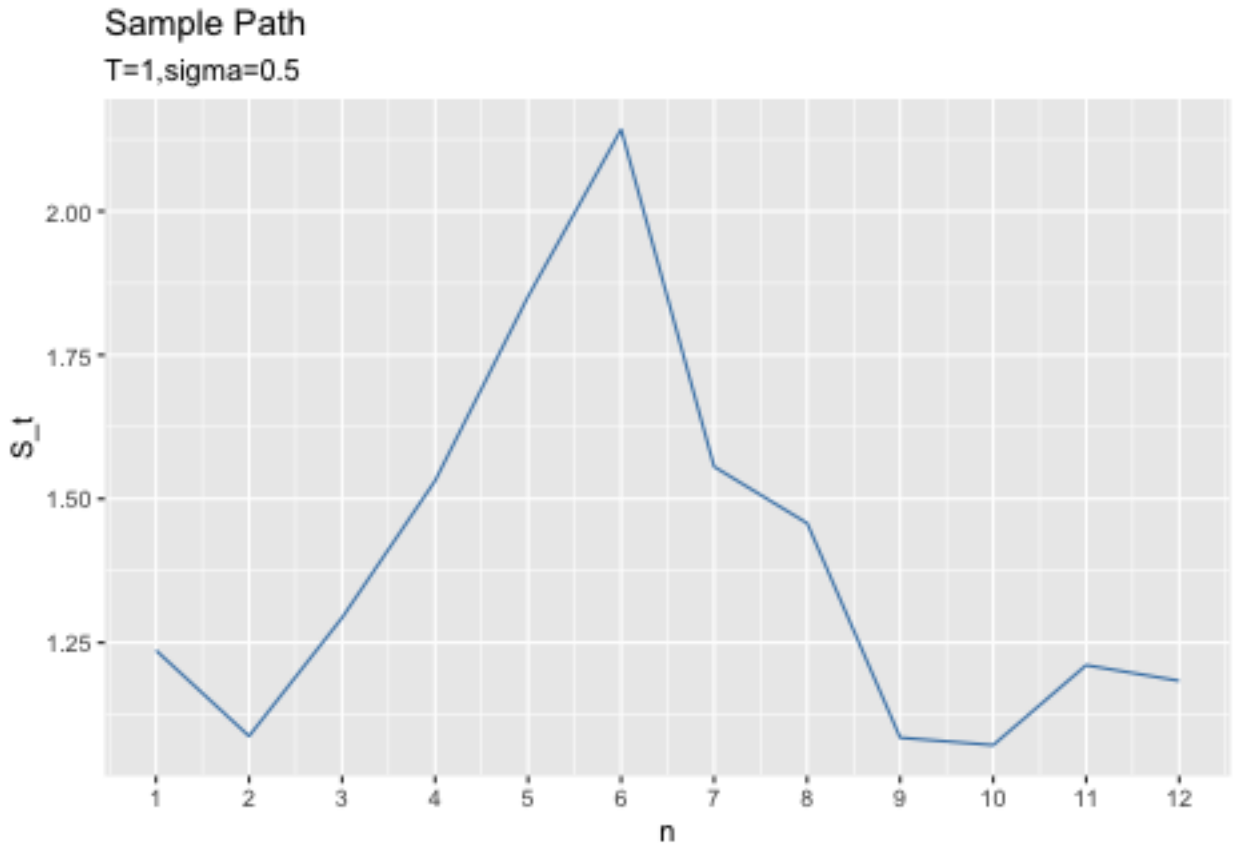
So

$$S(T) = S(t)e^{(r-\frac{1}{2}\sigma^2)(T-t)+\sigma\sqrt{T-t}N(0,1)}$$

As $S(0) = 1$, $r = 0.05$, and $n = 12$, we can calculate S(T)

```
r <- 0.05
n <- 12
S_t <- function(S,T,sigma){
  S*exp((r-sigma^2/2)*T/n+sigma*sqrt(T/n)*rnorm(1,0,1))
}
S_T <- function(T,sigma){
  data <- matrix(nrow = n)
```

```
  S <- 1
  for(i in 1:n){
    S <- data[i] <- S_t(S,T,sigma)
  }
  data
}
a <- S_T(1,0.5)

library(ggplot2)
plot1 <- ggplot(data.frame(n = seq(1:12),S_t = a),aes(x = n, y = S_t))+
        geom_line(col="steelblue", size=0.5)+scale_x_continuous(breaks=seq(1, 12, 1))+
        labs(title="Sample Path", subtitle="T=1,sigma=0.5")
plot1
```



2. let $\sigma = 0.5$ and $T = 1$ change K

We can hace that when K increase, the correlation coefficients decreases.

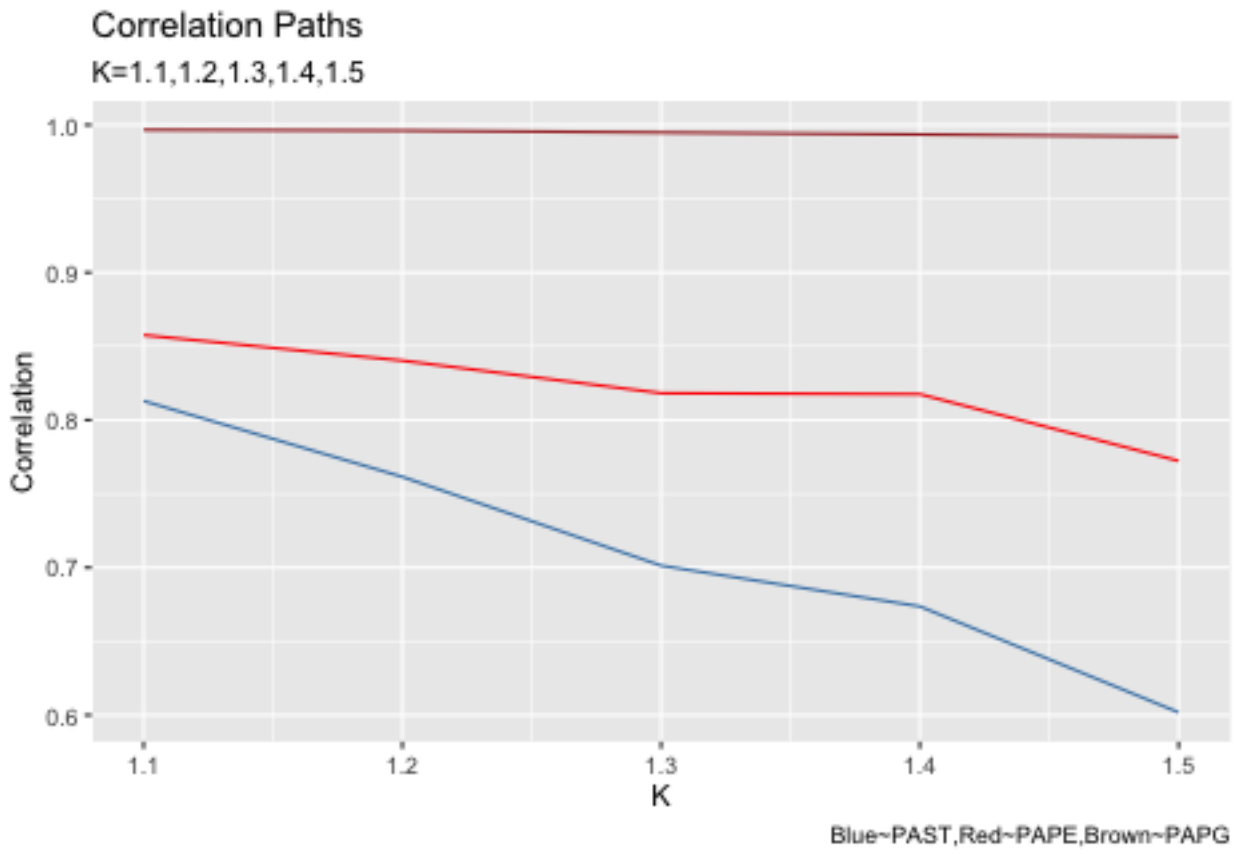
```
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
```

```
##
##     %+%, alpha
```

```
Cor <- function(K){
  data <- matrix(ncol= 4,nrow = 5000)
  for(i in 1:5000){
    ST <- S_T(1,0.5)
    data[i,1] <- ST[12]
    SA <- mean(ST)
    SG <- geometric.mean(ST)
    data[i,2] <- PA <- exp(-r)*max(0,SA-K)
    data[i,3] <- PE <- exp(-r)*max(0,ST[12]-K)
    data[i,4] <- PG <- exp(-r)*max(0,SG-K)
  }
  data
}
PAST <- matrix(nrow = 5)
PAPE <- matrix(nrow = 5)
PAPG <- matrix(nrow = 5)
K <- c(1.1,1.2,1.3,1.4,1.5)
for(i in 1:5){
  x <- Cor(K[i])
  PAST[i] <- cor(x[,1],x[,2])
  PAPE[i] <- cor(x[,3],x[,2])
  PAPG[i] <- cor(x[,4],x[,2])
}

p <- ggplot(data.frame(n = K),aes(x = n)) +
  geom_line(aes(y=PAST),col="steelblue", size=0.5)+
  geom_line(aes(y=PAPE),col="red", size=0.5)+
  geom_line(aes(y=PAPG),col="brown", size=0.5)+
  labs(x="K",y="Correlation",title="Correlation Paths", subtitle="K=1.1,1.2,1.3,1.4,1.5")+
  labs(caption="Blue~PAST,Red~PAPE,Brown~PAPG")+
  scale_x_continuous(breaks=seq(1, 1.5, 0.1))
p
```

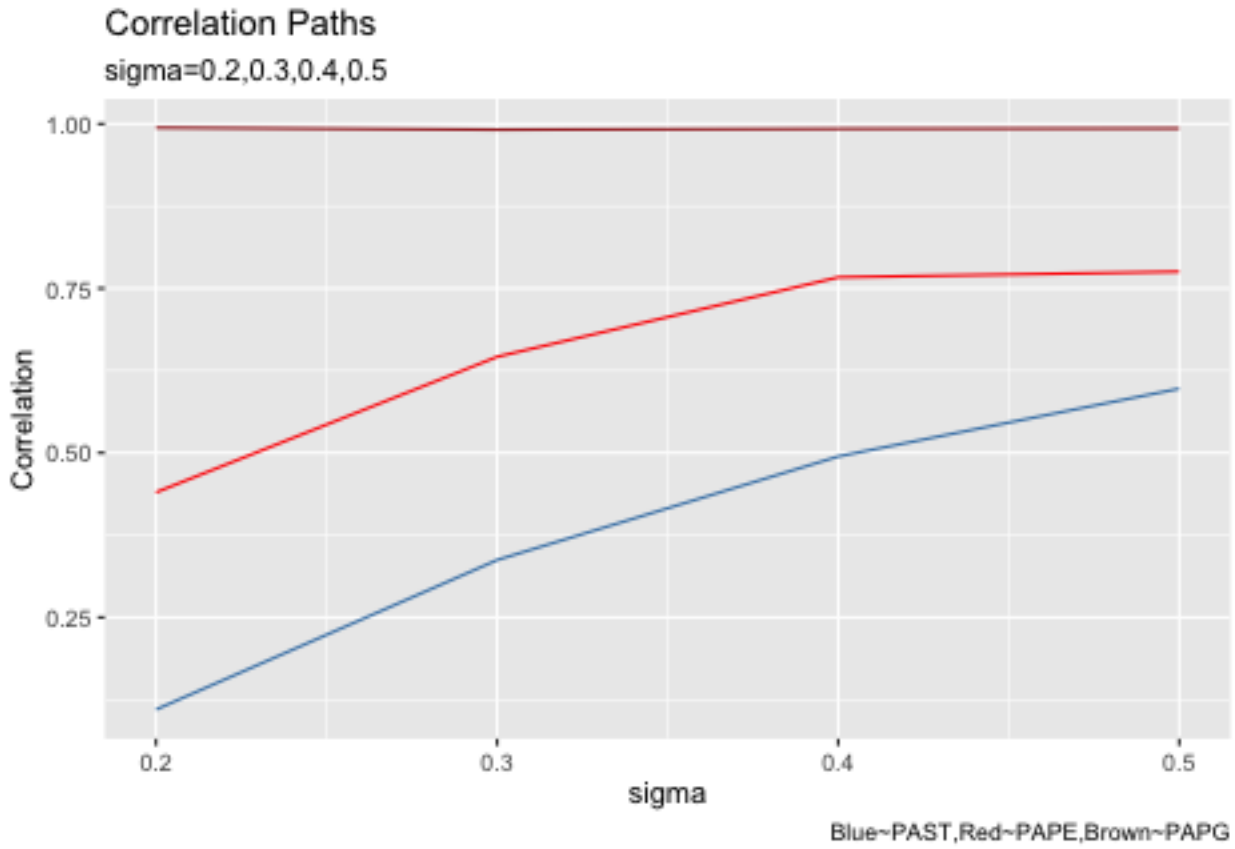


3.let $T = 1$ and $K = 1.5$ change sigma.

As sigma increase, the correlation coefficient also increases.

```
Cor <- function(sigma){
  data <- matrix(ncol= 4,nrow = 5000)
  for(i in 1:5000){
    ST <- S_T(1,sigma)
    data[i,1] <- ST[12]
    SA <- mean(ST)
    SG <- geometric.mean(ST)
    data[i,2] <- PA <- exp(-r)*max(0,SA-1.5)
    data[i,3] <- PE <- exp(-r)*max(0,ST[12]-1.5)
    data[i,4] <- PG <- exp(-r)*max(0,SG-1.5)
  }
  data
}
PAST <- matrix(nrow = 4)
PAPE <- matrix(nrow = 4)
PAPG <- matrix(nrow = 4)
sigma <- c(0.2,0.3,0.4,0.5)
for(i in 1:4){
  x <- Cor(sigma[i])
  PAST[i] <- cor(x[,1],x[,2])
  PAPE[i] <- cor(x[,3],x[,2])
```

```
  PAPG[i] <- cor(x[,4],x[,2])
}

p <- ggplot(data.frame(n = sigma),aes(x = n)) +
  geom_line(aes(y=PAST),col="steelblue", size=0.5)+
  geom_line(aes(y=PAPE),col="red", size=0.5)+
  geom_line(aes(y=PAPG),col="brown", size=0.5)+
  labs(x="sigma",y="Correlation",title="Correlation Paths", subtitle="sigma=0.2,0.3,0.4,0.5")+
  labs(caption="Blue~PAST,Red~PAPE,Brown~PAPG")+
  scale_x_continuous(breaks=seq(0.1, 0.6, 0.1))
p
```

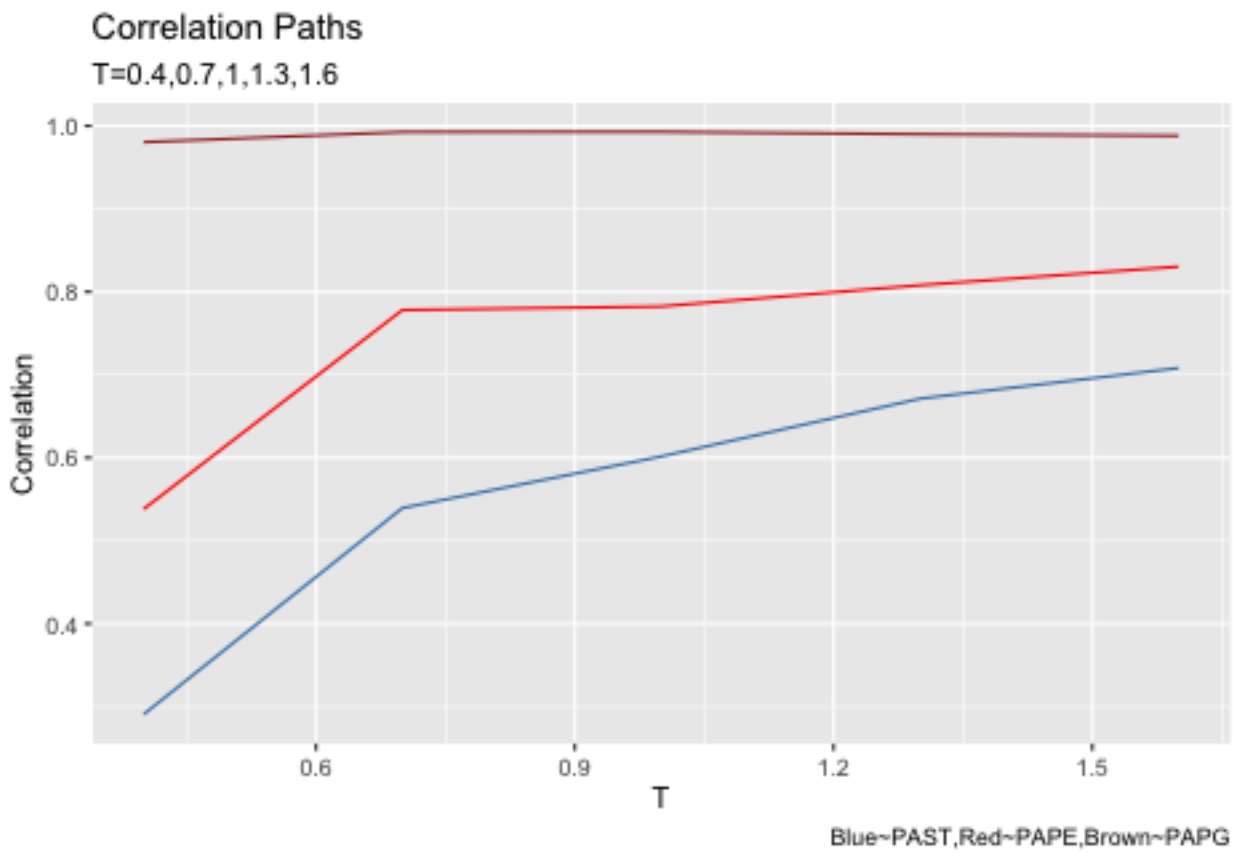


4. Let $\sigma = 0.5$ and $K = 1.5$ change T

As T increase, the correlation coefficient will also increase.

```
Cor <- function(T){
  data <- matrix(ncol= 4,nrow = 5000)
  for(i in 1:5000){
    ST <- S_T(T,0.5)
    data[i,1] <- ST[12]
    SA <- mean(ST)
    SG <- geometric.mean(ST)
    data[i,2] <- PA <- exp(-r)*max(0,SA-1.5)
```

```
    data[i,3] <- PE <- exp(-r)*max(0,ST[12]-1.5)
    data[i,4] <- PG <- exp(-r)*max(0,SG-1.5)
  }
  data
}
PAST <- matrix(nrow = 5)
PAPE <- matrix(nrow = 5)
PAPG <- matrix(nrow = 5)
T <- c(0.4,0.7,1,1.3,1.6)
for(i in 1:5){
  x <- Cor(T[i])
  PAST[i] <- cor(x[,1],x[,2])
  PAPE[i] <- cor(x[,3],x[,2])
  PAPG[i] <- cor(x[,4],x[,2])
}

p <- ggplot(data.frame(n = T),aes(x = n)) +
  geom_line(aes(y=PAST),col="steelblue", size=0.5)+
  geom_line(aes(y=PAPE),col="red", size=0.5)+
  geom_line(aes(y=PAPG),col="brown", size=0.5)+
  labs(x="T",y="Correlation",title="Correlation Paths", subtitle="T=0.4,0.7,1,1.3,1.6")+
  labs(caption="Blue~PAST,Red~PAPE,Brown~PAPG")+
  scale_x_continuous(breaks=seq(0.3, 1.8, 0.3))
p
```



5. $\sigma = 0.4$, $T = 1$ and $K = 1.5$

SD which using control variance is smaller than no control variance.

```
MC <- function() {
  data <- matrix(ncol= 3,nrow = 5000)
  for(i in 1:5000){
    ST <- S_T(1,0.4)
    data[i,1] <- ST[12]
    SA <- mean(ST)
    SG <- geometric.mean(ST)
    data[i,2] <- PA <- exp(-r)*max(0,SA-1.5)
    data[i,3] <- PG <- exp(-r)*max(0,SG-1.5)
  }
  mean_PA <- mean(data[,2])
  mean_PG <- mean(data[,3])
  return(c(mean_PA,mean_PG))
}

CV <- function(n){
  data <- matrix(ncol= 2,nrow = n)
  for(i in 1:n){
    x <- MC()
    data[i,1] <- x[1]
    data[i,2] <- x[2]
  }
  data
}
result <- CV(200)
meanpg <- mean(result[,2])
cov <- cov(result[,1],result[,2])
varpg <- var(result[,2])

fair <- function(n){
  data <- matrix(nrow = n)
  for(i in 1:n){
    ST <- S_T(1,0.4)
    SG <- geometric.mean(ST)
    data[i] <- PG <- exp(-r)*max(0,SG-1.5)
  }
  mean(data)
}
theta <- fair(5000*500)
sd_nocontrol <- sd(result[,1])
result[,1] <- result[,1] - cov/varpg*(result[,2]-theta)
sd_control <- sd(result[,1])
c(sd_nocontrol,sd_control)
```

```
## [1] 0.0007642068 0.0001130177
```