Financial_Engineering_HA3

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

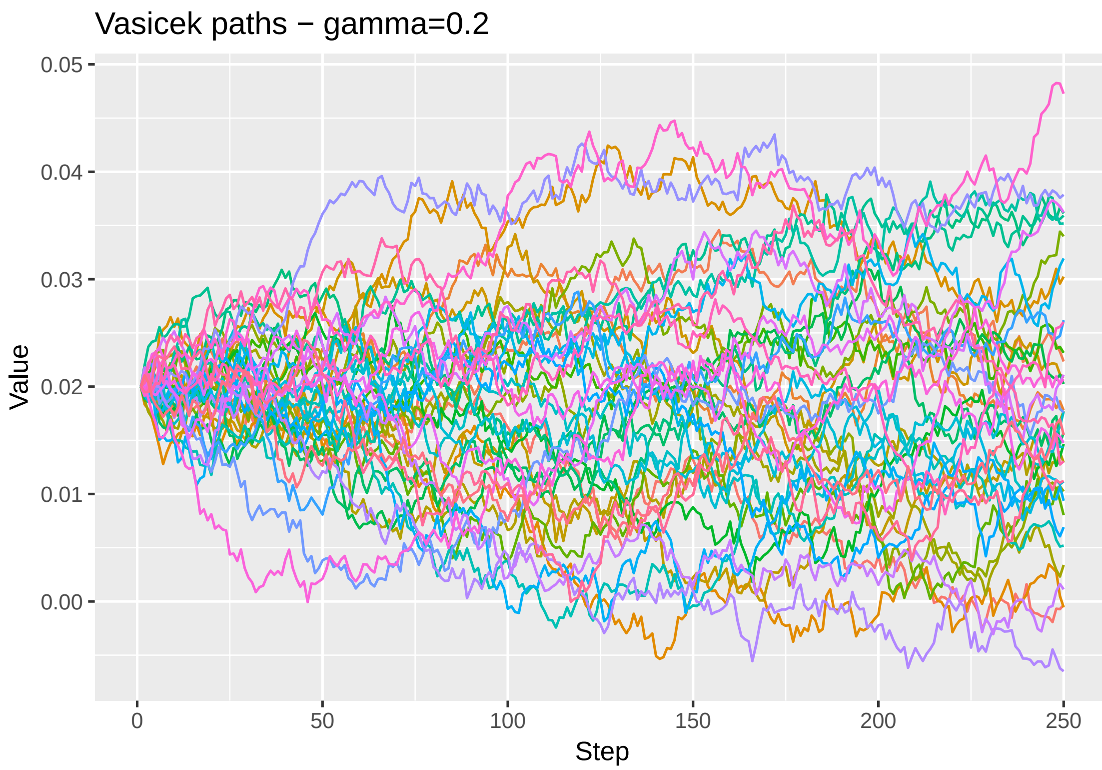
April 7, 2019

Exercise 1

Suppose we have short rate r_t following a Vasicek process with $r_0 = 0.02$, $\bar{r} = 0.05$, $\gamma = 0.2$, $\sigma = 0.015$. Take $T = 1$ and number of steps equal to 250.

We generate 40 paths and plot our results (you can use exact simulation or Euler discretization). Now keeping everything fixed we change the parameter γ to 5. Finally, we visualize the generated paths in a separate plots:

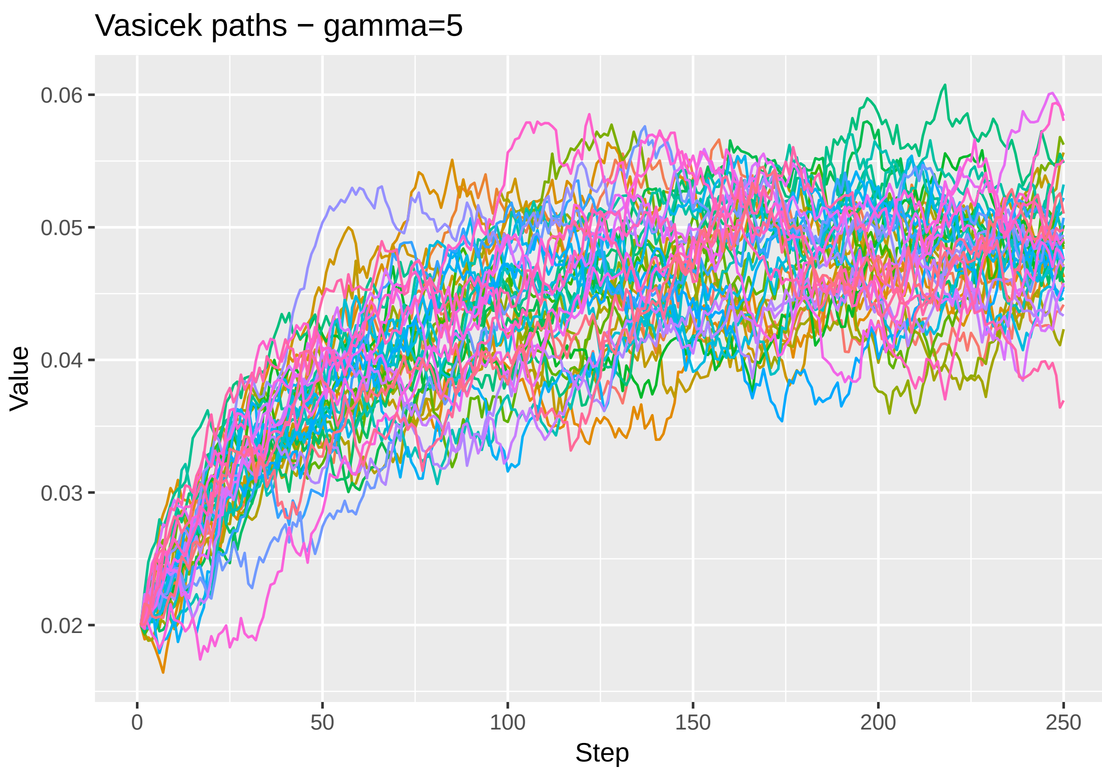
```
# Generation of Path following the Vasicek Model
paths.VAS<-function(r0=.02, r=.05, gamma=.2, sigma=.015, steps=250, n.paths=40){
  Tcap <- 1
  h <- Tcap/steps
  r_t <- as.data.frame(matrix(rep(c(r0,rep(NA,steps-1)),n.paths), ncol = n.paths))
  names(r_t) <- paste0(rep('Path.',n.paths), c(1:n.paths))
    for(j in 1:n.paths) {
      for(i in 2:250) {
        r_t[i,j] <- r_t[i-1,j]*exp(-gamma*h)+r*(1-exp(-gamma*h)) +
                    sigma*sqrt((1-exp(-2*gamma*h))/(2*gamma))*rnorm(1,0,1)
        }
    }
  A <- cbind(c(1:steps), r_t)
  names(A)[1] <- 'Step'
  return(A)
}

## Evaluation
set.seed(2019)
pl.dt.vas <- gather(paths.VAS(), 'Path', 'Value', 2:41)
set.seed(2019)
pl.dt.vas5 <- gather(paths.VAS(gamma=5), 'Path', 'Value', 2:41)

ggplot(pl.dt.vas,aes(Step,Value,group=Path))+
  geom_line(aes(colour=Path))+
  theme(legend.title = element_blank())+
  theme(legend.position = "none")+
  ggtitle('Vasicek paths - gamma=0.2')
```



In case of a very low speed of mean-reversion ($\gamma = 0.2$), we visually see that the process does not come back from the initial $r_0 = 0.02$ to the long-run mean $\bar{r} = 0.05$ within the 250 observations generated. Note that we can also observe some negative values for r_t in this case.

In case we change the speed of mean-reversion to be quite high ($\gamma = 5$), we visually see that the process comes back quite fast from the initial $r_0 = 0.02$ to the long-run mean $\bar{r} = 0.05$.

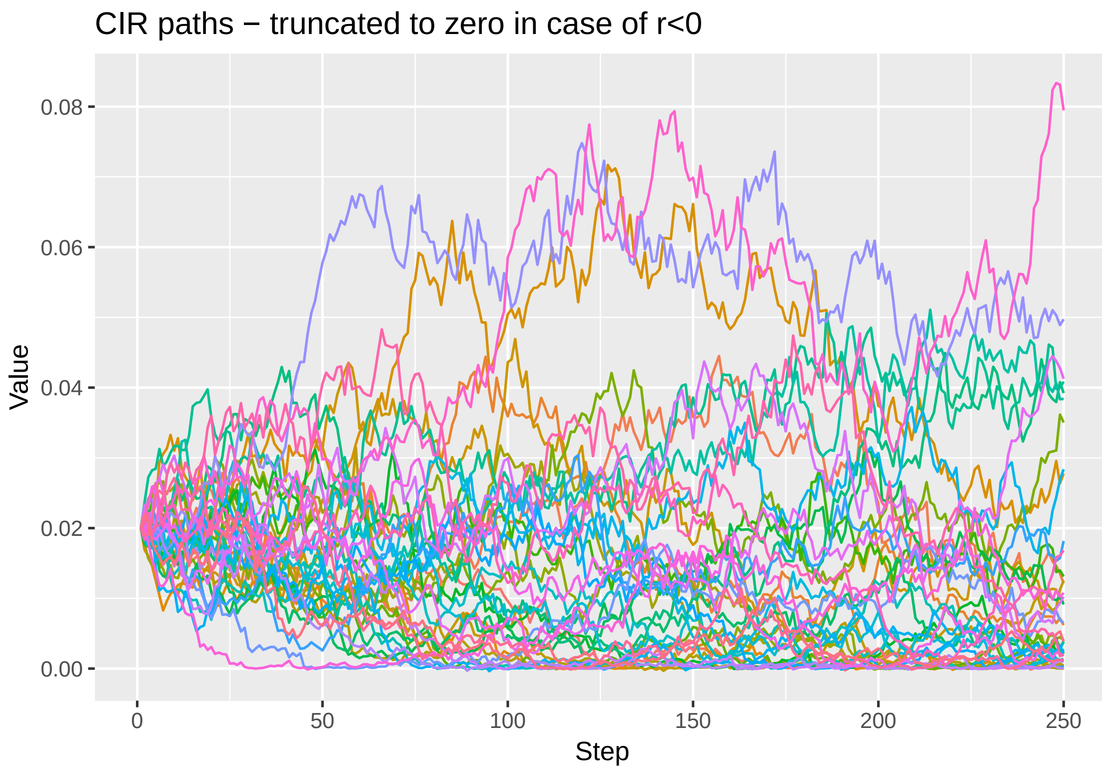
```
ggplot(pl.dt.vas5,aes(Step,Value,group=Path))+
  geom_line(aes(colour=Path))+
  theme(legend.title = element_blank())+
  theme(legend.position = "none")+
  ggtitle('Vasicek paths - gamma=5')
```



Exercise 2

Suppose we have a short rate r_t following CIR process with $r_0 = 0.02$, $\bar{r} = 0.05$, $\gamma = 1.2$, $\alpha = 0.04$. Take $T = 1$ and number of steps equal to 250. We use Euler discretization using truncation or reflection to avoid negative values. Finally, we generate 40 paths and plot our results:

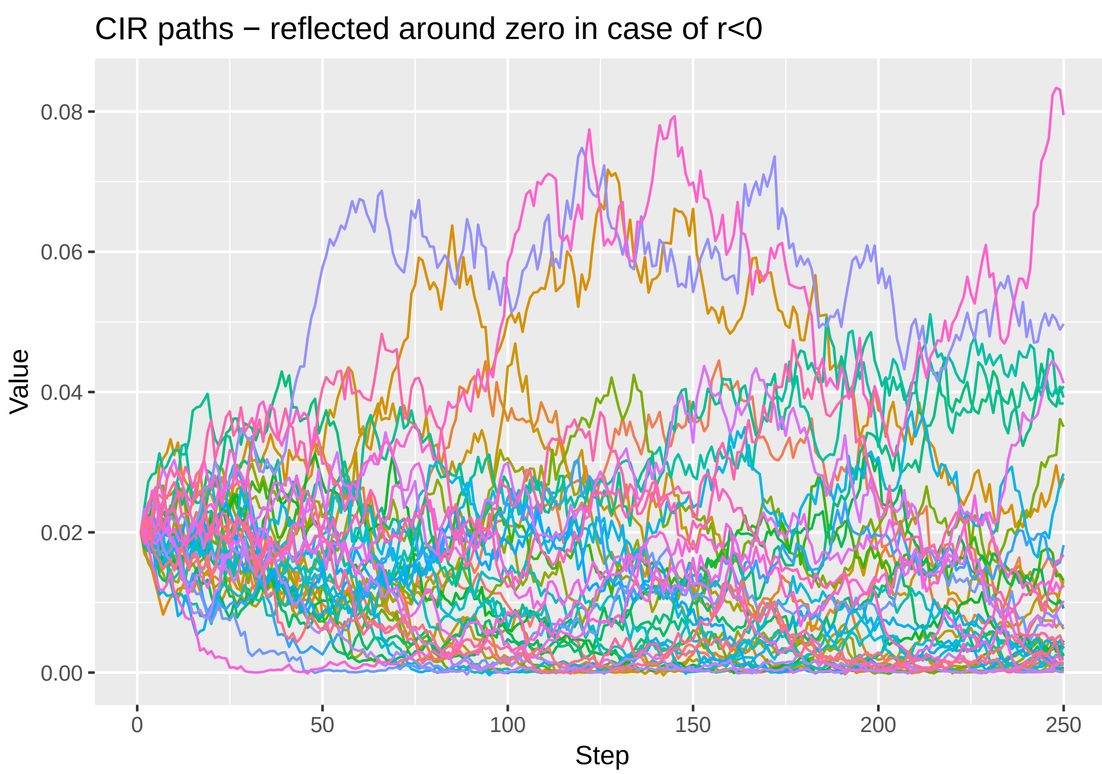
```
## Path Generation
paths.CIR<-function(r0=.02,r=.05, gamma=.2,alpha=.04,steps=250,n.paths=40,negs='truncate'){
  Tcap <- 1
  h <- Tcap/steps
  r_t <- as.data.frame(matrix(rep(c(r0,rep(NA,steps-1)),n.paths), ncol = n.paths))
  names(r_t) <- paste0(rep('Path.',n.paths), c(1:n.paths))
  if(negs=='truncate') {
      for(j in 1:n.paths) {
        for(i in 2:250){
          r_t[i,j] <- gamma*r*h+(1-gamma*h)*max(r_t[i-1,j],0) +
                      sqrt(alpha*max(r_t[i-1,j],0)*h)*rnorm(1,0,1)
          }
      }
  } else if(negs=='reflect'){
      for(j in 1:n.paths){
        for(i in 2:250){
          r_t[i,j] <- gamma*r*h+(1-gamma*h)*max(r_t[i-1,j],-r_t[i-1,j]) +
                      sqrt(alpha*max(r_t[i-1,j],-r_t[i-1,j])*h)*rnorm(1,0,1)
          }
      }
  } else {stop("Options for handling negative r's are 'truncate' and 'reflect'")}

  A <- cbind(c(1:steps), r_t)
  names(A)[1] <- 'Step'
  return(A)
}



## Evaluation
set.seed(2019)
pl.dt <- gather(paths.CIR(), 'Path', 'Value', 2:41)
set.seed(2019)
pl.dt.refl <- gather(paths.CIR(negs = 'reflect'), 'Path', 'Value', 2:41)

ggplot(pl.dt,aes(Step,Value,group=Path)) +
  geom_line(aes(colour=Path))+
  theme(legend.title = element_blank())+
  theme(legend.position = "none")+
  ggtitle('CIR paths - truncated to zero in case of r<0')
```



In the CIR model the diffusion coefficient, $\sqrt{\alpha r_t}$ avoids the possibility of negative interest rates for all parameter values.

As we use Euler discretization we use in this case truncation to avoid negative values.

```r
ggplot(pl.dt.refl,aes(Step,Value,group=Path))+
  geom_line(aes(colour=Path))+
  theme(legend.title = element_blank())+
  theme(legend.position = "none")+
  ggtitle('CIR paths - reflected around zero in case of r<0')
```



Here we use Euler discretization and reflection to avoid negative values.

Exercise 3

We price a Look-back call option for which we know that the maturity $T = 2$ years, the current stock price is $S_0 = 10$, the stock's volatility $\sigma = 0.2$ and assume that $r = 0.01$.

We use monthly monitoring, i.e. $\Delta = 1/12$ and Monte Carlo simulation with 10000 scenarios.

To do so we use naive Monte Carlo for a look-back call option and implement the following Algorithm (we have to generate the whole paths here as our option is path-dependent):

```
Call.naive.mc <- function(S0, sigma, r, delta, T_years, n) {
  m <- T_years/delta
  S <- matrix(rep(0, (m+1)*n), nrow=n)
  S[,1] <- rep(S0, n)
  SM <- payoff <- c(rep(0,n))
  Z <- matrix(rep(0, m*n), nrow=n)

  for (i in 1:n) {
    for (j in 1:m) {
      Z[i,j] <- rnorm(1, mean = 0, sd = 1)
      S[i,j+1] <- S[i,j]*exp((r-sigma^2/2)*delta +sigma*sqrt(delta)*Z[i,j])
      }
    SM[i] <- max(S[i,])
    payoff[i] <- exp(-r*T_years)*(SM[i]-S[i,m])
  }

  price <- mean(payoff)
  se <- sd(payoff)/sqrt(n)
  z.score <- qnorm(1-0.05/2, mean = 0, sd = 1)
  low.b <- price - z.score*se
  up.b <- price + z.score*se
  width <- up.b - low.b
  return(c(MC.naive=price, s.e.=se, Lower=low.b, Upper=up.b, ci.width = width))
}


# inputs
S0 <- 10
sigma <- 0.2
r <- 0.01 # assumed
delta <- 1/12
T_years <- 2
n <- 10000

# Evaluation
set.seed(2019)
Call.naive.mc(S0, sigma, r, delta, T_years, n)
```

```
##   MC.naive       s.e.      Lower      Upper   ci.width
## 1.99394405 0.01388484 1.96673026 2.02115783 0.05442757
```

Exercise 4

Part a)

Suppose we have the setting given in page 116 of the slide set. We solve this exercise by using Euler discretization and come up with a Monte Carlo estimate and the corresponding errors.

Setting: Call option pricing under Heston model

We have call option writen on a stock which is assumed to follow Heston model with parameters $T = 1$, $S_0 = K = 100$, $r = 0.05$, $V_0 = 0.04$, $\alpha = 1.2$ (speed of mean-reversion), $\bar{V} = 0.04$ (long-run volatility level), $\zeta = 0.3$ (diffusion coeff of volatility) and $\rho = -0.5$.

$\rightarrow$ Fourier inversion methods can be used which would yield the price 10.3009.

Now we price this option by using Monte Carlo with an Euler scheme:

- We take $n = 10, n = 50n = 100, n = 500$ and $n = 1000$ as number of (time) steps and $m = 20000$ as the number of paths.
- Additionally we compute the mean absolute error for each case.

```r
# Heston Call pricing (MC - Euler)
Call.Heston.mc.E<-function(S0,K,r,T_years,V0,Vbar,alpha,zeta,rho,m,n,Call.Fourier){
  delta <- T_years/n
  S <- V <- matrix(rep(0, (n+1)*m), nrow=m)
  S[,1] <- rep(S0, m)
  V[,1] <- rep(V0, m)
  payoff <- error <- c(rep(0,m))
  Z <- Z1 <- Zs <- matrix(rep(0, m*n), nrow=m)

  for (i in 1:m) {
    Z[i,] <- rnorm(n, mean = 0, sd = 1)
    Z1[i,] <- rnorm(n, mean = 0, sd = 1)
    Zs[i,] <- rho*Z[i,] + sqrt(1-rho^2)*Z1[i,]
    for (j in 1:n) {
      V[i,j+1] <- max(0, V[i,j]+alpha*delta*T_years*(Vbar-V[i,j])+
                         zeta*sqrt(V[i,j]*delta*T_years)*Z[i,j])
      S[i,j+1]<-max(0,S[i,j]*((1+r*delta*T_years)+sqrt(V[i,j]*delta*T_years)*Zs[i,j]))
      }
    payoff[i] <- exp(-r*T_years)*max(S[i,n]-K, 0)
    error[i] <- payoff[i]-Call.Fourier
  }

  a.m.error <- abs(mean(error))
  Price <- mean(payoff)
  se <- sd(payoff)/sqrt(n)
  z.score <- qnorm(1-0.05/2, mean = 0, sd = 1)
  low.b <- Price - z.score*se
  up.b <- Price + z.score*se
  width <- up.b - low.b
  return(c(MC.Euler=Price, s.e.=se, Lower=low.b, Upper=up.b,
           ci.width = width, abs.mean.error = a.m.error))
}


# inputs
S0 <- K <- 100
```

```r
r <- 0.05
T_years <- 1
V0 <- 0.04
Vbar <- 0.04
alpha <- 1.2
zeta <- 0.3
rho <- -0.5
m <- 20000 #1e06
Call.Fourier <- 10.3009 # Fourier inversion


# Evaluation
set.seed(10)
C.Hes.E.n10 <- Call.Heston.mc.E(S0, K, r, T_years, V0, Vbar, alpha, zeta, rho, m,
                                Call.Fourier, n = 10); C.Hes.E.n10
```

```
##       MC.Euler          s.e.         Lower         Upper      ci.width
##      9.3919497     3.7152938     2.1101076    16.6737918    14.5636842
## abs.mean.error
##      0.9089503
```

```r
set.seed(10)
C.Hes.E.n50 <- Call.Heston.mc.E(S0, K, r, T_years, V0, Vbar, alpha, zeta, rho, m,
                                Call.Fourier, n = 50); C.Hes.E.n50
```

```
##       MC.Euler          s.e.         Lower         Upper      ci.width
##    10.22488191    1.76692985    6.76176304   13.68800078    6.92623775
## abs.mean.error
##     0.07601809
```

```r
set.seed(10)
C.Hes.E.n100 <- Call.Heston.mc.E(S0, K, r, T_years, V0, Vbar, alpha, zeta, rho, m,
                                 Call.Fourier, n = 100); C.Hes.E.n100
```

```
##       MC.Euler          s.e.         Lower         Upper      ci.width
##     10.1942138     1.2555384     7.7334037    12.6550238     4.9216201
## abs.mean.error
##      0.1066862
```

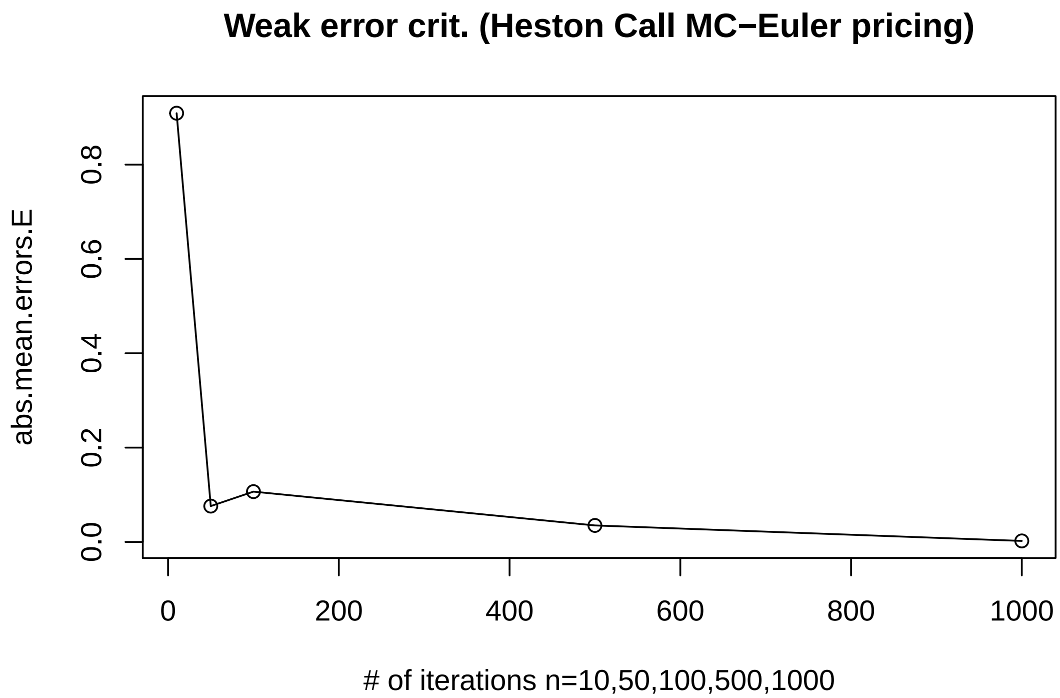
```r
set.seed(10)
C.Hes.E.n500 <- Call.Heston.mc.E(S0, K, r, T_years, V0, Vbar, alpha, zeta, rho, m,
                                 Call.Fourier, n = 500); C.Hes.E.n500
```

```
##       MC.Euler          s.e.         Lower         Upper      ci.width
##    10.26592989    0.56436234    9.15980002   11.37205976    2.21225974
## abs.mean.error
##     0.03497011
```

```r
set.seed(10)
C.Hes.E.n1000 <- Call.Heston.mc.E(S0, K, r, T_years, V0, Vbar, alpha, zeta, rho, m,
                                  Call.Fourier, n = 1000); C.Hes.E.n1000
```

```
##       MC.Euler          s.e.         Lower         Upper      ci.width
##   10.298716131   0.397097917   9.520418515  11.077013747   1.556595233
## abs.mean.error
##    0.002183869
```

```r
abs.mean.errors.E <- c(C.Hes.E.n10[6], C.Hes.E.n50[6], C.Hes.E.n100[6],
                       C.Hes.E.n500[6], C.Hes.E.n1000[6])
n <- c(10, 50, 100, 500, 1000)
plot(n, abs.mean.errors.E, type = "o"
     ,xlab= "# of iterations n=10,50,100,500,1000",
     main = "Weak error crit. (Heston Call MC-Euler pricing)")
```



Partb)

Now we repeat this exercise where we use the Milstein scheme to price the same call option. We also compute the errors corresponding to different n values and compare them with the ones we have obtained above in a) for the Euler discretization.

We apply the Milstein scheme to the variance process only, as this usually turns out to be the cause of potential problems (See e.g. https://dms.umontreal.ca/~bedard/Heston.pdf p.11)

```
# Heston Call pricing (MC - Milstein)
Call.Heston.mc.M<-function(S0,K,r,T_years,V0,Vbar,alpha,zeta,rho,m,n,Call.Fourier){
  delta <- T_years/n
  S <- V <- matrix(rep(0, (n+1)*m), nrow=m)
  S[,1] <- rep(S0, m)
  V[,1] <- rep(V0, m)
  payoff <- error <- c(rep(0,m))
  Z <- Z1 <- Zs <- matrix(rep(0, m*n), nrow=m)

  for (i in 1:m) {
    Z[i,] <- rnorm(n, mean = 0, sd = 1)
    Z1[i,] <- rnorm(n, mean = 0, sd = 1)
    Zs[i,] <- rho*Z[i,] + sqrt(1-rho^2)*Z1[i,]
    for (j in 1:n) {
      V[i,j+1] <- max(0, V[i,j]+alpha*delta*T_years*(Vbar-V[i,j])+
                        # Milstein scheme applied to variance process only
                        zeta*sqrt(V[i,j]*delta*T_years)*Z[i,j]+
                        # as it usually turns out to be the cause of potential problem
                        0.25*zeta^2*delta*T_years*((Z[i,j])^2-1))
      S[i,j+1] <- max(0, S[i,j]*((1+r*delta*T_years)
                                 +sqrt(V[i,j]*delta*T_years)*Zs[i,j]))
      }
    payoff[i] <- exp(-r*T_years)*max(S[i,n]-K, 0)
    error[i] <- payoff[i]-Call.Fourier
  }

  a.m.error <- abs(mean(error))
  Price <- mean(payoff)
  se <- sd(payoff)/sqrt(n)
  z.score <- qnorm(1-0.05/2, mean = 0, sd = 1)
  low.b <- Price - z.score*se
  up.b <- Price + z.score*se
  width <- up.b - low.b
  return(c(MC.Milstein=Price, s.e.=se, Lower=low.b, Upper=up.b,
         ci.width = width, abs.mean.error = a.m.error))
}


# inputs
S0 <- K <- 100
r <- 0.05
T_years <- 1
V0 <- 0.04
Vbar <- 0.04
alpha <- 1.2
zeta <- 0.3
```

```r
rho <- -0.5
m <- 20000 #1e06
Call.Fourier <- 10.3009 # Fourier inversion


# Evaluation
set.seed(10)
C.Hes.M.n10 <- Call.Heston.mc.M(S0, K, r, T_years, V0, Vbar, alpha, zeta, rho, m,
                                Call.Fourier, n = 10); C.Hes.M.n10
```

```
##    MC.Milstein           s.e.          Lower          Upper       ci.width
##      9.3312933      3.6853285      2.1081822     16.5544044     14.4462222
## abs.mean.error
##      0.9696067
```

```r
set.seed(10)
C.Hes.M.n50 <- Call.Heston.mc.M(S0, K, r, T_years, V0, Vbar, alpha, zeta, rho, m,
                                Call.Fourier, n = 50); C.Hes.M.n50
```

```
##    MC.Milstein           s.e.          Lower          Upper       ci.width
##    10.22096705     1.76750713     6.75671673    13.68521737     6.92850064
## abs.mean.error
##     0.07993295
```

```r
set.seed(10)
C.Hes.M.n100 <- Call.Heston.mc.M(S0, K, r, T_years, V0, Vbar, alpha, zeta, rho, m,
                                 Call.Fourier, n = 100); C.Hes.M.n100
```

```
##    MC.Milstein           s.e.          Lower          Upper       ci.width
##     10.1856116      1.2545317      7.7267746     12.6444487      4.9176741
## abs.mean.error
##      0.1152884
```

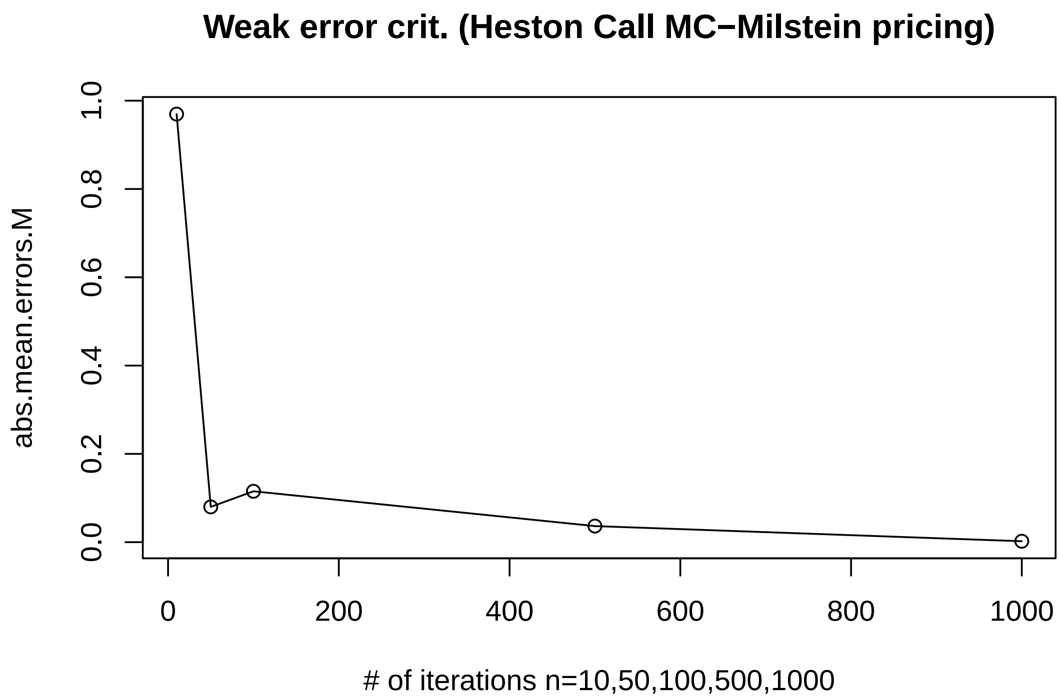
```r
set.seed(10)
C.Hes.M.n500 <- Call.Heston.mc.M(S0, K, r, T_years, V0, Vbar, alpha, zeta, rho, m,
                                 Call.Fourier, n = 500); C.Hes.M.n500
```

```
##    MC.Milstein           s.e.          Lower          Upper       ci.width
##    10.26441876     0.56424235     9.15852408    11.37031343     2.21178935
## abs.mean.error
##     0.03648124
```

```r
set.seed(10)
C.Hes.M.n1000 <- Call.Heston.mc.M(S0, K, r, T_years, V0, Vbar, alpha, zeta, rho, m,
                                  Call.Fourier, n = 1000); C.Hes.M.n1000
```

```
##    MC.Milstein           s.e.          Lower          Upper       ci.width
##   10.298737895    0.397147609    9.520342884   11.077132906    1.556790021
## abs.mean.error
##    0.002162105
```

```r
abs.mean.errors.M <- c(C.Hes.M.n10[6], C.Hes.M.n50[6], C.Hes.M.n100[6],
                       C.Hes.M.n500[6], C.Hes.M.n1000[6])
n <- c(10, 50, 100, 500, 1000)
plot(n, abs.mean.errors.M, type = "o"
     ,xlab= "# of iterations n=10,50,100,500,1000"
     ,main = "Weak error crit. (Heston Call MC-Milstein pricing)")
```

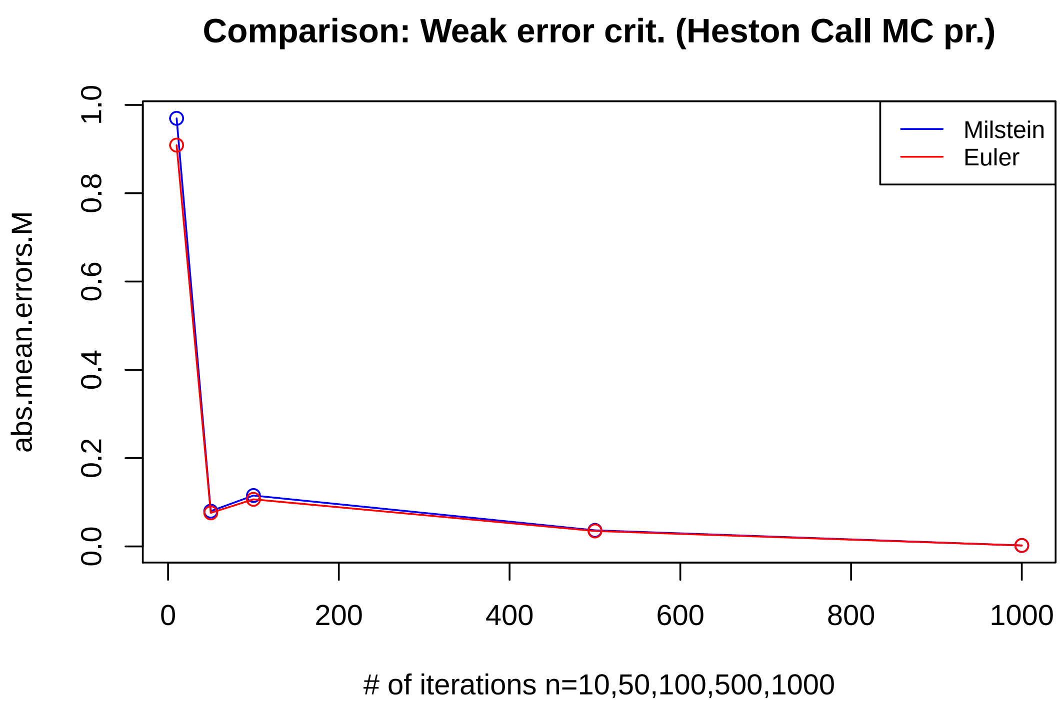


Weak error crit. (Heston Call MC−Milstein pricing)
abs.mean.errors.M
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000
# of iterations n=10,50,100,500,1000

Comparison of Milstein and Euler approximation:

We check the weak order of convergence to get a feeling for the behavior of the error. We see a general decrease in the mean absolute error as the number of time steps increases. The various conditions (on both the option payoff and the SDE) that are required to guarantee a given order of convergence of the schemes is not satisfied, i.e. sometimes Euler will perform better and sometimes Milstein (as we can see in the graph, however the difference is quite small). Overall, the outcome would depend highly on the generated paths.

```
plot(n, abs.mean.errors.M, type = "o", col = "blue"
     ,xlab= "# of iterations n=10,50,100,500,1000"
     ,main = "Comparison: Weak error crit. (Heston Call MC pr.)")
lines(n, abs.mean.errors.E, type = "o", col = "red" )
legend("topright", legend=c("Milstein", "Euler"),
       col=c("blue", "red"), lty=1, cex=0.8)
```



Exercise 5

Part a)

Suppose we have the setting given in page 129 (T=2, S_0=100, σ=0.2, r=0.01, K=100, Δ=1/12, n=10000) of the slide set. We price this Asian option by using control variates method where we choose the standard European call option as the control variate ($Y1 = e^{-rT}(S_T - K)^+$). Moreover, we provide the corresponding 95% confidence intervals and also compute the price by using only the naive MC for comparison.

We know that $m = T/\Delta$. Furthermore, we chose to take $k = 10,000$ for the pilot simulations and $n = 100,000$ for the main simulation (which have to be carried out independently):

```
## Naive MC
Call.mc <- function(S0, K, vol, T_years, delta, r, n) {
  m <- T_years / delta
  S1 <- matrix(rep(0, (m+1)*n), nrow=n)
  S1[,1] <- rep(S0, n)
  payoff.A1 <- c(rep(0,n))
  A1 <- c()
  Z4 <- matrix(rep(0, n*m), nrow=n)
  for (i in 1:n) {
    Z4[i,] <- rnorm(m, mean = 0, sd = 1)
    for (j in 1:m) {
      S1[i,j+1] <- S1[i,j]*exp((r-vol^2/2)*delta + vol*sqrt(delta)*Z4[i,j])
    }
    A1[i] <- mean(S1[i,])
    payoff.A1[i] <- exp(-r*T_years)*max(A1[i]-K, 0)
  }

  Price <- mean(payoff.A1) # MC-estimate
  se <- sd(payoff.A1)/sqrt(n)
  z.score <- qnorm(1-0.05/2, mean = 0, sd = 1)
  low.b <- Price - z.score*se
  up.b <- Price + z.score*se
  width <- up.b - low.b
  return(c(MC.est=Price, s.e.=se, Lower=low.b, Upper=up.b, ci.width = width))
}


## MC with control variates
Call.mc.cv <- function(S0, K, vol, T_years, delta, r, k, n) {
  ## Pilot simulation
  m <- T_years / delta
  S <- matrix(rep(0, (m+1)*k), nrow=k)
  S[,1] <- rep(S0, k)
  payoff.A <- c(rep(0,k))
  payoff.E.C <- c()
  A <- c()
  Z1 <- matrix(rep(0, k*m), nrow=k)
  for (i in 1:k) {
    Z1[i,] <- rnorm(m, mean = 0, sd = 1)
    for (j in 1:m) {
      S[i,j+1] <- S[i,j]*exp((r-vol^2/2)*delta
                              +vol*sqrt(delta)*Z1[i,j]) # Simulate Si values (GBM)
```

```r
    }
    A[i] <- mean(S[i,])
    payoff.A[i] <- exp(-r*T_years)*pmax(A[i]-K, 0) # Arithmetic-average Asian Call
    payoff.E.C[i] <- exp(-r*T_years)*pmax(S[i,m]-K, 0)# Eur. Call payoff: control var.
  }
  a <- -cov(payoff.A, payoff.E.C)/var(payoff.E.C) # estimate of c*
  corr.X.Y <- cor(payoff.E.C, payoff.A)

  ## Main simulation (indep.)
  S2 <- matrix(rep(0, (m+1)*n), nrow=n)
  S2[,1] <- rep(S0, n)
  payoff.A2 <- c(rep(0,n))
  payoff.E.C2 <- c()
  A2 <- c()
  Z2 <- matrix(rep(0, n*m), nrow=n)
  for (i in 1:n) {
    Z2[i,] <- rnorm(m, mean = 0, sd = 1)
    for (j in 1:m) {
      S2[i,j+1] <- S2[i,j]*exp((r-vol^2/2)*delta + vol*sqrt(delta)*Z2[i,j])
    }
    A2[i] <- mean(S2[i,])
    payoff.A2[i] <- exp(-r*T_years)*max(A2[i]-K, 0)
    payoff.E.C2[i] <- exp(-r*T_years)*pmax(S2[i,m]-K, 0)
  }

  payoff_cv <- payoff.A2 + a*(payoff.E.C2 - mean(payoff.E.C2)) # controlled estimator
  Price <- mean(payoff_cv) # MC-CV-estimate
  se <- sd(payoff_cv)/sqrt(n)
  z.score <- qnorm(1-0.05/2, mean = 0, sd = 1)
  low.b <- Price - z.score*se
  up.b <- Price + z.score*se
  width <- up.b - low.b
  return(
  c(MC.cv=Price,s.e.=se,Lower=low.b,Upper=up.b,ci.width=width,corr.X.Y=corr.X.Y)
  )
}

## Input
delta <- (1/12)
T_years <- 2
S0 <- 100
vol <- 0.2
r <- 0.01
K <- 100
n <- 100000
k <- 10000

## Evaluations & Comparison
set.seed(10)
naive.mc <- Call.mc(S0, K, vol, T_years, delta, r, n); naive.mc # naive MC
```

```
##     MC.est       s.e.      Lower      Upper   ci.width
## 6.88043126 0.03450448 6.81280373 6.94805880 0.13525506
```

```r
# MC CV with European Call
set.seed(10)
naive.mc.cv.E <- Call.mc.cv(S0, K, vol, T_years, delta, r, k, n); naive.mc.cv.E
```

```
##      MC.cv       s.e.      Lower      Upper   ci.width   corr.X.Y
## 6.86569028 0.01740628 6.83157459 6.89980597 0.06823138 0.86438523
```

Part b)

Now we take the geometric-average Asian option as control variate ($Y2 = e^{-rT} max(A_{dg} - K, 0)^+$) and compute the price of the arithmetic-average Asian option from before. Again we provide the corresponding 95% confidence intervals and compare this with the result we have obtained above:

```
# define geom mean function
gm_mean <- function(x, na.rm=TRUE){
  exp(sum(log(x[x > 0]), na.rm=na.rm) / length(x))
}

## MC with control variates
Call.mc.cv2 <- function(S0, K, vol, T_years, delta, r, k, n) {
  ## Pilot simulation
  m <- T_years / delta
  S1 <- matrix(rep(0, (m+1)*k), nrow=k)
  S1[,1] <- rep(S0, k)
  payoff.G.A <- c(rep(0,k))
  payoff.A.A <- c(rep(0,k))
  A1 <- A2 <- c()
  Z1 <- matrix(rep(0, k*m), nrow=k)
  for (i in 1:k) {
    Z1[i,] <- rnorm(m, mean = 0, sd = 1)
    for (j in 1:m) {
     S1[i,j+1]<-S1[i,j]*exp((r-vol^2/2)*delta + vol*sqrt(delta)*Z1[i,j])
     #Simulate Si's (GBM by drawing from a standard normal dist.)
     }
    A1[i] <- gm_mean(S1[i,])
    payoff.G.A[i]<-exp(-r*T_years)*pmax(A1[i]-K, 0)#Geom-avg Asian Call:payoffs is CV
    A2[i] <- mean(S1[i,])
    payoff.A.A[i]<-exp(-r*T_years)*pmax(A2[i]-K, 0)#Arithmetic-average Asian Call
  }
  a <- -cov(payoff.A.A, payoff.G.A)/var(payoff.G.A) # estimate of c*
  corr.X.Y <- cor(payoff.G.A, payoff.A.A)

  # Main simulation (indep.)
  S2 <- matrix(rep(0, (m+1)*n), nrow=n)
  S2[,1] <- rep(S0, n)
  payoff.G.A2 <- c(rep(0,n))
  payoff.A.A2 <- c(rep(0,n))
  A3 <- A4 <- c()
  Z2 <- matrix(rep(0, n*m), nrow=n)
  for (i in 1:n) {
    Z2[i,] <- rnorm(m, mean = 0, sd = 1)
    for (j in 1:m) {
      S2[i,j+1] <- S2[i,j]*exp((r-vol^2/2)*delta + vol*sqrt(delta)*Z2[i,j])
    }
    A3[i] <- gm_mean(S2[i,])
    payoff.G.A2[i] <- exp(-r*T_years)*pmax(A3[i]-K, 0)
    A4[i] <- mean(S2[i,])
    payoff.A.A2[i] <- exp(-r*T_years)*max(A4[i]-K, 0)
  }

  payoff_cv<-payoff.A.A2+a*(payoff.G.A2-mean(payoff.G.A2))#controlled estimator
```

```r
  Price <- mean(payoff_cv) # MC-CV-estimate
  se <- sd(payoff_cv)/sqrt(n)
  z.score <- qnorm(1-0.05/2, mean = 0, sd = 1)
  low.b <- Price - z.score*se
  up.b <- Price + z.score*se
  width <- up.b - low.b
  return(
  c(MC.cv=Price,s.e.=se,Lower=low.b,Upper=up.b,ci.width=width,corr.X.Y=corr.X.Y)
  )
}

## Input
delta <- (1/12)
T_years <- 2
S0 <- 100
vol <- 0.2
r <- 0.01
K <- 100
n <- 100000
k <- 10000

## Evaluations & Comparison
set.seed(10)
naive.mc <- Call.mc(S0, K, vol, T_years, delta, r, n); naive.mc # naive MC
```

```
##     MC.est       s.e.      Lower      Upper   ci.width
## 6.88043126 0.03450448 6.81280373 6.94805880 0.13525506
```

```r
# MC CV with European Call
set.seed(10)
naive.mc.cv.E <- Call.mc.cv(S0, K, vol, T_years, delta, r, k, n); naive.mc.cv.E
```

```
##      MC.cv       s.e.      Lower      Upper   ci.width    corr.X.Y
## 6.86569028 0.01740628 6.83157459 6.89980597 0.06823138 0.86438523
```

```r
# MC CV with Geometric-average Asian Call
set.seed(10)
naive.mc.cv.GA <- Call.mc.cv2(S0, K, vol, T_years, delta, r, k, n); naive.mc.cv.GA
```

```
##       MC.cv        s.e.       Lower       Upper    ci.width    corr.X.Y
## 6.865690279 0.001321318 6.863100543 6.868280015 0.005179472 0.999314020
```

Comparison:

The effectiveness of a control variate, as measured by the variance reduction ratio is determined by the strength of the correlation between the quantity of interest X and the control Y. Therefore, the higher the correlation the better the control variant (ceteris paribus).

Theoretically, when using the standard European Call as control variate, we capture the nonlinearity in the option payoff, but we do not capture the essential features of the payoff, as we only concentrate on the final value of the stock.

When using the geometric-average Asian option as control variate we capture both, the essential features of the option payoff, as well as the nonlinearity in the option payoff.

This also coincides with our empirical outcomes (compare the two correlations and widths of the confidence intervals).

Exercise 6:

Part a)

```
# Barrier Option Naive Monte Carlo
Euler.Barrier.MC<-function(S0=60,K=50,b=40,r=0.01,sigma=0.4,Tcap=1/4,m=65,n=10000){
#65 is the num. of days in the current quarter
# iterations for naive monte carlo - plain vanilla
Ps<-rep(NA,n)
S<-as.data.frame(matrix(rep(c(S0,rep(NA,m)),n),ncol=n))
names(S)<-paste0('Path.',c(1:n))
exting<-0 # records the number of knock-outs
for(i in 1:n){
  for(j in 1:(m)){
  S[j+1,i]<-S[j,i]*exp((r-sigma^2/2)*Tcap/m+sigma*sqrt(Tcap/m)*rnorm(1,0,1))
  }
  I_S<-1*(min(S[,i])>b*exp(-0.5826*sigma*sqrt(Tcap/m)))
  exting<-exting+(1-I_S)
  Ps[i]<-exp(-r*Tcap)*(I_S*max(K-S[m+1,i],0))
}
S<-cbind(c(0:m),S)
names(S)[1]<-'Day'
P_est<-mean(Ps)
return(list( P_do=P_est
            ,MC_SE=(sqrt(sum((Ps-P_est)^2)/(n-1)))/sqrt(n)
            ,Knock.Outs=exting
            ,Paths.Frame=S)
       )
}

tic()
n<-5000
Put<-Euler.Barrier.MC()
toc()
```

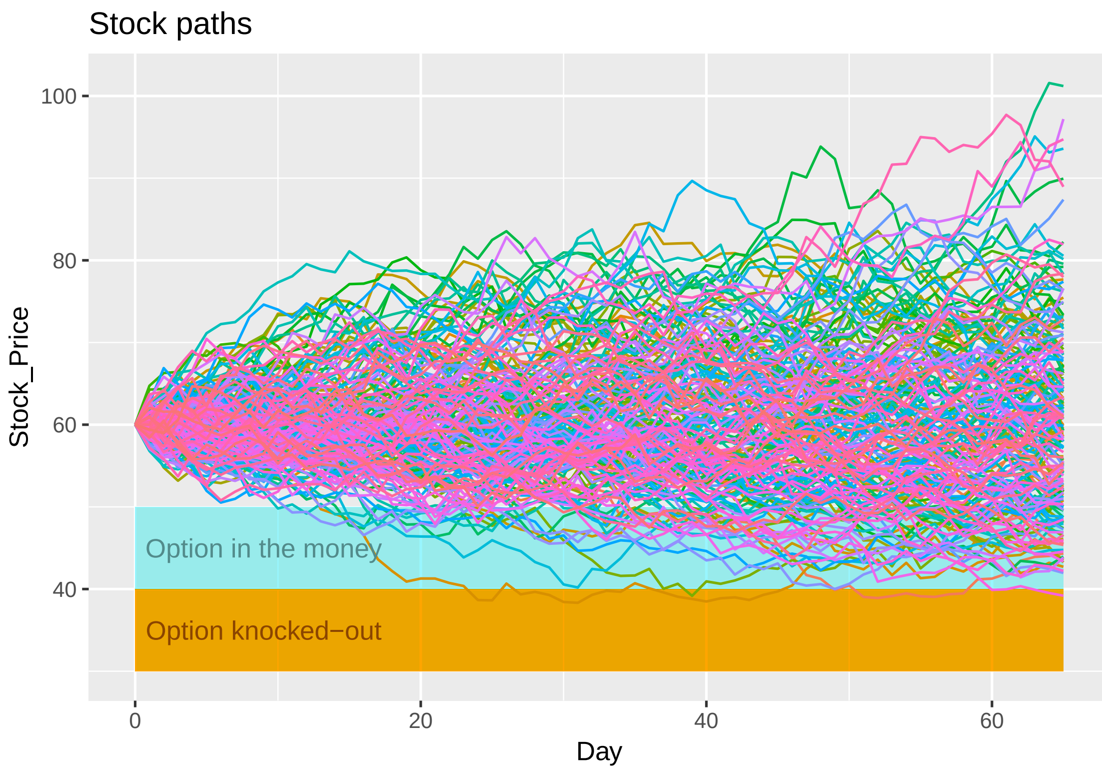
```
## 463.04 sec elapsed
```

```
# Monte Carlo Estmate of Option Price (in $)
Put$P_do
```

```
## [1] 0.6334201
```

```
# Estimator's Standard Error
Put$MC_SE
```

```
## [1] 0.01787216
```




```
Put.200<-Euler.Barrier.MC(n = 200)
s.pl<-gather(Put.200$Paths.Frame,'Path','Stock_Price',2:201)
ggplot(s.pl,aes(Day,Stock_Price,group=Path))+
geom_line(aes(colour=Path))
```

Part b)

Importance sampling is a useful variance reduction technique for the cases of barrier option pricing, where the event of crossing the set knock-out/knock-in price is highly unlikely. Consider an up-and-in option as an example. In such case, if the barrier level is set high above the initial level of an underlying, naive Monte Carlo will result in most of the simulations wasted, as the resulting pay-off will simply be zero.

In similar instances, it is possible to increase efficiency by introducing some drift c towards the barrier price level and then correcting for this change by multiplication of our estimator by the corresponding likelihood ratio. By assuming the stock price follows Geometric Brownian Motion, the price increments will be log-normally distributed:

$$logS_t - logS_{t-\Delta} = Z_t \sim N\left(\mu, \sigma^2\Delta\right)$$

With $\mu = \left(r - \frac{\sigma^2}{2}\right)\Delta$.

In our step-wise generation in the manner : $S_{j+1} = Sj \cdot e^{(\mu+\sigma*\sqrt{\Delta}*Z_j)}$ this would amount to setting $\tilde{\mu} = \mu - c$ (with $c > 0$ for down-options and $c < 0$ for up-options) and to multiplication of the Monte Carlo Estimator $e^{-rT}(I(\mathbf{S})(\pm K \mp S_M)^+)$ [1] by the ratio of two joint density functions:

$$\hat{\Theta}_{IS} = \frac{f(z_1, .., z_M)}{g(z_1, .., z_M)}\mathbb{E}_g\left[I(\mathbf{S})(\pm K \mp S_M)^+)\right]$$

[1] with $+K - S_M$ for put option and $-K + S_M$ for the call option