Ayush Sharma 1 February 2018

(150123046)

S(0) = 100, K = 100, T = 1, M = 100, r = 8%, vol = 20%

Use the following set of u and d for your program:

; .

Here , with being the number of subintervals in the time interval . Use the continuous compounding convention in your calculations (i.e., both in and in the pricing formula).

Note : The payoff of the look-back option is given by

, where .

Question 1.

- American Options

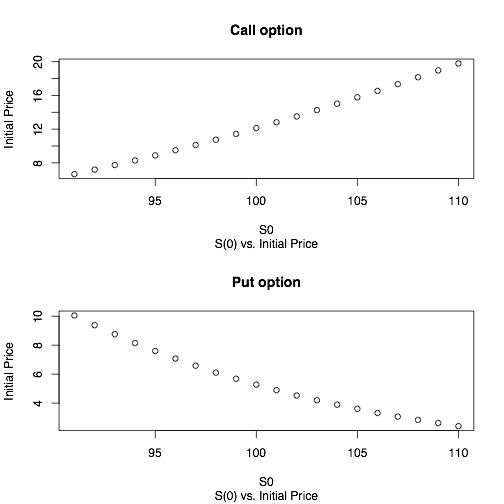
For the given information:

Initial call option price = 12.12305 .

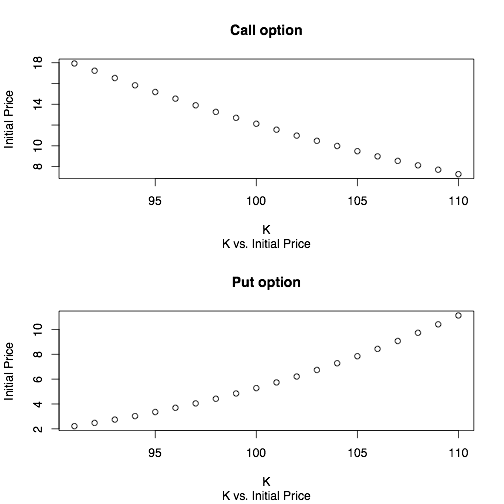
Initial put option price = 5.279837 .

Now, plot of the initial prices of both call and put options by varying one of the parameters at a time (as given below) while keeping the other parameters fixed (as given above) :

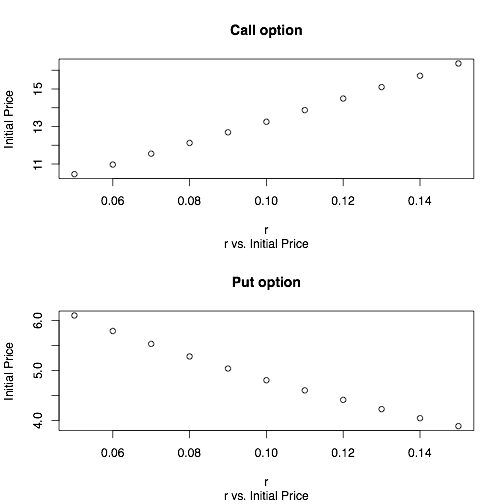
1. S(0)



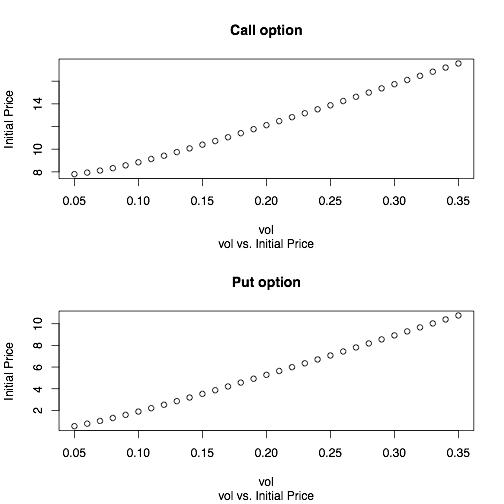
1. K

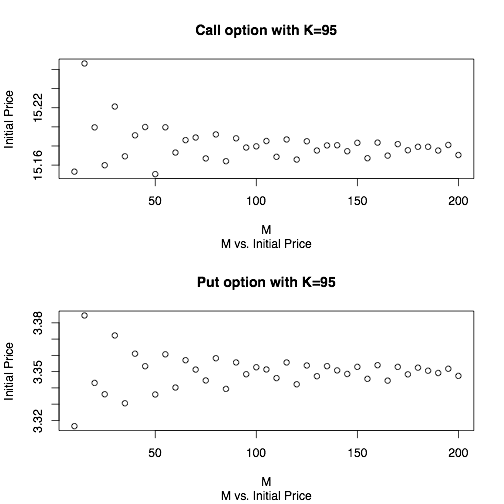
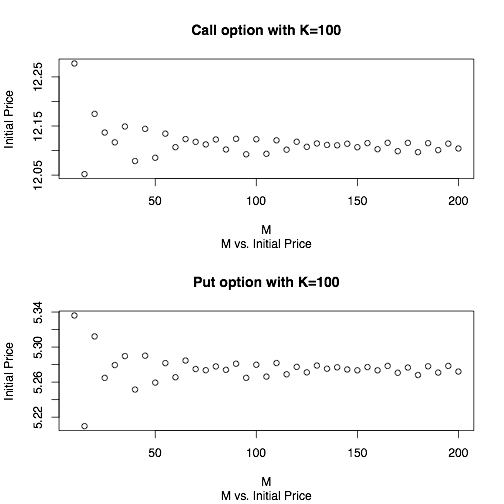
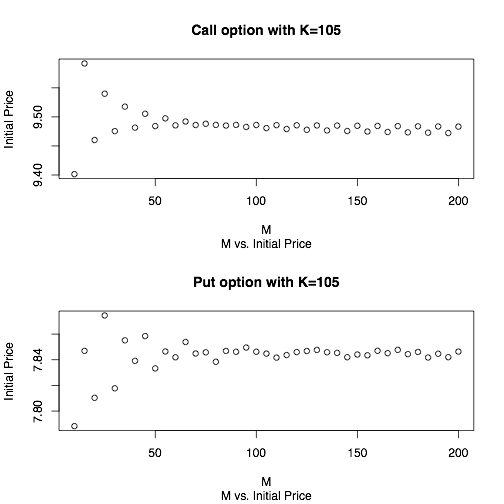


1. r



1. vol



1. M (Do this for three values of K , K = 95; 100; 105 ).
   * K = 95
   * K = 100
   * K = 105

Question 2.

- Look-back Options

For M = 5 , Initial option price = 9.119299 .

For M = 10 , Initial option price = 10.08058 .

For M = 25 , Initial option price = 11.0035 .

For M = 50 ,

Error in matrix(0, nrow = (2^M), ncol = (M + 1)) :

invalid 'nrow' value (too large or NA)

In addition: Warning message:

In matrix(0, nrow = (2^M), ncol = (M + 1)) :

NAs introduced by coercion to integer range

Here, a noticeable point is that for M = 25, two 6.5 GB matrices are declared, which can be minimised, thereby, increasing time taken (due to time-complexity and space complexity tradeoff), when using brute force approach (i.e. taking care of all cases separately).

The values of options at time t = 0, for the above values of M that I have taken, are different, i.e. look-back options have different initial values for different number of subintervals of the time interval [0, T], and follow an increasing pattern with M.

The values of the options at all intermediate time points for M = 5.

| Time →  Possibility ↓ | **0.0** | **0.2** | **0.4** | **0.6** | **0.8** | **1.0** |
| --- | --- | --- | --- | --- | --- | --- |
| **1** | 9.119298986 | 9.027951166 | 8.548076184 | 7.416771005 | 5.501638814 | 0 |
| **2** |  | 9.504839866 | 9.799118754 | 9.955271273 | 9.571391532 | 11.18141312 |
| **3** |  |  | 7.147915757 | 6.201916454 | 4.600479678 | 0 |
| **4** |  |  | 12.16866466 | 13.71286297 | 15.63185188 | 19.45269154 |
| **5** |  |  |  | 6.201916454 | 4.600479678 | 0 |
| **6** |  |  |  | 8.32461467 | 8.003613781 | 9.349916553 |
| **7** |  |  |  | 7.148418208 | 6.680842999 | 6.374517471 |
| **8** |  |  |  | 17.58206271 | 21.18808935 | 25.39456348 |
| **9** |  |  |  |  | 4.600479678 | 0 |
| **10** |  |  |  |  | 8.003613781 | 9.349916553 |
| **11** |  |  |  |  | 3.846928884 | 0 |
| **12** |  |  |  |  | 13.07138097 | 16.26637356 |
| **13** |  |  |  |  | 3.846928884 | 0 |
| **14** |  |  |  |  | 10.68090443 | 13.5780025 |
| **15** |  |  |  |  | 10.68090443 | 13.5780025 |
| **16** |  |  |  |  | 25.05122946 | 29.48259712 |
| **17** |  |  |  |  |  | 0 |
| **18** |  |  |  |  |  | 9.349916553 |
| **19** |  |  |  |  |  | 0 |
| **20** |  |  |  |  |  | 16.26637356 |
| **21** |  |  |  |  |  | 0 |
| **22** |  |  |  |  |  | 7.81841603 |
| **23** |  |  |  |  |  | 5.330382286 |
| **24** |  |  |  |  |  | 21.23497691 |
| **25** |  |  |  |  |  | 0 |
| **26** |  |  |  |  |  | 7.81841603 |
| **27** |  |  |  |  |  | 2.901350497 |
| **28** |  |  |  |  |  | 18.80594512 |
| **29** |  |  |  |  |  | 2.901350497 |
| **30** |  |  |  |  |  | 18.80594512 |
| **31** |  |  |  |  |  | 18.80594512 |
| **32** |  |  |  |  |  | 32.10539404 |

Question 3.

- Look-back Options

{Markov based computationally efficient binomial memoized algorithm}

For M = 5 , Initial option price = 9.119299 .

For M = 10 , Initial option price = 10.08058 .

For M = 25 , Initial option price = 11.0035 .

For M = 50 ,

Time limit exceeded, 20 minutes passed.

Aliter,

{Monte Carlo Simulation based algorithm}

For M = 5 , Initial option price = 9.126311 .

For M = 10 , Initial option price = 10.06328 .

For M = 25 , Initial option price = 10.99132 .

For M = 50 , Initial option price = 11.52002 .

The values of options at time t = 0, for the above values of M that I have taken, are different, i.e. look-back options have different initial values for different number of subintervals of the time interval [0, T], and follow an increasing pattern with M.

However, while using the computationally efficient algorithm it is difficult to tabulate the values of the options at all intermediate time points for M = 5 (or for any other M).

Comparatively, only the Monte Carlo Simulation based algorithm is able to handle the case of M = 50, as it approximates and not calculates. Hence, it also has the least time complexity and space complexity.

Code (R)

*### Script for question 1.*

#American Options

rm(list = ls());

pos <- function(x){

ind = which(x < 0)

z = x

z[ind] <- 0 ## z now contains the x^+

return(z)

}

greater <- function(x, y){

ind = which(x < y)

z = x

z[ind] <- y[ind] ## z now contains the max(x,y) iterative.

return(z)

}

binopt <- function( S0, K, r, t, M, vol, Flag ){

dt = t/M;

time <- seq(0, t, by=dt);

u = exp(vol\*sqrt(dt) + (r-((vol^2)/2))\*dt);

d = exp(-vol\*sqrt(dt) + (r-((vol^2)/2))\*dt);

#Continuous Compounding so "exp(r\*dt)".

if ((d > exp(r\*dt)) | (exp(r\*dt) > u)){

stop('ArbitargePossible as "d < exp(r\*dt) < u" not true.');

}

AssetPrice <- matrix(0, nrow = (M+1), ncol = (M+1));

OptionValue <- matrix(0, nrow = (M+1), ncol = (M+1));

AssetPrice[1,1] = S0;

for (i in 2:(M+1)){

AssetPrice[1, i] <- AssetPrice[1, (i-1)]\*u;

AssetPrice[2:i, i] <- AssetPrice[1:(i-1), (i-1)]\*d;

}

#Flag = 1 for a call option, or Flag = 0 for a put option.

if (Flag == 1){

OptionValue[, M+1] <- pos(AssetPrice[, M+1] - K);

}

else if (Flag == 0){

OptionValue[, M+1] <- pos(K - AssetPrice[, M+1]);

}

#Continuous Compounding so "exp(r\*dt)".

p\_ = (exp(r\*dt) - d)/(u-d);

q\_ = (u - exp(r\*dt))/(u-d);

for (i in seq(M, 1, by=-1)){

# for European Options:

# OptionValue[1:i, i] <- (p\_\*OptionValue[1:i, i+1] + q\_\*OptionValue[2:(i+1), i+1])/exp(r\*dt);

# for American Options:

if (Flag == 1){

OptionValue[1:i, i] <- greater(pos(AssetPrice[1:i, i] - K), (p\_\*OptionValue[1:i, i+1] + q\_\*OptionValue[2:(i+1), i+1])/exp(r\*dt));

}

else if (Flag == 0){

OptionValue[1:i, i] <- greater(pos(K - AssetPrice[1:i, i]), (p\_\*OptionValue[1:i, i+1] + q\_\*OptionValue[2:(i+1), i+1])/exp(r\*dt));

}

}

result <- list("AssetPrice" = AssetPrice, "OptionValue" = OptionValue, "time" = time);

return(result);

}

S0 = 100;

K = 100;

t = 1;

M = 100;

r = 0.08;

vol = 0.2;

cat("Initial call option price =", (binopt( S0, K, r, t, M, vol, 1 )$OptionValue)[1,1], ".\n");

cat("Initial put option price =", (binopt( S0, K, r, t, M, vol, 0 )$OptionValue)[1,1], ".\n");

##Part a.

S0 = 91:110;

ac <- 1:length(S0); ap <- 1:length(S0);

for (i in 1:length(S0)) {

ac[i] <- (binopt( S0[i], K, r, t, M, vol, 1 )$OptionValue)[1,1];

ap[i] <- (binopt( S0[i], K, r, t, M, vol, 0 )$OptionValue)[1,1];

}

pdf("1a.pdf");

par(mfrow=c(2,1));

plot(S0,ac, main="Call option", sub="S(0) vs. Initial Price",

xlab="S0", ylab="Initial Price");

plot(S0,ap, main="Put option", sub="S(0) vs. Initial Price",

xlab="S0", ylab="Initial Price");

dev.off();

S0 = 100;

#\*#

##Part b.

K = 91:110;

bc <- 1:length(K); bp <- 1:length(K);

for (i in 1:length(K)) {

bc[i] <- (binopt( S0, K[i], r, t, M, vol, 1 )$OptionValue)[1,1];

bp[i] <- (binopt( S0, K[i], r, t, M, vol, 0 )$OptionValue)[1,1];

}

pdf("1b.pdf");

par(mfrow=c(2,1));

plot(K,bc, main="Call option", sub="K vs. Initial Price",

xlab="K", ylab="Initial Price");

plot(K,bp, main="Put option", sub="K vs. Initial Price",

xlab="K", ylab="Initial Price");

dev.off();

K = 100;

#\*#

##Part c.

r = seq(0.05, 0.15, by=0.01);

cc <- 1:length(r); cp <- 1:length(r);

for (i in 1:length(r)) {

cc[i] <- (binopt( S0, K, r[i], t, M, vol, 1 )$OptionValue)[1,1];

cp[i] <- (binopt( S0, K, r[i], t, M, vol, 0 )$OptionValue)[1,1];

}

pdf("1c.pdf");

par(mfrow=c(2,1));

plot(r,cc, main="Call option", sub="r vs. Initial Price",

xlab="r", ylab="Initial Price");

plot(r,cp, main="Put option", sub="r vs. Initial Price",

xlab="r", ylab="Initial Price");

dev.off();

r = 0.08;

#\*#

##Part d.

vol = seq(0.05, 0.35, by=0.01);

dc <- 1:length(vol); dp <- 1:length(vol);

for (i in 1:length(vol)) {

dc[i] <- (binopt( S0, K, r, t, M, vol[i], 1 )$OptionValue)[1,1];

dp[i] <- (binopt( S0, K, r, t, M, vol[i], 0 )$OptionValue)[1,1];

}

pdf("1d.pdf");

par(mfrow=c(2,1));

plot(vol,dc, main="Call option", sub="vol vs. Initial Price",

xlab="vol", ylab="Initial Price");

plot(vol,dp, main="Put option", sub="vol vs. Initial Price",

xlab="vol", ylab="Initial Price");

dev.off();

vol = 0.2;

#\*#

##Part e.

M = seq(10, 200, by=5);

ec\_k95 <- 1:length(M); ec\_k100 <- 1:length(M); ec\_k105 <- 1:length(M);

ep\_k95 <- 1:length(M); ep\_k100 <- 1:length(M); ep\_k105 <- 1:length(M);

for (i in 1:length(M)) {

ec\_k95[i] <- (binopt( S0, 95, r, t, M[i], vol, 1 )$OptionValue)[1,1];

ep\_k95[i] <- (binopt( S0, 95, r, t, M[i], vol, 0 )$OptionValue)[1,1];

ec\_k100[i] <- (binopt( S0, 100, r, t, M[i], vol, 1 )$OptionValue)[1,1];

ep\_k100[i] <- (binopt( S0, 100, r, t, M[i], vol, 0 )$OptionValue)[1,1];

ec\_k105[i] <- (binopt( S0, 105, r, t, M[i], vol, 1 )$OptionValue)[1,1];

ep\_k105[i] <- (binopt( S0, 105, r, t, M[i], vol, 0 )$OptionValue)[1,1];

}

pdf("1e\_k95.pdf");

par(mfrow=c(2,1));

plot(M,ec\_k95, main="Call option with K=95", sub="M vs. Initial Price",

xlab="M", ylab="Initial Price");

plot(M,ep\_k95, main="Put option with K=95", sub="M vs. Initial Price",

xlab="M", ylab="Initial Price");

dev.off();

pdf("1e\_k100.pdf");

par(mfrow=c(2,1));

plot(M,ec\_k100, main="Call option with K=100", sub="M vs. Initial Price",

xlab="M", ylab="Initial Price");

plot(M,ep\_k100, main="Put option with K=100", sub="M vs. Initial Price",

xlab="M", ylab="Initial Price");

dev.off();

pdf("1e\_k105.pdf");

par(mfrow=c(2,1));

plot(M,ec\_k105, main="Call option with K=105", sub="M vs. Initial Price",

xlab="M", ylab="Initial Price");

plot(M,ep\_k105, main="Put option with K=105", sub="M vs. Initial Price",

xlab="M", ylab="Initial Price");

dev.off();

M = 100;

#\*#

rm(list = ls())

*### Script for Question 2.*

#Lookback Options

rm(list = ls());

pos <- function(x){

ind = which(x < 0)

z = x

z[ind] <- 0 ## z now contains the x^+

return(z)

}

greater <- function(x, y){

ind = which(x < y)

z = x

z[ind] <- y[ind] ## z now contains the max(x,y) iterative.

return(z)

}

binopt <- function( S0, r, t, M, vol ){

dt = t/M;

time <- seq(0, t, by=dt);

u = exp(vol\*sqrt(dt) + (r-((vol^2)/2))\*dt);

d = exp(-vol\*sqrt(dt) + (r-((vol^2)/2))\*dt);

#Continuous Compounding so "exp(r\*dt)".

if ((d > exp(r\*dt)) | (exp(r\*dt) > u)){

stop('ArbitargePossible as "d < exp(r\*dt) < u" not true.');

}

AssetPrice <- matrix(0, nrow = (2^M), ncol = (M+1));

OptionValue <- matrix(0, nrow = (2^M), ncol = (M+1));

MaxAsset <- matrix(0, nrow = (2^M), ncol = (M+1));

AssetPrice[1,1] = S0; MaxAsset[1,1] = S0;

for (i in 2:(M+1)){

AssetPrice[seq(1, 2^(i-1), 2), i] <- AssetPrice[(1:2^(i-2)), (i-1)]\*u;

AssetPrice[seq(2, 2^(i-1), 2), i] <- AssetPrice[(1:2^(i-2)), (i-1)]\*d;

MaxAsset[seq(1, 2^(i-1), 2), i] <- greater(AssetPrice[seq(1, 2^(i-1), 2), i], MaxAsset[(1:2^(i-2)), i-1]);

MaxAsset[seq(2, 2^(i-1), 2), i] <- greater(AssetPrice[seq(2, 2^(i-1), 2), i], MaxAsset[(1:2^(i-2)), i-1]);

}

OptionValue[, M+1] <- (MaxAsset[, M+1] - AssetPrice[, M+1]);

#Continuous Compounding so "exp(r\*dt)".

p\_ = (exp(r\*dt) - d)/(u-d);

q\_ = (u - exp(r\*dt))/(u-d);

for (i in seq(M, 1, by=-1)){

#for European Options:

#OptionValue[1:i, i] <- (p\_\*OptionValue[1:i, i+1] + q\_\*OptionValue[2:(i+1), i+1])/exp(r\*dt);

#for American Options:

#if (Flag == 1){

# OptionValue[1:i, i] <- greater(pos(AssetPrice[1:i, i] - K), (p\_\*OptionValue[1:i, i+1] + q\_\*OptionValue[2:(i+1), i+1])/exp(r\*dt));

#}

#else if (Flag == 0){

# OptionValue[1:i, i] <- greater(pos(K - AssetPrice[1:i, i]), (p\_\*OptionValue[1:i, i+1] + q\_\*OptionValue[2:(i+1), i+1])/exp(r\*dt));

#}

#for Lookback Options:

OptionValue[1:2^(i-1), i] <- (p\_\*OptionValue[seq(1, 2^i, 2), i+1] + q\_\*OptionValue[seq(2, 2^i, 2), i+1])/exp(r\*dt);

}

result <- list("AssetPrice" = AssetPrice, "OptionValue" = OptionValue, "time" = time);

return(result);

}

S0 = 100;

t = 1;

M = c(5, 10, 25, 50);

r = 0.08;

vol = 0.2;

# Time <- as.character(binopt( S0, r, t, 5, vol )$time);

OptionValue <- (binopt( S0, r, t, 5, vol )$OptionValue);

# write.csv(OptionValue, file = "2.csv", dec = ".", col.names = Time);

write.csv(OptionValue, file = "2.csv");

for (i in 1:length(M)){

cat("For M = ", M[i],", ");

cat("Initial option price =", (binopt( S0, r, t, M[i], vol )$OptionValue)[1,1], ".\n");

}

# OptionValue <- (binopt( S0, r, t, 5, vol )$OptionValue);

rm(list = ls())

*### Script for Question 3.*

*{Markov based computationally efficient binomial memoized algorithm}*

#Lookback Options - Efficient

# library("functools");

library("memoise");

rm(list = ls());

pos <- function(x){

ind = which(x < 0)

z = x

z[ind] <- 0 ## z now contains the x^+

return(z)

}

greater <- function(x, y){

ind = which(x < y)

z = x

z[ind] <- y[ind] ## z now contains the max(x,y) iterative.

return(z)

}

v <- function( N, n, s, m ){

if (N == n){

return(m - s);

}

else{

return( ( p\_\*v(N, n+1, u\*s, greater(m, u\*s))

+

q\_\*v(N, n+1, d\*s, greater(m, d\*s))

)

/

R

);

}

}

memo\_v <- memoise(v);

# memo\_v <- Memoise(v);

markov\_binopt <- function( S0, r, t, M, vol ){

dt = t/M;

time <- seq(0, t, by=dt);

#Continuous Compounding so "exp(r\*dt)".

R <<- exp(r\*dt);

u <<- exp(vol\*sqrt(dt) + (r-((vol^2)/2))\*dt);

d <<- exp(-vol\*sqrt(dt) + (r-((vol^2)/2))\*dt);

if ((d > R) | (R > u)){

stop('ArbitargePossible as "d < exp(r\*dt) < u" not true.');

}

p\_ <<- (R - d)/(u-d);

q\_ <<- (u - R)/(u-d);

result <- memo\_v(M, 0, S0, S0);

return(result);

}

S0 = 100;

t = 1;

#M = c(5, 10, 25, 50);

M = c(5, 10, 25);

r = 0.08;

vol = 0.2;

for (i in 1:length(M)){

cat("For M = ", M[i],", ");

cat("Initial option price =", markov\_binopt( S0, r, t, M[i], vol ), ".\n");

}

rm(list = ls())

{Monte Carlo Simulation based algorithm}

#Lookback Options - MC

rm(list = ls());

set.seed(47);

sample <- 1000000;

pos <- function(x){

ind = which(x < 0)

z = x

z[ind] <- 0 ## z now contains the x^+

return(z)

}

greater <- function(x, y){

ind = which(x < y)

z = x

z[ind] <- y[ind] ## z now contains the max(x,y) iterative.

return(z)

}

v <- function(M, S0, u, d, R){

rv <- runif(sample\*M);

{

ind = which(rv < 0.5);

rv[ind] <- u;

rv[-ind] <- d;

}

m <- c(rep(1, sample), rv);

m <- matrix(data = m, nrow = sample, ncol = (M+1));

for (i in 3:(M+1)){

m[,i] <- m[,(i-1)]\*m[,i];

}

max <- apply(m, 1, max);

{

max <- S0\*max;

SM <- S0\*m[,(M+1)];

}

return(mean(max-SM)/(R^M));

}

mc\_binopt <- function( S0, r, t, M, vol ){

dt = t/M;

time <- seq(0, t, by=dt);

#Continuous Compounding so "exp(r\*dt)".

R <<- exp(r\*dt);

u <<- exp(vol\*sqrt(dt) + (r-((vol^2)/2))\*dt);

d <<- exp(-vol\*sqrt(dt) + (r-((vol^2)/2))\*dt);

if ((d > R) | (R > u)){

stop('ArbitargePossible as "d < exp(r\*dt) < u" not true.');

}

p\_ <<- (R - d)/(u-d);

q\_ <<- (u - R)/(u-d);

result <- v(M, S0, u, d, R);

return(result);

}

S0 = 100;

t = 1;

M = c(5, 10, 25, 50);

r = 0.08;

vol = 0.2;

for (i in 1:length(M)){

cat("For M = ", M[i],", ");

cat("Initial option price =", mc\_binopt( S0, r, t, M[i], vol ), ".\n");

}

rm(list = ls())