Ayush Sharma
(150123046)

1 February 2018

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

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

$$u = e^{\sigma\sqrt{\Delta t} + (r - \frac{1}{2}\sigma^2)\Delta t}; \qquad d = e^{-\sigma\sqrt{\Delta t} + (r - \frac{1}{2}\sigma^2)\Delta t}.$$

Here $\Delta t = \dfrac{T}{M}$, with M being the number of subintervals in the time interval $[0, T]$. Use the continuous compounding convention in your calculations (i.e., both in $\tilde{\mathbb{P}}$ and in the pricing formula).

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

$$V = \max_{0 \le i \le M} \{S(i)\} - S(M), \qquad \text{where } S(i) = S(i\Delta t).$$

QUESTION 1.

- American Options

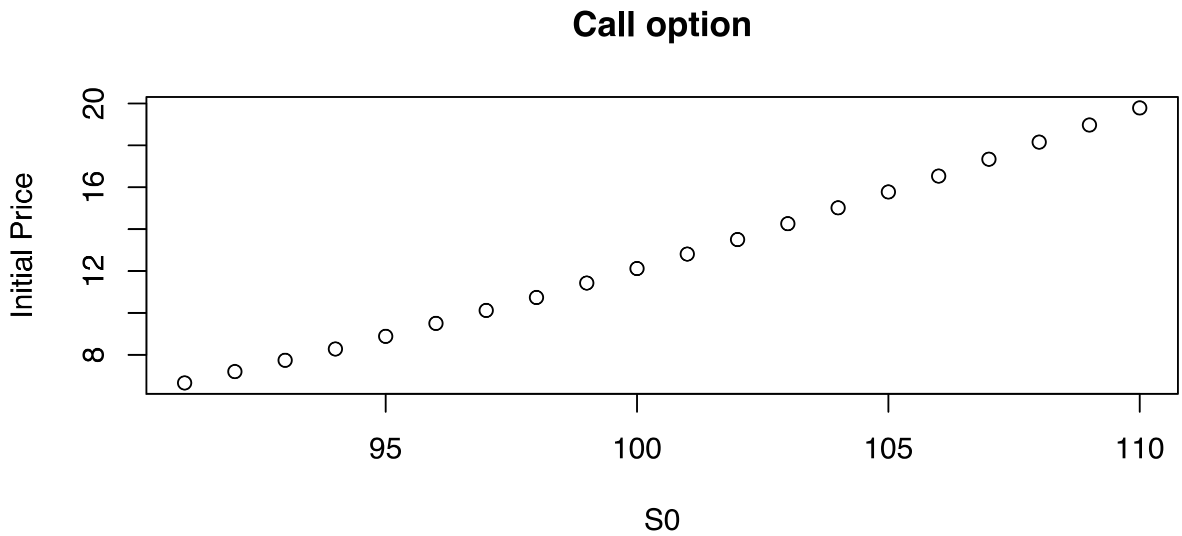
For the given information:

Initial call option price = 12.12305 .

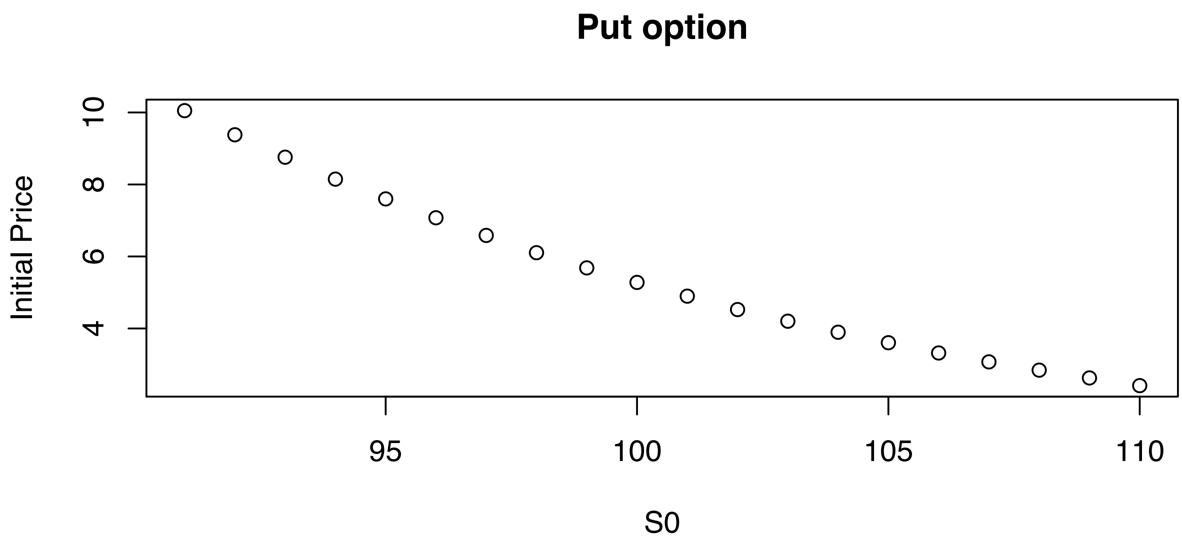
Initial put option price = 5.279837 .

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) :

A. S(0)

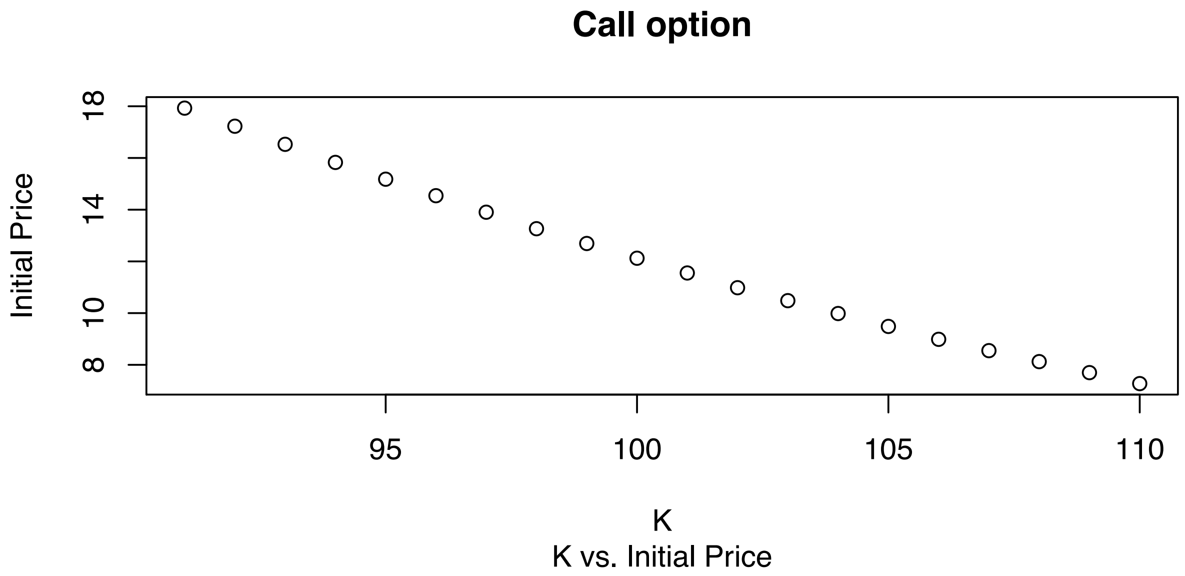


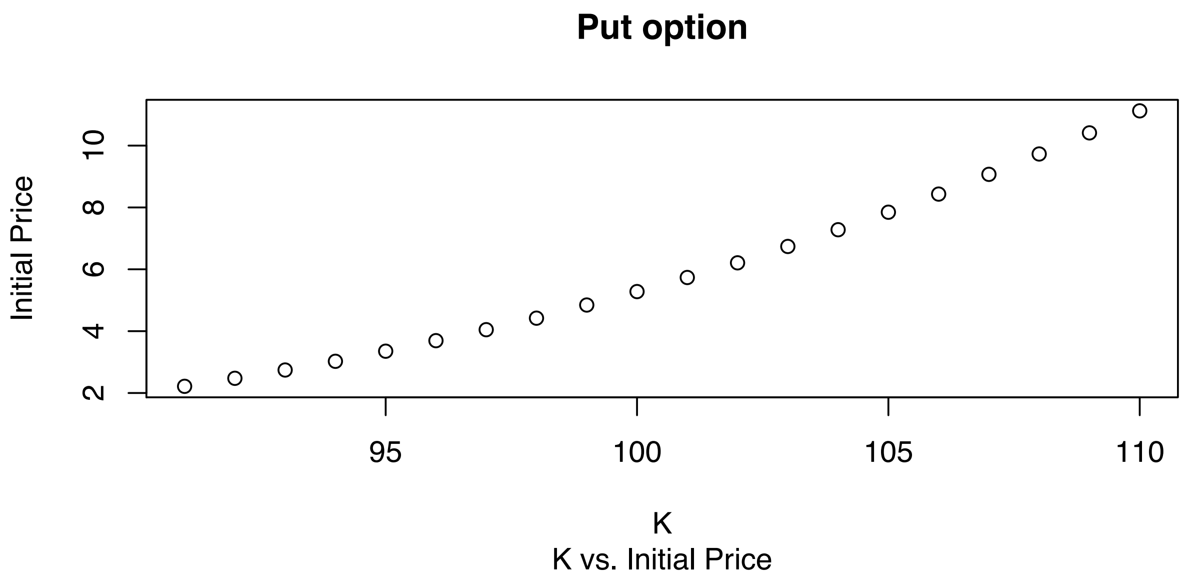
S(0) vs. Initial Price



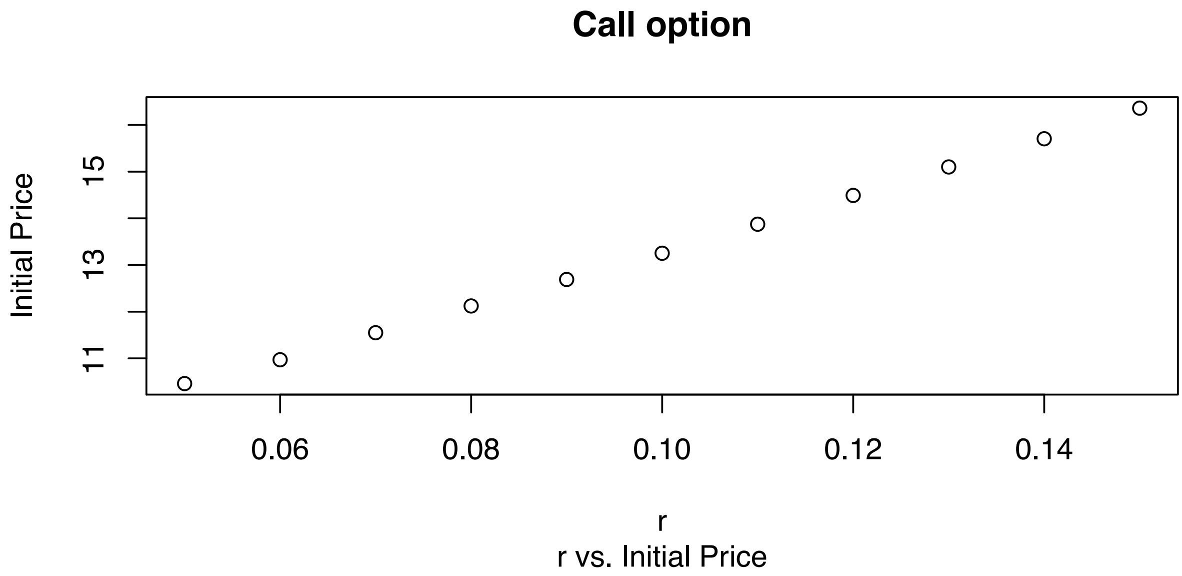
S(0) vs. Initial Price

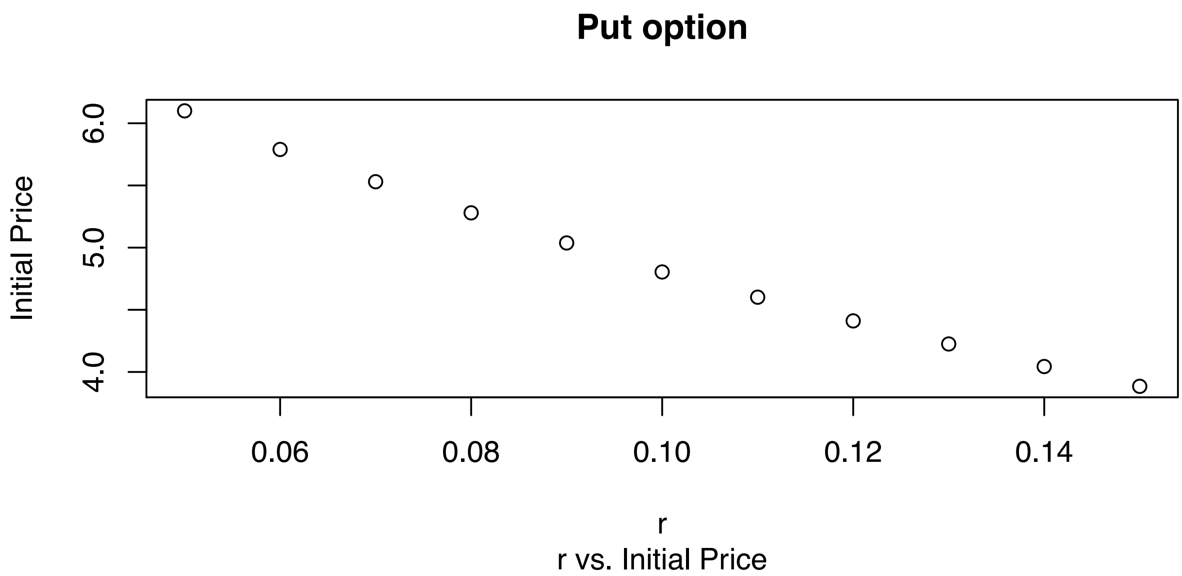
B. K


Call option
Initial Price
18
14
10
8
95
100
105
110
K
K vs. Initial Price

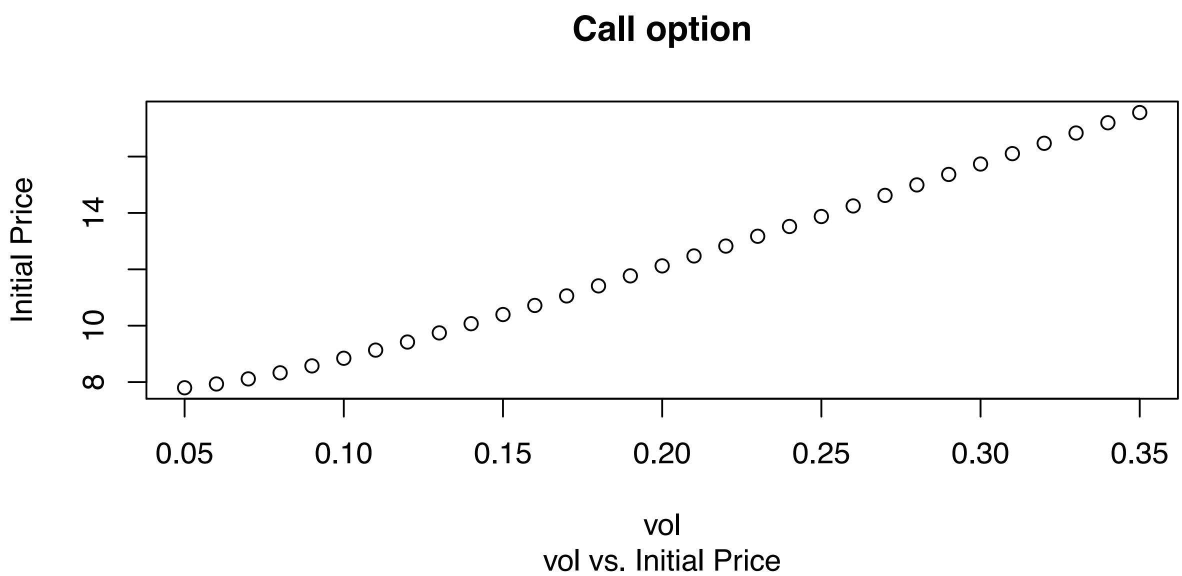

Put option
Initial Price
10
8
6
4
2
95
100
105
110
K
K vs. Initial Price

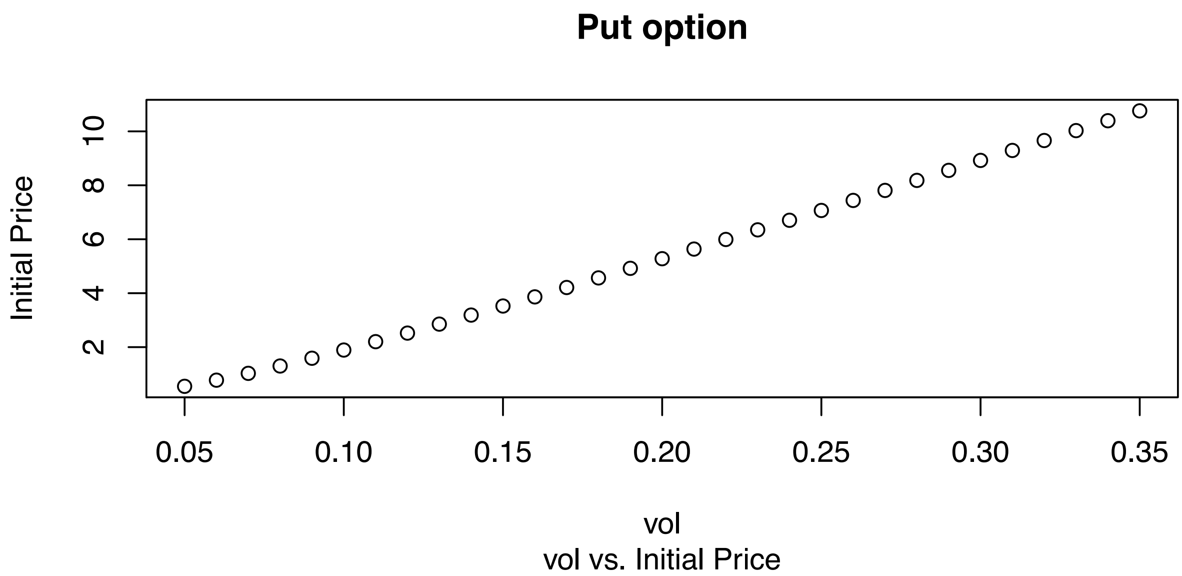
C. r


Call option
Initial Price
11
13
15
0.06
0.08
0.10
0.12
0.14
r
r vs. Initial Price


Put option
Initial Price
4.0
5.0
6.0
0.06
0.08
0.10
0.12
0.14
r
r vs. Initial Price

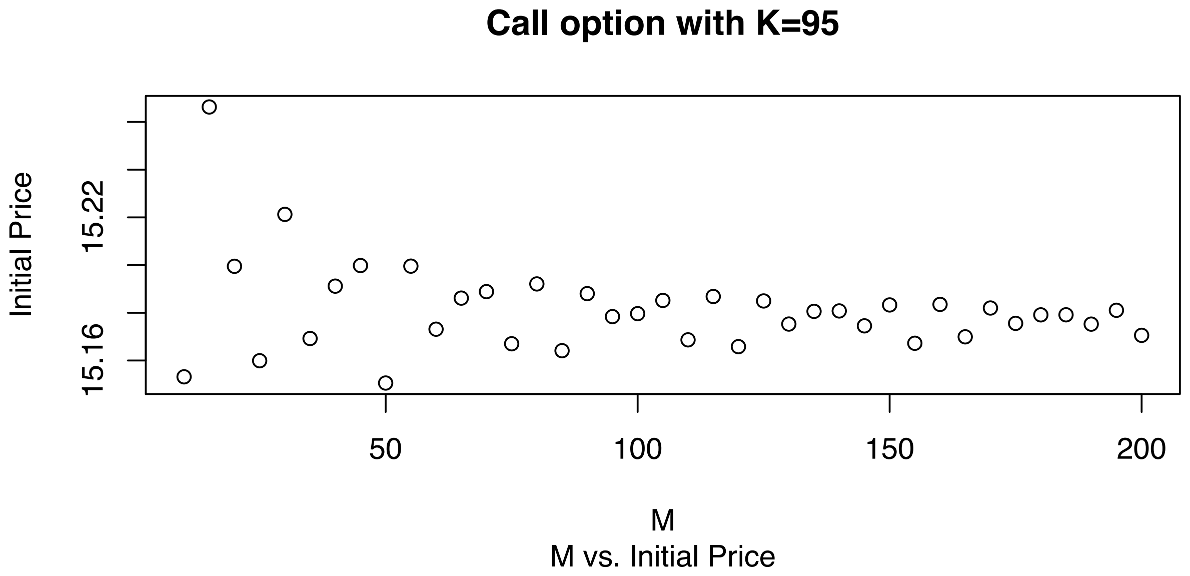
D. vol

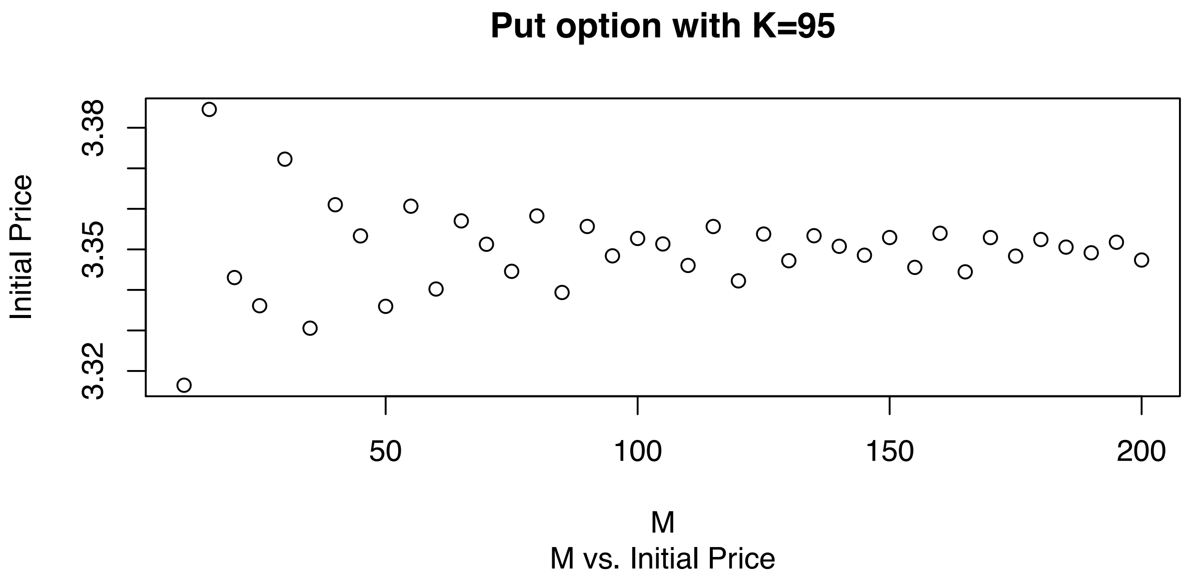

Call option
Initial Price
8
10
14
0.05
0.10
0.15
0.20
0.25
0.30
0.35
vol
vol vs. Initial Price


Put option
Initial Price
2
4
6
8
10
0.05
0.10
0.15
0.20
0.25
0.30
0.35
vol
vol vs. Initial Price

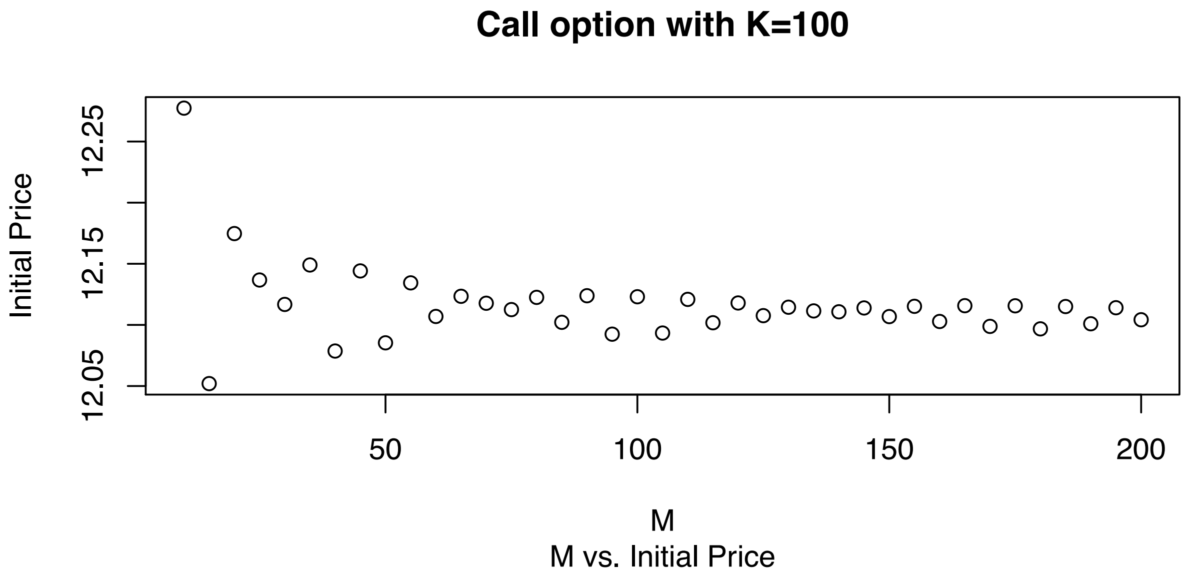
E. M (Do this for three values of K , K = 95; 100; 105).

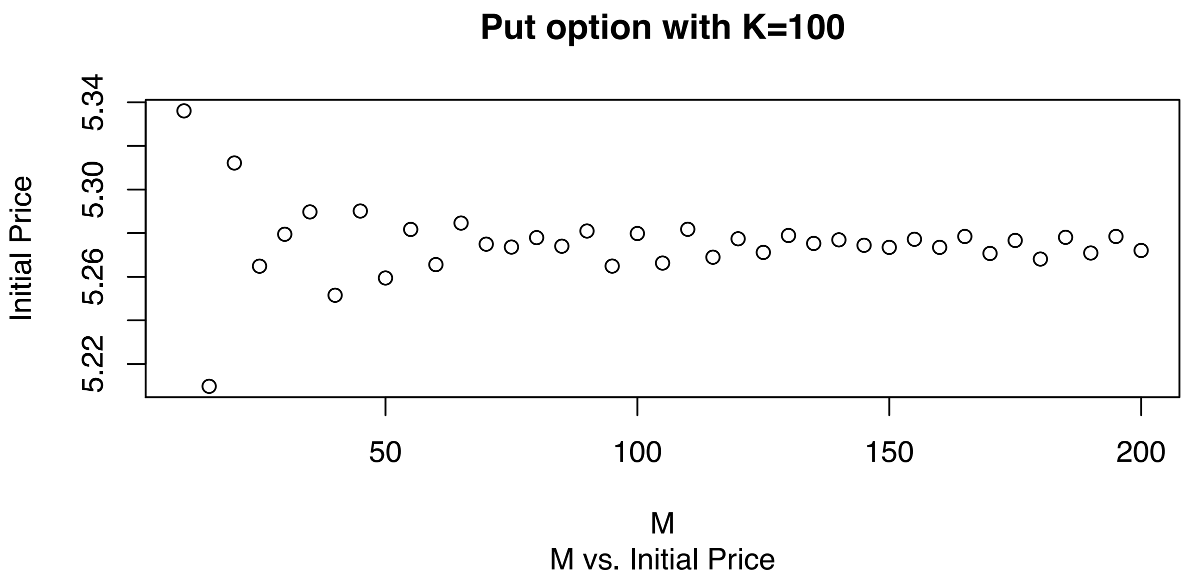
- K = 95



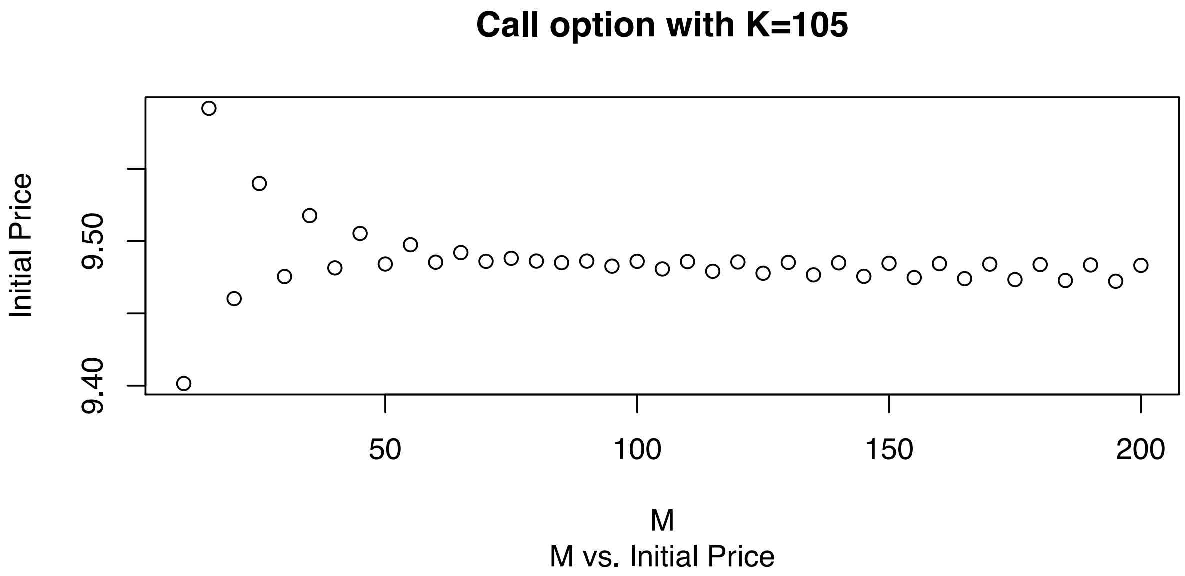


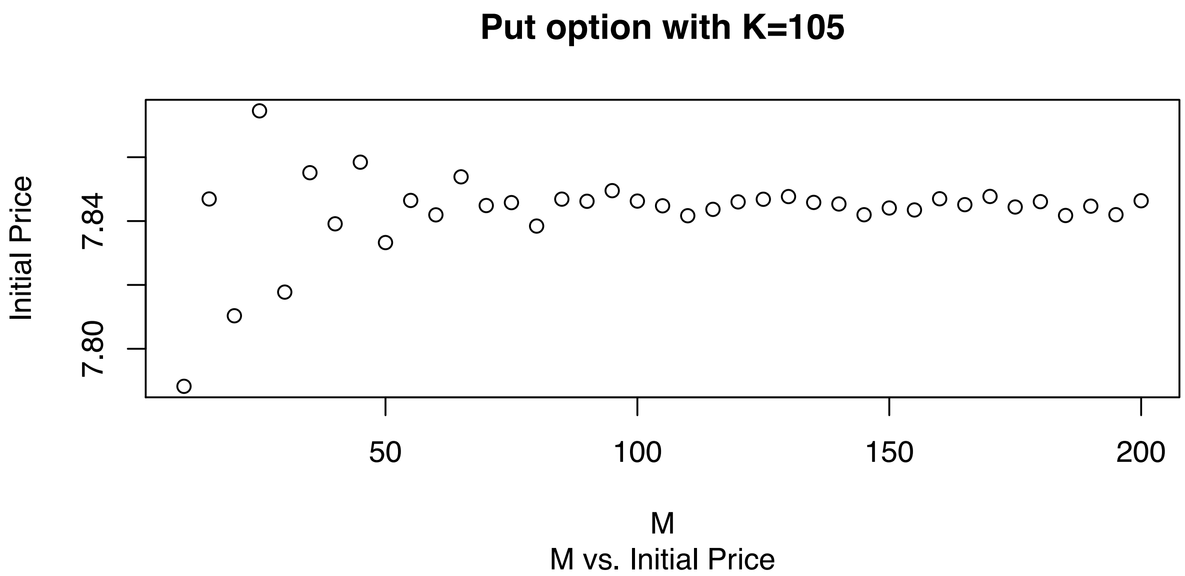
- K = 100


Call option with K=100
Initial Price
12.05
12.15
12.25
50
100
150
200
M
M vs. Initial Price


Put option with K=100
Initial Price
5.22
5.26
5.30
5.34
50
100
150
200
M
M vs. Initial Price

- K = 105


Call option with K=105
Initial Price
9.40
9.50
50
100
150
200
M
M vs. Initial Price


Put option with K=105
Initial Price
7.80
7.84
50
100
150
200
M
M vs. Initial Price

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	9.02795116(	8.54807618	7.41677100	5.50163881	0
2		9.50483986(	9.79911875	9.95527127	9.57139153	11.1814131
3			7.14791575	6.20191645	4.60047967	0
4			12.1686646(	13.7128629	15.6318518	19.4526915
5				6.20191645	4.60047967	0
6				8.32461467	8.00361378	9.34991655
7				7.14841820	6.68084299	6.37451747
8				17.5820627	21.1880893	25.3945634
9					4.60047967	0
10					8.00361378	9.34991655
11					3.84692888	0
12					13.0713809	16.2663735(
13					3.84692888	0
14					10.6809044	13.5780025
15					10.6809044	13.5780025
16					25.0512294(	29.4825971
17						0
18						9.34991655
19						0
20						16.2663735(
21						0
22						7.81841603
23						5.33038228(
24						21.2349769
25						0
26						7.81841603
27						2.90135049
28						18.8059451

Time → Possibility ↓	0.0	0.2	0.4	0.6	0.8	1.0
29						2.90135049
30						18.8059451
31						18.8059451
32						32.1053940

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