Project title:
The Application of Monte Carlo Simulation to Option Pricing

Name:
Tao Wang

Course:
EE512 Stochastic Process

Instructor:
Dr. Osonde Osoba

Date:
Dec/02/2018

Project Objective:

In this project, Monte Carlo Simulation and Brownian Motion will be used to help simulate an Option Pricing Model.

Mechanism and Tools:

The prices of underlying assets will be generated by Brownian Motion, STD Brownian Motion and Geometric Brownian Motion. Then, an estimation of prices of any kind of options (EU, America, Asian and so on) will be achieved by using the option pricing formula and LLN. The simulated option prices will be compared with the results of other two known simulation methods(TurnbullWakeman Simulation and Levy Approximation). The programming language will be R.

Notations:

Most of path dependent options can be priced by using Monte Carlo Simulation. In this project, the prices of underlying assets will be generated from Brownian Motion processes and limited sequences of simulation paths will be built up. At the exercise or expiry time, the payoff of options will be discounted(continuous compounding) back to present and take the average of all discounted prices as the estimate of the option price(by law of large numbers: LLN). The precision of the estimate is provided by the standard error $\sigma/\sqrt{N}$.

Code:

The code for simulation is listed below. For the test of simulation, some specific criteria have been set for simulations. Time to maturity is 5 years, small time interval (time increment) is 1/365,current price of underlying assets(stock) is \$50, strike price of the option is \$50, risk-free rate is 0.10, volatility of underlying assets is 0.40. The type of options can be ‘call’ or ‘put’ for the user’s choice.

Code(details):

#GBM Simulation for option price based on average price of underlying assets. r

#(Risk-neutral GBM Simulation)

#Compared with TurnbullWakeman and Levy Approximation

```
gbmm = function(T1,N,ty)

{
  set.seed(1)
  m =365*T1      # number of assets' prices in every path (days)
  life = T1      # maturity or remaining time to maturity (years)
  dt = 1/365    # simulation unit time

  s0 = 50    # underlying asset current price
  k=50       # option strike price
  sigma =0.40    # annual volatility of underlying assets
  r = 0.10      #risk-free rate



  sg = sqrt(dt)*sigma    # daily vola
  mu = r - 0.5*sigma^2    # annual drift
  mg = mu*dt      # drift per day

  inc0= log(s0)
  x0 =rep(inc0,N)          # one row with N values
  incs = rnorm(m*N,mean=mg,sd=sg)    #one column of m*N values
  mat = matrix(incs,m,N)             # m by N matrix

  d2 = data.frame(mat)
```

```
x = rbind(x0,d2)
lofs = cumsum(x)
s = exp(lofs)          # simulate the underlying assets' prices

sT = apply(s,2,mean)   # average price for each path,by using LLN to apraximate the future underlying price

if(ty=='c')
{
   payoff = pmax(0,sT-k)}
else if(ty =='p'){
   payoff = pmax(0,k-sT)
}
else{print('NA') }

dpayoff = payoff*exp(-r*life)


oprice =mean(dpayoff)        #simulated option price by mean
serror = sd(dpayoff)/sqrt(N)
lb = oprice - qnorm(0.95)*serror
ub = oprice + qnorm(0.95)*serror


cat(' Sim price of option', oprice, '\n','std.error', serror)
cat('\n CI for the option price',lb,ub,'\n')


matplot(s,xlab = 'days',ylab = 'B(t)',type='l',lty=1:N,las=1)        #simulated prices of assets
```

```
   # mean function      (mean function graph)
   xaxis = 1:(m+1)
   mg =r*dt
   yaxis =rep(mg,m)
   yaxis = c(log(s0),yaxis)
   yaxis = cumsum(yaxis)
   yaxis =exp(yaxis)
   lines(xaxis,yaxis,lwd =4,col='red')

   #average curve

   yaxis = rowMeans(s)
   lines(xaxis,yaxis,lwd=4,col='blue')
   grid()

}

#Compared tests
S = 50        # Stock price at t=0
SA = 50       # Observed average price (currently)
X =50          # strike
Time = 5      # lifetime
time = 5      #remaining lifetime
tau = 0       #time after signature
r = 0.10
q = 0           # divident yield
b = r-q
sigma = 0.40
```

TurnbullWakemanAsianApproxOption('c',S,SA,X,Time,time,tau,r,b,sigma)@price

LevyAsianApproxOption('c',S,SA,X,Time,time,r,b,sigma)@price

TurnbullWakemanAsianApproxOption('p',S,SA,X,Time,time,tau,r,b,sigma)@price

LevyAsianApproxOption('p',S,SA,X,Time,time,r,b,sigma)@price

Test:

gbmm (Geometric Brownian Motion)

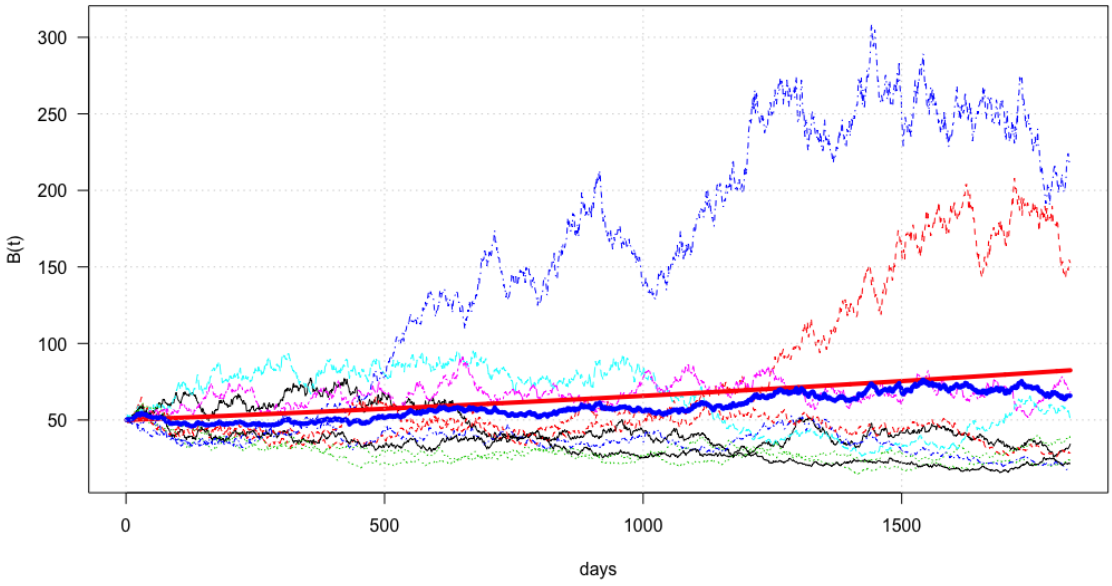
Call option pricing

> gbmm(5,10,'c')

Sim price of option 10.09972

std.error 6.340147

CI for the option price -0.3288896 20.52834

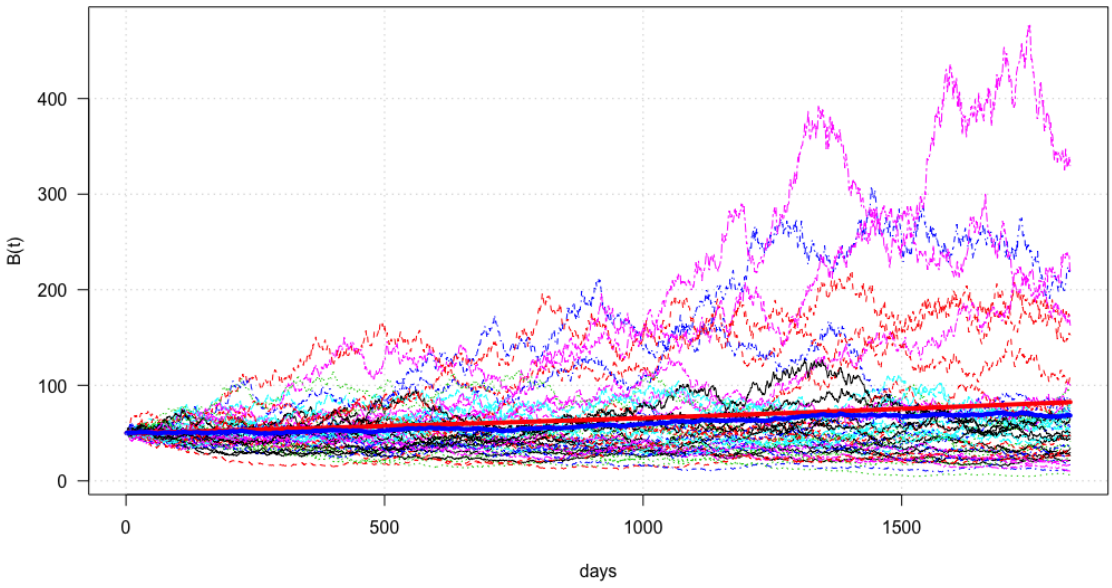


> gbmm(5,50,'c')

Sim price of option 9.763577 #simulated option price

std.error 2.689046 #standard errors

CI for the option price 5.340489 14.18666 #Confidential interval

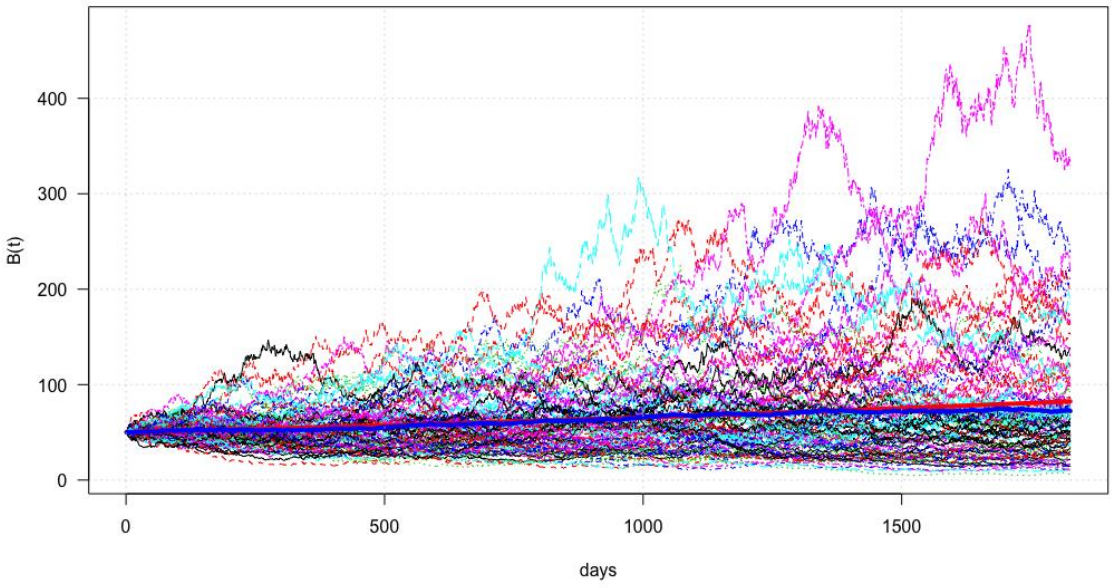


> gbmm(5,100,'c')

Sim price of option 11.37342

std.error 1.7782

CI for the option price 8.448539 14.2983



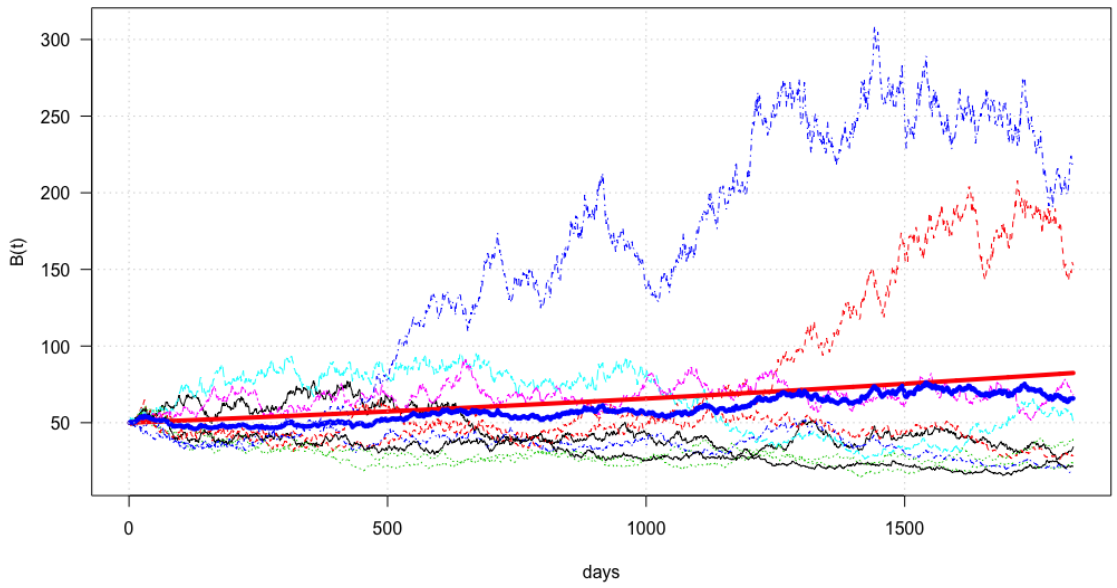
Put option pricing

> gbmm(5,10,'p')

Sim price of option 4.914547

std.error 1.79992

CI for the option price 1.953942 7.875153

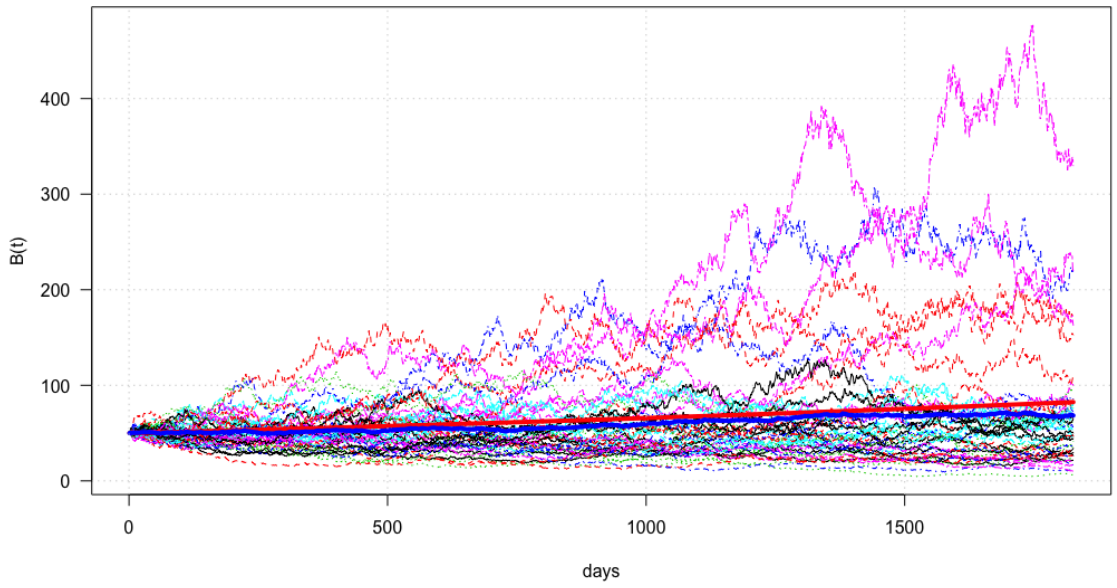


> gbmm(5,50,'p')

Sim price of option 4.100675

std.error 0.799191

CI for the option price 2.786123 5.415227

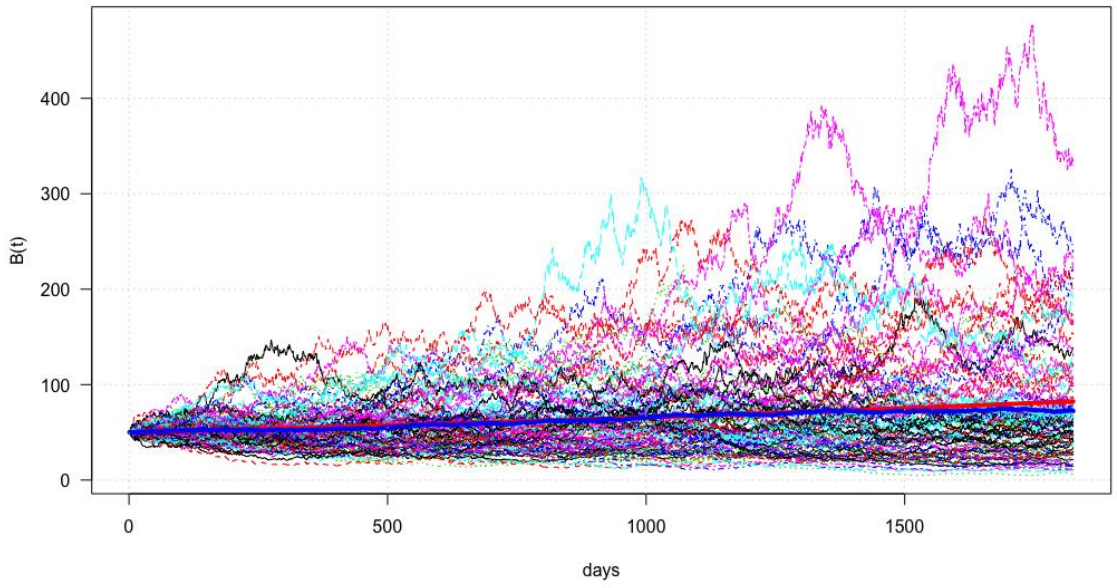


> gbmm(5,100,'p')

Sim price of option 3.47834

std.error 0.5269305

CI for the option price 2.611616 4.345063



Compared with GBM method

```
> TurnbullWakemanAsianApproxOption('c',S,SA,X,Time,time,tau,r,b,sigma)@price
[1] 13.05696      #simulated option price through TurnbullWakeman simulation
> LevyAsianApproxOption('c',S,SA,X,Time,time,r,b,sigma)@price
[1] 13.05696        #simulated option price through Levy Simulation
> TurnbullWakemanAsianApproxOption('p',S,SA,X,Time,time,tau,r,b,sigma)@price
[1] 4.036559        #simulated option price through TurnbullWakeman simulation
> LevyAsianApproxOption('p',S,SA,X,Time,time,r,b,sigma)@price
[1] 4.036559        #simulated option price through TurnbullWakeman simulation
>
```

Performance Analysis:

In the test, as we can see, the graph shows the simulated asset's prices generated from Geometric Brownian Motion by Monte Carlo Simulation (X-axis as time and Y-axis as asset price). The thick blue line is the average curve of the assets and the red line is the mean function curve. Those simulated asset's prices are moving around these two lines, which indicates the simulation has a good shape. Besides, from the test results, the simulated option price is almost the same, compared to the other two simulation methods (TurnbullWakeman simulation and Levy simulation). Even if there are some differences when the simulating steps are not sufficient, the standard error also can cover the difference. The simulation process and the final pricing result also follow the rules of statistics. The more sample data is involved, the estimated result is more accurate. (The B(t) of the y-axis is the simulated prices of the underlying asset.)

sbmm

#sbmm1. r

```
sbmm = function(T1,N,ty)

{
  set.seed(1)
  m =365*T1      # number of assets' prices every path (days)
  life = T1      # maturity or remaining time to maturity (years)
  dt = 1/365    # simulation unit time

  s0 = 50    # underlying asset current price
  k=50        # option strike price
  sigma =0.40    # annual volatility of underlying assets
  r = 0.10      #risk-free rate

  sg = sqrt(dt)*sigma    # daily vola
  mu = 0.0    # annual drift(since it's std BM)
  mg = mu*dt      # drift per day

  inc0= s0
  x0 =rep(inc0,N)          # one row with N values
  incs = rnorm(m*N,mean=mg,sd=sg)    #one column of m*N values
  mat = matrix(incs,m,N)              # m by N matrix

  d2 = data.frame(mat)
  x = rbind(x0,d2)
  lofs = cumsum(x)
  s = lofs
```

sT = apply(s,2,mean) # average price for each path,by using LLN to apraximate the future underlying price

```
if(ty=='c')
{
   payoff = pmax(0,sT-k)}
else if(ty =='p'){
   payoff = pmax(0,k-sT)
}
else{print('NA') }

dpayoff = payoff*exp(-r*life)


oprice =mean(dpayoff)        #simulated option price by mean
serror = sd(dpayoff)/sqrt(N)
lb = oprice - qnorm(0.95)*serror
ub = oprice + qnorm(0.95)*serror


cat('Sim price of option', oprice, 'std.error', serror)
cat('\n CI for the option price',lb,ub,'\n')




matplot(s,xlab = 'days',ylab = 'B(t)',type='l',lty=1:N,las=1)
# mean function
abline(inc0,mg,lty=2,col='red')
```

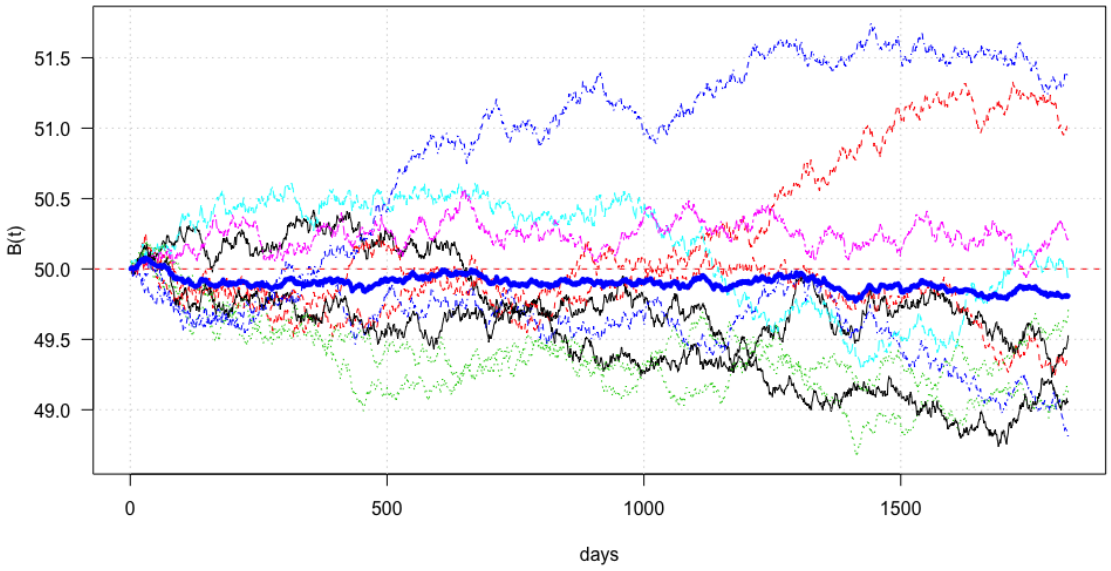
```
    grid()


    #average curve
    xaxis = 1:(m+1)
    yaxis = rowMeans(s)
    lines(xaxis,yaxis,lwd=4,col='blue')


}
```

Test results:

sbmm(Standard Brownian Motion)

Call option pricing

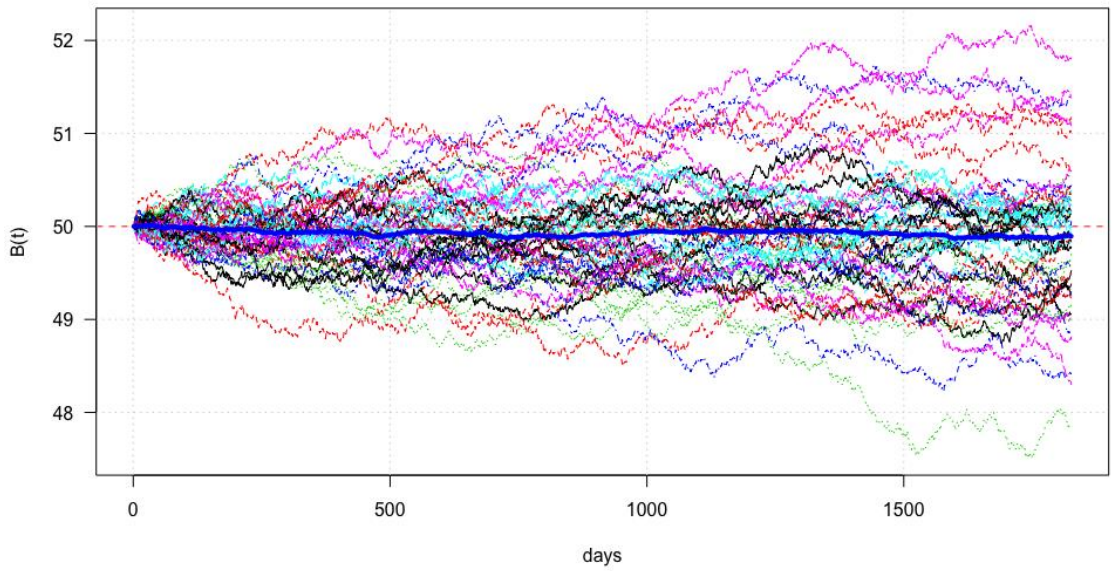
```
> sbmm(5,10,'c')
Sim price of option 0.09021621 std.error 0.05323069
 CI for the option price 0.002659519 0.1777729
```



> sbmm(5,50,'c')

Sim price of option 0.09335949 std.error 0.02424172

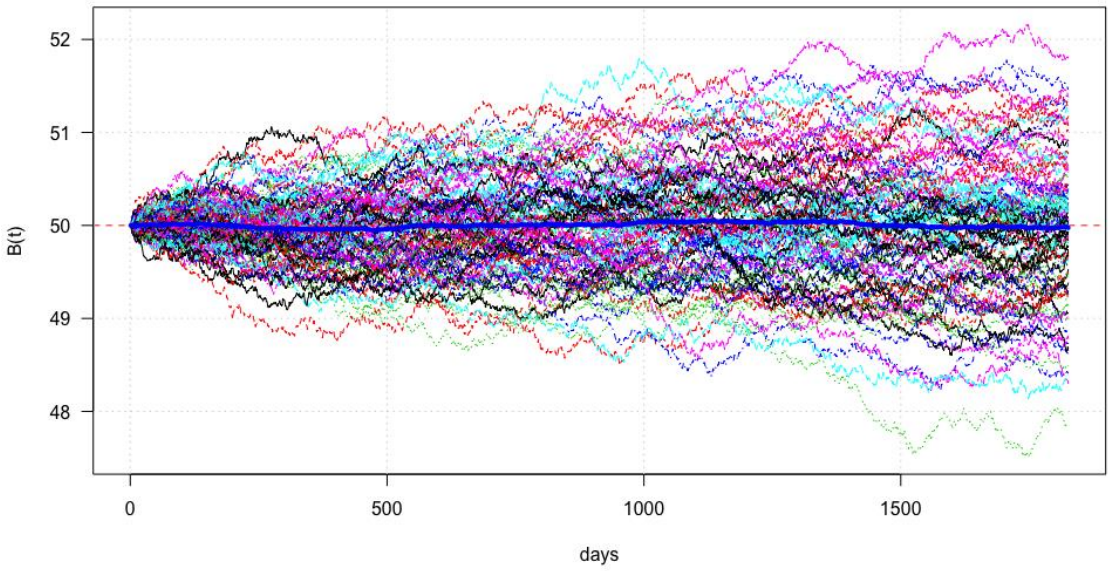
CI for the option price 0.05348541 0.1332336



> sbmm(5,100,'c')

Sim price of option 0.1166929 std.error 0.01715126

CI for the option price 0.08848157 0.1449042

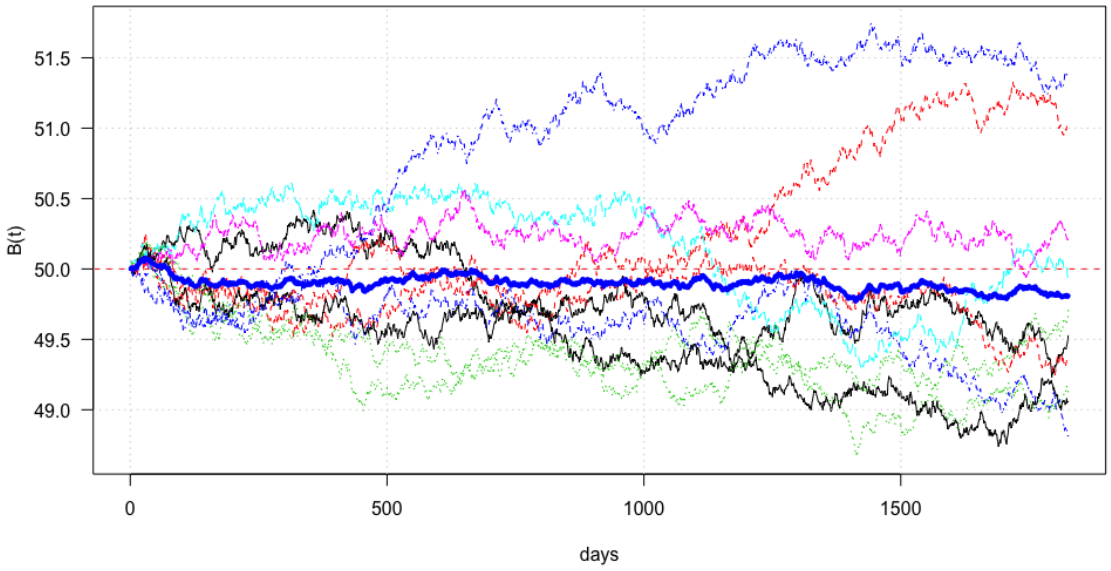


Put option pricing

> sbmm(5,10,'p')

Sim price of option 0.1514814 std.error 0.05279292

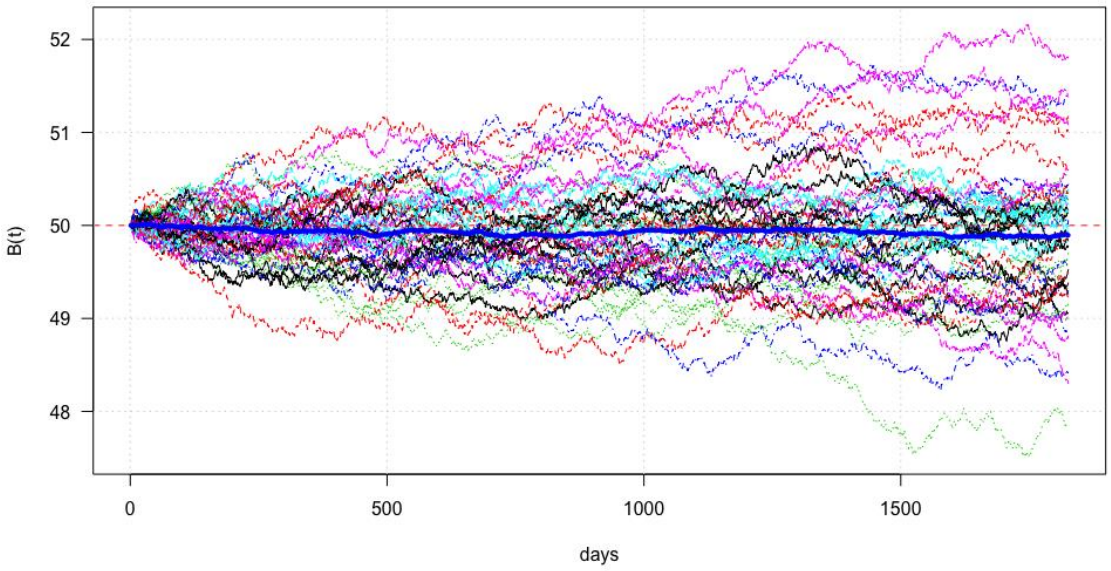
CI for the option price 0.06464472 0.238318



> sbmm(5,50,'p')

Sim price of option 0.1360768 std.error 0.02587999

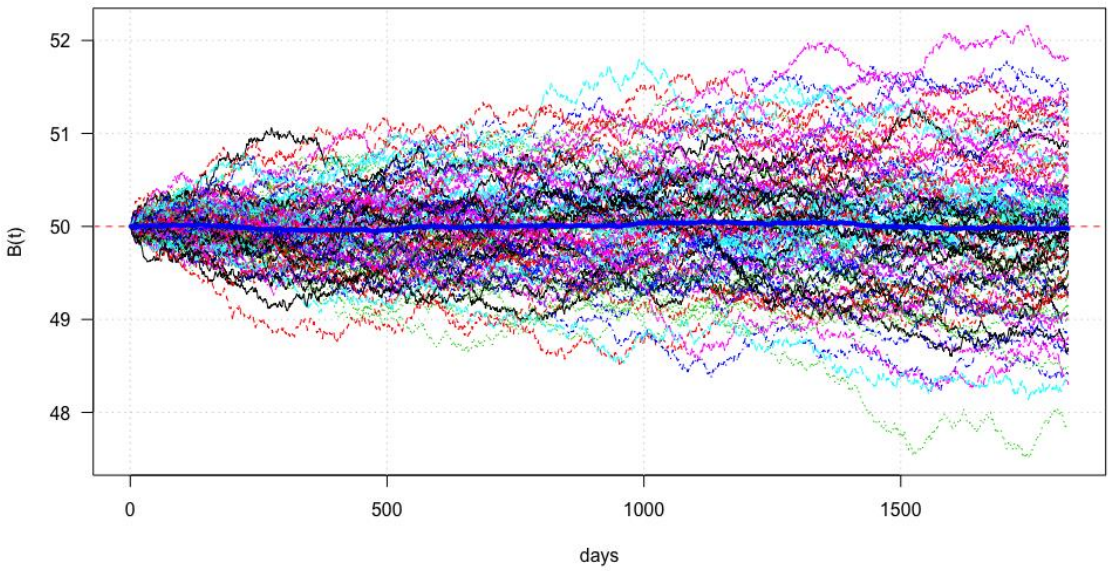
CI for the option price 0.09350796 0.1786456



> sbmm(5,100,'p')

Sim price of option 0.116015 std.error 0.0170369

CI for the option price 0.08799177 0.1440382



bmm

#bmm1(Risk-Neutral). r

```
bmm = function(T1,N,ty)

{
  set.seed(1)
  m =365*T1
  life = T1
  dt = 1/365
  s0 = 50
  k=50
  r = 0.10
  sigma = 0.40    # annual vola

  mu = r    # annual drift
  mg = mu*dt      # drift per day
  sigma = 0.80    # annual vola
  sg = sqrt(dt)*sigma    # daily vola

  inc0= s0
  x0 =rep(inc0,N)          # one row with N values
  incs = rnorm(m*N,mean=mg,sd=sg)    #one column of m*N values
  mat = matrix(incs,m,N)              # m by N matrix

  d2 = data.frame(mat)
  x = rbind(x0,d2)
  lofs = cumsum(x)
  s = lofs
```

sT = apply(s,2,mean) # average price for each path,by using LLN to apraximate the future underlying price

```
if(ty=='c')
{
payoff = pmax(0,sT-k)}
else if(ty =='p'){
payoff = pmax(0,k-sT)
}
else{print('NA') }

dpayoff = payoff*exp(-r*life)


oprice =mean(dpayoff)        #simulated option price by mean
serror = sd(dpayoff)/sqrt(N)
lb = oprice - qnorm(0.95)*serror
ub = oprice + qnorm(0.95)*serror


cat('Sim price of option', oprice, 'std.error', serror)
cat('\n CI for the option price',lb,ub,'\n')




matplot(s,xlab = 'days',ylab = 'B(t)',type='l',lty=1:N,las=1)
# mean function
abline(inc0,mg,lty=2,col='red')
grid()
```

```
    #average curve
    xaxis = 1:(m+1)
    yaxis = rowMeans(s)
    lines(xaxis,yaxis,lwd=4,col='blue')


}
```

Test:

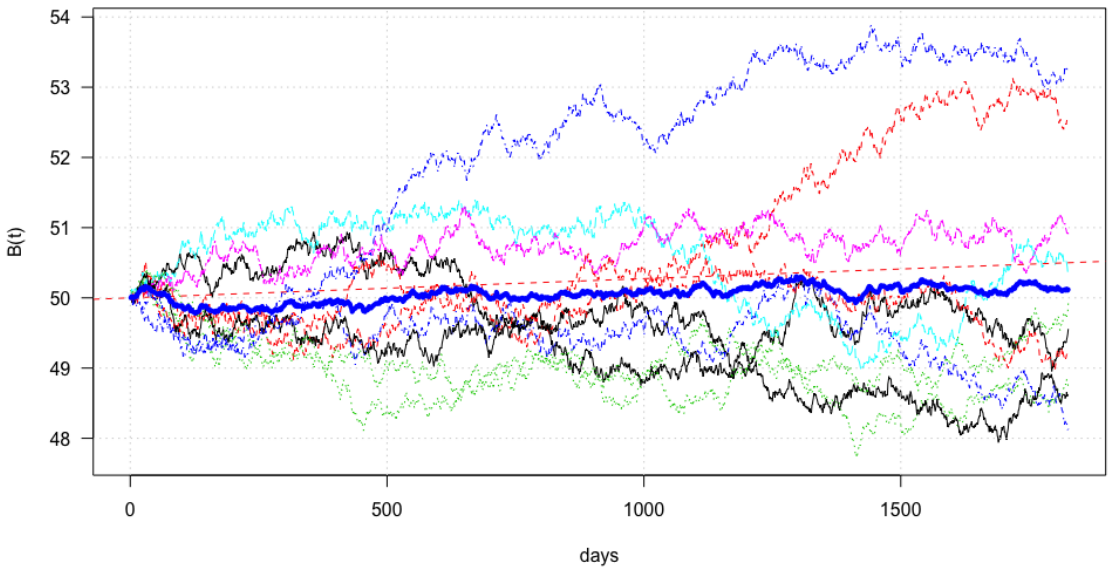
bmm (General Brownian Motion)

Call option pricing

> bmm(5,10,'c')

Sim price of option 0.2410855 std.error 0.1248803

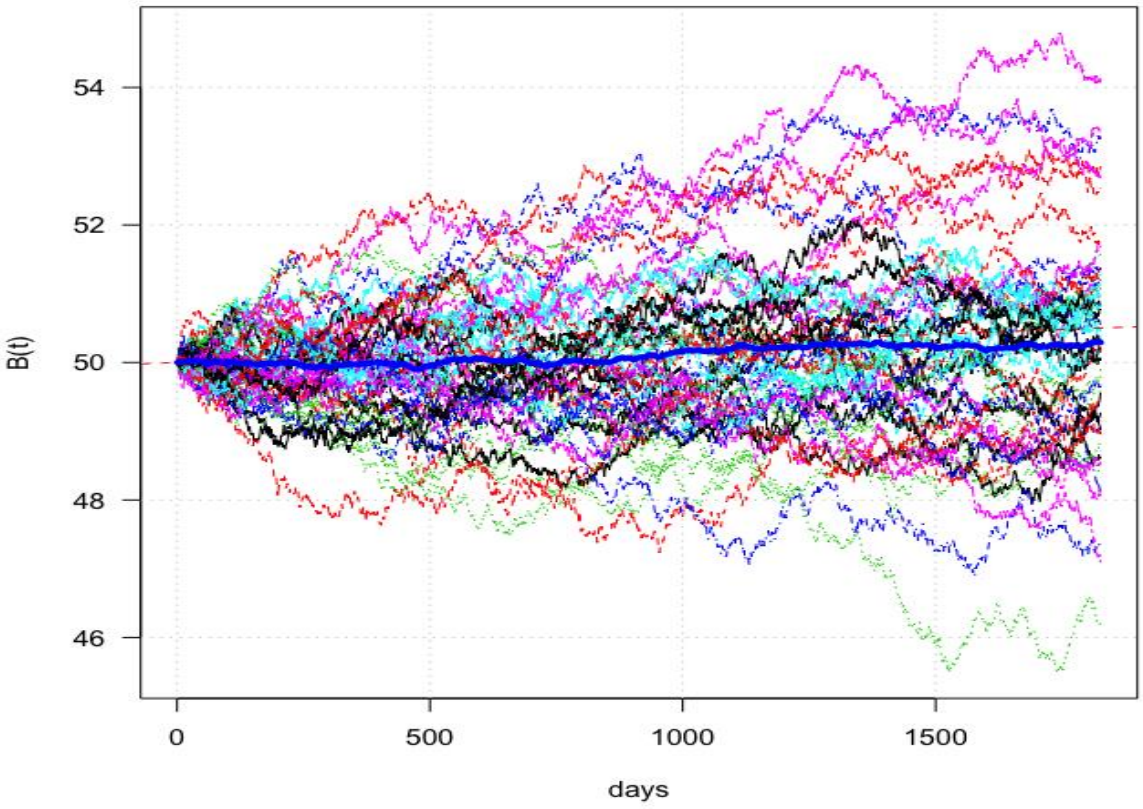
CI for the option price 0.03567571 0.4464953



> bmm(5,50,'c')

Sim price of option 0.2638801 std.error 0.05500371

CI for the option price 0.1734071 0.3543532

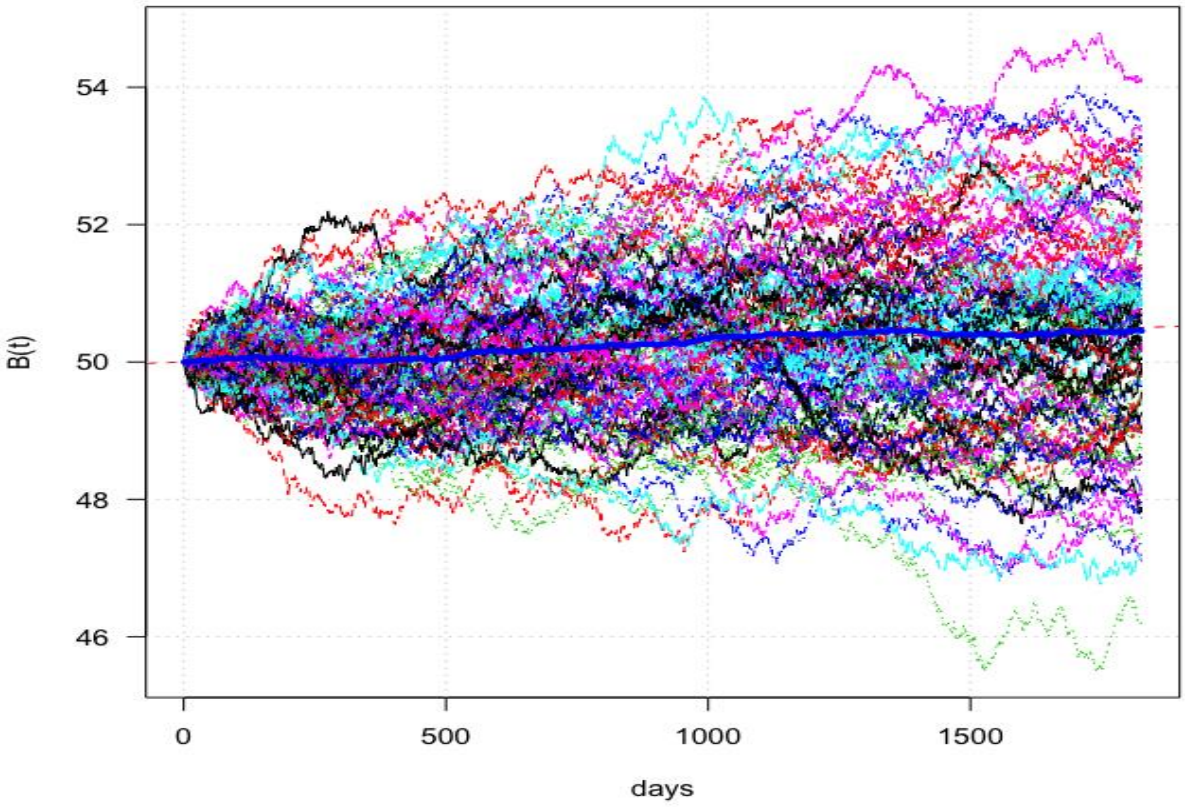


> bmm(5,100,'c')

Sim price of option 0.3196146 std.error 0.03921261

CI for the option price 0.2551156 0.3841136

>

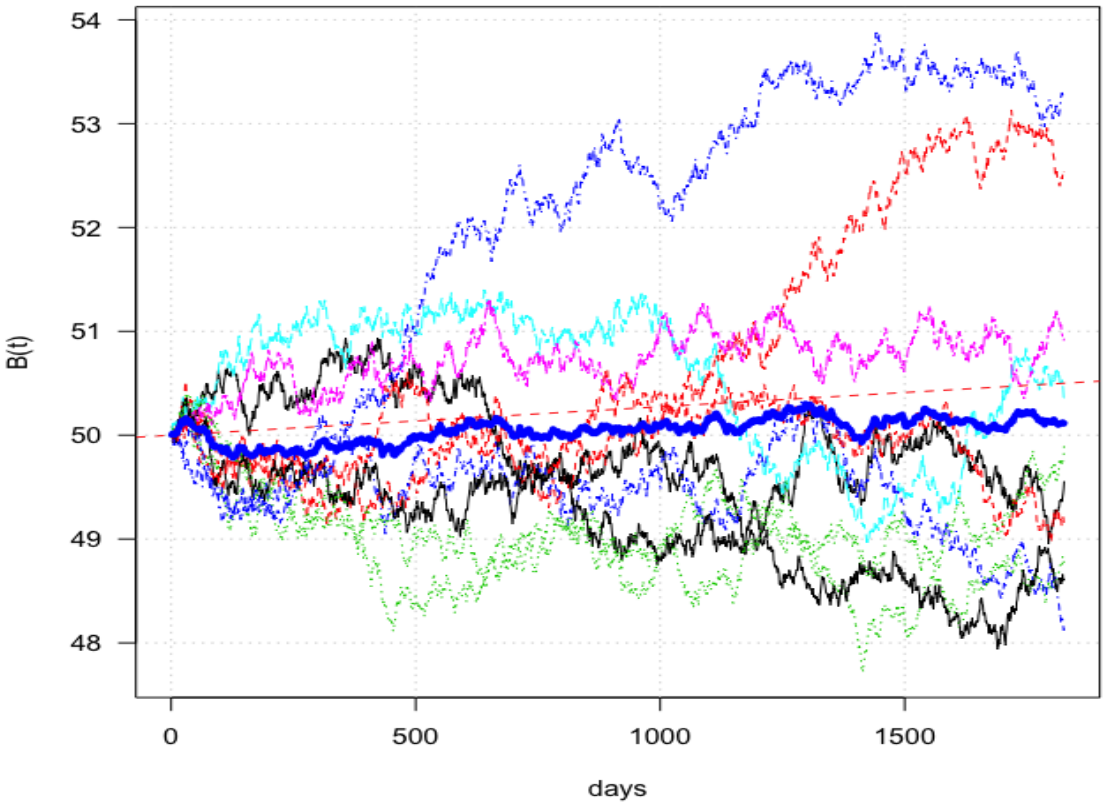


Put option pricing

> bmm(5,10,'p')

Sim price of option 0.2119831 std.error 0.08762439

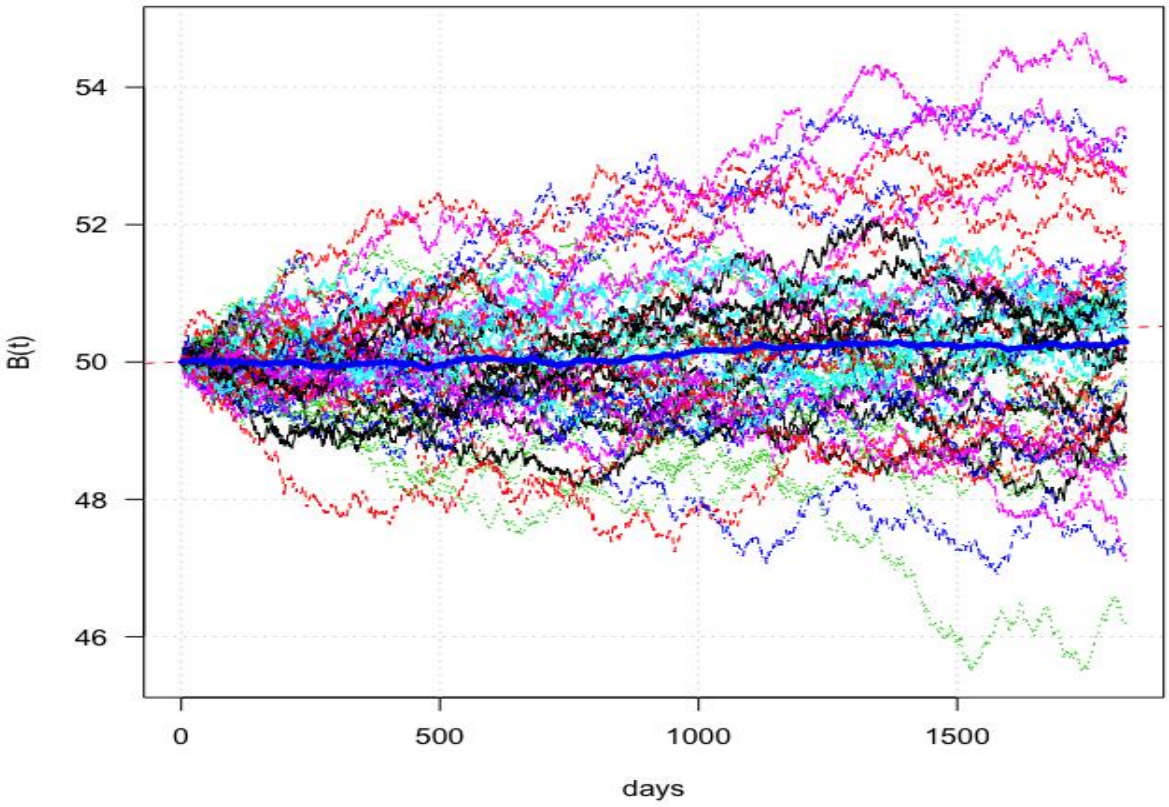
CI for the option price 0.06785382 0.3561124



> bmm(5,50,'p')

Sim price of option 0.197682 std.error 0.04415095

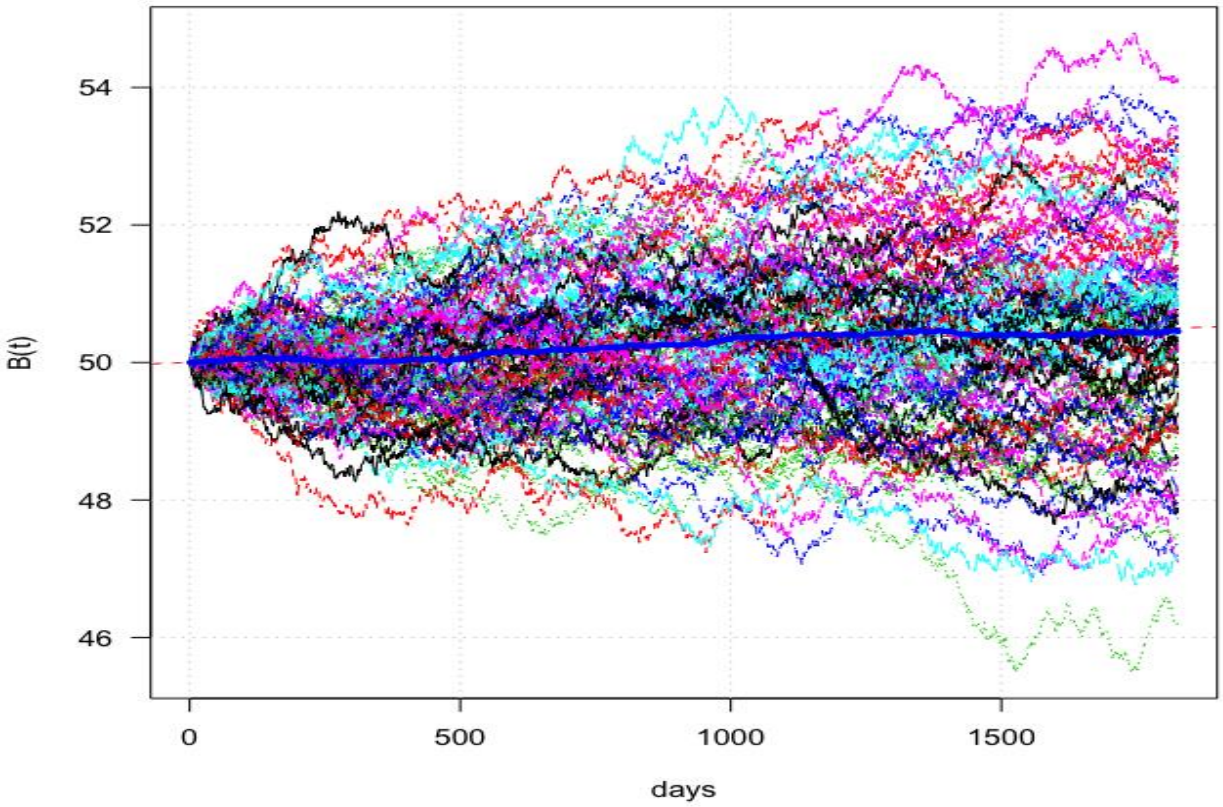
CI for the option price 0.1250601 0.2703038



> bmm(5,100,'p')

Sim price of option 0.1666261 std.error 0.02860361

CI for the option price 0.1195773 0.2136748



Performance Analysis

As we can see, the simulation prices of underlying assets are in a good shape but the option prices simulated are having a conspicuous difference from the results of Geometric Brownian motion under the same financial conditions. Based on this performance, the option prices simulated by the general Brownian motion and standard Brownian motion are pretty small, which might have undervalued the tested option.

Summary

Based on the simulation and analysis above, the simulation performance of Geometric Brownian Motion by Monte Carlo simulation has delivered the closest result, compared to the Turnbull Wakeman Simulation and Levy Approximation. The simulations of Standard Brownian Motion and General Brownian Motion are in the good shape but the simulated prices are having a conspicuous disparity from the simulated results of the Turnbull Wakeman Simulation and Levy Approximation. Thus, for the path dependent options, the Monte Carlo Simulation through Geometric Brownian Motion might be one of the best ways to price options.

Reference:

1. Hull, J C, 2018, Options, Futures and Other Derivatives, 10th edition, Pearson Prentice Hall, New Jersey.
2. Glasserman, P, 2003, Monte Carlo Methods in Financial Engineering, Cambridge University Press, Cambridge.
3. Bennett M., Hugen D., Financial Analytics with R, Cambridge, 2016