Pricing Financial Derivatives Project 1

Angus McKay, Laura Roman, Euan Dowers, Veronika Kyuchukova

February 2, 2017

Exercise 1: Pricing and hedging under the Binomial model

Consider a binomial model with $r = 0.04, T = 4, d = 0.98, S_0 = 20$ and a European call with maturity $T = 4$ and strike price $K = 20$.

(i) **Check if the no-arbitrage condition is satisfied in this market and compute the risk neutral probability**

In the lecture notes we are given that in the binomial model the no-arbitrage condition is satisfied if and only if

$$d < 1 + r < u$$

so as $0.98 < 1.04 < 1.06$, the no-arbitrage condition is satisfied.

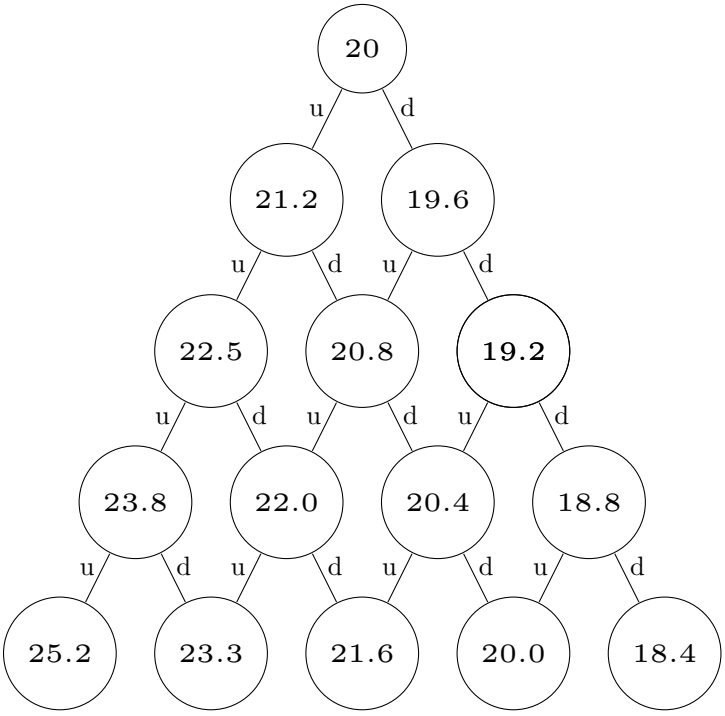
Furthemore, we are given that the risk-neutral probability is defined by

$$q = \frac{1 + r - d}{u - d} = 0.75$$

So the risk-neutral probability is 0.75.

(ii) **Construct the binomial tree for different values of S_n**

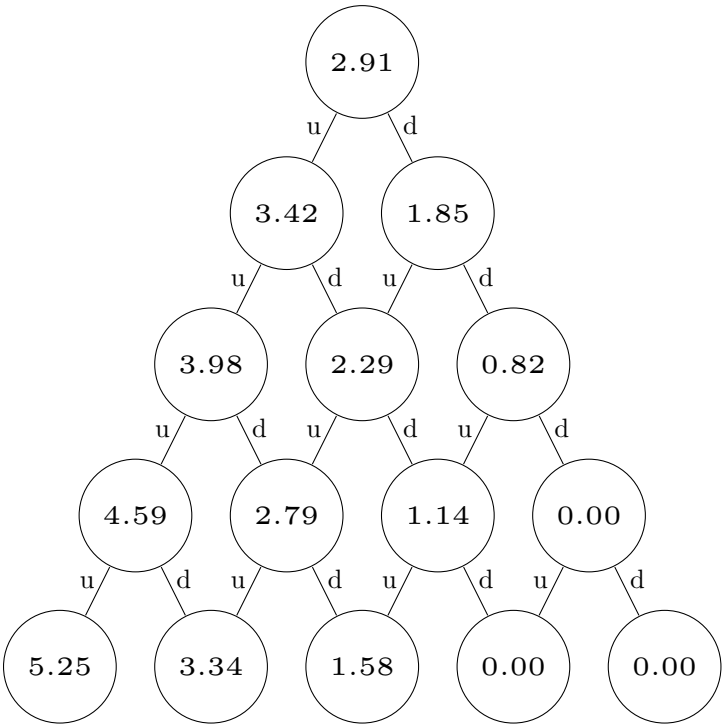
The following tree represents the binomial tree of S_n for $n \in \{0, 1, 2, 3, 4\}$. Note that prices are rounded to 1 decimal place for presentation, but rounding is done after evaluation to avoid numerical errors.



(iii) **Determine the different values of the hedging portfolio** The values of the hedging portfolio at each stage are calculated using the backward recursion formula:

$$V_n = (1+r)^{n-T}\widetilde{E}[h(S_T)|F_n]$$

The values of the hedging portfolio are shown in the binomial tree below:



(iv) **Compute the premium fee using the formula of the expectation of the payoff** The premium fee is calculated using the formula for the expectation of the payoff:

$$V_0 = (1+r)^{-T}\widetilde{E}[h(S_T)]$$

Which gives:

$$V_0 = 1.04^{-4} \times (0.75^4 \times 5.25 + 4 \times 0.75^3 \times 0.25 \times 3.34 + 6 \times 0.75^2 \times 0.25^2 \times 1.58 + 0 + 0) = 2.91$$

This is the same value as obtained for V_0 using backwards recursion.

(v) **Compute the hedging strategy assuming the values of S_n taken are:** $S_0 = 20, S_1 = 21.2, S_2 = 20.776, S_3 = 22.022, S_4 = 21.582$

n	a_n	B_n	b_n	S_n	V_n
0	2.9	1	0.00	20	2.9
0+	-16.75	1	0.98	20	2.9
1	-16.75	1.04	0.98	21.2	3.42
1+	-17.75	1.04	1.00	21.2	3.42
2	-17.75	1.0816	1.00	20.78	2.28
2+	-18.35	1.0816	0.99	20.78	2.28
3	-18.35	1.125	0.99	22.02	2.79
3+	-19.23	1.125	1.00	22.02	2.79
4	-19.23	1.1698	1.00	21.58	**1.58**

Exercise 2: Simulation of a Brownian motion

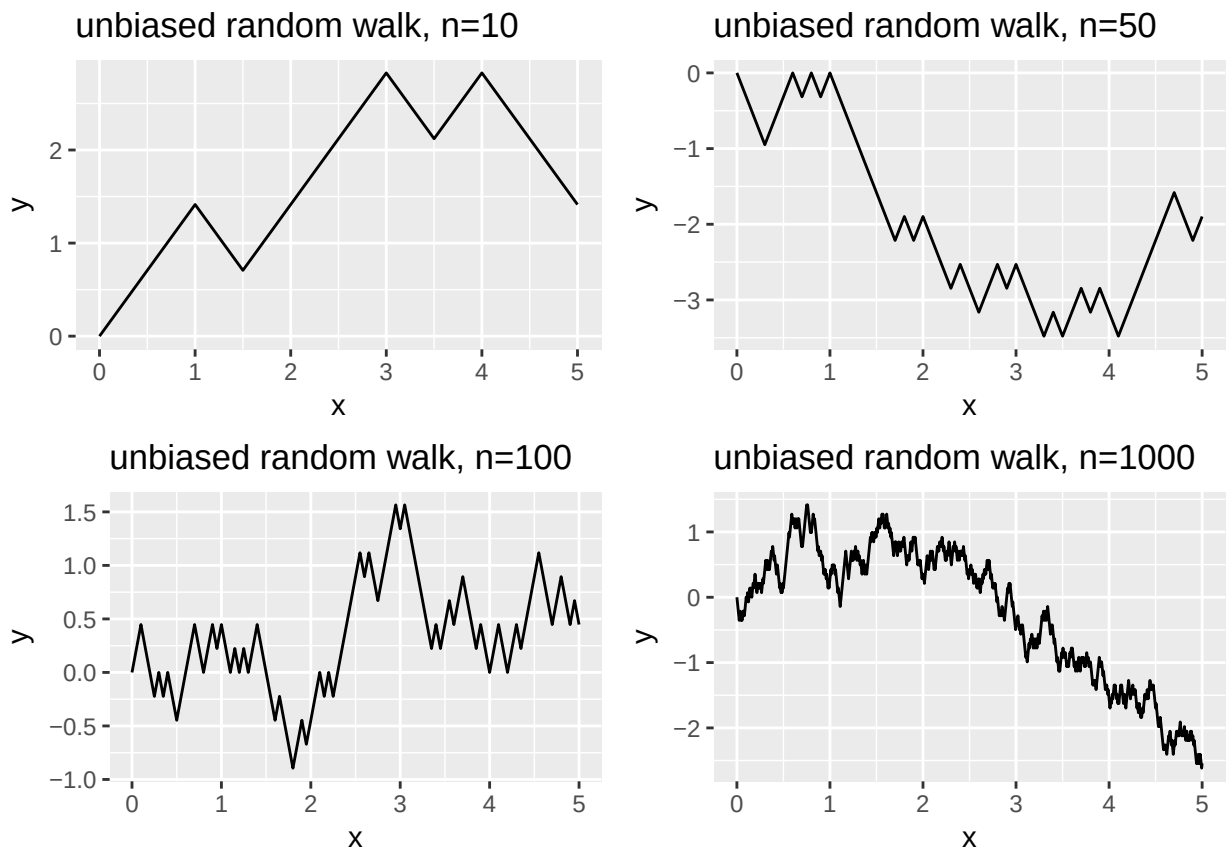
(i) **Simulate one trajectory of the continous time process $S_n(t), t \in [0,5]$ for $n = 10, 50, 100, 1000$ and comment what you observe.**

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```
#_____________________________________________________
#Part 1

Sn <- function(n,T = 5){
  # Create function that produces random walk with step size sqrt(T/n)
  stepsize <- sqrt(T/n)
  Z <- rbinom(n+1,1,0.5)
  Z[Z==1] <- stepsize
  Z[Z==0] <- -stepsize
  X <- rep(0,n+1)
  for ( i in 1:n+1) {
    X[i] = X[i-1] + Z[i] }
  return(X)
}

# Now plot the process Sn(t) for n = 10,50,100,1000
y1 <- Sn(10)
x1 <- seq(0,5,length.out = length(y1))
dd1 <- as.data.frame(x1)
dd1 <- cbind(dd1,y1)
q1 <- ggplot(data=dd1, aes(x=x1)) +
  geom_line(aes(y=y1)) +
  ggtitle("unbiased random walk, n=10")+
  ylab("y") +xlab ("x")


y2 <- Sn(50)
x2 <- seq(0,5,length.out = length(y2))
dd2 <- as.data.frame(x2)
dd2 <- cbind(dd2,y2)
q2 <- ggplot(data=dd2, aes(x=x2)) +
  geom_line(aes(y=y2)) +
  ggtitle("unbiased random walk, n=50")+
  ylab("y") +xlab ("x")


y3 <- Sn(100)
x3 <- seq(0,5,length.out = length(y3))
dd3 <- as.data.frame(x3)
dd3 <- cbind(dd3,y3)
q3 <- ggplot(data=dd3, aes(x=x3)) +
  geom_line(aes(y=y3)) +
  ggtitle("unbiased random walk, n=100")+
  ylab("y") +xlab ("x")
```

```r
y4 <- Sn(1000)
x4 <- seq(0,5,length.out = length(y4))
dd4 <- as.data.frame(x4)
dd4 <- cbind(dd4,y4)
q4 <- ggplot(data=dd4, aes(x=x4)) +
  geom_line(aes(y=y4)) +
  ggtitle("unbiased random walk, n=1000")+
  ylab("y") +xlab ("x")

multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {
  library(grid)

  # Make a list from the ... arguments and plotlist
  plots <- c(list(...), plotlist)

  numPlots = length(plots)

  # If layout is NULL, then use 'cols' to determine layout
  if (is.null(layout)) {
    # Make the panel
    # ncol: Number of columns of plots
    # nrow: Number of rows needed, calculated from # of cols
    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                     ncol = cols, nrow = ceiling(numPlots/cols))
  }

  if (numPlots==1) {
    print(plots[[1]])

  } else {
    # Set up the page
    grid.newpage()
    pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))

    # Make each plot, in the correct location
    for (i in 1:numPlots) {
      # Get the i,j matrix positions of the regions that contain this subplot
      matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

      print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
                                      layout.pos.col = matchidx$col))
    }
  }
}

multiplot(q1,q3,q2,q4, cols=2)
```



Let $S_n(t), t \in [0, T]$, where n is fixed, and the process is constructed as follows:

- $S_n(0) = 0$
- n independent steps take place equally spaced in the time interval $[0, T]$
- the size of the step is $\sqrt{\frac{T}{n}}$
- each step is interpolated with a line

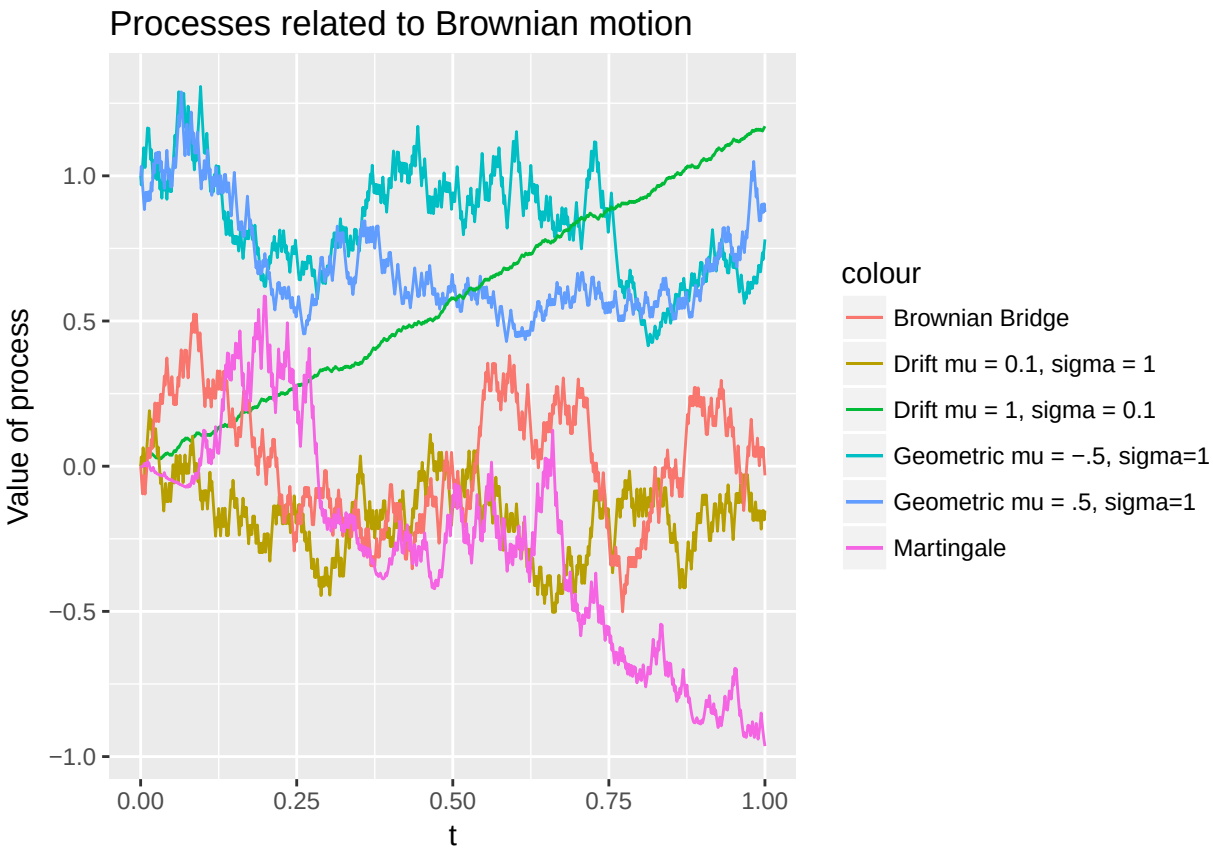
So we obtain an unbiased random walk that moves continuously in time. For different number of n steps, we observe that the simulated random walk resembles, to the Brownian motion.

Indeed, we saw in class the proof of the convergence of the unbiased random walk to the Brownian motion, that by the central limit theorem ensures that:

$(S_n(t), t \in [0, T]) \longrightarrow (B_t, t \in [0, T])$ as $n \to \infty$.

(ii) **Simulate two trajectories of the geometric Brownian motion on the interval** $[0,1]$ **and** $n = 1000$ **with** $(\mu, \sigma) = \{(-0.5, 1), (0.5, 1)\}$**; two trajectories of the Brownian motion with drift on the interval** $[0,1]$ **and with** $(\mu, \sigma) = \{(1, 0.1), (0.1, 1)\}$**, a trajectory of the Brownian bridge and for the martingale** $B_t^2 - t$**. Plot them and comment what you observe.**

```
#_____________________________________________________
#Part 2

geometric.brownian <- function(n,T,sigma,mu){
  B <- Sn(n,T)
  t <- seq(0,T,length.out = n+1)
  X <- sigma * B + mu * t
  return(exp(X))
}

brownian.drift <- function(n,T,sigma,mu){
  B <- Sn(n,T)
  t <- seq(0,T,length.out = n+1)
  X <- sigma * B + mu * t
  return(X)
}

brownian.bridge <- function(n,T){
  B <- Sn(n,T)
  t <- seq(0,1,length.out = n+1)
  X <- B - t * B[n]
}

martingale <- function(n,T) {
  B <- Sn(n,T)
  t <- seq(0,T,length.out = n+1)
  return(B^2 - t)
}

n <- 1000; T <- 1; t <- seq(0,T,length.out = n+1)

simulations <- as.data.frame(
cbind( t,geometric.brownian(n,T,sigma = 1, mu = -0.5),
geometric.brownian(n,T,sigma = 1, mu =0.5),brownian.drift(n,T,sigma = 0.1, mu = 1),
brownian.drift(n,T,sigma = 1, mu = 0.1), brownian.bridge(n,T), martingale(n,T)
  ))

ggplot(data = simulations,aes(x=t,y=V2)) +
  labs(x='t', y='Value of process', title = 'Processes related to Brownian motion') +
  geom_line(aes(x=t,y=V2,colour='Geometric mu = -.5, sigma=1')) +
  geom_line(aes(x=t,y=V3,colour='Geometric mu = .5, sigma=1')) +
  geom_line(aes(x=t,y=V4,colour='Drift mu = 1, sigma = 0.1')) +
  geom_line(aes(x=t,y=V5,colour='Drift mu = 0.1, sigma = 1')) +
  geom_line(aes(x=t,y=V6,colour='Brownian Bridge')) +
  geom_line(aes(x=t,y=V7,colour='Martingale'))
```


Processes related to Brownian motion
Value of process
1.0
0.5
0.0
−0.5
−1.0
0.00
0.25
0.50
0.75
1.00
t
colour
Brownian Bridge
Drift mu = 0.1, sigma = 1
Drift mu = 1, sigma = 0.1
Geometric mu = −.5, sigma=1
Geometric mu = .5, sigma=1
Martingale

Annex

```
#Code for Exercise 1

# part 1
r <- 0.04
u <- 1.06
d <- 0.98
(1+r-d)/(u-d)

# part 2
S0 <- c(20)
S1 <- c(21.2,  19.6)
S2 <- c(22.472,  20.776,  19.208)
S3 <- c(23.82032,  22.02256, 20.36048, 18.82384)
S4 <- c(25.24954,   23.34391,  21.58211, 19.95327, 18.44736)

# part 3
p <- c(0.75, 0.25)
V4 <- c((25.24953-20), (23.34391-20), (21.58211-20), 0, 0)
V3 <- c(p%*%V4[1:2], p%*%V4[2:3], p%*%V4[3:4], p%*%V4[4:5])/(1+r)
V2 <- c(p%*%V3[1:2], p%*%V3[2:3], p%*%V3[3:4])/(1+r)
V1 <- c(p%*%V2[1:2], p%*%V2[2:3])/(1+r)
V0 <- c(p%*%V1[1:2])/(1+r)

# part 4
premFee <- (p[1]^4*V4[1] + p[1]^3*p[2]*4*V4[2] + p[1]^2*p[2]^2*6*V4[3] +
  p[1]*p[2]^3*4*V4[4] + p[2]^4*V4[5])*(1+r)^(-4)

# part 5
a4 <- (u*V4[3] - d*V4[2]) / ((1+r)*(u-d))
b4 <- (V4[2] - V4[3])/(S3[2]*(u-d))

a3 <- (u*V3[3] - d*V3[2]) / ((1+r)*(u-d))
b3 <- (V3[2] - V3[3])/(S2[2]*(u-d))

a2 <- (u*V2[2] - d*V2[1]) / ((1+r)*(u-d))
b2 <- (V2[1] - V2[2])/(S1[1]*(u-d))

a1 <- (u*V1[2] - d*V1[1]) / ((1+r)*(u-d))
b1 <- (V1[1] - V1[2])/(S0[1]*(u-d))
```