Class Project

Hukai Luo

04 December 2018

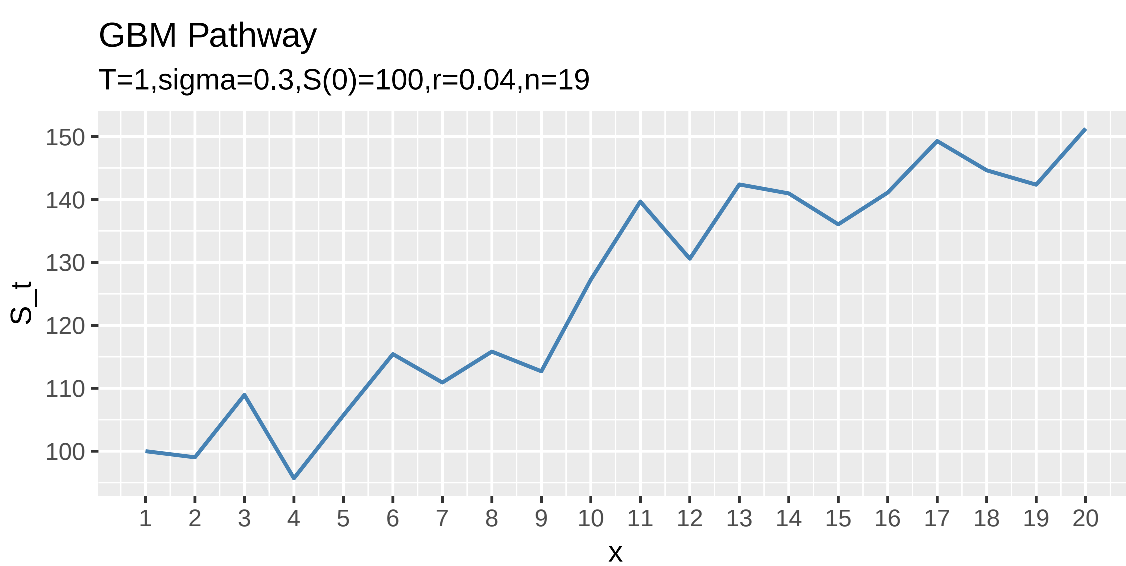
1 Simulate Geometric Brownian Motion

Geometric Brownian Motion solution

$$S(T) = S(t)e^{(\mu-\frac{1}{2}\sigma^2)(T-t)+\sigma\sqrt{T-t}z}$$

Now, given some initial data $S(0) = 100$, $r = 0.04$, $\sigma = 0.3$, $T = 1$, let's simulate the Geometric Brownian Motion pathways.

```
s0 <- 100
r <- 0.04
sigma <- 0.3                                    # input initial data
T <- 1
S_T <- function(s0,r,sigma,T,n=19){             # build the GBM function below
  data <- double(0)
  data[1] <- s0
  for(i in 1:19){
    s0 <- s0*exp((r-0.5*sigma^2)*(T/n)+sigma*sqrt(T/n)*rnorm(1,0,1))
    data[i+1] <- s0
  }
  return(data)
}
data <- S_T(s0,r,sigma,T)                       # use the function the get each path's stock value S_T
library(ggplot2)                                # plot the pathway
plot1 <- ggplot(data.frame(x=seq(1:20),S_t = data),aes(x=x,y=S_t))+
  geom_line(col="steelblue", size=0.7)+scale_x_continuous(breaks=seq(1, 20, 1))+
  labs(title="GBM Pathway", subtitle="T=1,sigma=0.3,S(0)=100,r=0.04,n=19")
plot1
```

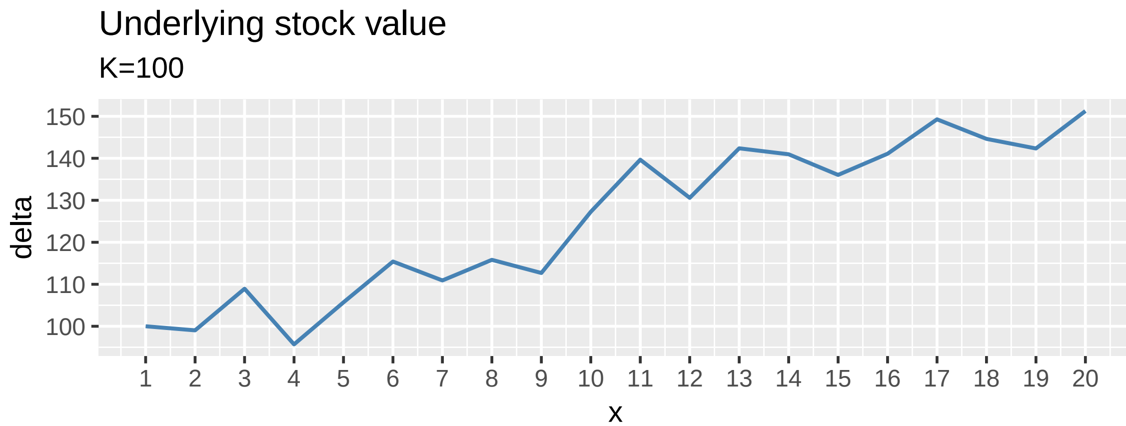


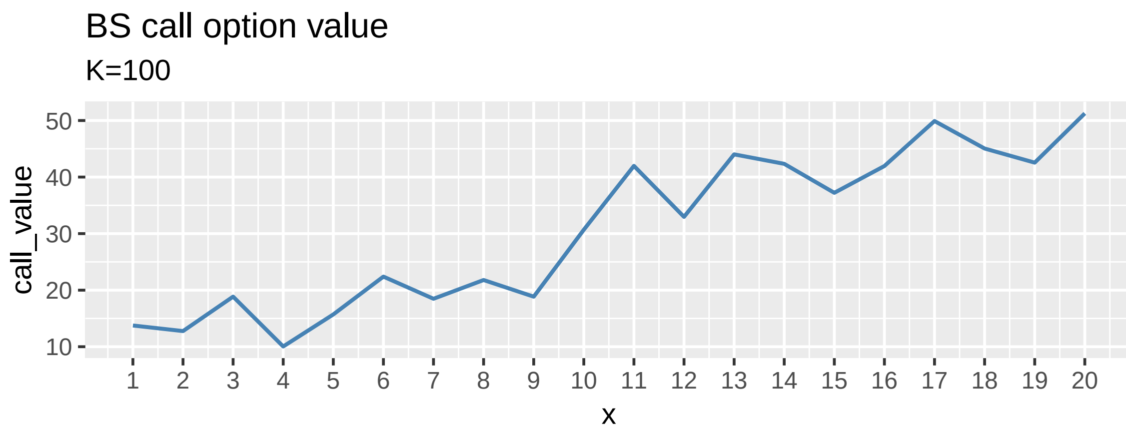
2 Vanilla black-scholes european call option

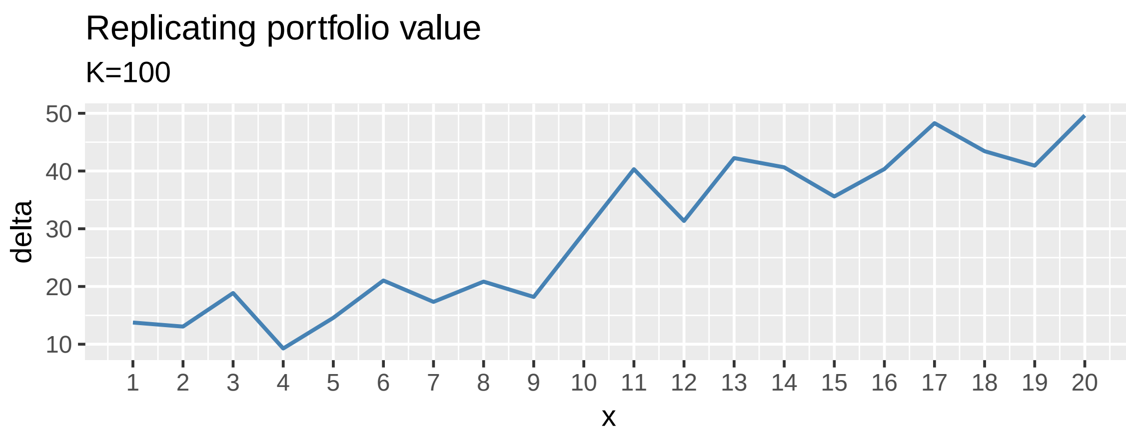
Set the $K = 100$

```
K=100                                # Excercise price K
n <- 19
B_S <- function(data,K){             # Build the Bs call option function
  call_value <- double(0)            # calculate d1,d2,call option value,cashflow,replicating value
  d1 <- double(0)
  d2 <- double(0)
  for(i in 1:20){
    d1[i] <- (log(data[i]/K)+(r+0.5*sigma^2)*(20-i)*T/n)/(sigma*sqrt((20-i)*T/n))
    d2[i] <- (log(data[i]/K)+(r-0.5*sigma^2)*(20-i)*T/n)/(sigma*sqrt((20-i)*T/n))
    call_value[i] <- data[i]*pnorm(d1[i])-K*exp(-r*(20-i)*T/n)*pnorm(d2[i])
    }
  cashflow <- double(0)              # calculate the cash flow and replicating value
  bt <- double(0)
  replicate <- double(0)
  bt[1] <- K*exp(-r*T)*pnorm(d2[1])
  cashflow[1] <- 0
  replicate[1] <- -bt[1]+data[1]*pnorm(d1[1])
  for(i in 2:20){
    cashflow[i] <- data[i]*(pnorm(d1[i])-pnorm(d1[i-1]))
    bt[i] <- bt[i-1]*exp(r*T/n)+cashflow[i]
    replicate[i] <- data[i]*pnorm(d1[i])- bt[i]
  }
  return(data.frame(stock=data,d1=d1,delta=pnorm(d1),Bt=bt,replicate=replicate,call_option=call_value))
}
call <- B_S(data,100)
print(call)
```

```
##        stock         d1     delta        Bt replicate call_option
## 1  100.00000 0.28333333 0.6115393  47.40067 13.753265    13.75326
## 2   99.02972 0.24238508 0.5957591  45.93785 13.060002    12.76205
## 3  108.94251 0.56983412 0.7156049  59.09097 18.868824    18.85494
## 4   95.68877 0.09992665 0.5397987  42.39282  9.259851    10.03892
## 5  105.69820 0.45965023 0.6771163  56.99639 14.573590    15.70654
## 6  115.42197 0.80016027 0.7881910  69.93697 21.037590    22.39477
## 7  110.91155 0.65170297 0.7427036  65.03928 17.335131    18.47451
## 8  115.82349 0.84130925 0.7999126  71.80250 20.846177    21.79056
## 9  112.69258 0.73906800 0.7700671  68.59046 18.190399    18.84690
## 10 127.25761 1.31306891 0.9054201  85.95971 29.261894    30.68665
## 11 139.66762 1.81310047 0.9650918  94.47507 40.317007    41.97431
## 12 130.58328 1.55461634 0.9399813  91.39515 31.350686    32.96305
## 13 142.37984 2.11234810 0.9826717  97.66602 42.246614    44.01509
## 14 140.95759 2.19550970 0.9859365  98.33205 40.643182    42.33717
## 15 136.03936 2.14522045 0.9840324  98.28025 35.586883    37.21219
## 16 141.08948 2.63073264 0.9957399 100.13919 40.349248    41.95489
## 17 149.27865 3.47347378 0.9997431 100.94782 48.292487    49.90943
## 18 144.62669 3.88289274 0.9999484 101.19025 43.428973    45.04703
## 19 142.34173 5.19485144 0.9999999 101.41084 40.930876    42.55203
## 20 151.25733        Inf 1.0000000 101.62457 49.632759    51.25733
```







The value of the replicating portfolio at time T

$$C = \triangle S + B$$

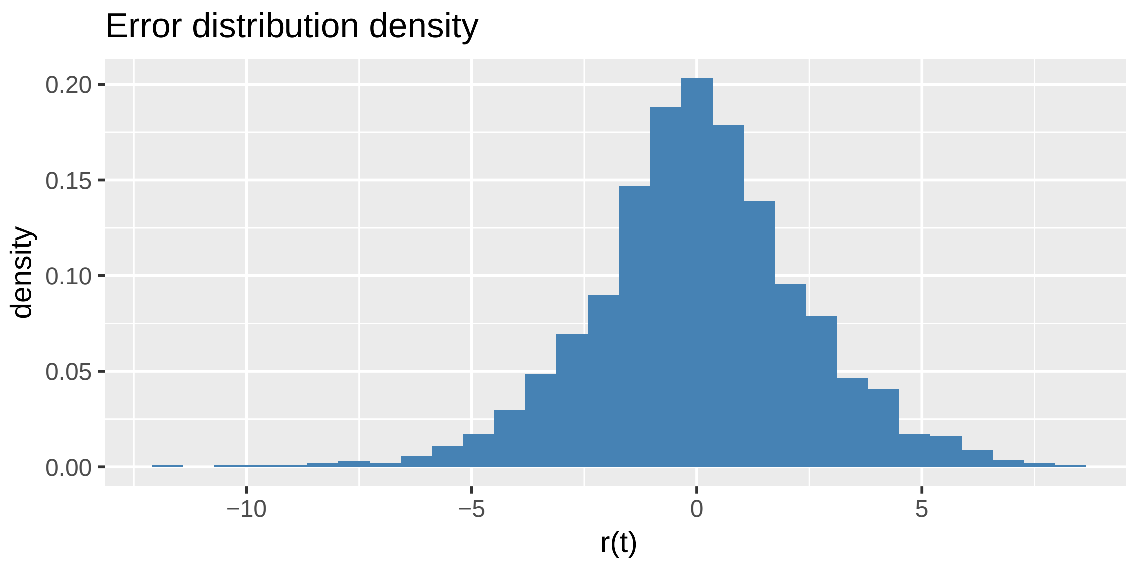
The terminal value of the call at time T

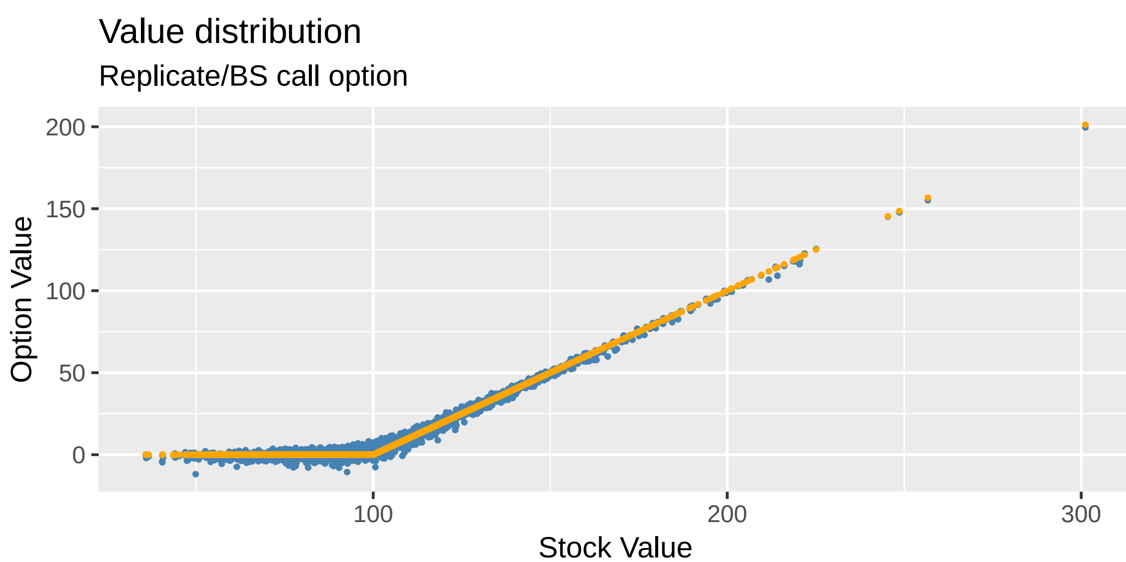
$$C = max(S - K, 0)$$

Now repeat above process for 2000 times to generate the distribution of replicating error.

```
stock <- call <- replicate <- error <- double(0)
for(i in 1:2000){
```

```
  t <- B_S(S_T(s0,r,sigma,T),100)
  error[i] <- t$replicate[20]-t$call_option[20]
  replicate[i] <- t$replicate[20]
  call[i] <- t$call_option[20]
  stock[i] <- t$stock[20]
}
```





3 Cash-or-nothing call option

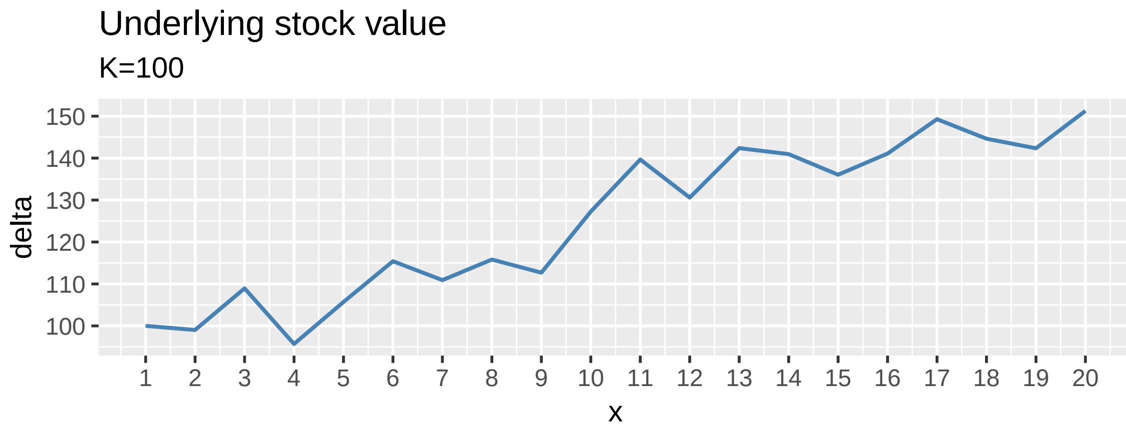
Cash-or-nothing call option either pays you a fixed amount of money or nothing at all.

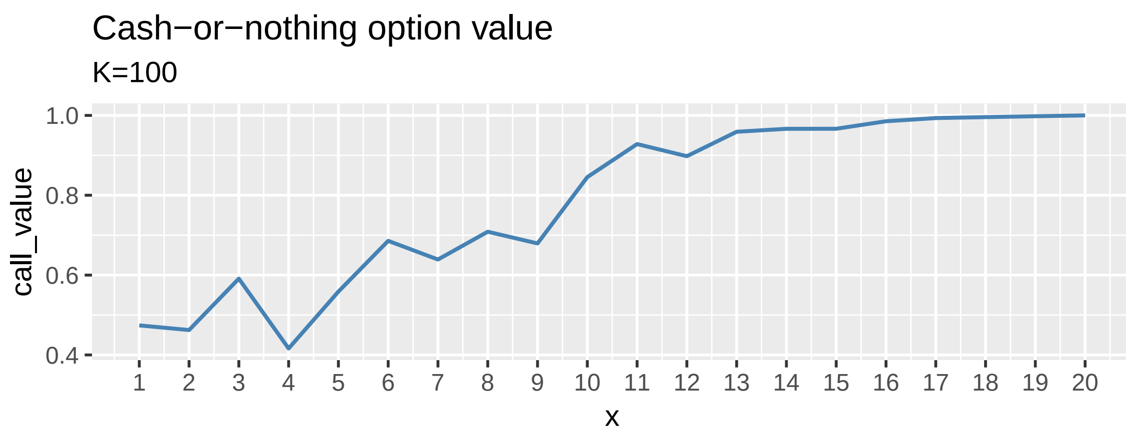
$$C_{cn} = ke^{-r\tau}N(d2)$$

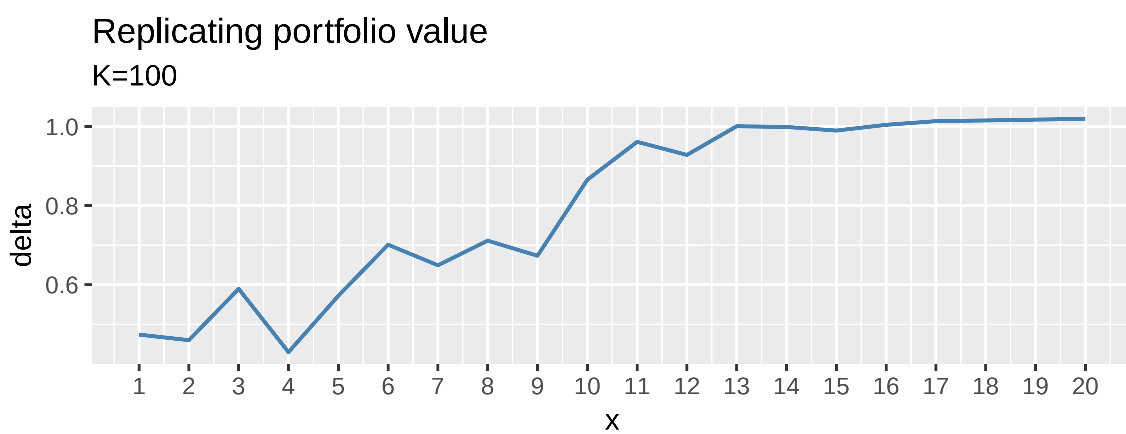
where S the initial stock price, K the strike price, T the time to maturity,σ the volatility and r the risk free interest rate, in thie section, we will set $k = 1$ for convenience.

```r
t <- 1e-6
C_N <- function(data,K){
  call_value <- double(0)           # calculate d1,d2,call option value,cashflow,replicating value
  d1 <- double(0)
  d2 <- double(0)
  delta <- double(0)
  for(i in 1:20){
    d1[i] <- (log(data[i]/K)+(r+0.5*sigma^2)*(20-i)*T/n)/(sigma*sqrt((20-i)*T/n))
    d2[i] <- (log(data[i]/K)+(r-0.5*sigma^2)*(20-i)*T/n)/(sigma*sqrt((20-i)*T/n))
    call_value[i] <- exp(-r*(20-i)*T/n)*pnorm(d2[i])
    delta[i] <- exp(-r*(20-i)*T/n)*dnorm(d2[i])/(data[i]*sigma*sqrt((20-i)*T/n))
  }
  delta[20] <- 0
  cashflow <- double(0)             # calculate the cash flow and replicating value
  bt <- double(0)
  replicate <- double(0)
  bt[1] <- -call_value[1]+data[1]*delta[1]
  cashflow[1] <- 0
  replicate[1] <- -bt[1]+data[1]*delta[1]
  for(i in 2:20){
    cashflow[i] <- data[i]*(delta[i]-delta[i-1])
    bt[i] <- bt[i-1]*exp(r*T/n)+cashflow[i]
    replicate[i] <- data[i]*delta[i]- bt[i]
  }
  return(data.frame(stock=data,delta=delta,bt=bt,replicate=replicate,CON=call_value))
}
call1 <- C_N(data,100)
print(call1)
```

```
##        stock        delta         bt replicate       CON
## 1  100.00000 1.277488e-02  0.8034810 0.4740067 0.4740067
## 2   99.02972 1.326697e-02  0.8539065 0.4599181 0.4623581
## 3  108.94251 1.195174e-02  0.7124211 0.5896313 0.5910485
## 4   95.68877 1.441907e-02  0.9500187 0.4297246 0.4161376
## 5  105.69820 1.346609e-02  0.8512921 0.5720493 0.5586344
## 6  115.42197 1.124791e-02  0.5970592 0.7011962 0.6857979
## 7  110.91155 1.300073e-02  0.7927263 0.6492052 0.6389990
## 8  115.82349 1.174567e-02  0.6490315 0.7113934 0.7085812
## 9  112.69258 1.330022e-02  0.8255850 0.6732509 0.6793396
## 10 127.25761 7.740539e-03  0.1198134 0.8652290 0.8453495
## 11 139.66762 3.734298e-03 -0.4394762 0.9610367 0.9281776
## 12 130.58328 6.120857e-03 -0.1287577 0.9280393 0.8978279
## 13 142.37984 2.353436e-03 -0.6654338 1.0005156 0.9589755
## 14 140.95759 2.125119e-03 -0.6990192 0.9985709 0.9663806
## 15 136.03936 2.596422e-03 -0.6363766 0.9895922 0.9665494
## 16 141.08948 9.105692e-04 -0.8755739 1.0040456 0.9853354
## 17 149.27865 8.030067e-05 -1.0013605 1.0133477 0.9933087
## 18 144.62669 2.181488e-05 -1.0119295 1.0150845 0.9957220
## 19 142.34173 8.000656e-08 -1.0171559 1.0171672 0.9978968
## 20 151.25733 0.000000e+00 -1.0193116 1.0193116 1.0000000
```







The value of the replicating portfolio at time T

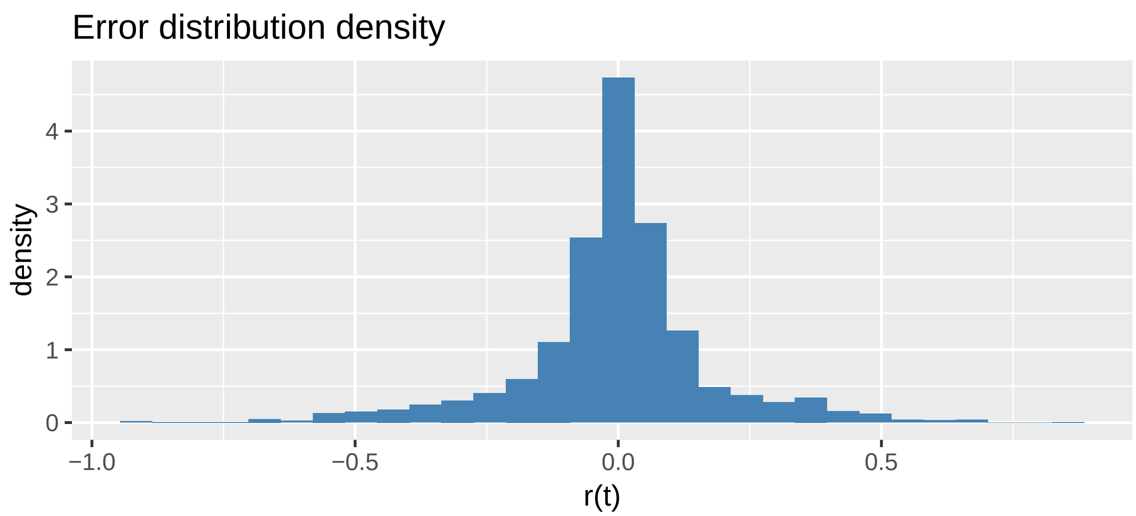
$$C = \triangle S + B$$

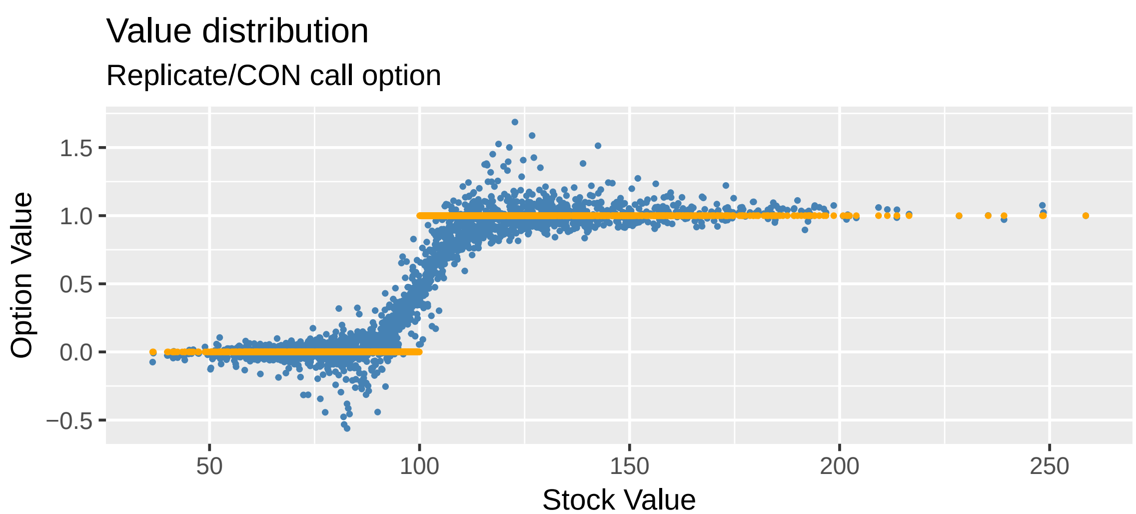
The terminal value of the call at time T

$$C = \frac{max(S-K,0)}{|S-K|}$$

Now repeat above process for 2000 times to generate the distribution of replicating error.

```
stock <- call <- replicate <- error <- double(0)
for(i in 1:2000){
  t <- C_N(S_T(s0,r,sigma,T),100)
  error[i] <- t$replicate[20]-t$CON[20]
  replicate[i] <- t$replicate[20]
  call[i] <- t$CON[20]
  stock[i] <- t$stock[20]
}
```





4 Asset-or-nothing call option

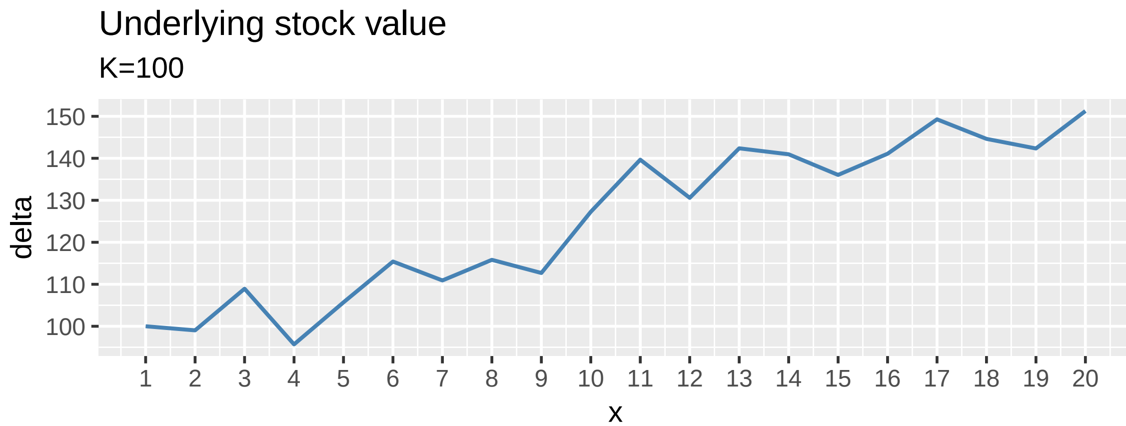
The asset-or-nothing option is basically the same, but your payment equals the price of the asset underlying the option.

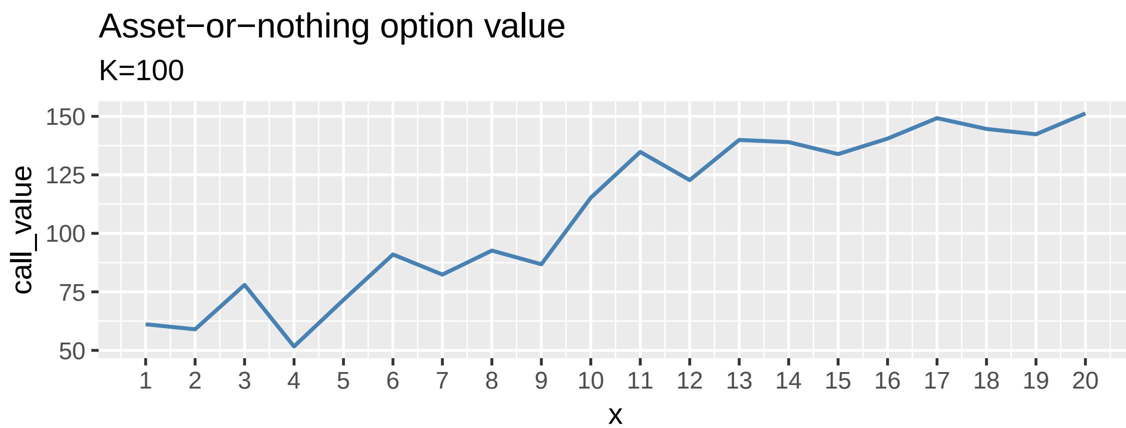
$$C_{an} = SN(d1)$$

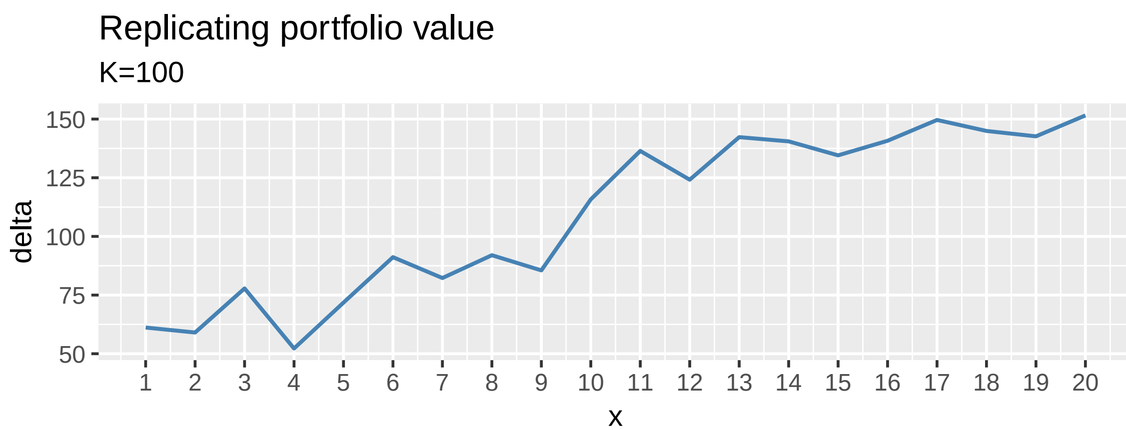
In order to replicate this option, we can buy call at $K = 100$ and buy K units cash-or-nothing call at $K = 100$.

```r
epsilon <- 0.01
A_N <- function(data,K){
  call_value <- double(0)            # calculate d1,d2,call option value,cashflow,replicating value
  d1 <- double(0)
  d2 <- double(0)
  delta <- double(0)
  for(i in 1:20){
    d1[i] <- (log(data[i]/K)+(r+0.5*sigma^2)*(20-i)*T/n)/(sigma*sqrt((20-i)*T/n))
    d2[i] <- (log(data[i]/K)+(r-0.5*sigma^2)*(20-i)*T/n)/(sigma*sqrt((20-i)*T/n))
    call_value[i] <- data[i]*pnorm(d1[i])
    delta[i] <- pnorm(d1[i])+dnorm(d1[i])/(sigma*sqrt((20-i)*T/n))
  }
  delta[20] <- pnorm(d1[i])
  cashflow <- double(0)              # calculate the cash flow and replicating value
  bt <- double(0)
  replicate <- double(0)
  bt[1] <- -call_value[1]+data[1]*delta[1]
  cashflow[1] <- 0
  replicate[1] <- -bt[1]+data[1]*delta[1]
  for(i in 2:20){
    cashflow[i] <- data[i]*(delta[i]-delta[i-1])
    bt[i] <- bt[i-1]*exp(r*T/n)+cashflow[i]
    replicate[i] <- data[i]*delta[i]- bt[i]
  }
  return(data.frame(stock=data,delta=delta,bt=bt,replicate=replicate,AON=call_value))
}
call2 <- A_N(data,100)
print(call2)
```

```
##        stock    delta           bt replicate       AON
## 1  100.00000 1.889027 127.748765831  61.15393  61.15393
## 2   99.02972 1.922456 131.328498331  59.05181  58.99785
## 3  108.94251 1.910779 130.333073947  77.83195  77.95979
## 4   95.68877 1.981706 137.394692820  52.23231  51.65268
## 5  105.69820 2.023725 142.125607654  71.77852  71.56998
## 6  115.42197 1.912982 129.642887142  91.15721  90.97456
## 7  110.91155 2.042777 144.311912310  82.25566  82.37441
## 8  115.82349 1.974480 136.705645021  91.98552  92.64868
## 9  112.69258 2.100089 151.148959879  85.51549  86.78085
## 10 127.25761 1.679474  97.941049985 115.78480 115.22160
## 11 139.66762 1.338522  50.527453281 136.42068 134.79208
## 12 130.58328 1.552067  78.519384523 124.15461 122.74584
## 13 142.37984 1.218015  31.122646762 142.29818 139.91264
## 14 140.95759 1.198448  28.430122696 140.50027 138.97523
## 15 136.03936 1.243675  34.642592501 134.54610 133.86713
## 16 141.08948 1.086797  12.581798139 140.75381 140.48844
## 17 149.27865 1.007773   0.811765741 149.62725 149.24030
## 18 144.62669 1.002130  -0.002696013 144.93742 144.61922
## 19 142.34173 1.000008  -0.304747950 142.64760 142.34171
## 20 151.25733 1.000000  -0.306584863 151.56392 151.25733
```







The value of the replicating portfolio at time T

$$C = \triangle S + B$$

The terminal value of the call at time T

$$C = \frac{max(S-K,0)}{|S-K|}S$$

Now repeat above process for 2000 times to generate the distribution of replicating error.

```
stock <- call <- replicate <- error <- double(0)
for(i in 1:2000){
  t <- A_N(S_T(s0,r,sigma,T),100)
  error[i] <- t$replicate[20]-t$AON[20]
  replicate[i] <- t$replicate[20]
  call[i] <- t$AON[20]
  stock[i] <- t$stock[20]
}
```