Class Project Math 5660

Chamundeswari Koppisetti

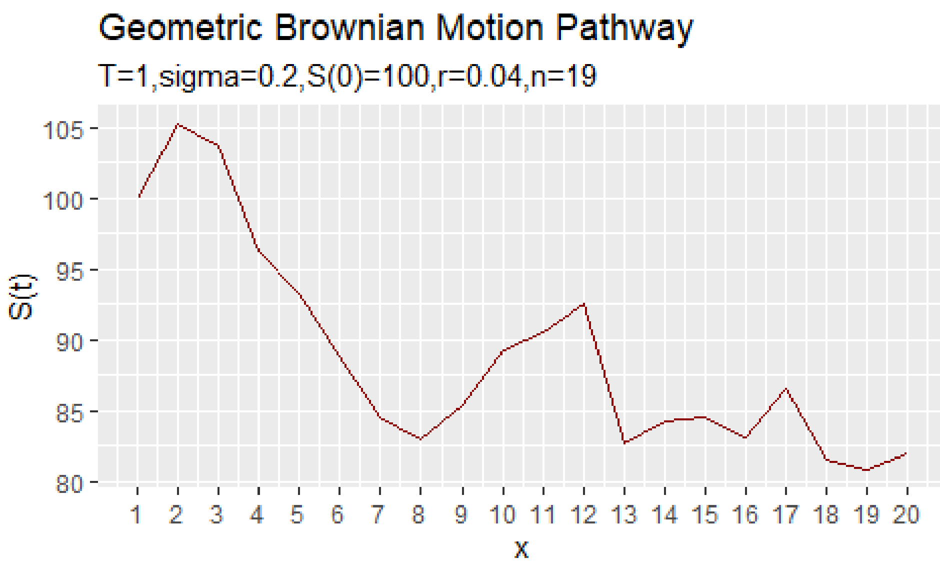
11/25/2019

1 Simulate Geometric Brownian Motion

Geometric Brownian Motion:

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

```
# input initial values
S_0 <- 100
r <- 0.04
sigma <- 0.2
n = 19
T <- 1

# Geometric brownian motion function
S_T <- function(S_0,r,sigma,T,n=19){
  data <- double(0)
  data[1] <- S_0
  for(i in 1:19){
    S_0 <- S_0*exp((r-0.5*sigma^2)*(T/n)+sigma*sqrt(T/n)*rnorm(1,0,1))
    data[i+1] <- S_0
  }
  return(data)
}
#function the get each path's stock value S_T
data <- S_T(S_0,r,sigma,T)
library(ggplot2)
plot1 <- ggplot(data.frame(x=seq(1:20),S_t = data),aes(x=x,y=S_t))+
  geom_line(col="darkred", size=0.7)+scale_x_continuous(breaks=seq(1, 20,
1))+
  labs(title="Geometric Brownian Motion Pathway",
subtitle="T=1,sigma=0.2,S(0)=100,r=0.04,n=19",y="S(t)")
plot1 #GBM pathway
```



2 Value of Vanilla Black-Scholes European call option

```
#input exercise price
K=100

#Black Scholes call option function: calculating d1, d2 and call value
C_BS <- function(data,K){
  call_value <- double(0)
  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])
    }
# calculating cash flow, B(t) and C replicated (t)
  cashflow <- double(0)
  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,
```

```r
call_option=call_value))
}
call1 <- C_BS(data,100)
print(call1)
```

```
##        stock          d1        delta         Bt  replicate  call_option
## 1  100.00000  0.30000000 6.179114e-01 51.8660885  9.9250537 9.925054e+00
## 2  105.23116  0.55393079 7.101869e-01 61.6856498 13.0481375 1.308463e+01
## 3  103.65448  0.47349947 6.820716e-01 58.9013708 11.7984056 1.165939e+01
## 4   96.25909  0.06756025 5.269321e-01 44.0919262  6.6300813 6.841539e+00
## 5   93.28164 -0.12480496 4.503390e-01 37.0401125  4.9682479 5.079949e+00
## 6   88.86698 -0.42998245 3.336042e-01 26.7443060  2.9020935 3.071346e+00
## 7   84.51751 -0.76863777 2.210542e-01 17.2882224  1.3947266 1.642419e+00
## 8   83.04530 -0.93044944 1.760692e-01 13.5888666  1.0328536 1.166937e+00
## 9   85.43601 -0.80607122 2.101009e-01 16.5250357  1.4251467 1.440107e+00
## 10  89.26560 -0.56497431 2.860456e-01 23.3391131  2.1949195 2.148528e+00
## 11  90.51408 -0.51757368 3.023779e-01 24.8665989  2.5028559 2.234040e+00
## 12  92.60420 -0.39739322 3.455388e-01 28.9158839  3.0824557 2.594000e+00
## 13  82.66216 -1.38640333 8.281189e-02  7.2592530 -0.4138433 3.613691e-01
## 14  84.26452 -1.35476042 8.774695e-02  7.6904022 -0.2964477 3.665320e-01
## 15  84.51015 -1.48647420 6.857687e-02  6.0865430 -0.2911015 2.518964e-01
## 16  83.14553 -1.87373940 3.048318e-02  2.9320500 -0.3975100 8.744719e-02
## 17  86.59279 -1.69216841 4.530693e-02  4.2218598 -0.2986059 1.246075e-01
## 18  81.52034 -3.05141393 1.138832e-03  0.6301588 -0.5373208 1.655524e-03
## 19  80.80097 -4.57735099 2.354505e-06  0.5396583 -0.5394681 1.742128e-06
## 20  81.94503        -Inf 0.000000e+00  0.5406027 -0.5406027 0.000000e+00
```

#Deliverables: 2 description for Vanilla Black-Scholes European call option

The value of the replicating portfolio at time T

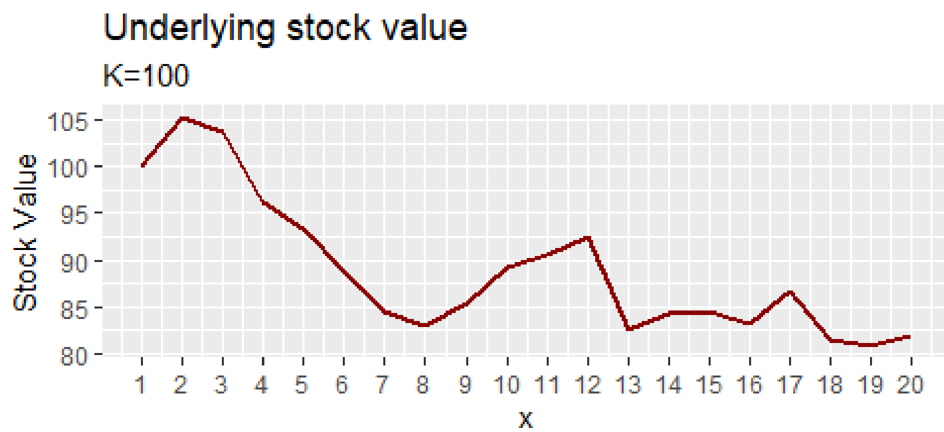
$$C = \triangle S + B$$

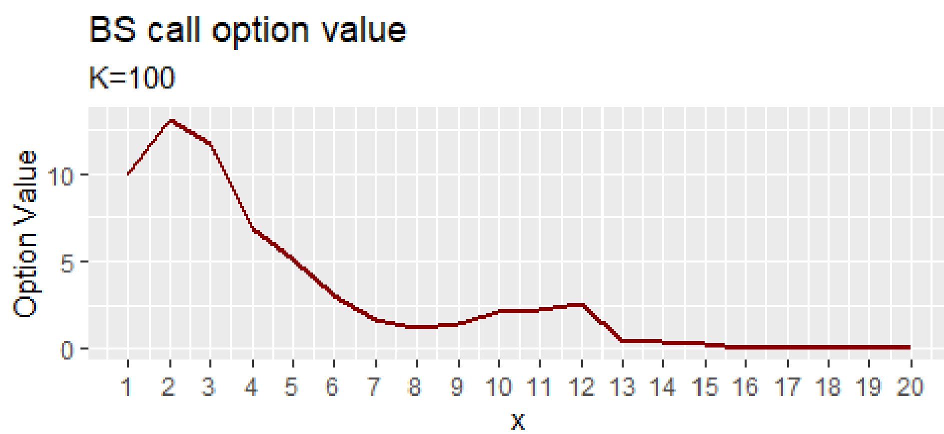
The terminal value of the call at time T

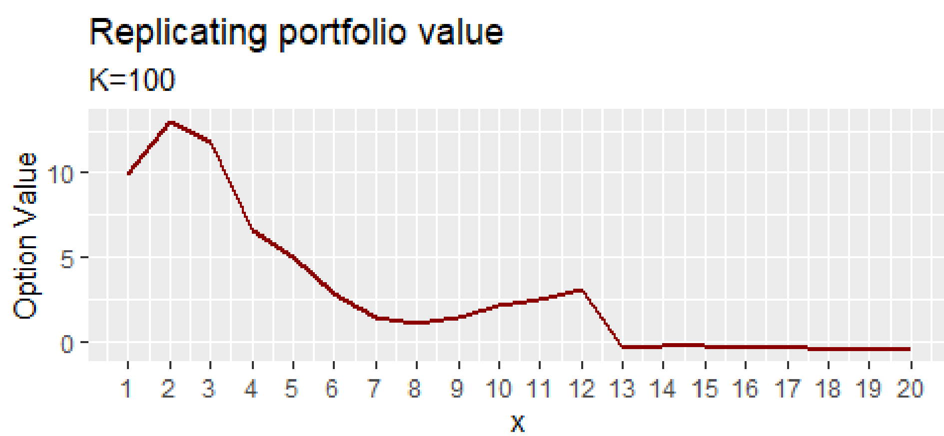
$$C = max(S - K, 0)$$

#Deliverables: (3-4) plots for Vanilla Black-Scholes European call option

```
## Loading required package: grid
```



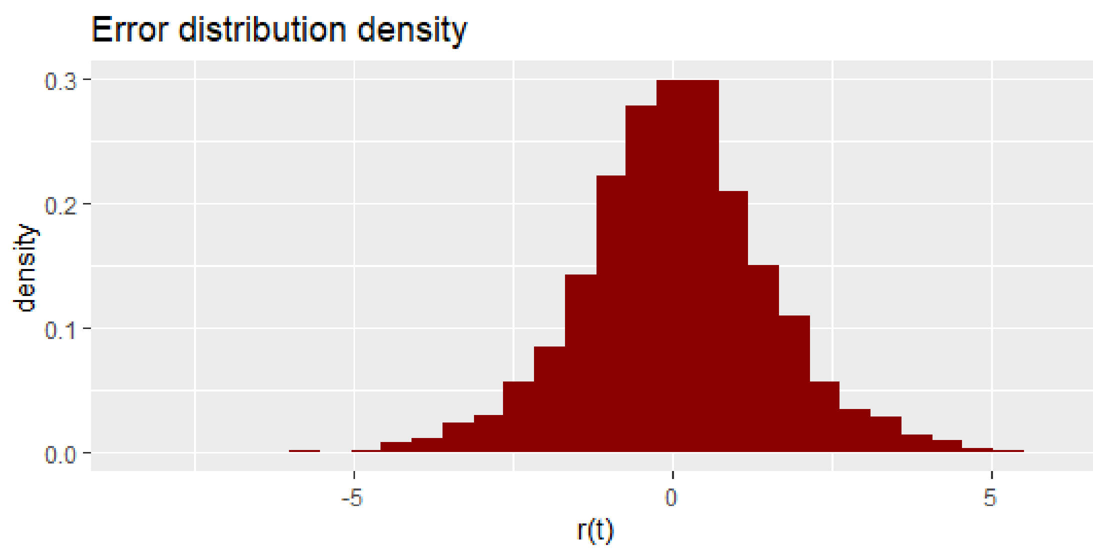


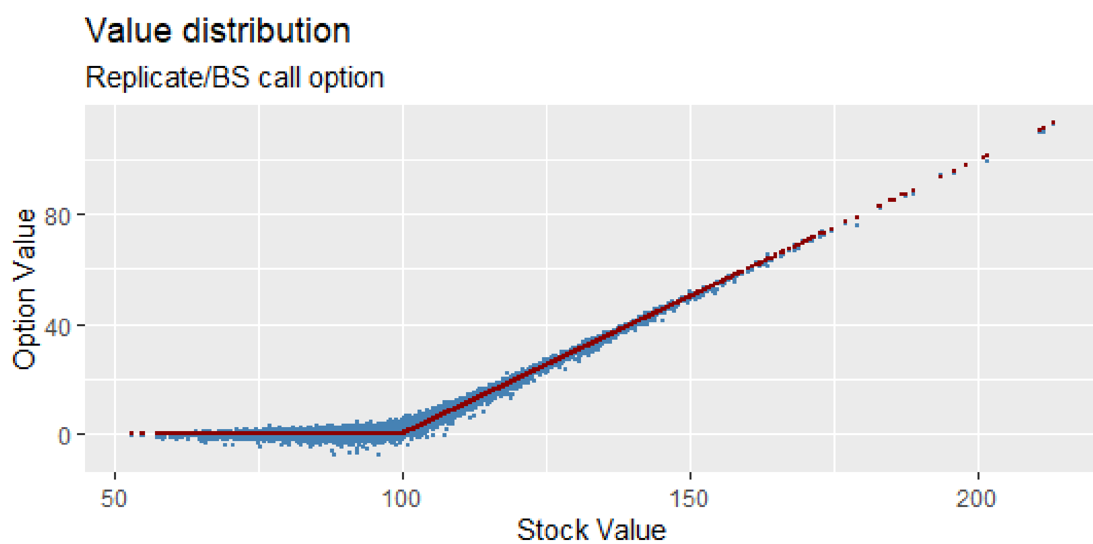


Repeat above process for 5000 times to generate the distribution of replicating error
#Deliverables: 1 plot, distribution of relication errors for Vanilla Black-Scholes European call option

```
stock <- call <- replicate <- error <- double(0)
for(i in 1:5000){
  t <- C_BS(S_T(S_0,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]
}

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



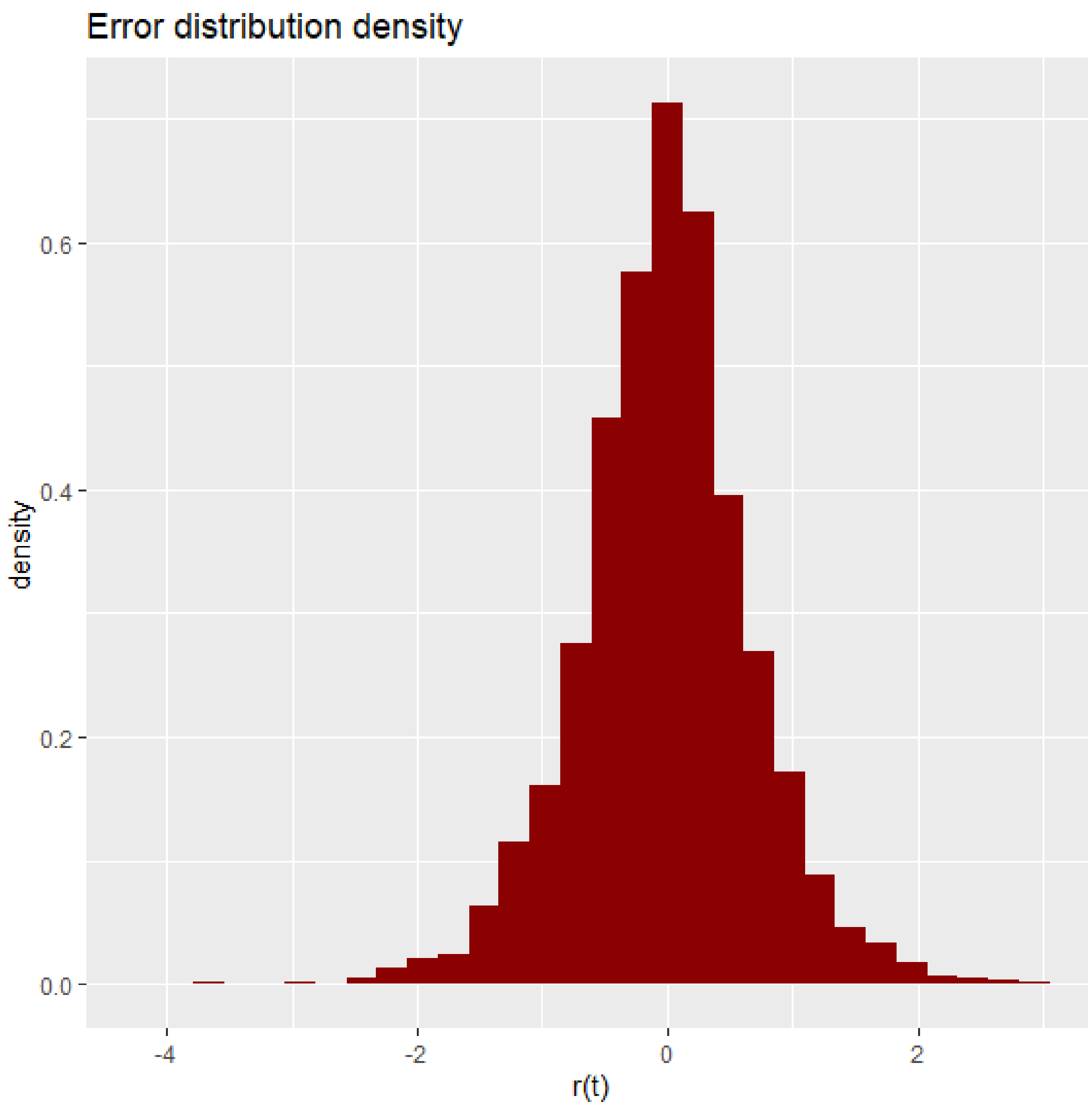


It is well understood that even if we have perfect foreknowledge of the quadratic variation, you will still experience replication errors, as we could not continuously rebalance the hedging position. If we adjust the delta hedge N times during the life of the option, the time interval between the hedge rebalances will be $\Delta t = T / N$.

Magnitude of this replicating error is inversely related to the frequency of rebalancing under the BS assumptions. The smaller the adjustment period of the portfolio, the higher the kurtosis for asset price returns

Have plotted with n=99, below is the graph

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



3 Value of Cash-or-nothing European call option

Cash-or-nothing call option has a binary outcome. It pays out either a fixed amount, if the underlying stock exceeds a predetermined threshold or strike price, or pays out nothing. Has a discontinuous pay off

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

We will set K=1 for convenience

```
t <- 1e-6

# Call option function: calculating d1, d2 and call value
C_CN <- function(data,K){
  call_value <- double(0)
```

```
  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

# calculating cash flow, B(t) and C replicated (t)
  cashflow <- double(0)
  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_v
alue))
}
call2 <- C_CN(data,100)
print(call2)
```

```
##        stock        delta          Bt replicate       CON
## 1  100.00000 0.0190693908  1.38827819 0.5186609 0.5186609
## 2   97.73747 0.0201842776  1.50017017 0.4725900 0.4736376
## 3  112.64348 0.0138990261  0.79533917 0.7702955 0.7385328
## 4  112.81671 0.0140743085  0.81679011 0.7710271 0.7474179
## 5  117.24588 0.0114306235  0.50855030 0.8316433 0.8114517
## 6  115.03922 0.0130589507  0.69694355 0.8053480 0.7927363
## 7  116.64434 0.0120378636  0.57930833 0.8248403 0.8217541
## 8  110.14850 0.0175407518  1.18666409 0.7454234 0.7353361
## 9  114.54706 0.0139896429  0.78239586 0.8200766 0.8145375
## 10 122.68745 0.0073205380 -0.03417074 0.9323089 0.9114069
## 11 126.33700 0.0047229480 -0.36241448 0.9590976 0.9433595
## 12 119.51326 0.0089881951  0.14657531 0.9276332 0.9094011
## 13 123.85017 0.0049656345 -0.35131062 0.9663053 0.9516988
## 14 136.82002 0.0004469656 -0.97029536 1.0314492 0.9852589
## 15 120.50648 0.0055633229 -0.35578603 1.0262025 0.9590677
## 16 120.02067 0.0045336092 -0.48012278 1.0242496 0.9708425
```

```
## 17 121.65135 0.0017753275 -0.81668332 1.0326543 0.9876351
## 18 117.13051 0.0024790825 -0.73597327 1.0263495 0.9890563
## 19 117.01003 0.0001952429 -1.00475647 1.0276018 0.9976137
## 20 121.25717 0.0000000000 -1.03054857 1.0305486 1.0000000
```

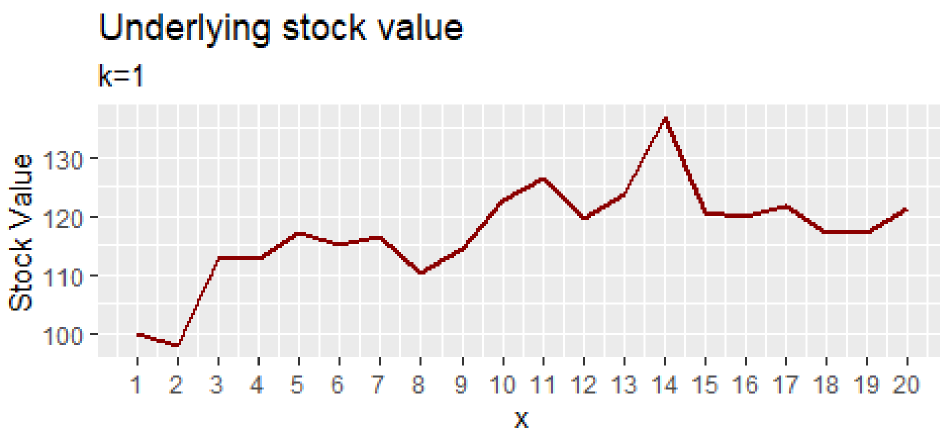
#Deliverables: 2 description for Cash-or-Nothing European call option The value of the replicating portfolio at time T

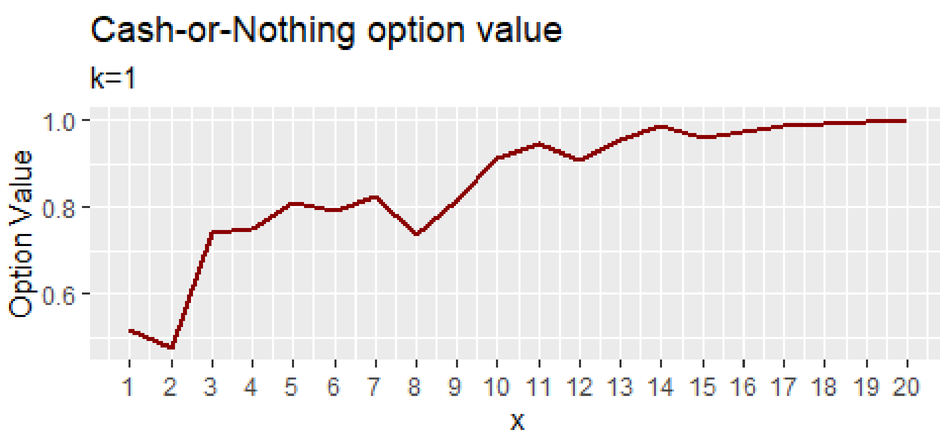
$$C = \triangle S + B$$

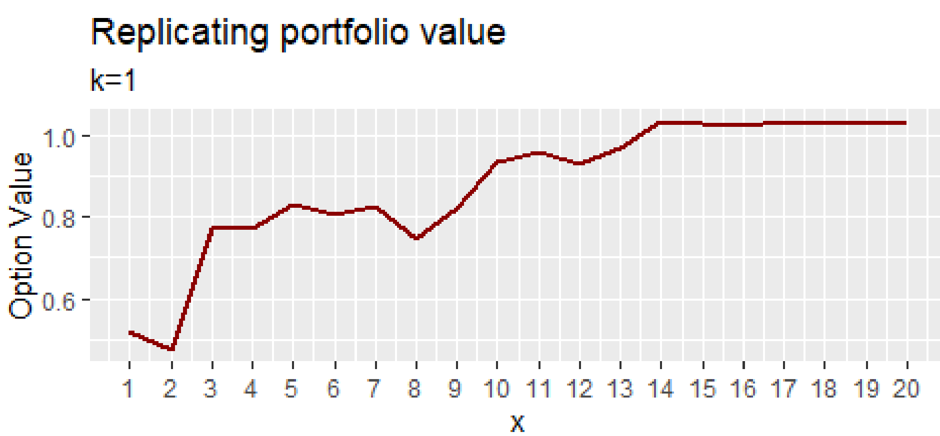
The terminal value of the call at time T

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

#Deliverables: (3-4) plots for Cash-or-Nothing European call option



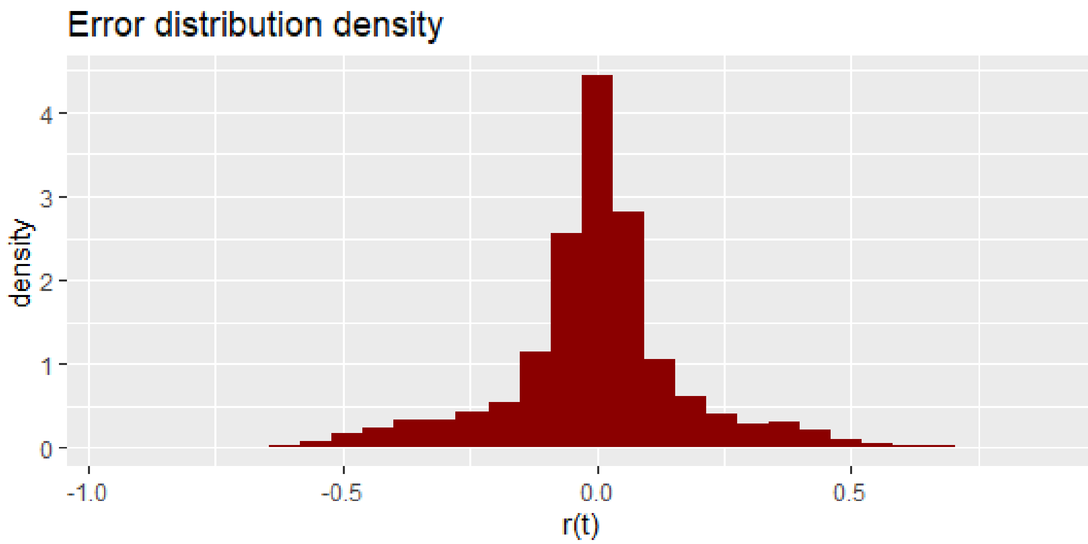


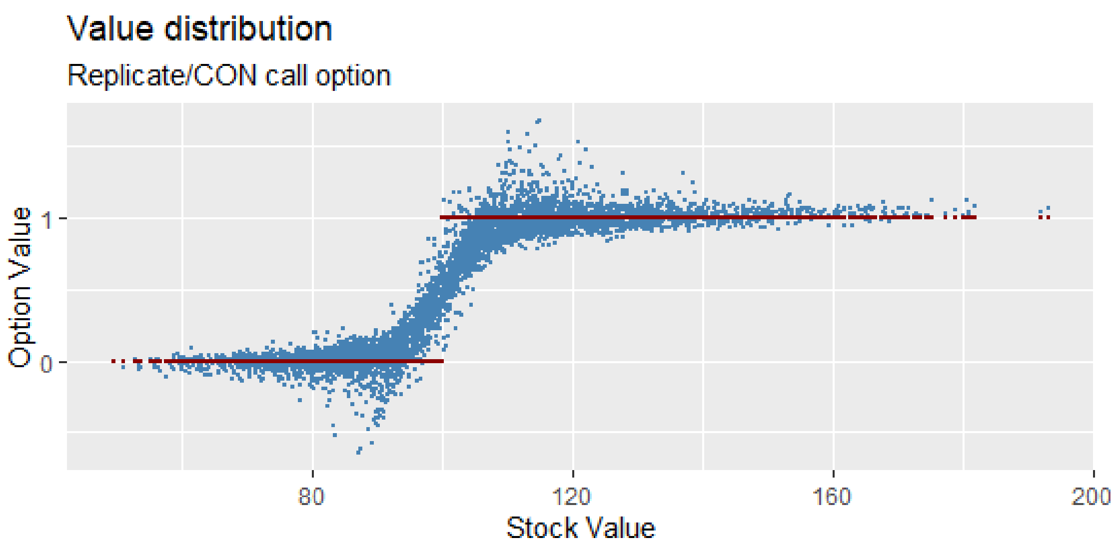


Repeat above process for 5000 times to generate the distribution of replicating error
#Deliverables: 1 plot, distribution of relication errors for Cash-or-Nothing European call option

```
stock <- call <- replicate <- error <- double(0)
for(i in 1:5000){
  t <- C_CN(S_T(S_0,r,sigma,T),100)
  error[i] <- t$replicate[20]-t$CON[20]
  replicate[i] <- t$replicate[20]
  call[i] <- t$CON[20]
```

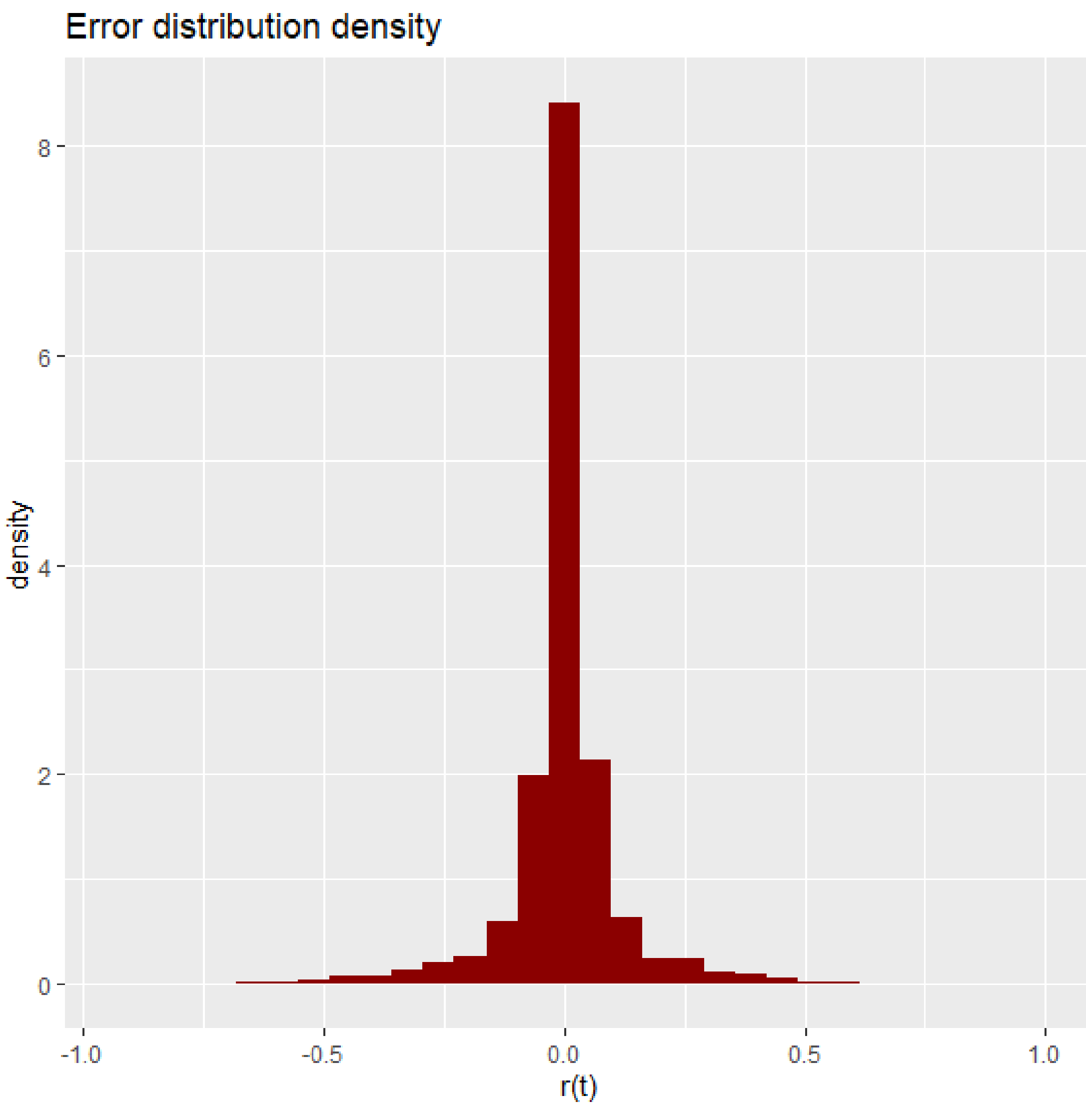
```
  stock[i] <- t$stock[20]
}

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```





Have plotted with n=99, below is the graph

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



3 Value of Asset-or-nothing European call option

Asset-or-nothing call is a type of digital option whose payout is fixed after the underlying asset exceeds the predetermined threshold or strike price. The payout depends only on whether or not the underlying asset closes above the strike price - in the money - at the expiration date.

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

Special type of financial derivative with a non-linear discontinuous pay off function

```
epsilon <- 0.01

# Call option function: calculating d1, d2 and call value
C_AN <- function(data,K){
```

```
  call_value <- double(0)
  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])

# calculating cash flow, B(t) and C replicated (t)
  cashflow <- double(0)
  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_v
alue))
}
call3 <- C_AN(data,100)
print(call3)
```

```
##        stock    delta        Bt replicate      AON
## 1  100.00000 2.524850 190.69391  61.79114 61.79114
## 2   99.74733 2.581018 196.69835  60.75131 60.83376
## 3  104.79935 2.533432 192.12586  73.37611 73.62002
## 4  102.36989 2.660711 205.56038  66.81636 67.20544
## 5   99.97596 2.771971 217.11693  60.01358 60.44285
## 6   98.90685 2.857386 226.02260  56.59247 57.04096
## 7   87.24532 2.324230 179.98355  22.79464 24.61156
## 8   87.10043 2.321706 180.14305  22.07855 23.01059
## 9   87.12882 2.333618 181.56059  21.76482 21.70882
## 10  93.51770 3.072227 251.01622  36.29141 37.73589
## 11  95.90373 3.345752 277.77728  43.09283 44.23201
## 12 100.10557 3.591843 302.99775  56.56571 58.09655
## 13  97.41668 3.771096 321.09861  46.26910 47.40644
## 14 105.04136 3.681627 312.37729  74.34577 76.45015
## 15 106.77506 3.625776 307.07215  80.07028 83.93207
## 16 100.60291 4.839074 429.78063  57.04429 58.39918
```

```
## 17  95.71268 4.905149 437.01054  32.47436 31.85388
## 18  96.40574 5.833851 527.46379  34.95294 30.88153
## 19  97.18942 7.754296 715.22229  38.41318 28.21436
## 20  97.66915 0.000000 -40.62582  40.62582  0.00000
```

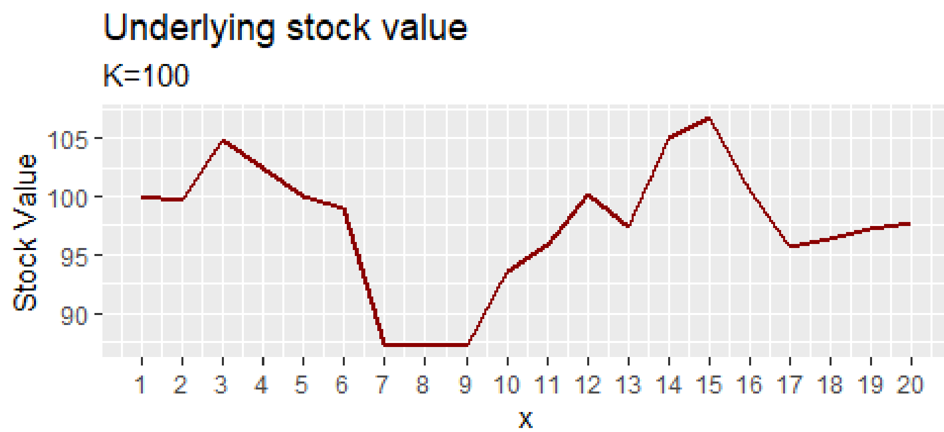
#Deliverables: 2 description for Asset-or-Nothing European call option The value of the replicating portfolio at time T

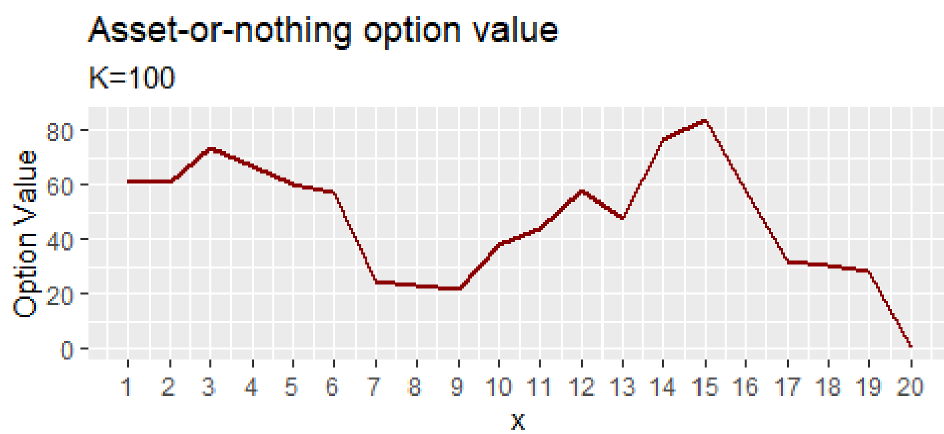
$$C = \triangle S + B$$

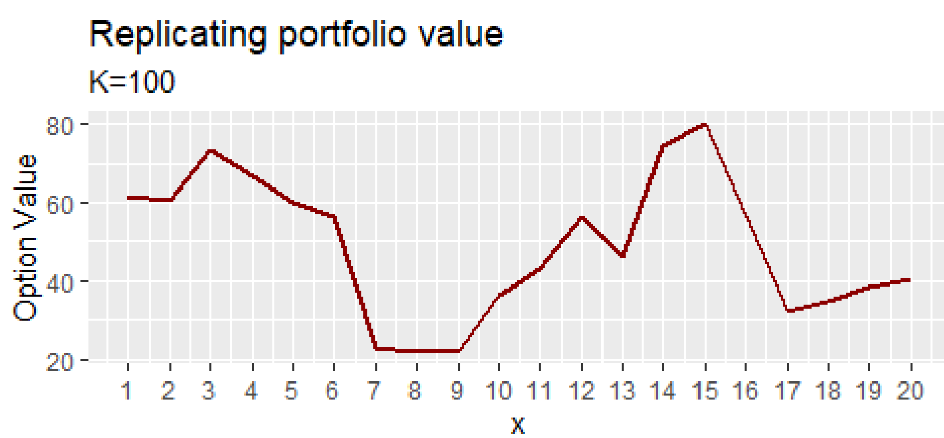
The terminal value of the call at time T

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

#Deliverables: (3-4) plots for Asset-or-Nothing European call option



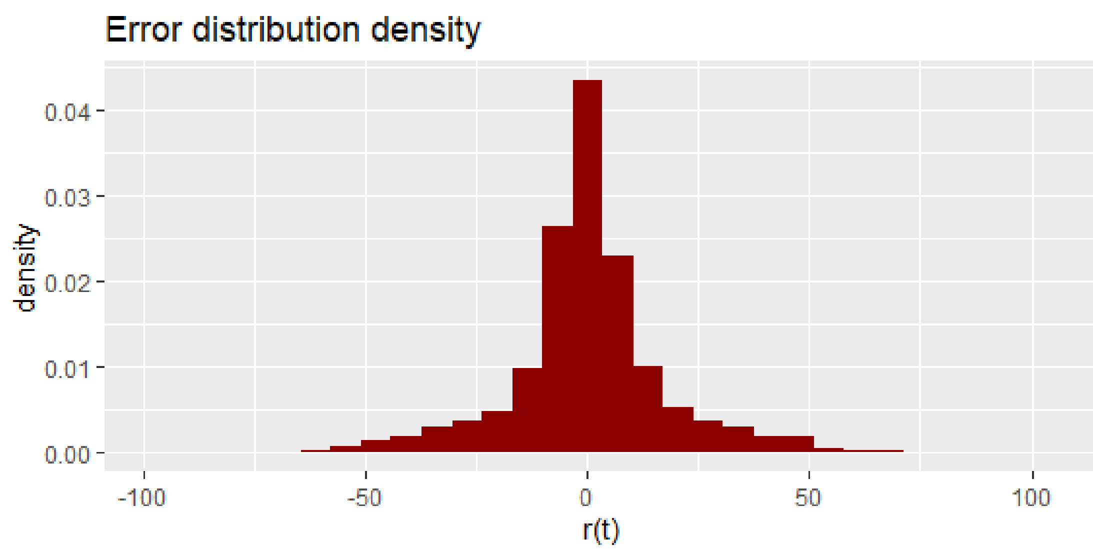


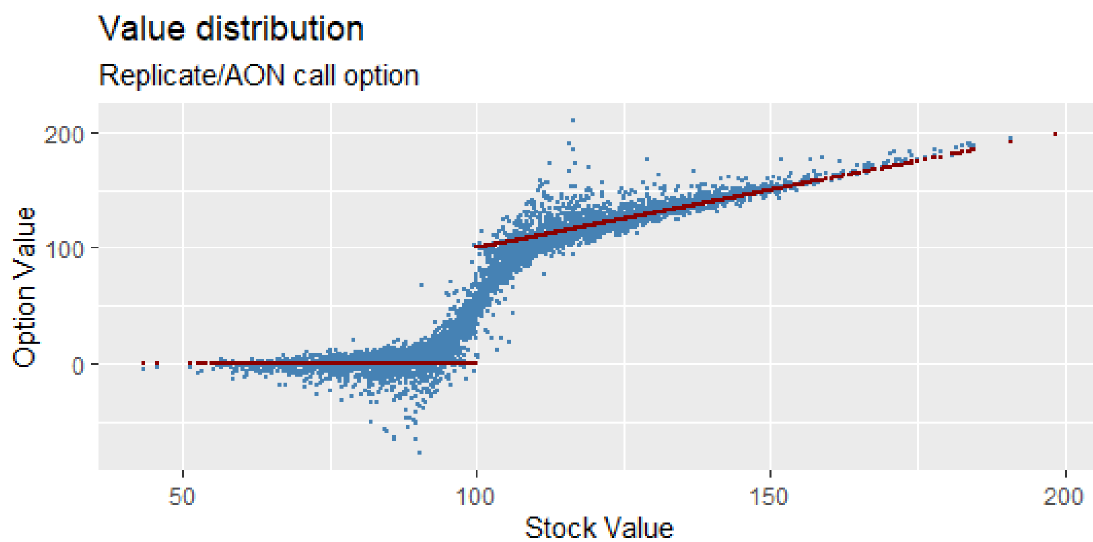


Repeat above process for 5000 times to generate the distribution of replicating error #Deliverables: 1 plot, distribution of relication errors for Asset-or-Nothing call option

```
stock <- call <- replicate <- error <- double(0)
for(i in 1:5000){
  t <- C_AN(S_T(S_0,r,sigma,T),100)
  error[i] <- t$replicate[20]-t$AON[20]
  replicate[i] <- t$replicate[20]
  call[i] <- t$AON[20]
```

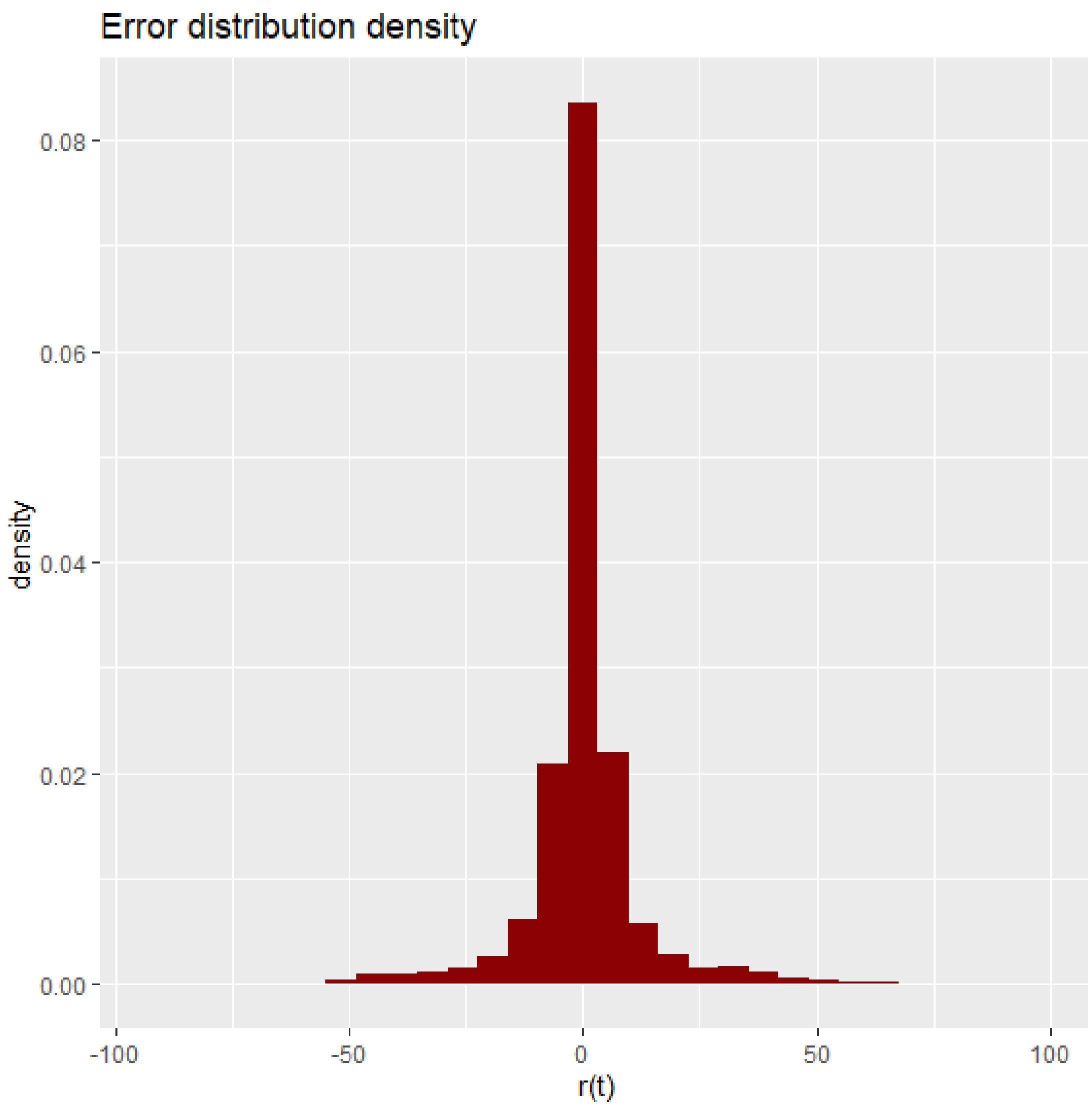
```
  stock[i] <- t$stock[20]
}

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```





Have plotted with n=99, below is the graph

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```


Error distribution density
0.08
0.06
0.04
0.02
0.00
density
-100
-50
0
50
100
r(t)