Homework4

Problem 1

(1)

```
library(Sim.DiffProc)

model.select <- function(data){
  #model 1
  fx1 <- expression(theta[1]*x)
  gx1 <- expression(theta[2]*x^theta[3])
  mod1 <- fitsde(data=as.ts(data),drift=fx1,diffusion=gx1,
                 start = list(theta1=1,theta2=1,theta3=1),pmle="ozaki")
  AIC(mod1)
  #model2
  fx2 <- expression(theta[1]+theta[2]*x)
  gx2 <- expression(theta[3]*x^theta[4])
  mod2 <- fitsde(data=as.ts(data),drift=fx2,diffusion=gx2,
                 start = list(theta1=1,theta2=1,theta3=1,theta4=1),pmle="ozak
i")
  AIC(mod2)
  #model3
  fx3 <- expression(theta[1]+theta[2]*x)
  gx3 <- expression(theta[3]*sqrt(x))
  mod3 <- fitsde(data=as.ts(data),drift=fx3,diffusion=gx3,
                 start = list(theta1=1,theta2=1,theta3=1),pmle="ozaki")

  #model4
  fx4 <- expression(theta[1])
  gx4 <- expression(theta[2]*x^theta[3])
  mod4 <- fitsde(data=as.ts(data),drift=fx4,diffusion=gx4,
                 start = list(theta1=1, theta2=1,theta3=1),pmle="ozaki")

  #model5
  fx5 <- expression(theta[1]*x)
  gx5 <- expression(theta[2]+theta[3]*x^theta[4])
  mod5 <- fitsde(data=as.ts(data),drift=fx5,diffusion=gx5,
                 start = list(theta1=1, theta2=1,theta3=1,theta4=1),pmle="oza
ki")

  ## Computes AIC
  AIC <- c(AIC(mod1),AIC(mod2),AIC(mod3),AIC(mod4),AIC(mod5))
  Test <- data.frame(AIC,row.names = c("mod1","mod2","mod3","mod4","mod5"))
  Bestmod <- rownames(Test)[which.min(Test[,1])]
  cat("best model: ",Bestmod,"\n")
  print(Test)
```

```
}
col1.AIC <- model.select(sample_data$stock1)  #mod3
col2.AIC <- model.select(sample_data$stock2)  #mod1
col3.AIC <- model.select(sample_data$stock3)  #mod4
col4.AIC <- model.select(sample_data$stock4)  #mod3
col5.AIC <- model.select(sample_data$stock5)  #mod1
```

Comment: By using "ozaki" method, we can select the best model for each column,the result shows as follow:

Stock1	Stock2	Stock3	Stock4	Stock5
"mod3"	"mod1"	"mod4"	"mod3"	"mod1"

(2)

```
pmle <- eval(formals(fitsde.default)$pmle)
#model1
fx1 <- expression(theta[1]*x)
gx1 <- expression(theta[2]*x^theta[3])
fitres1 <- lapply(1:4, function(i) fitsde(as.ts(sample_data$stock2),drift=fx1,
diffusion=gx1,pmle=pmle[i],start = list(theta1=1,theta2=1,theta3=1)))
Coef.model1 <- data.frame(do.call("cbind",lapply(1:4,function(i) coef(fitres1
               [[i]]))))
Info1 <- data.frame(do.call("rbind",lapply(1:4,function(i) logLik(fitres1
         [[i]]))), do.call("rbind",lapply(1:4,function(i) AIC(fitres1[[i]]))),
         do.call("rbind",lapply(1:4,function(i) BIC(fitres1[[i]]))), row.name
         s=pmle)
names(Coef.model1) <- c(pmle)
names(Info1) <- c("logLik","AIC","BIC")
Coef.model1 #coefficients of model 1
```

	euler	kessler	ozaki	Shoji
Theta1	5.19156299	0.6069917	1.825129e-05	1.813951e-05
Theta2	-0.03302663	0.9322179	7.829446e-03	7.805915e-03
Theta3	-4.02625030	0.6164141	5.142185e-01	5.147164e-01

```
Info1  #calculate log likelihood function value and AIC,BIC value
```

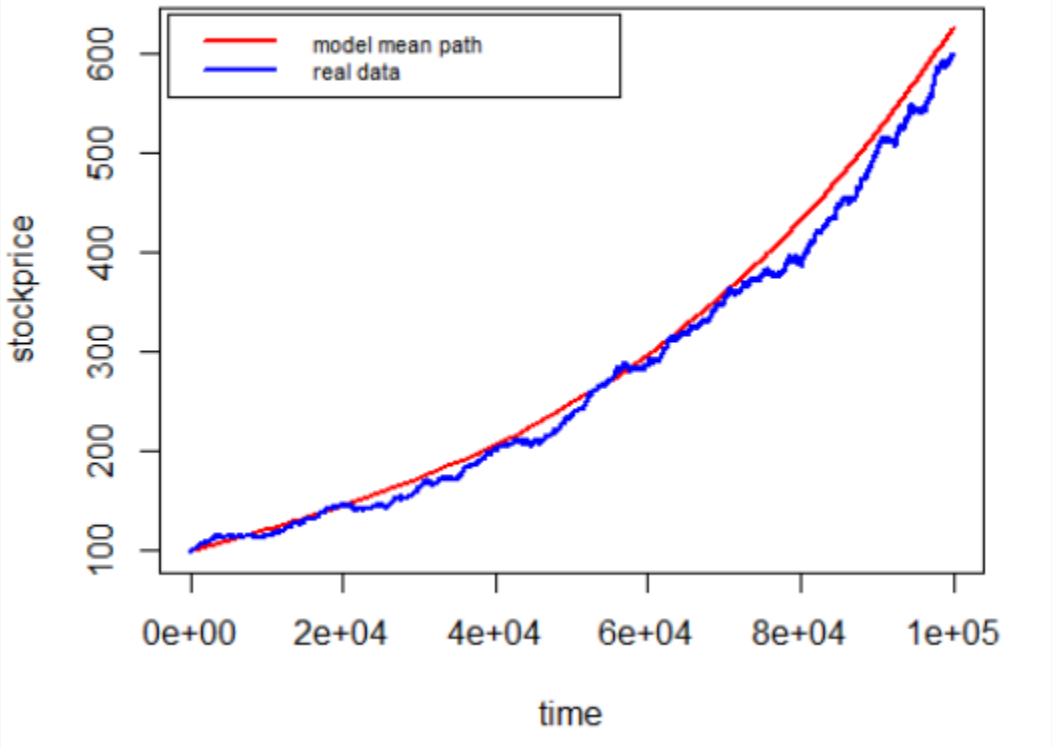
	logLik	AIC	BIC
Euler	0.00	6.0	23.02587
Kessler	0.00	6.0	23.02587
Ozaki	62557.56	-125109.1	-125092.09978
shoji	62565.69	-125125.4	-125108.34471

```
f <- expression(0.00001825*x)
g <- expression(0.007829*x^0.5142)
```

```
mod <- snssde1d(drift=f,diffusion=g,x0=sample_data$stock2[1],M=100, N=length
       (sample_data$stock2),t0 = 1,T = 100001)
mod
```

```
Ito Sde 1D:
        | dX(t) = 1.825e-05 * X(t) * dt + 0.007829 * X(t)^0.5142 * dW(t)
Method:
        | Euler scheme of order 0.5
Summary:
        | Size of process       | N  = 100002.
        | Number of simulation  | M  = 100.
        | Initial value         | x0 = 100.
        | Time of process       | t in [1,100001].
        | Discretization        | Dt = 0.99999.
```

```
plot(time(mod)[-1],mean(mod)[-1],type = "l", col = "red",lwd = 2, xlab = "tim
     e", ylab = "stockprice")
lines(time(mod)[-1],sample_data$stock2,col = "blue",lwd = 2)
legend("topleft",c("model mean path","real data"),inset = .01,col=c("red","bl
      ue"),lwd=2,cex=0.7)
```




```
#model3
fx3 <- expression(theta[1]+theta[2]*x)
gx3 <- expression(theta[3]*sqrt(x))
fitres3 <- lapply(1:4, function(i) fitsde(as.ts(sample_data$stock1),drift=fx3,
diffusion=gx3,pmle=pmle[i],start = list(theta1=1,theta2=1,theta3=1)))
Coef.model3 <- data.frame(do.call("cbind",lapply(1:4,function(i) coef(fitres3
```

```
              [[i]])))
Info2 <- data.frame(do.call("rbind",lapply(1:4,function(i) logLik(fitres3
        [[i]]))), do.call("rbind",lapply(1:4,function(i) AIC(fitres3[[i]]))),
        do.call("rbind",lapply(1:4,function(i) BIC(fitres3[[i]]))), row.name
        s=pmle)
names(Coef.model3) <- c(pmle)
names(Info2) <- c("logLik","AIC","BIC")
Coef.model3 #coefficients of model 3
```

	euler	kessler	ozaki	Shoji
Theta1	0.9813911	1	2.911066e-04	-5.111348e-03
Theta2	0.2786251	1	2.188168e-05	3.854424e-05
Theta3	-13.0835906	1	4.013635e-03	-4.017660e-03

```
Info2 #calculate log likelihood function value and AIC,BIC value
```

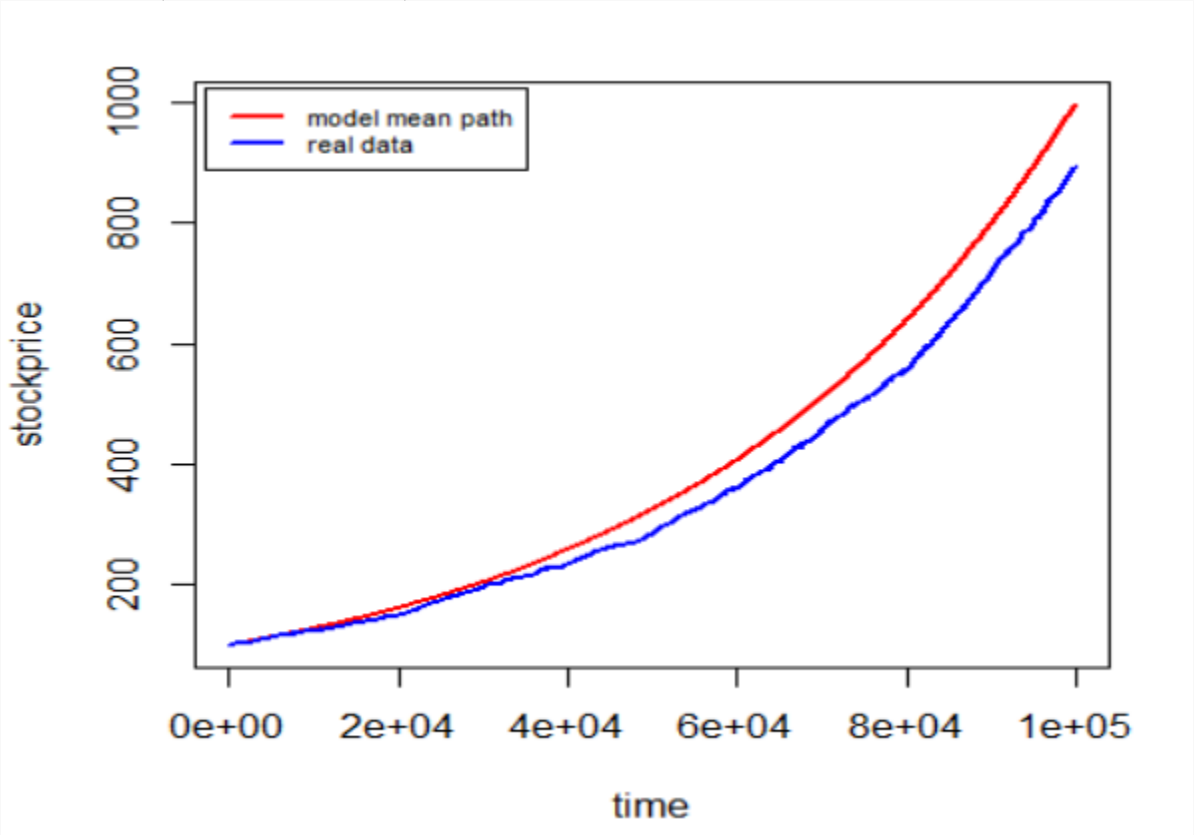
	logLik	AIC	BIC
Euler	0.00	6.0	23.02587
Kessler	0.00	6.0	23.02587
Ozaki	127742.4	-255478.9	-255461.82520
shoji	127637.6	-255269.2	-255252.20795

```
f <- expression( (0.0002911+0.00002188*x) )
g <- expression( 0.004014*sqrt(x) )
mod <- snssde1d(drift=f,diffusion=g,x0=sample_data$stock1[1],M=100, N=length
(sample_data$stock1),t0 = 1,T = 100001)
mod
```

```
Ito Sde 1D:
        | dX(t) = (0.0002911 + 2.188e-05 * X(t)) * dt + 0.004014 * sqrt(X(t)) * dw(t)
Method:
        | Euler scheme of order 0.5
Summary:
        | Size of process       | N  = 100002.
        | Number of simulation  | M  = 100.
        | Initial value         | x0 = 100.
        | Time of process       | t in [1,100001].
        | Discretization        | Dt = 0.99999.
```

```
plot(time(mod)[-1],mean(mod)[-1],type = "l", col = "red",lwd = 2, xlab = "tim
     e", ylab = "stockprice")
lines(time(mod)[-1],sample_data$stock1,col = "blue",lwd = 2)
legend("topleft",c("model mean path","real data"),inset = .01,col=c("red","bl
```

```
        ue"),lwd=2,cex=0.7)
```




```
#model4
fx4 <- expression(theta[1])
gx4 <- expression(theta[2]*x^theta[3])
fitres4 <- lapply(1:4, function(i) fitsde(as.ts(sample_data$stock3),drift=fx4,
diffusion=gx4,pmle=pmle[i], start = list(theta1=1,theta2=1,theta3=1)))
Coef.model4 <- data.frame(do.call("cbind",lapply(1:4,function(i) coef(fitres4
               [[i]]))))
Info3 <- data.frame(do.call("rbind",lapply(1:4,function(i) logLik(fitres4
         [[i]]))), do.call("rbind",lapply(1:4,function(i) AIC(fitres4[[i]]))),
         do.call("rbind",lapply(1:4,function(i) BIC(fitres4[[i]]))), row.name
         s=pmle)
names(Coef.model4) <- c(pmle)
names(Info3) <- c("logLik","AIC","BIC")
Coef.model4 #coefficients of model 4
```

	euler	kessler	ozaki	Shoji
Theta1	-0.1684631	-0.022723635	1	1
Theta2	-0.0434471	0.006183543	1	1
Theta3	0.1308275	0.691080989	1	1

```
Info3 #calculate log likelihood function value and AIC,BIC value
```

	logLik	AIC	BIC
Euler	0.00	6.00	23.02587
Kessler	-23402.22	46810.44	46827.46413
Ozaki	0.00	6.00	23.02587
shoji	0.00	6.00	23.02587

```
f <- expression(1)
g <- expression(1*x^1)
mod <- snssde1d(drift=f,diffusion=g,x0=sample_data$stock3[1],M=100, N=length
       (sample_data$stock3),t0 = 1, T = 100001)
mod
```

```
Ito Sde 1D:
        | dX(t) = 1 * dt + 1 * X(t)^1 * dW(t)
Method:
        | Euler scheme of order 0.5
Summary:
        | Size of process       | N  = 100002.
        | Number of simulation  | M  = 100.
        | Initial value         | x0 = 100.
        | Time of process       | t in [1,100001].
        | Discretization        | Dt = 0.99999.
```

```
plot(time(mod)[-1],mean(mod)[-1],type = "l", col = "red",lwd = 2, xlab = "tim
     e", ylab = "stockprice")
lines(time(mod)[-1],sample_data$stock3,col = "blue",lwd = 2)
legend("topleft",c("model mean path","real data"),inset = .01,col=c("red","bl
      ue"),lwd=2,cex=0.7)
```

Comment: Because it takes too long to knit r-markdown, so I input the results by hand. Because I choose model 1,3,4 in question (1), I only estimate parameter for this three model.

From the coefficient table for every model, we can observe that the coefficients of euler and Kessler are very similar, Also, the coefficients of ozaki and shoji are very similar.

I input coefficient of ozaki method to the formula and simulate the stock price. From the plot for each model we can see that simulation value is very close to the real value.

(3)

Comment: From the coefficient, the best method always be the method that you choose in the model select at question(1).

So here the ozaki model has the maximum log-likelihood function value and minimum AIC and BIC. I choose ozaki be the best model.(but some times AIC value become very weird so that we cannot make exactly conclusion)

Problem 2

(1)

```
data <- read.csv("2017_2_15_mid.csv")

#choose five year col to finish the question
num<-seq(from=1,to=nrow(data),by=2)
sigma.mkt <- data$X1Yr[num]/100
strike.mkt <- data$X1Yr[-num]/100

beta <- 0.5 #beta parameter
F <- strike.mkt #spot price equals to strike price because at the money
K <- strike.mkt
T <- 1

sigma.ATM <- function(x){

  A = (1-beta)^2/24*x[1]^2/F^(2-2*beta)
  B = 1/4*x[2]*beta*x[3]*x[1]/F^(1-beta)
  C = (2-3*x[2]^2)/24*x[3]^2
  term1 = x[1]*(1 + (A + B + C)*T)
  term2 = F^(1-beta)

  return(term1/term2)
}

SSE <- function(x){
  return(sum((sigma.mkt-sigma.ATM(x))^2))
}

library(stats)
x0 <- c(1,0,0)
opt1 <- optim(x0,SSE)
Alpha1 <- opt1$par[1]
Rho1 <- opt1$par[2]
nu1 <- opt1$par[3] #vol of vol
data.frame(Alpha1,Rho1,nu1)

##      Alpha1      Rho1       nu1
## 1 0.1058173 0.5858817 0.1701602
```

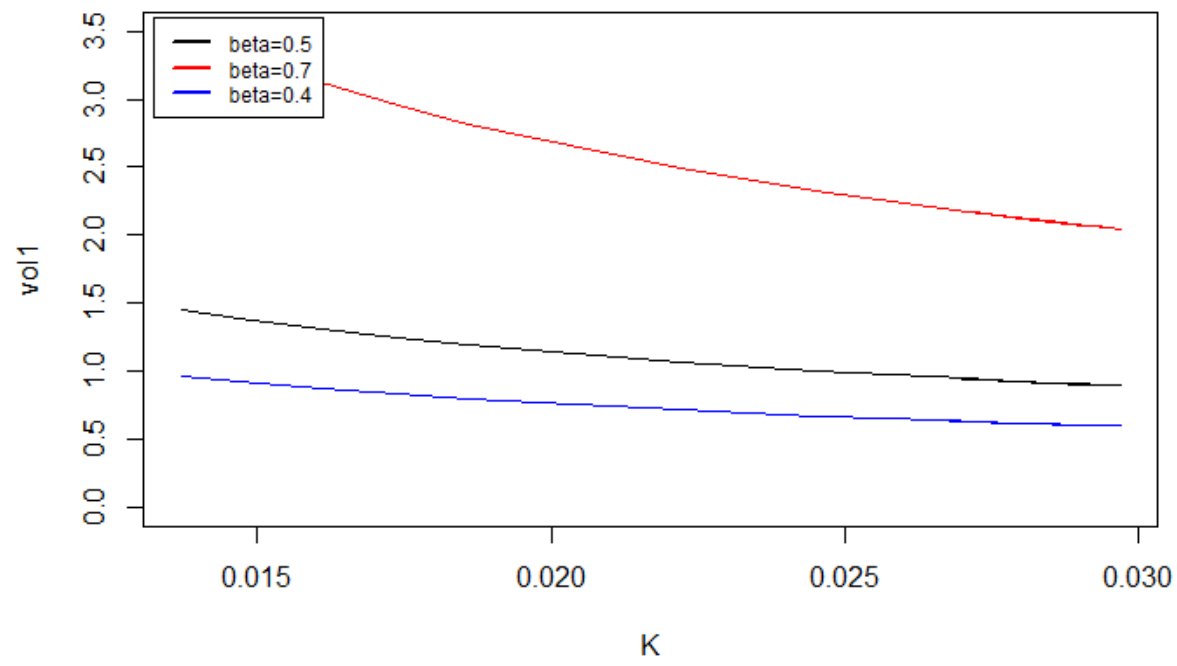
(2)

```
beta <- 0.7
opt2 <- optim(x0,SSE)
Alpha2 <- opt2$par[1]
Rho2 <- opt2$par[2]
nu2 <- opt2$par[3] #vol of vol
data.frame(Alpha2,Rho2,nu2)
```

```
##      Alpha2       Rho2         nu2
## 1 0.2346604 0.05334457 -0.02053435

beta <- 0.4
opt3 <- optim(x0,SSE)
Alpha3 <- opt3$par[1]
Rho3 <- opt3$par[2]
nu3 <- opt3$par[3] #vol of vol
data.frame(Alpha3,Rho3,nu3)

##       Alpha3      Rho3       nu3
## 1 0.07144836 0.5727171 0.1876424
```

(3)

```
vol1 <- sigma.ATM(opt1$par)
plot(K,vol1,type = "l",ylim = c(0,1))

vol2 <- sigma.ATM(opt2$par)
lines(K,vol2,col = "red")

vol3 <- sigma.ATM(opt3$par)
lines(K,vol3,col = "blue")
legend("topleft",c("beta=0.5","beta=0.7","beta=0.4"),inset = .01,
       col=c("black","red","blue"),lwd=2,cex=0.75)
```



Comment: From the plot, we can observe that the choice of beta has little effect on the resulting shape of the volatility curve

produced by the SABR model, but volatility become bigger when beta become bigger.

(4)

```
#show the minimum object function value for beta=0.5,0.7,0.4
data.frame(opt1$value,opt2$value,opt3$value)

##   opt1.value opt2.value opt3.value
## 1   1.643249   1.224507   1.878109
```

Comment: From the result table above, we know that we can obtain minimum value for the function when beta=0.7

(5)

```
sigma.mkt <- data$X20Yr[num]/100
strike.mkt <- data$X20Yr[-num]/100
beta <- 0.7 #beta parameter
F <- strike.mkt #spot price equals to strike price because at the money
K <- strike.mkt
T <- 20

x <- c(Alpha2,Rho2,nu2)
cat("estimate volatility: ",sigma.ATM(x),"\n")

## estimate volatility:  0.7205645 0.7188378 0.7171285 0.7154363 0.7137609 0.
7088327 0.7056261 0.7040457 0.7024802 0.7024802 0.7024802 0.7024802 0.7032611
 0.7040457 0.706422 0.7104594 0.7188378 0.7249594 0.7322347

cat("market volatility; ",sigma.mkt)

## market volatility;  0.7312 0.7747 0.8439 0.8208 0.8226 0.8418 0.8263 0.813
5 0.8339 0.7835 0.7929 0.7478 0.7256 0.7286 0.6845 0.6202 0.5591 0.6161 0.609
2

sum((sigma.mkt-sigma.ATM(x))^2) #calculate SSE between estimate and market vo
latility

## [1] 0.1827733
```

Comment: we choose beta=0.7 from the analysis in question(4), then we calculate at-the-money volatility using the parameters I obtained and compare the numbers I obtain to the benchmark values and observe these two are very similar to each other.

Problem 3

(1)

```
i <- sqrt(-1+0i)
S0 <- 1
```

```r
r <- 0.05
T <- 30/252
sigma <- 0.2

alpha <- 3
lambda <- 0.001
N <- 1024
eta <- 2*pi/N/lambda


#character function
chara_fun <- function(u){
  exp(i*(log(S0)+(r-sigma^2/2)*T)*u - 0.5*sigma^2*T*u^2)
}

psi_fun <-function(nu){
  term1 = exp(-r*T)*chara_fun(nu-(alpha+1)*i)
  term2 = alpha^2+alpha-nu^2+i*(2*alpha+1)*nu
  term1/term2
}

#delta function
Kronecker.delta <- function(x){
  if(x==0)
    return(1)
  else
    return(0)
}

#price option
call_price <- function(u){
  #substitute some parameters
  a = N*eta
  b = 1/2*N*lambda #strike level from -b to b
  ku = -b+lambda*(u-1)

  term = c()
  #Using the trapezoid rule for the integral
  for(j in 1:N){
    nu.j = eta*(j-1)
    term1 = exp(-i*2*pi/N*(j-1)*(u-1)) * exp(i*b*nu.j)
    term2 = psi_fun(nu.j) * eta/3 * (3+(-1)^j-Kronecker.delta(j-1))
    term = c(term,term1*term2)
  }
  return(Re(exp(-alpha*ku)/pi * sum(term)))
}
#calculate option price for k=0, which is K=1,u=513
cat("option price by FFT: ",call_price(513),"\n")

## option price by FFT:  0.03050403
```

```r
#calculate European call option price by BS model
library(RQuantLib)
cat("option price by BS: ",EuropeanOption("call", underlying = 1, strike = 1,
                                          dividendYield = 0, riskFreeRate =
                            0.05, maturity = 30/252, volatility = 0.2)$value)

## option price by BS:  0.03056762
```

Comment: option price by FFT: 0.03050403

option price by BS: 0.03056762

(2) bonus

```r
S0 <- 1
K <- c(0.5,0.75,1,1.25,1.5)
r <- 0
T <- 5
div <- 0
sigma.nu <- 0.2
k <- log(K)

alpha <- 2
lambda <- 0.01
N <- 1024
eta <- 2*pi/N/lambda

mu <- 0.1
nu0 <- 0.1
kappa <- (4+2+1)/3
rho <- -0.3

#character function
chara_fun_heston <- function(u){
  zeta <- -(u^2+i*u)/2
  gamma <- kappa - i*rho*sigma.nu*u
  theta <- sqrt(gamma^2-2*sigma.nu^2*zeta)
  A = i*u*(log(S0)+(r-div)*T)
  B = 2*zeta*(1-exp(-theta*T))/(2*theta-(theta-gamma)*(1-exp(-theta*T)))
  C = 2*log((2*theta-(theta-gamma)*(1-exp(-theta*T)))/(2*theta))
  E = (theta-gamma)*T
  H = kappa*mu/sigma.nu^2*(C + E)
  exp(A + B*nu0 - H)
}

psi_fun_heston <-function(nu){
  term1 = exp(-r*T)*chara_fun_heston(nu-(alpha+1)*i)
  term2 = alpha^2+alpha-nu^2+i*(2*alpha+1)*nu
  term1/term2
}
```

```r
#price option
call_price_heston <- function(u){
  #substitute some parameters
  a = N*eta
  b = N*lambda/2 #strike level from -b to b
  ku = -b+lambda*(u-1)
  #u =-(k+b)/lambda+1

  term <- c()
  #Using the trapezoid rule for the integral
  for(j in 1:N){
    nu.j = eta*(j-1)
    term1 = exp(-i*2*pi/N*(j-1)*(u-1)) * exp(i*b*nu.j)
    term2 = psi_fun_heston(nu.j) * eta/3 * (3+(-1)^j-Kronecker.delta(j-1))
    term = c(term,term1*term2)
  }
  return(Re(exp(-alpha*ku)/pi * sum(term)))
}
b = N*lambda/2 #strike level from -b to b
u <- (k+b)/lambda+1
heston.FFT <- c()
for(a in 1:5){
  heston.FFT <- c(heston.FFT, call_price_heston(u[a]))
}
heston.BS <- c(0.5315591,0.3771046,0.2655691,0.1859287,0.1284557)
heston.price <- c(0.545298,0.387548,0.275695,0.197629,0.143341)
DF <- data.frame(heston.FFT,heston.BS,heston.price,
                 row.names = c("k=0.25","k=0.5","k=1","k=1.25","k=1.5"))
DF
```

```
##        heston.FFT heston.BS heston.price
## k=0.25  0.5426880 0.5315591     0.545298
## k=0.5   0.3853756 0.3771046     0.387548
## k=1     0.2742748 0.2655691     0.275695
## k=1.25  0.1969069 0.1859287     0.197629
## k=1.5   0.1429345 0.1284557     0.143341
```

Comment: the first column is the results of FFT method, the second column is the results of BS method, the third column is the results of the paper. We can see that these option prices are very similar to each other