Problem 2

Q1

In [2]:
```
library(quantmod)
library(fpp)
```

```
Warning message:
"package 'quantmod' was built under R version 3.6.2"
Loading required package: xts

Warning message:
"package 'xts' was built under R version 3.6.2"
Loading required package: zoo


Attaching package: 'zoo'


The following objects are masked from 'package:base':

    as.Date, as.Date.numeric


Loading required package: TTR

Warning message:
"package 'TTR' was built under R version 3.6.2"
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo

Version 0.4-0 included new data defaults. See ?getSymbols.

Warning message:
"package 'fpp' was built under R version 3.6.2"
Loading required package: forecast

Warning message:
"package 'forecast' was built under R version 3.6.2"
Registered S3 methods overwritten by 'forecast':
  method             from
  fitted.fracdiff    fracdiff
  residuals.fracdiff fracdiff

Loading required package: fma

Warning message:
"package 'fma' was built under R version 3.6.2"
Loading required package: expsmooth

Warning message:
"package 'expsmooth' was built under R version 3.6.2"
Loading required package: lmtest

Loading required package: tseries

Warning message:
"package 'tseries' was built under R version 3.6.2"
```

In [3]:
```
getSymbols("SPY",from='2015-01-01',to='2020-02-22',src="yahoo")
dim(SPY)
```

```
'getSymbols' currently uses auto.assign=TRUE by default, but will
use auto.assign=FALSE in 0.5-0. You will still be able to use
'loadSymbols' to automatically load data. getOption("getSymbols.env")
and getOption("getSymbols.auto.assign") will still be checked for
alternate defaults.

This message is shown once per session and may be disabled by setting
options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.


Warning message:
"'indexClass<-' is deprecated.
Use 'tclass<-' instead.
See help("Deprecated") and help("xts-deprecated")."
```

'SPY'

1293 · 6

From the dimention, we can see that this time series have 1293 days and later we will seperate it into a training set(contains 1193 observations) and a test set (contains 100 observations).

In [3]:
```
head(SPY)
```

	SPY.Open	SPY.High	SPY.Low	SPY.Close	SPY.Volume	SPY.Adjusted
2015-01-02	206.38	206.88	204.18	205.43	121465900	186.1593
2015-01-05	204.17	204.37	201.35	201.72	169632600	182.7974
2015-01-06	202.09	202.72	198.86	199.82	209151400	181.0756
2015-01-07	201.42	202.72	200.88	202.31	125346700	183.3320
2015-01-08	204.01	206.16	203.99	205.90	147217800	186.5852
2015-01-09	206.40	206.42	203.51	204.25	158567300	185.0900

In [4]:
```
tail(SPY)
```

	SPY.Open	SPY.High	SPY.Low	SPY.Close	SPY.Volume	SPY.Adjusted
2020-02-13	335.86	338.12	335.56	337.06	54501900	337.06
2020-02-14	337.51	337.73	336.20	337.60	64582200	337.60
2020-02-18	336.51	337.67	335.21	336.73	57226200	336.73
2020-02-19	337.79	339.08	337.48	338.34	48814700	338.34
2020-02-20	337.74	338.64	333.68	336.95	74163400	336.95
2020-02-21	335.47	335.81	332.58	333.48	113788200	333.48

In [5]:
```
data=SPY[,6]
```

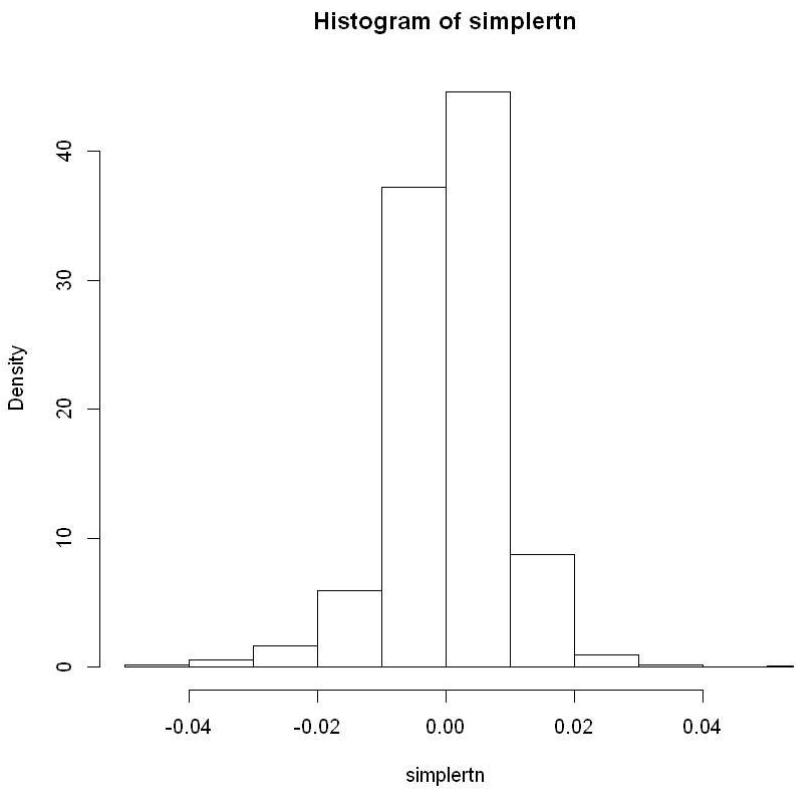
In [6]:
```
logrtn=dailyReturn(data,type="log")
simplertn=dailyReturn(data)
```

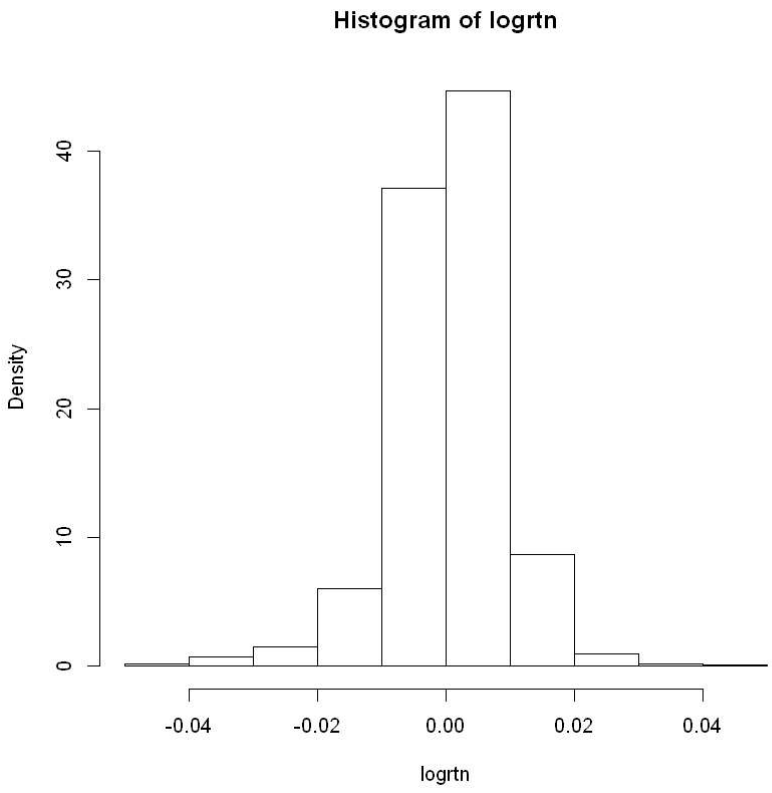
In [7]:
```
adf.test(logrtn)
```

```
Warning message in adf.test(logrtn):
"p-value smaller than printed p-value"

        Augmented Dickey-Fuller Test

data:  logrtn
Dickey-Fuller = -11.731, Lag order = 10, p-value = 0.01
alternative hypothesis: stationary
```

We can see from the outcome of adftest that the p-value=0.01<0.05, so the logrtn time series is stationary at the confidence 95%.

In [8]:
```
hist(simplertn, freq=F,xlim=c(-0.05,0.05))
hist(logrtn,freq=F,xlim=c(-0.05,0.05))
par(mfrow=c(2,1))
```





For this time series, we use log to make it stationary.

In [9]:
```
data1=logrtn[1:1193]
auto.arima(data1)
```

```
Series: data1
ARIMA(1,0,1) with non-zero mean

Coefficients:
         ar1      ma1   mean
      0.9622  -0.9832  4e-04
s.e.  0.0203   0.0135  1e-04

sigma^2 estimated as 7.346e-05:  log likelihood=3986.57
AIC=-7965.14   AICc=-7965.11   BIC=-7944.81
```

In [10]:
```
m1=arima(data1,order=c(1,0,1))
```

From the auto.arima, we get that our model is arima(1,0,1). This is m1.

In [47]:
```
count1=0
for (i in 1:100)
{

data=logrtn[i:1192+i,]

model=arima(data,order=c(1,0,1))

pre=predict(model,1)[1]

if(as.numeric(pre)*(as.numeric(logrtn[1193+i,1]))>0){
     count1=count1+1
    }
}
```

```
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
```

```
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
Warning message in arima(data, order = c(1, 0, 1)):
"possible convergence problem: optim gave code = 1"
```

In [48]:
```
count1
```

49

In [49]:
```
acc1=count1/100
acc1
```

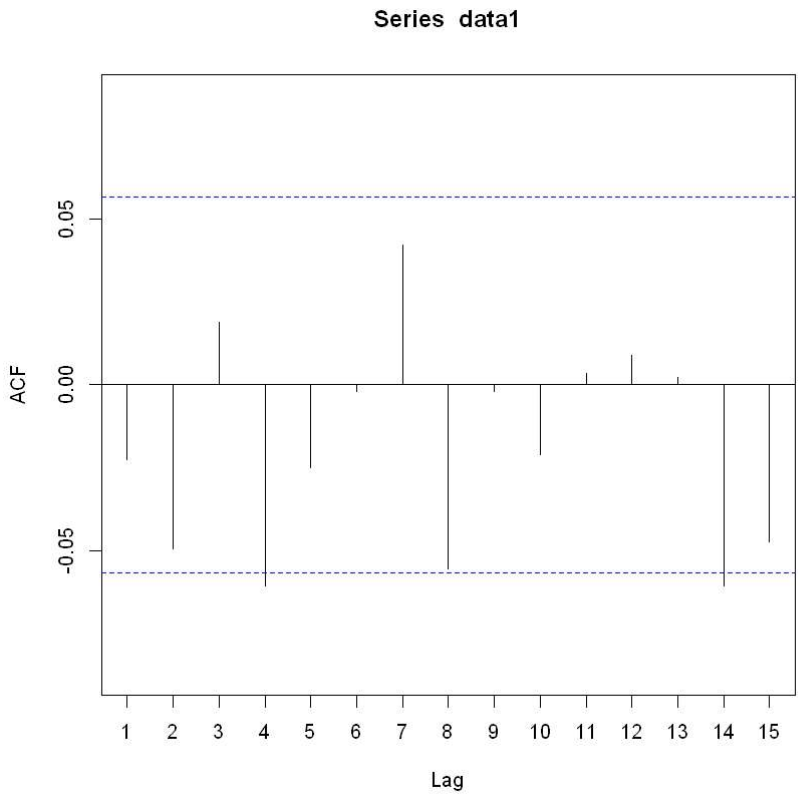
0.49

m1=arima(1,0,1) has an accuracy of 0.49.

Q2

MA model

In [14]:
```
Acf(data1,lag=15)
```



From the ACF plot, we get two significant order, one is 4, the other is 14. We will compare two models with AIC and error criterion.

In [15]:
```
Box.test(logrtn,lag=10)

Box.test(logrtn,lag=10,type="Ljung")
```

```
        Box-Pierce test

data:  logrtn
X-squared = 14.892, df = 10, p-value = 0.1361

        Box-Ljung test

data:  logrtn
X-squared = 14.975, df = 10, p-value = 0.133
```

In [16]:
```
m2=arima(data1,c(0,0,4))
```

In [17]:
```
m2
```

```
Call:
arima(x = data1, order = c(0, 0, 4))

Coefficients:
          ma1      ma2     ma3      ma4  intercept
      -0.0247  -0.0531  0.0216  -0.0734      4e-04
s.e.   0.0289   0.0291  0.0295   0.0310      2e-04

sigma^2 estimated as 7.312e-05:  log likelihood = 3987.87,  aic = -7963.73
```

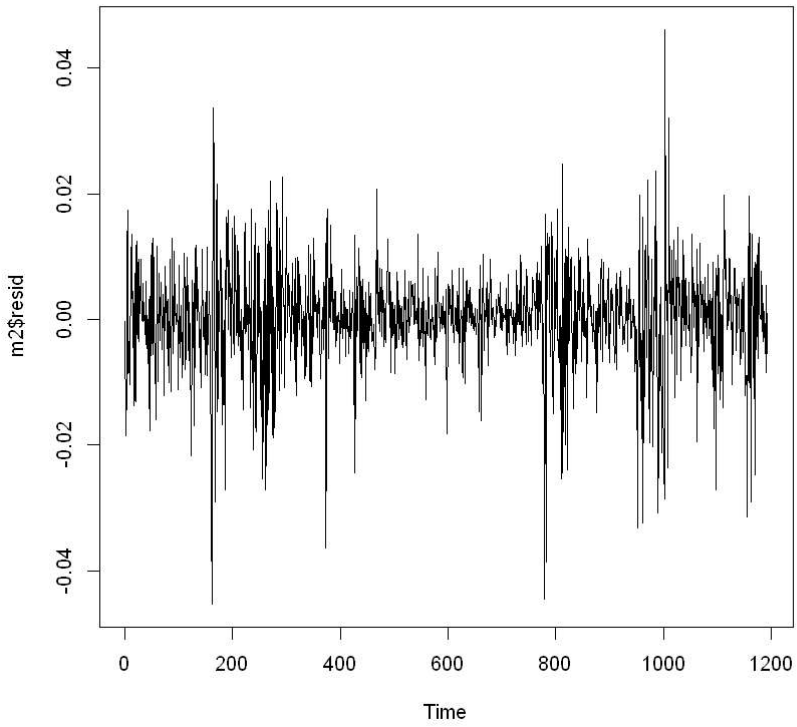
In [18]:
```
Box.test(m2$residuals,lag=10,type="Ljung")
```

```
        Box-Ljung test

data:  m2$residuals
X-squared = 7.35, df = 10, p-value = 0.692
```

In [19]:
```
res2=m2$resid
res2=res2[!is.na(res2)]
sd(res2)
```

0.00855488918440655

In [20]:
```
plot(m2$resid)
```



In [21]:
```
m2test=arima(data1,c(0,0,14))
```

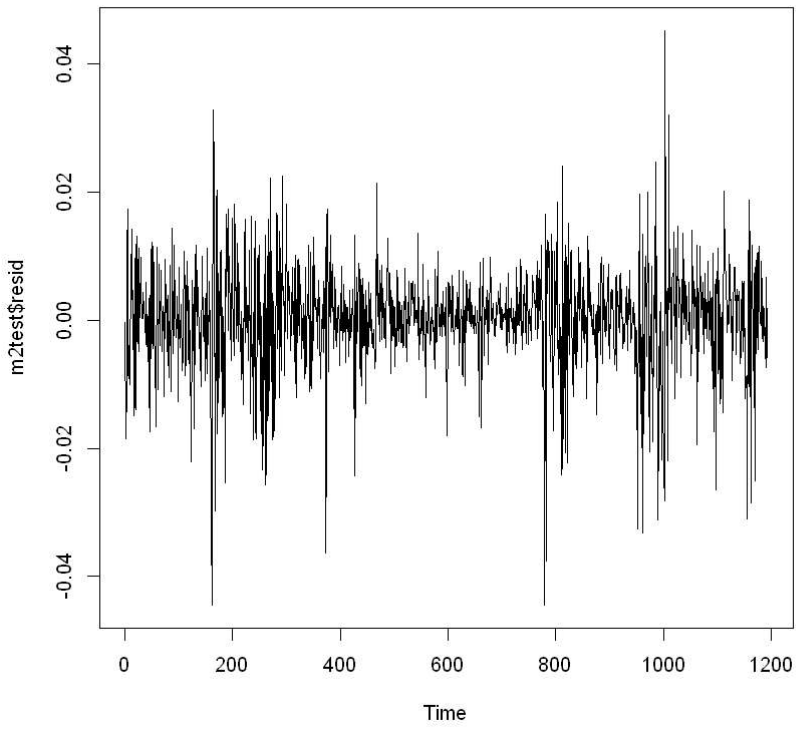
In [22]:
```
m2test
```

```
Call:
arima(x = data1, order = c(0, 0, 14))

Coefficients:
         ma1      ma2     ma3      ma4      ma5      ma6     ma7      ma8
      -0.028  -0.0454  0.0275  -0.0675  -0.0320  -0.0080  0.0519  -0.0512
s.e.   0.029   0.0291  0.0294   0.0297   0.0294   0.0297  0.0297   0.0284
          ma9     ma10     ma11    ma12    ma13     ma14  intercept
      -0.0094  -0.0263  -0.0042  0.0203  0.0010  -0.0718      4e-04
s.e.   0.0288   0.0298   0.0311  0.0293  0.0296   0.0305      2e-04

sigma^2 estimated as 7.229e-05:  log likelihood = 3994.61,  aic = -7957.21
```

In [23]:
```
res2test=m2test$resid
res2test=res2test[!is.na(res2test)]
sd(res2test)
```

0.0085062038048902

In [24]:
```
plot(m2test$resid)
```



From the AIC and standard variance, we can see that MA(4) is better than MA(14). So we choose q=4 and m2=MA(4).

In [26]:
```
count2=0
for (i in 1:100)
{

data=logrtn[i:1192+i,]

model=arima(data,order=c(0,0,4))

pre=predict(model,1)[1]

if(as.numeric(pre)*(as.numeric(logrtn[1193+i,1]))>0){
     count2=count2+1
    }
}
```

In [27]:
```
count2
```

54

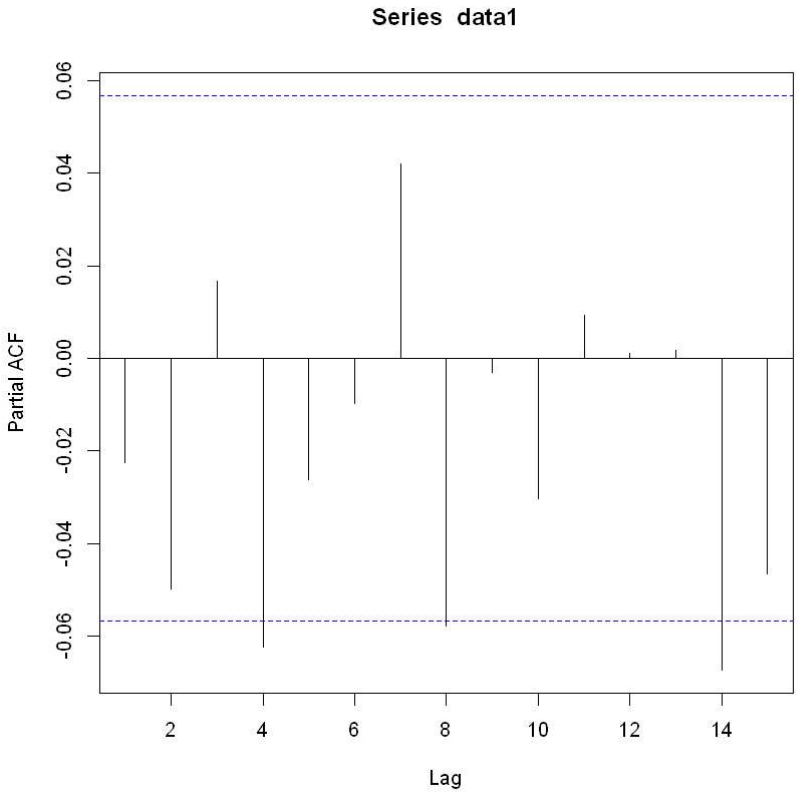
In [28]:
```
acc2=count2/100
acc2
```

0.54

m2=MA(4) has an accuracy of 0.54.

AR model

In [29]:
```
pacf(data1,lag=15)
```



From the pacf, we can see that order=4 might be the order of AR. Besides, we will use other functions to help us decide the best order.

In [30]:
```
m3=ar(as.vector(data1), method="mle")
```

In [31]:
```
m3
```

```
Call:
ar(x = as.vector(data1), method = "mle")

Coefficients:
      1        2        3        4
-0.0218  -0.0526   0.0152  -0.0626

Order selected 4  sigma^2 estimated as  7.32e-05
```

The ar function will automatically fit an AR model using AIC criterion. And the order it selected is 4.

In [32]:
```
m3=arima(data1,c(4,0,0))
m3
```

```
Call:
arima(x = data1, order = c(4, 0, 0))

Coefficients:
          ar1      ar2     ar3      ar4  intercept
      -0.0218  -0.0526  0.0151  -0.0627      4e-04
s.e.   0.0289   0.0289  0.0289   0.0290      2e-04

sigma^2 estimated as 7.319e-05:  log likelihood = 3987.3,  aic = -7962.61
```

In [35]:
```
count3=0
for (i in 1:100)
{

data=logrtn[i:1192+i,]

model=arima(data,order=c(4,0,0))

pre=predict(model,1)[1]

if(as.numeric(pre)*(as.numeric(logrtn[1193+i,1]))>0){
     count3=count3+1
    }
}
```

In [36]:
```
acc3=count3/100
acc3
```

0.55

As we can see, acc1=0.49, acc2=0.54, acc3=0.55, so we choose m3=AR(4) as our final model.

Q3

In [37]:
```
data=logrtn[1:1193,]

model=m3

pre=predict(model,1)[1]
```

In [38]:
```
pre
```

$pred = A Time Series:
0.00124209852240935

In [39]:
```
s=100000
for (i in 1:100)
{

data=logrtn[i:1192+i,]

model=arima(data,order=c(4,0,0))

pre=predict(model,1)[1]

if(as.numeric(pre)>0){
     s=s*(1+as.numeric(simplertn[1193+i]))
    }
}
```

In [40]:
```
s
```

106803.970424357

Q4

In [41]:
```
mean=mean(logrtn[1:1193])
mean
```

0.000382919523466322

In [42]:
```
s1=100000
for (i in 1:100)
{

data=logrtn[i:1192+i,]

model=arima(data,order=c(4,0,0))

pre=predict(model,1)[1]


if(as.numeric(pre)>mean){
     s1=s1*(1+as.numeric(simplertn[1193+i]))
    }
}
```

In [43]:
```
s1
```

104927.815377925

Q5

In [44]:
```
mean1=mean(logrtn[1:1193])+0.25*sd(logrtn[1:1193])
mean1
```

0.00253039405171044

In [45]:
```
s2=100000
for (i in 1:100)
{

data=logrtn[i:1192+i,]

model=arima(data,order=c(4,0,0))

pre=predict(model,1)[1]

if(as.numeric(pre)>mean1){
     s2=s2*(1+as.numeric(simplertn[1193+i]))
    }
}
```

In [46]:
```
s2
```

1e+05

Q6

In [16]:
```
try1=0.75*mean(logrtn[1:1193])
try1
```

0.000287189642599741

In [44]:
```
n=1
i=0
s=vector()
while(i<1){
    s3=100000
    try=i*mean(logrtn[1:1193])
    for (j in 1:100){
        data=logrtn[j:1192+j,]
        model=arima(data,order=c(4,0,0))
        pre=predict(model,1)[1]

        if(as.numeric(pre)>try){
            s3=s3*(1+as.numeric(simplertn[1193+j]))
            }
    }
    s=c(s,s3)
    n=n+1
    i=i+0.1
}
```

In [45]:
```
s
```

106803.970424357 · 106803.970424357 · 106150.873439916 · 104397.198249337 · 103838.837710216 · 106244.553473954 · 105915.323519614 · 105431.207545448 · 104749.539121005 · 105420.231477863 · 104927.815377925

Firstly, we try to write a loop to find out at what level of mean we can get the best return. And we can get above output. As we can see, when pre>0 or pre>0.1 * mean, the returns are best.So this might be the strategy to find one criterion related to mean. The reason why we don't contain sd is that the standard variance is not at the same quantative level with mean.

In [28]:
```
s4=100000
for (i in 1:100)
{

data=logrtn[(892+i):(1192+i),]

model=arima(data,order=c(4,0,0))

pre=predict(model,1)[1]

if(as.numeric(pre)>0){
     s4=s4*(1+as.numeric(simplertn[1193+i]))
    }
}
```

In [29]:
```
s4
```

106762.247409162

Secondly, we use the past 300 days to predict and when the pre is bigger than 0, we long. From the output we can see that our return is good. The rational is that the return might more related to recent data so this might be better for our forecast.