AMS586 Project

Yunlong Pan 113061415

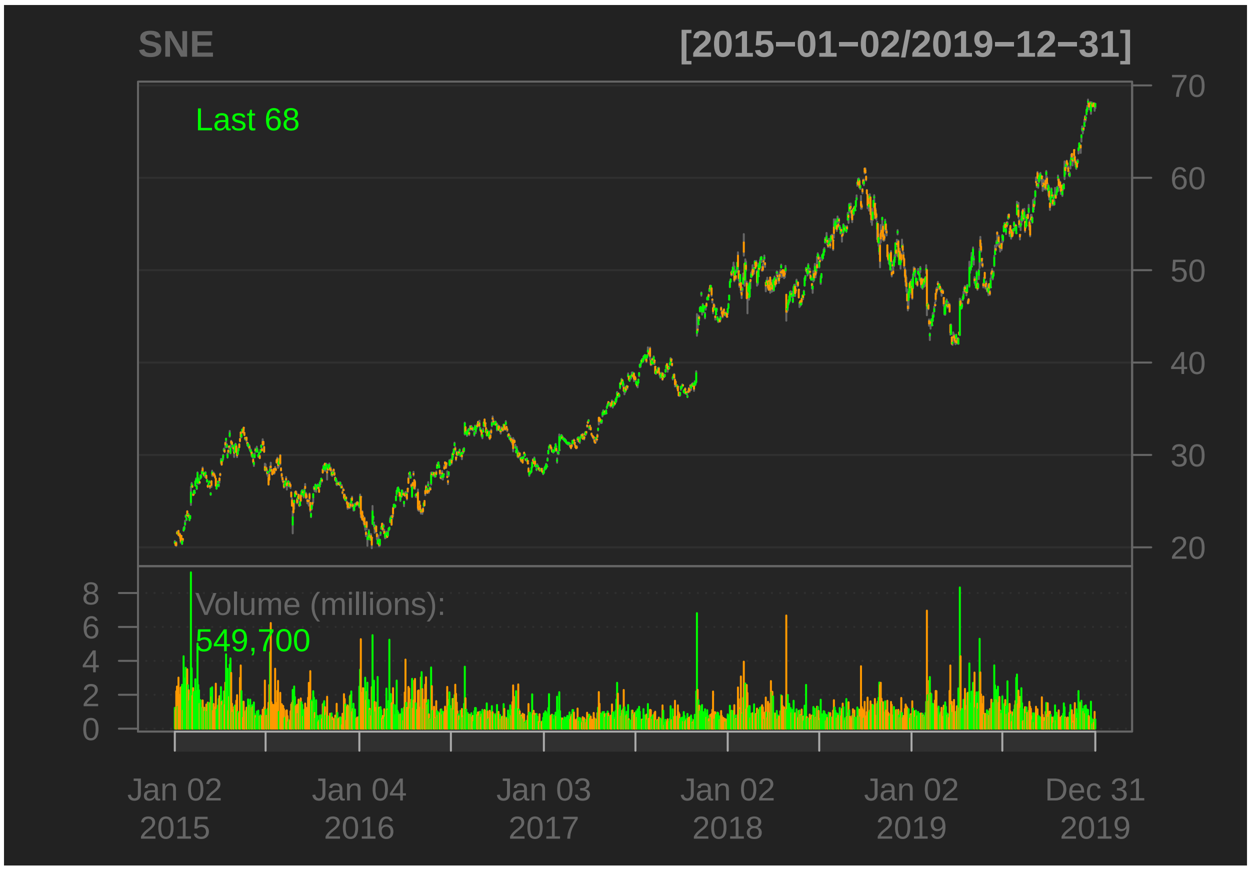
5/9/2020

Report for Fitting a Time Series Model

1. Introduction

In this report, I choose the stock price of Sony Corporation (SNE) as my data to analyze. The time period for the data is from Jan 2015 to Dec 2019. The total number of data points is 1258. The line chart for the close price during that period is shown as below.

```
chartSeries(SNE,subset='2015::2019')
```

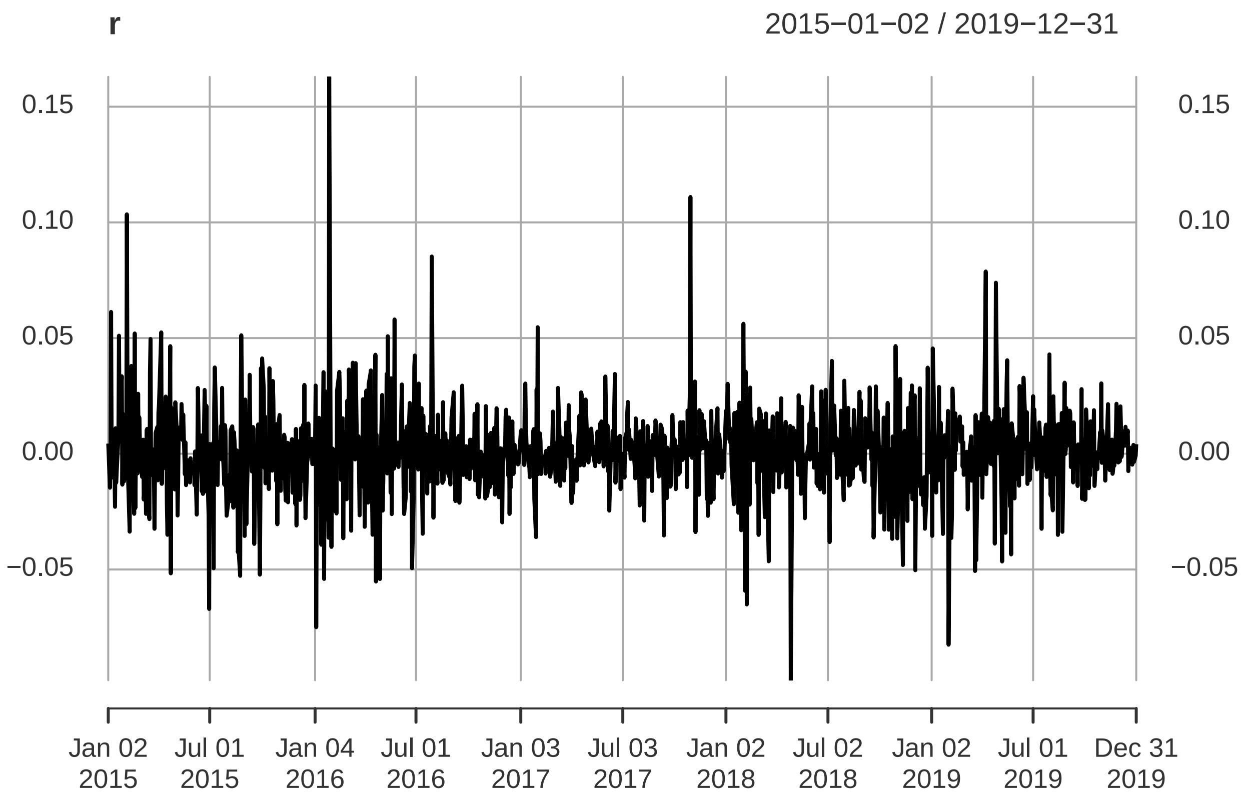


The structure of this report is as follows: in the second section, I will illustrate the method of modeling, which including the discussion about trend and seasonality and fitting the models for the random component; then, it will be followed by the diagnostic and comparison of the models in the third section; in the last section, there will be a short conclusion.

2.Method of modeling

To analyze the stock price, we usually calculate the logged return of the stock to make the data stationary. The following plot shows the daily logged return of SNE.

```
plot(r)
```



From the above plot, it seems that there is no trend or seasonality for this time series and the data has mean 0.

a.Trend

To test whether there is a drift or a trend for the return, I used the Augmented Dickey–Fuller (ADF) test. The model for the ADF test is

$$\Delta X_t = \alpha + \beta t + \gamma X_{t-1} + \delta_1 \Delta X_{t-1} + ... + \delta_{p-1} \Delta X_{t-p-1} + \varepsilon_t$$

Imposing the constraints $\alpha = 0$ and $\beta = 0$ corresponds to modelling a random walk without a drift, and using the constraint $\beta = 0$ corresponds to modelling a random walk with a drift. The lag p of the difference can be determined by AIC of the fitted AR models. The result of the test is as following.

```
##Check for the trend
summary(ur.df(r, type='trend', lags=20, selectlags="BIC"))
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
```

```
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.098396 -0.009419  0.000083  0.009253  0.160699
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.685e-04  1.104e-03   0.606    0.545
## z.lag.1     -1.054e+00  4.095e-02 -25.745   <2e-16 ***
## tt           3.802e-07  1.507e-06   0.252    0.801
## z.diff.lag   1.929e-02  2.847e-02   0.678    0.498
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01893 on 1233 degrees of freedom
## Multiple R-squared:  0.5173, Adjusted R-squared:  0.5162
## F-statistic: 440.5 on 3 and 1233 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -25.7455 220.9434 331.4151
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.96 -3.41 -3.12
## phi2  6.09  4.68  4.03
## phi3  8.27  6.25  5.34
```

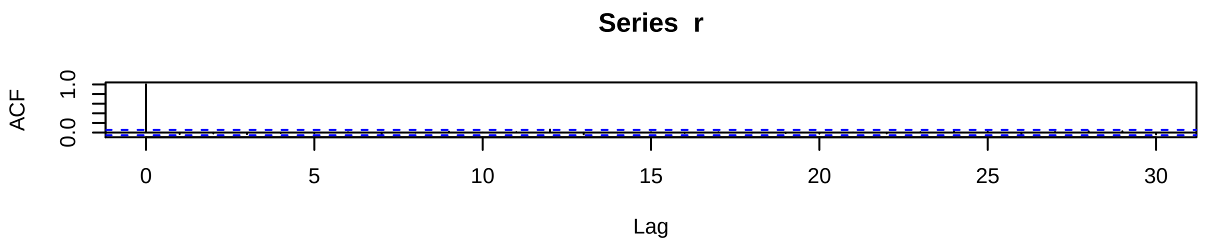
From the result, we can see that the intercept, which is α, is significantly different from 0. It means that the mean of the time series is not 0, in other words, there is a drift. Also, there is no linear trend for this time series, since the coefficient for tt is not significant.

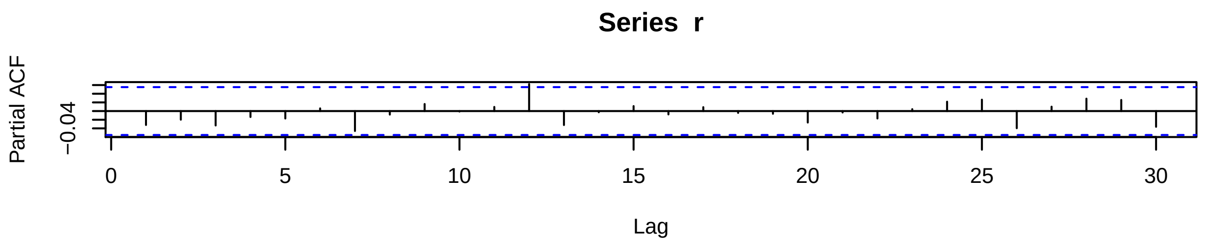
Additionally, this test also indicates there is no unit root present since the null hypothesis of $\gamma = 0$ is rejected. This means the model is not a random walk. We also notice that the $\hat{\gamma} = -1.054$ which means the AR part of the model is stationary.

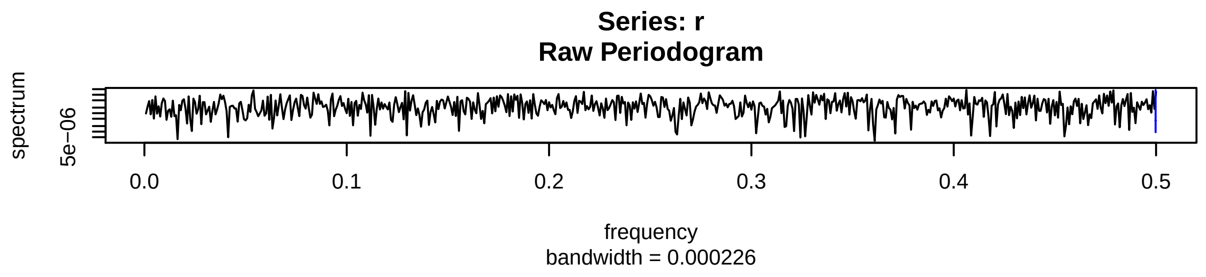
b. Seasonality

By viewing the ACF, PACF, and spectrum Periodgram, we cannot find an evidence for seasonality.

```r
##Check for the seasonality
par(mfrow=c(3,1))
acf(r)
pacf(r)
spec.pgram(r)
```





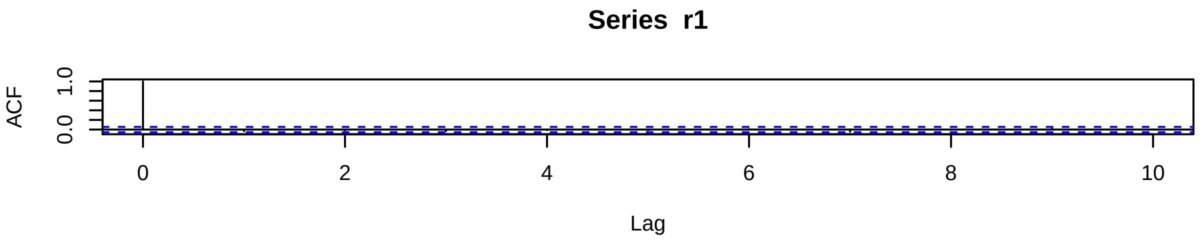


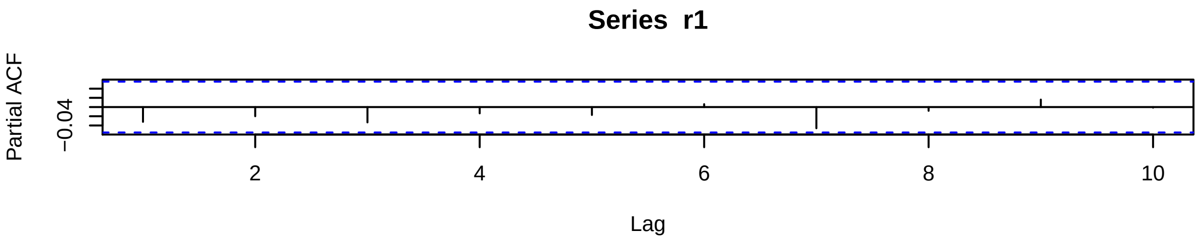
c. Random component

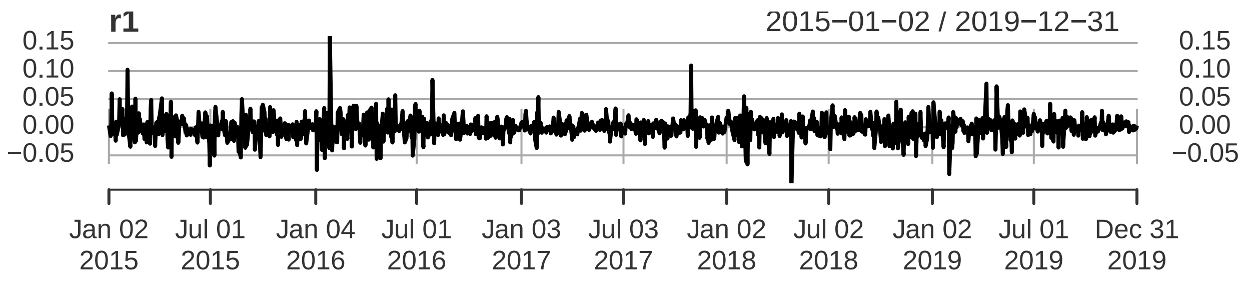
To remove the drift from the time series, we can use the two different methods: demeaning the data and making the difference.

First, I try to demean the data and fit the demeaned data with a time series model. After deducting the mean of the series, the time plot, ACF and PACF are as following.

```r
##1.Demean
r1=r-mean(r)
par(mfrow=c(3,1))
acf(r1,lag=10)
pacf(r1,lag=10)
plot(r1)
```







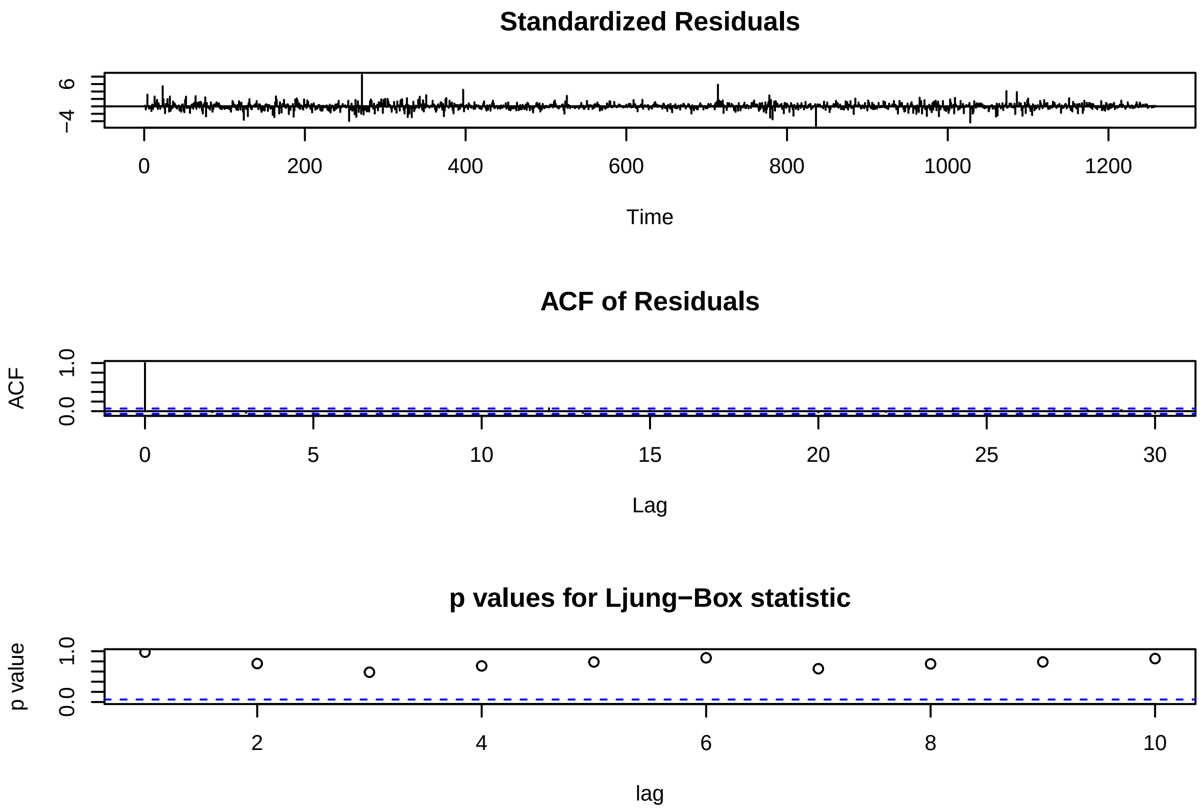
Since the Partial ACF cut off after the first lag, it seems that the demeaned logged return follows AR(1) model.

```
fit = arima(r,order=c(1,0,0))
summary(fit)
```

```
##
## Call:
## arima(x = r, order = c(1, 0, 0))
##
## Coefficients:
##           ar1  intercept
##       -0.0321      1e-03
## s.e.   0.0282      5e-04
##
## sigma^2 estimated as 0.0003591:  log likelihood = 3204.09,  aic = -6402.19
##
## Training set error measures:
##                        ME       RMSE        MAE  MPE MAPE      MASE
## Training set 5.38518e-07 0.01895071 0.01317968 -Inf  Inf 0.6738715
##                       ACF1
## Training set -0.0006643895
```

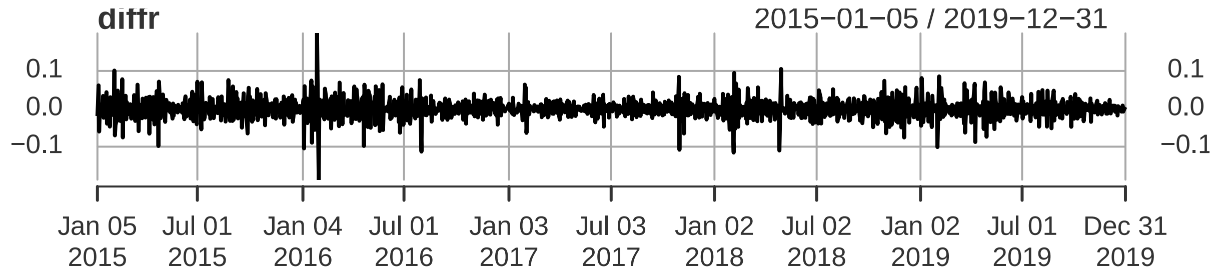
The diagnostic of residuals of the AR(1) model with drift is summarized in the following. From the ACF plot and the Ljung-Box statistics, we can see that the residuals are almost uncorrelated.

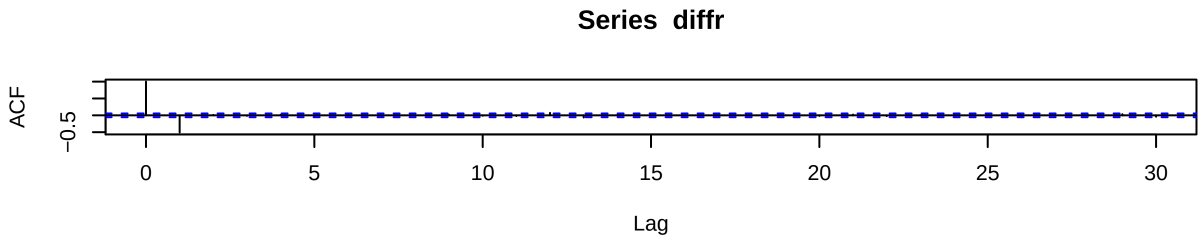
```
tsdiag(fit)
```

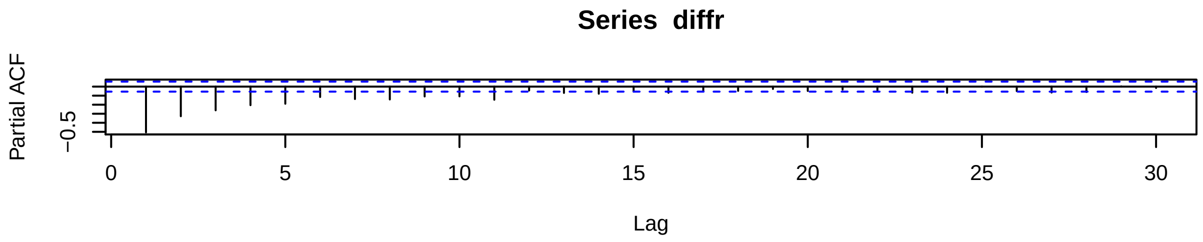


We can also get rid of the drift by making first difference. Then the time plot, ACF and PACF are listed below.

```
##2.Difference
diffr = na.omit(diff(r))
par(mfrow=c(3,1))
plot(diffr)
acf(diffr)
pacf(diffr)
```





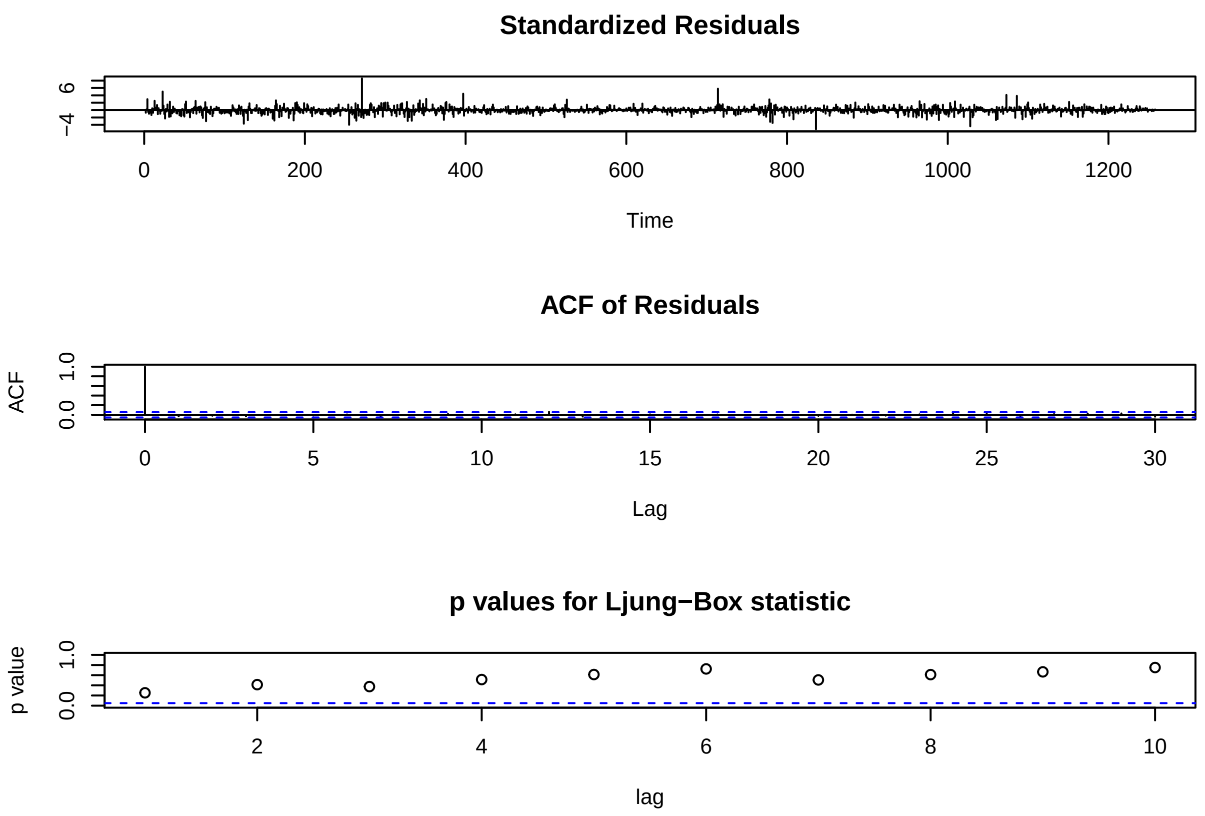


Since the ACF cut off after the first lag and the Partial ACF decrease gradually, it seems that the differenced logged return follows MA(1) model.

```
fit1 = arima(r, order=c(0,1,1))
summary(fit1)
```

```
##
## Call:
## arima(x = r, order = c(0, 1, 1))
##
## Coefficients:
##           ma1
##       -1.0000
## s.e.   0.0025
##
## sigma^2 estimated as 0.0003598:  log likelihood = 3196.83,  aic = -6389.66
##
## Training set error measures:
##                         ME      RMSE        MAE  MPE MAPE      MASE        ACF1
## Training set -0.0004476975 0.0189605 0.01323159 -Inf  Inf 0.6765257 -0.03220321
```

```
tsdiag(fit1)
```

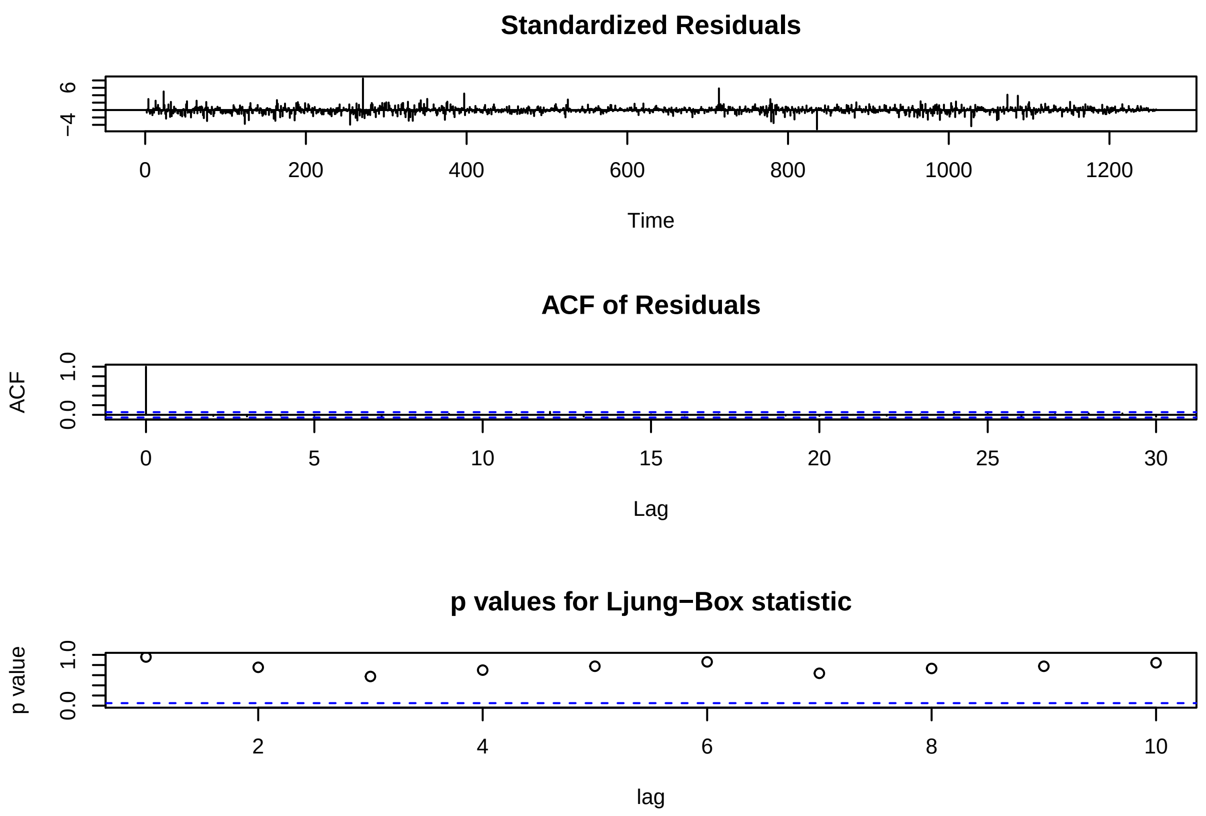


The residual seems to be not independent from each other. To solve this, I tried ARIMA(1,1,1).

```
fit2 = arima(r, order=c(1,1,1))
summary(fit2)
```

```
##
## Call:
## arima(x = r, order = c(1, 1, 1))
##
## Coefficients:
##           ar1      ma1
##       -0.0313  -1.0000
## s.e.   0.0282   0.0025
##
## sigma^2 estimated as 0.0003594:  log likelihood = 3197.45,  aic = -6388.9
##
## Training set error measures:
##                         ME       RMSE        MAE  MPE MAPE      MASE
## Training set -0.0004615528 0.01895072 0.01322398 -Inf  Inf 0.6761365
##                      ACF1
## Training set -0.001493171
```

```
tsdiag(fit2)
```



The residuals for ARIMA(1,1,1) are much better than ARIMA(0,1,1).

3. Model selection and diagnostic

To compare the two models, I make a comparison of the information criteria. From the following table, we can see that the AR(1) model with intercept is much better than that of ARIMA(1,1,1).

```
compare
```

```
##             model Log-likelihood      AIC
## 1 AR(1)+intercept        3204.09 -6402.19
## 2    ARIMA(1,1,1)        3197.45 -6388.89
```

Thus the final model is

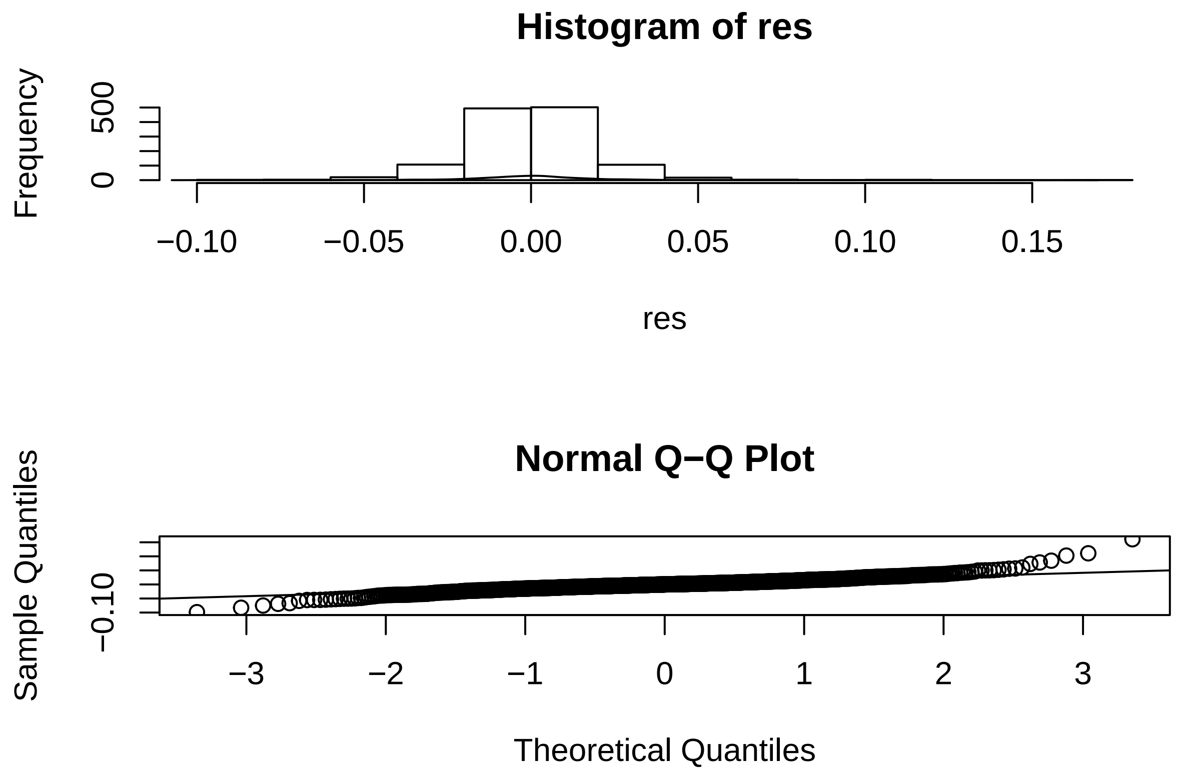
$$X_t = 0.0015 - 0.0321X_{t-1} + \varepsilon_t$$

The diagnostic for the independence of the residual has been shown in the previous section. Besides that, we can also check the normality of the residuals. From the histogram and qq-plot of the residual, we can see that the residuals are not normal distirbuted. The flat density curve and the invert S-shaped qq-plot indicate that the denisty of the residual should be fat tailed.

```
##Diagnostic
res=residuals(fit)
shapiro.test(res)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.93374, p-value < 2.2e-16
```

```
par(mfrow=c(2,1))
hist(res)
lines(density(res))
qqnorm(res)
qqline(res)
```



4. Forecasting

Using the AR(1) model with drift:

$$X_t = 0.0015 - 0.0321X_{t-1} + \varepsilon_t$$

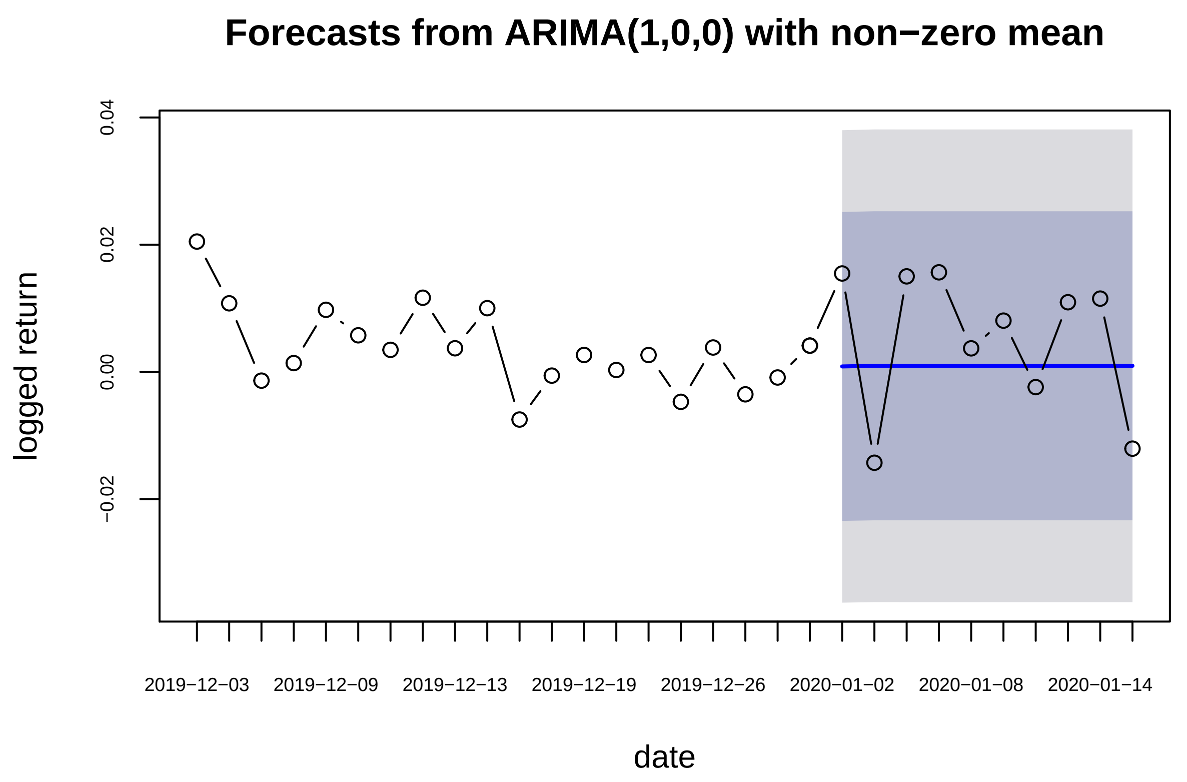
to forecast the logged return, I get the following result.

```
##Forecasting of lag 10
l=10 # number of lags for forecasting
h=20 # number of training data shown in the plot

fore <- forecast(fit,l)
summary(fore)
```

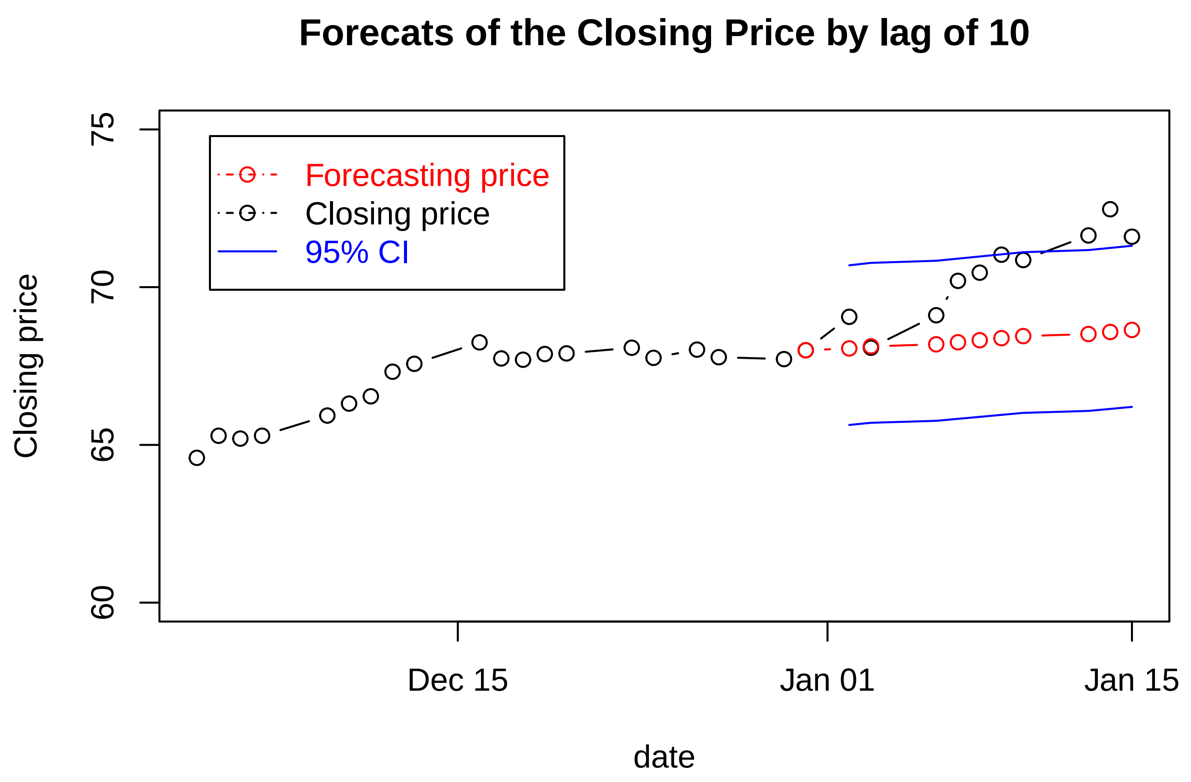
```
##
## Forecast method: ARIMA(1,0,0) with non-zero mean
##
## Model Information:
##
## Call:
## arima(x = r, order = c(1, 0, 0))
```

```
##
## Coefficients:
##           ar1  intercept
##       -0.0321      1e-03
## s.e.   0.0282      5e-04
##
## sigma^2 estimated as 0.0003591:  log likelihood = 3204.09,  aic = -6402.19
##
## Error measures:
##                        ME       RMSE        MAE  MPE MAPE      MASE
## Training set 5.38518e-07 0.01895071 0.01317968 -Inf  Inf 0.6738715
##                        ACF1
## Training set -0.0006643895
##
## Forecasts:
##      Point Forecast       Lo 80      Hi 80       Lo 95      Hi 95
## 1259   0.0008518857 -0.02343442 0.02513820 -0.03629082 0.03799459
## 1260   0.0009569980 -0.02334182 0.02525582 -0.03620484 0.03811884
## 1261   0.0009536236 -0.02334521 0.02525246 -0.03620824 0.03811548
## 1262   0.0009537320 -0.02334510 0.02525257 -0.03620813 0.03811559
## 1263   0.0009537285 -0.02334511 0.02525256 -0.03620813 0.03811559
## 1264   0.0009537286 -0.02334511 0.02525256 -0.03620813 0.03811559
## 1265   0.0009537286 -0.02334511 0.02525256 -0.03620813 0.03811559
## 1266   0.0009537286 -0.02334511 0.02525256 -0.03620813 0.03811559
## 1267   0.0009537286 -0.02334511 0.02525256 -0.03620813 0.03811559
## 1268   0.0009537286 -0.02334511 0.02525256 -0.03620813 0.03811559
```

```r
##Plot the forecasting logged return and the real value
plot(fore,h,axes=FALSE,ylab="logged return",xlab="date",type="b")
lines(c(n+0:l),ret["2019-12-31::"][1+0:l],type="b")

#combine the time period of last h terms in training data and the testing data of length l
date=c(index(r[n-0:(h-1)]),index(ret["2020-01-01::"][1:l]))
#add x-axis and y-axis
axis(1, at = c(n-h+1:(h+l)), labels = date, cex.axis=0.6)
axis(2,cex.axis=0.6)
box()
```



```
##Calculating the Forecasts of closing price
fore.mean <- as.vector(fore$mean) #Change the estimated mean to a vector
#change the last closing price in the training data to a number
lastprice <- as.numeric(SNE$SNE.Close["2019-12-31"])
fore.price <- Reduce(function(x,y){x*exp(y)},fore.mean, init=lastprice, accumulate=T)


#95% Upper and Lower bond for closing price
lower=fore.price[c(1+1:l)]*exp(fore$lower[,2])
upper=fore.price[c(1+1:l)]*exp(fore$upper[,2])

##Plot the forecasting closing price and the real value
plot(date,SNE$SNE.Close[date],type="b",ylab="Closing price",ylim=c(60,75), main="Forecats of the Closin
period=index(ret["2019-12-31::"][1+0:l]) #the forecast period
lines(period,fore.price,type="b",col="red")
lines(period[1+1:l],upper,col="blue")
lines(period[1+1:l],lower,col="blue")
legend("topleft", c("Forecasting price","Closing price","95% CI"),col=c("red","black","blue"), text.col
```

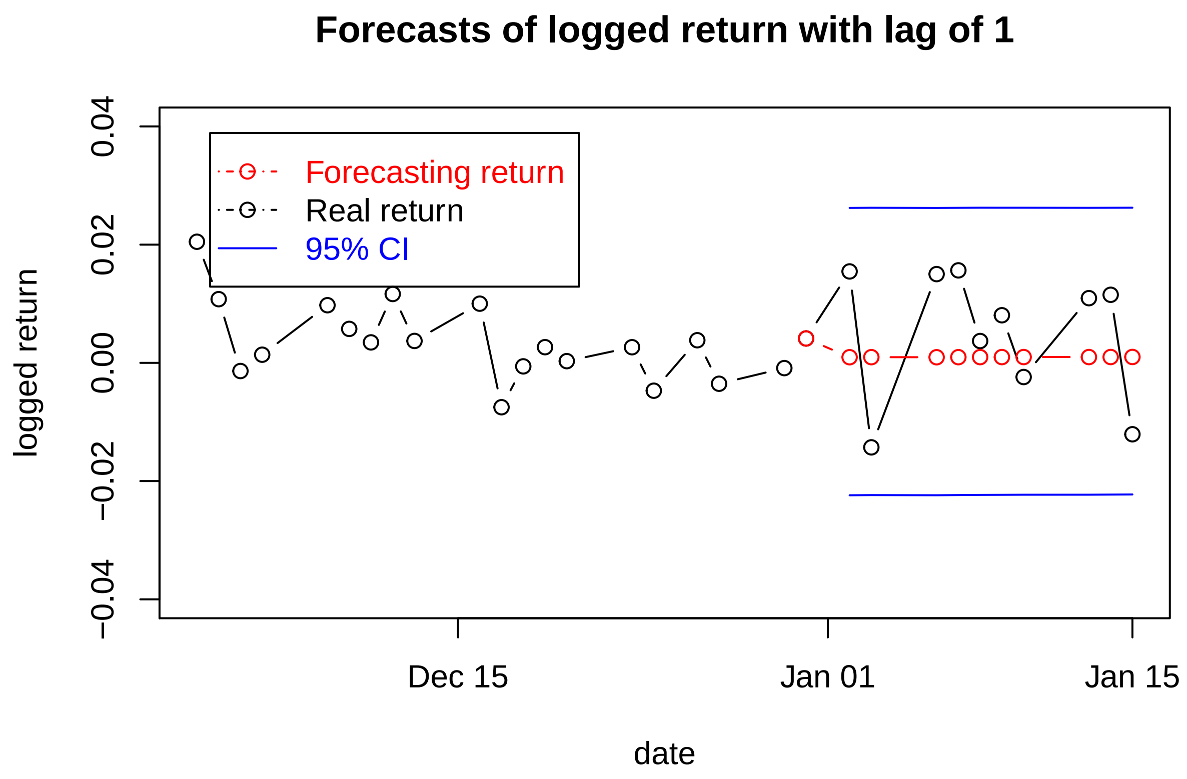


From the results and the figures above, we can see that the forecasts tend to be the same after several steps. This can be explained by the ACF plot. Since the autocorrelation is quite small after the 1^{st} lag, the h-step-ahead forecast is not reliable. So I tried to do 1-step-ahead forecast, then re-fit the time series with newly added observations and then predict the next one.

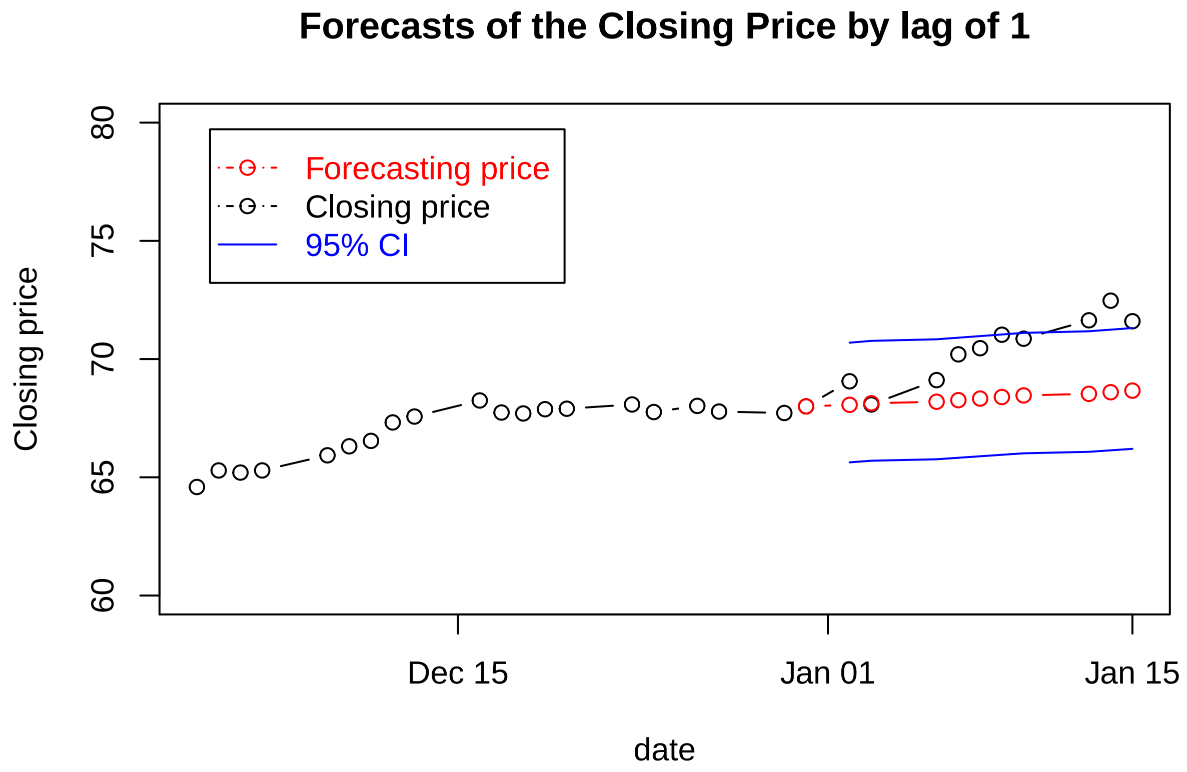
```
##Forecasting with lag of 1
r2=c(r,ret["2020-01-01::"][1:l])
fore2.mean=ret["2019-12-31::"][1+0:l]
fore2.upper=vector()
fore2.lower=vector()

## Loop to overlay early forecasts

for (j in seq(0, l-1, by=1)) {

  b.fit <-auto.arima(r2[1:(n+j)])

  b.pred <- forecast(b.fit, 1)

  fore2.mean[j+2]=b.pred$mean

  fore2.upper=rbind(fore2.upper,b.pred$upper)

  fore2.lower=rbind(fore2.lower,b.pred$lower)

}
fore2 <- cbind(fore2.mean[1+1:l],fore2.upper,fore2.lower)
colnames(fore2) <- c("Forecasts","H80","H95","L80","L95")
fore2
```

```
##                  Forecasts        H80         H95         L80         L95
## 2020-01-02 0.0009543302 0.02526285 0.03813100 -0.02335419 -0.03622234
## 2020-01-03 0.0009658581 0.02527036 0.03813639 -0.02333865 -0.03620468
## 2020-01-06 0.0009537485 0.02525485 0.03811907 -0.02334735 -0.03621157
## 2020-01-07 0.0009649002 0.02526165 0.03812358 -0.02333185 -0.03619378
## 2020-01-08 0.0009765356 0.02526943 0.03812931 -0.02331636 -0.03617624
## 2020-01-09 0.0009786895 0.02526215 0.03811704 -0.02330477 -0.03615966
## 2020-01-10 0.0009842895 0.02525948 0.03810999 -0.02329090 -0.03614141
## 2020-01-13 0.0009816172 0.02524751 0.03809310 -0.02328427 -0.03612986
## 2020-01-14 0.0009894891 0.02524844 0.03809036 -0.02326946 -0.03611138
## 2020-01-15 0.0009977998 0.02525013 0.03808854 -0.02325453 -0.03609294
```

```r
##Plotting

plot(date,r2[date],type="b",ylab="logged return",ylim=c(-0.04,0.04),main="Forecasts of logged return wi
lines(period,fore2.mean,type="b",col="red")
lines(period[1+1:l],fore2.mean[1+1:l]+fore2.upper[,1],col="blue")
lines(period[1+1:l],fore2.mean[1+1:l]+fore2.lower[,1],col="blue")
legend("topleft", c("Forecasting return","Real return","95% CI"), col=c("red","black","blue"),text.col=
```



```r
##Calculating the Forecasts of closing price
fore2.mean2=as.vector(fore2.mean[1+1:l])
fore2.price <- Reduce(function(x,y){x*exp(y)},fore2.mean2, init=lastprice, accumulate=T)

lower2=fore2.price[c(1+1:l)]*exp(fore2.lower[,2])
upper2=fore2.price[c(1+1:l)]*exp(fore2.upper[,2])

plot(date,SNE$SNE.Close[date],type="b",ylab="Closing price",ylim=c(60,80),main="Forecasts of the Closin
period=index(ret["2019-12-31::"][1+0:l]) #the forecast period

lines(period,fore2.price,type="b",col="red")
lines(period[1+1:l],upper,col="blue")
lines(period[1+1:l],lower,col="blue")

legend("topleft", c("Forecasting price","Closing price","95% CI"), col=c("red","black","blue"),text.col=
```



From the plots, it seems that the forecasts of the logged return of lag 1 are more close to the real data. However, by comparing the sum squared error, we can get the opposite conclusion. This can be explained by the small sample size we are using, only 10 forecast values. If we do testing based on larger sample size, the forecast result for the 1-step-ahead forecasting should be much better.

```
##Calculating the sum square error
sum((fore.mean-as.vector(ret[period[1+1:l]]))^2)#SSE of lag10
```

```
## [1] 0.001310385
```

```
sum((fore2.mean2-as.vector(ret[period[1+1:l]]))^2)#SSE of lag1
```

```
## [1] 0.001306906
```

5. Conclusion

I fit the logged return of SBUX data in a AR(1) model with drift:

$$X_t = 0.0015 - 0.0321X_{t-1} + \varepsilon_t$$

The are independent but not normally distributed. Its density must be fat tail.