Report for Fitting a Time Series Model

1. Introduction

In this report, I choose the stock price of Starbucks Corporation (SBUX) as my data to analyze. The time period for the data is from Jan 2009 to Dec 2013. The total number of data points is 1258. The line chart for the close price during that period is shown as below.

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 SBUX.
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

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

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.

```
# ADF Test
Call:
lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
Residuals:    Min      1Q  Median      3Q     Max
             -0.19918 -0.019548  0.009427  0.009960  0.164317
Coefficients:     Estimate     Std. Error t value Pr(>|t|)
(Intercept)    3.225343      1.201450   2.685 0.00736 **
z.lag.1       -1.061600      4.175402  -0.254 < 2e-16    ***
tt             -2.209093      1.634303  -1.346 0.17535
z.diff.lag    -1.068992      2.846402  -0.372 0.70966
---
Signif. codes:  < 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.0205 on 1233 degrees of freedom
Multiple R-squared: 0.5372, Adjusted R-squared: 0.5361
F-statistic: 477.1 on 3 and 1233 DF, p-value: < 2.2e-16
```

From the result, we can see that the intercept, which is $\alpha$, 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 $\hat{\gamma} \approx -1.063$ 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.

![Series r](image1)

![Series r](image2)

![Series r Raw Periodogram](image3)

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.

![Series #1](image4)

![Series #1](image5)

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

![Standardized Residuals](image)

![ACF of Residuals](image)

![p values for Ljung-Box statistic](image)

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

![Series differ](image)

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

```r
> fit1 = arima(r, order=c(0,1,1))
> summary(fit1)
Series: r
ARIMA(0,1,1)
Coefficients:
  ma1     -1.0000
s.e.    0.0042
sigma^2 estimated as 0.0004277:  log likelihood=3088.16
AIC=6172.33  AICC=6172.32  BIC=6162.06
Training set error measures:
      ME  RMSE  MAE  MPE  MAPE  MASE
Training set -0.0007476267 0.02067255 0.01441089 NaN   Inf 0.999337
```

```r
> fit2 = arima(r, order=c(1,1,1))
> summary(fit2)
Series: r
ARIMA(1,1,1)
Coefficients:
  ar1   ma1
-0.0720  -0.9995
s.e.    0.0282   0.0030
sigma^2 estimated as 0.0004256:  log likelihood=3091.4
AIC=6176.79  AICC=6176.77  BIC=6161.38
Training set error measures:
      ME  RMSE  MAE  MPE  MAPE  MASE
Training set -0.0007747377 0.02062276 0.01439253 NaN   Inf 0.9992019
```
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).

<table>
<thead>
<tr>
<th>model</th>
<th>Log-likelihood</th>
<th>AIC</th>
<th>AICC</th>
<th>BIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR(1)+intercept</td>
<td>3098</td>
<td>-6189.99</td>
<td>-6189.97</td>
<td>-6174.58</td>
</tr>
<tr>
<td>ARIMA(1,1,1)</td>
<td>3091.4</td>
<td>-6176.79</td>
<td>-6176.77</td>
<td>-6161.38</td>
</tr>
</tbody>
</table>

Thus the final model is

\[ X_t = 0.0017 - 0.0724 X_{t-1} + \varepsilon_t \]

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 distributed. The flat density curve and the invert S-shaped qq-plot indicate that the density of the residual should be fat tailed.
4. **Forecasting**

Using the AR(1) model with drift:

\[ X_t = 0.0017 - 0.0724X_{t-1} + \varepsilon_t \]

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

**Forecasts from ARIMA(1,0,0) with non-zero mean**

![Histogram of residuals](image1)

![Normal Q-Q Plot](image2)

<table>
<thead>
<tr>
<th>Point Forecast</th>
<th>Lo 90</th>
<th>Hi 90</th>
<th>Lo 95</th>
<th>Hi 95</th>
</tr>
</thead>
<tbody>
<tr>
<td>1259</td>
<td>0.001948632</td>
<td>-0.02447479</td>
<td>0.02837206</td>
<td>-0.03846251</td>
</tr>
<tr>
<td>1260</td>
<td>0.001697666</td>
<td>-0.02483290</td>
<td>0.02815243</td>
<td>-0.03885727</td>
</tr>
<tr>
<td>1261</td>
<td>0.001680691</td>
<td>-0.02481234</td>
<td>0.02817372</td>
<td>-0.03883690</td>
</tr>
<tr>
<td>1262</td>
<td>0.001679175</td>
<td>-0.02481386</td>
<td>0.02817221</td>
<td>-0.03883842</td>
</tr>
<tr>
<td>1263</td>
<td>0.001679285</td>
<td>-0.02481375</td>
<td>0.02817232</td>
<td>-0.03883831</td>
</tr>
<tr>
<td>1264</td>
<td>0.001679277</td>
<td>-0.02481375</td>
<td>0.02817231</td>
<td>-0.03883832</td>
</tr>
<tr>
<td>1265</td>
<td>0.001679278</td>
<td>-0.02481375</td>
<td>0.02817231</td>
<td>-0.03883832</td>
</tr>
<tr>
<td>1266</td>
<td>0.001679278</td>
<td>-0.02481375</td>
<td>0.02817231</td>
<td>-0.03883832</td>
</tr>
<tr>
<td>1267</td>
<td>0.001679278</td>
<td>-0.02481375</td>
<td>0.02817231</td>
<td>-0.03883832</td>
</tr>
<tr>
<td>1268</td>
<td>0.001679278</td>
<td>-0.02481375</td>
<td>0.02817231</td>
<td>-0.03883832</td>
</tr>
</tbody>
</table>
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 1st 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.
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.

```r
> sum((fore.mean-as.vector(ret[period[1:1;i]]))^2) #sse of lag10
[1] 0.002043399
> sum((fore2.mean2-as.vector(ret[period[1:1;i]]))^2) #sse of lag1
[1] 0.002054035
```

5. Conclusion

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

\[ X_t = 0.0017 - 0.0724X_{t-1} + \epsilon_t \]

The \( \epsilon_t \) are independent but not normally distributed. Its density must be fat tail.
R code:

```r
require(quantmod)
require(forecast)
require(urca)
require(tseries)

##load the data
getSymbols('SBUX')
chartSeries(SBUX,subset='2009::2013')

##Calculate the log-return
ret = na.omit(diff(log(SBUX$SBUX.Close)))
r = ret["2009::2013"]
n=length(r)
plot(r)

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

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

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

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

##2. Difference
differ = na.omit(diff(r))
par(mfrow=c(3,1))
plot(differ)
```
## Fit the ARIMA model
fit1 = arima(r, order=c(0,1,1))
summary(fit1)
tsdiag(fit1)

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

## Check by auto.arima
auto.arima(r)

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

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

###########################################################
## HW4 Forcasting
###########################################################
dev.off()

## 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)

###########################################################

## 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["2013-12-31::"])[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["2014-01-01::"][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$mean) #Change the estimated mean to a vector
#change the last closing price in the training data to a number
lastprice <- as.numeric(SBUX$SBUX.Close["2013-12-31"])

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,SBUX$SBUX.Close[date],type="b",ylab="Closing price",ylim=c(75,83.5),
     main="Forecats of the Closing Price by lag of 10")
period=index(ret["2013-12-31::"][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=c("red","black","blue"),lty=c(4,4,1), pch = c(1,1,NA),inset = .05)

##Forecasting with lag of 1
r2=c(r,ret["2014-01-01::"][1:l])
fore2.mean=ret["2013-12-31::"][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

#######################################################
##Plotting
plot(date,r2[date],type="b",ylab="logged return",
     ylim=c(-0.04,0.04),main="Forecasts of logged return with lag of 1")
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=c("red","black","blue"),lty=c(4,4,1), pch = c(1,1,NA),inset = .05)

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

#######################################################
##Plot the forecasting closing price and the real value
plot(date,SBUX$SBUX.Close[date],type="b",ylab="Closing price",
     ylim=c(75,83.5),
     main="Forecasts of the Closing Price by lag of 1")
period=index(ret["2013-12-31::"][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=c("red", "black", "blue"), lty=c(4,4,1), pch = c(1,1,NA), inset = .05)

## Calculating the sum square error
sum((fore.mean-as.vector(ret[period[1+1:l]]))^2)#SSE of lag10
sum((fore2.mean2-as.vector(ret[period[1+1:l]]))^2)#SSE of lag1

sum((fore.price[1+1:l]-as.vector(SBUX$SBUX.Close[period[1+1:l]]))^2)
sum((fore2.price[1+1:l]-as.vector(SBUX$SBUX.Close[period[1+1:l]]))^2)