HuYuDataInsight LLC

Zhaowei Cai

2024-05-21

(a)

```
setwd("D:/AMS-SBU/HuYuDataInsight/20240521")
library(quantmod)
```

```
## Warning: package 'quantmod' was built under R version 4.2.3

## Loading required package: xts

## Warning: package 'xts' was built under R version 4.2.3

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 4.2.3

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: TTR

## Warning: package 'TTR' was built under R version 4.2.3

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(urca)
```

```
## Warning: package 'urca' was built under R version 4.2.3
```

```
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.2.3
```

```
library(tseries)
```

```
## Warning: package 'tseries' was built under R version 4.2.3
```

```
library(fGarch)
```

```
## Warning: package 'fGarch' was built under R version 4.2.3

## NOTE: Packages 'fBasics', 'timeDate', and 'timeSeries' are no longer
## attached to the search() path when 'fGarch' is attached.
##
## If needed attach them yourself in your R script by e.g.,
##         require("timeSeries")

##
## Attaching package: 'fGarch'

## The following object is masked from 'package:TTR':
##
##     volatility
```

```
library(zoo)
library(tseries)
library(rugarch)
```

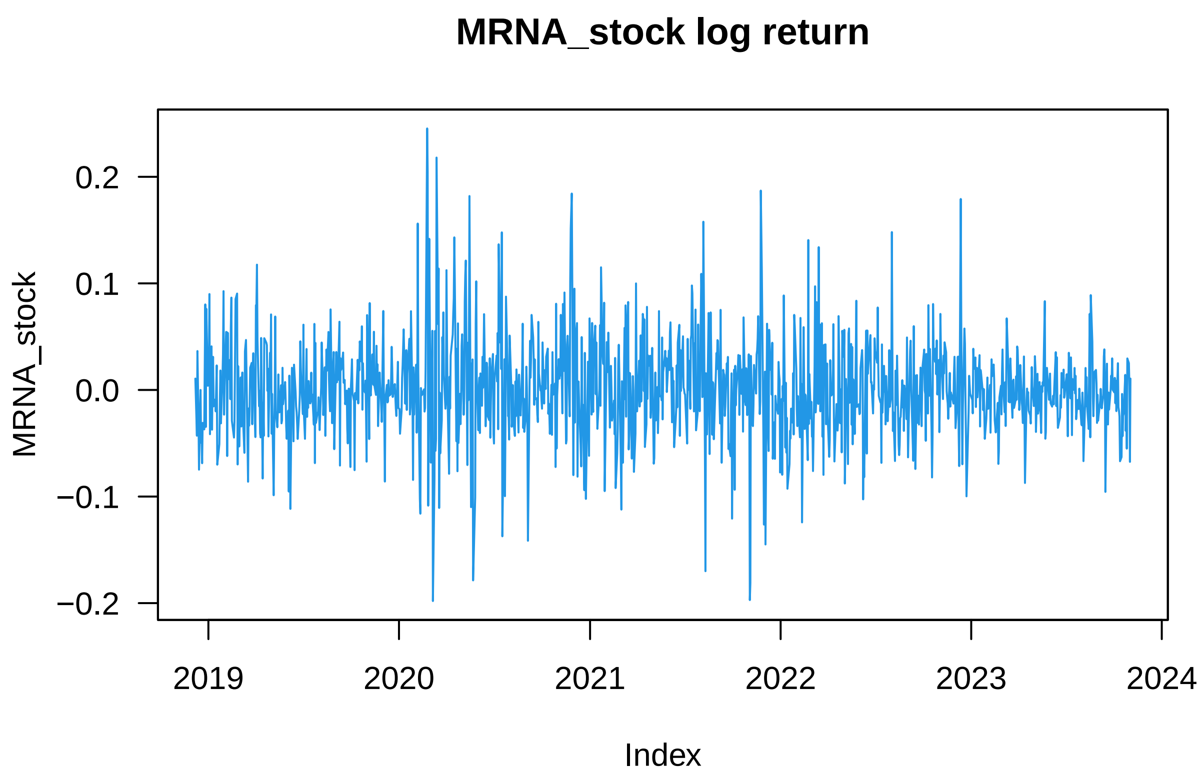
```
## Warning: package 'rugarch' was built under R version 4.2.3

## Loading required package: parallel

##
## Attaching package: 'rugarch'

## The following object is masked from 'package:stats':
##
##     sigma
```

```
library(stringr)
library(PerformanceAnalytics)
```

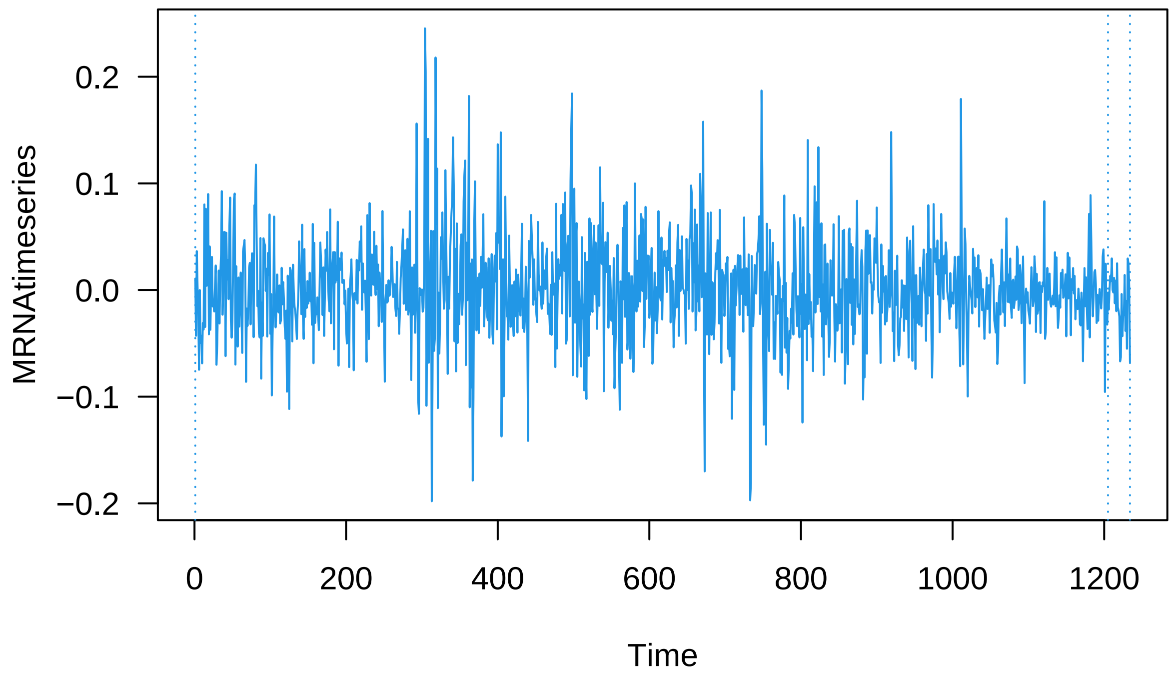
```
## Warning: package 'PerformanceAnalytics' was built under R version 4.2.3

##
## Attaching package: 'PerformanceAnalytics'

## The following object is masked from 'package:graphics':
##
##     legend
```

```
library(xts)
```

```
data = read.csv('Moderna 5 Years.csv')
closing = data$Close # closing price
log_closing = log(data$Close)
log_return = na.omit(diff(log(data$Close))) # log return

# Visualize the data
time = as.Date(data$Date)
df = data.frame(datefield = time[2:length(time)], MRNA = log_return)
MRNA_stock = with(df, zoo(MRNA, order.by = time))
plot.zoo(MRNA_stock, col = 4, las = 1, main = "MRNA_stock log return")
```



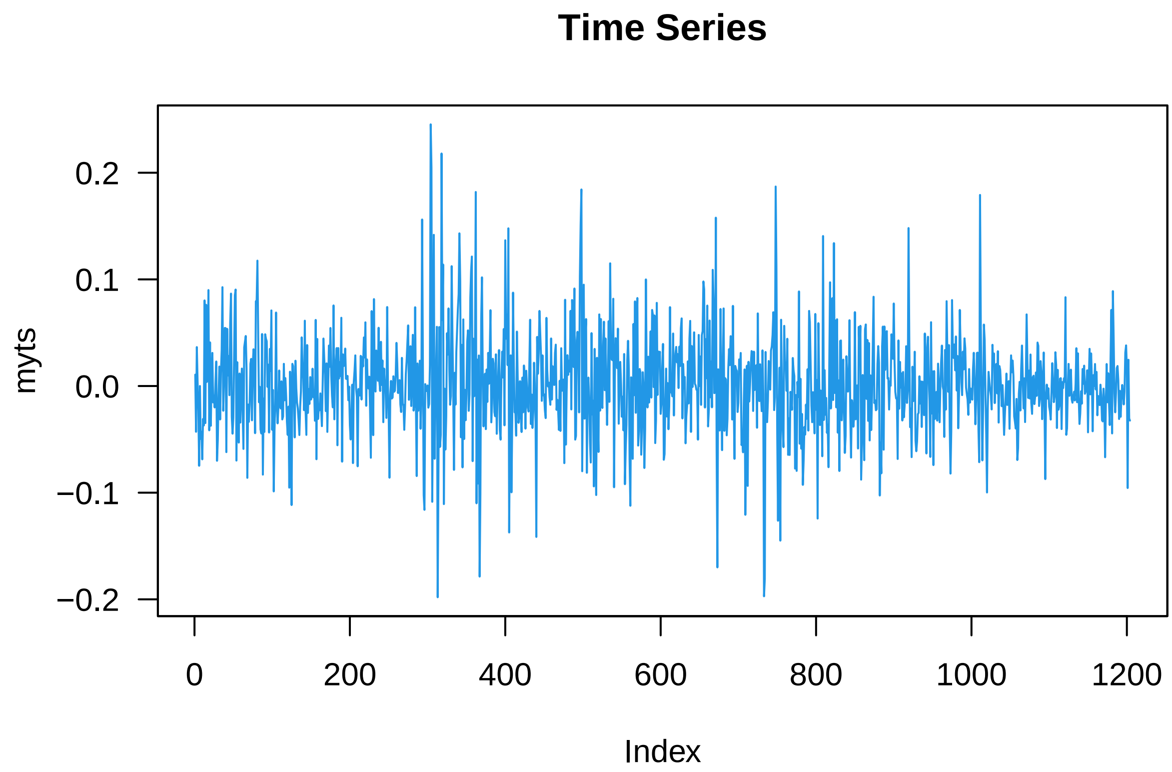

```
MRNAtimeseries <- ts(log_return, frequency = 1)
plot(MRNAtimeseries, col=4, las=1)
abline(v=c(1, 1205, 1234), lty="dotted", col=4)
```




```
data_len = length(MRNAtimeseries)
myts = subset(MRNAtimeseries, subset=rep(c(TRUE, FALSE), times=c(1204,30)))
```

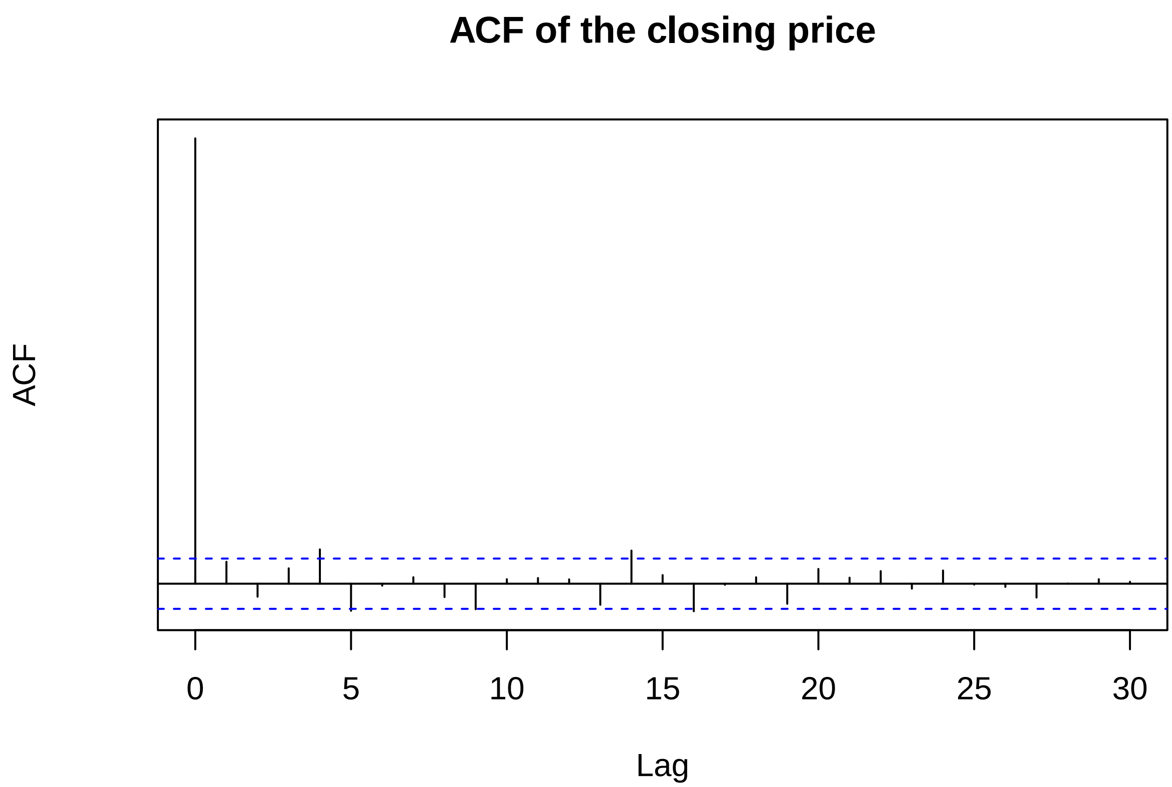
Step 1: visualize myts

```
plot.zoo(myts, col=4, las=1, main="Time Series")
```



Step 2: unit root test (augmented Dickey-Fuller) of myts

```
n = length(myts)
acf(myts,main="ACF of the closing price",yaxt="n")
```

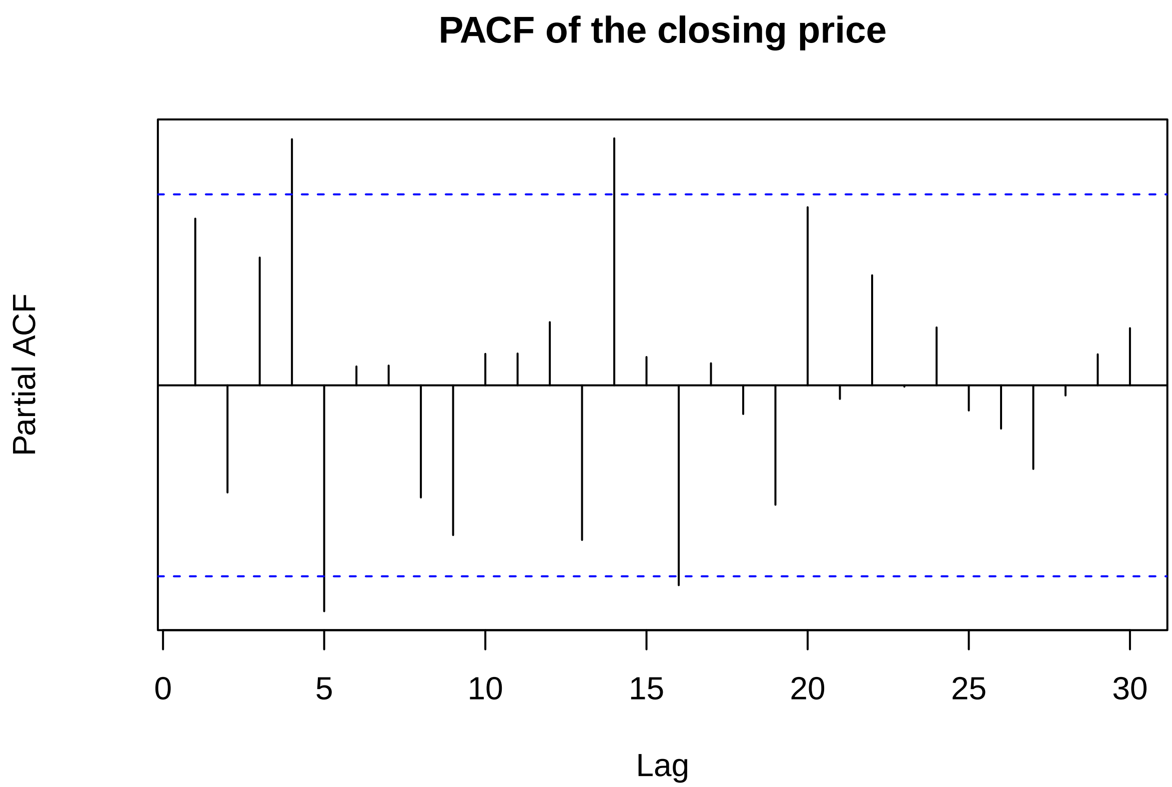



```
adf.test(myts, alternative = 'stationary')
```

```
## Warning in adf.test(myts, alternative = "stationary"): p-value smaller than
## printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  myts
## Dickey-Fuller = -10.925, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary
```

```
pacf(myts,main="PACF of the closing price",yaxt="n")
```



P-value less than 0.05, reject H0, and it is stationary.

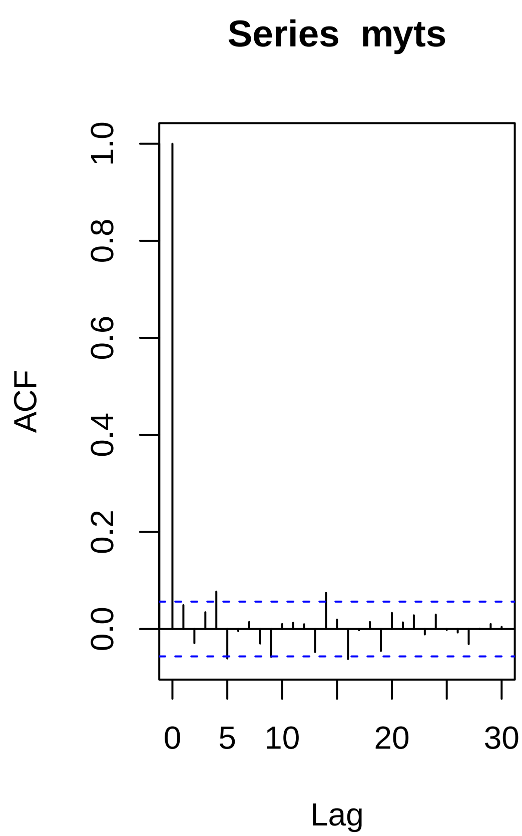
```r
summary(ur.df(myts, type='trend', lags=20, selectlags="BIC"))
```

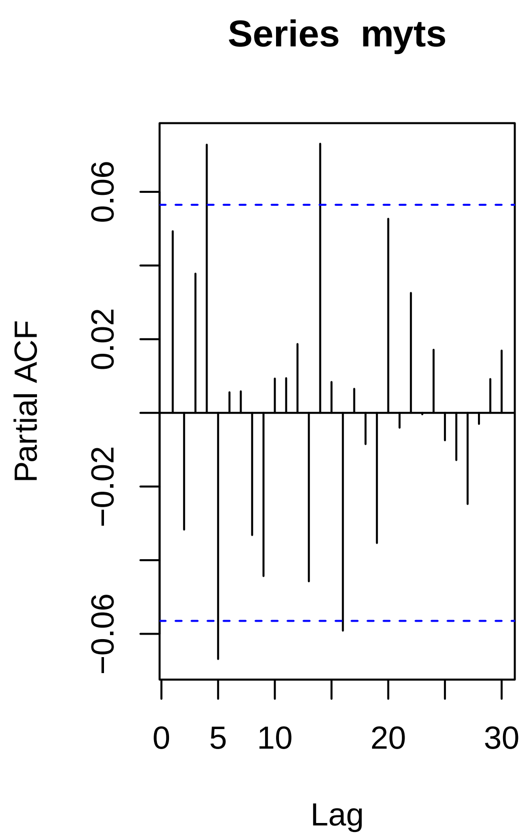
```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.20355 -0.02596 -0.00258  0.02500  0.24021
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.301e-03  2.819e-03   1.881   0.0602 .
## z.lag.1     -9.898e-01  4.011e-02 -24.679   <2e-16 ***
## tt          -6.177e-06  4.018e-06  -1.537   0.1244
## z.diff.lag   4.235e-02  2.909e-02   1.456   0.1457
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04711 on 1179 degrees of freedom
## Multiple R-squared:  0.4758, Adjusted R-squared:  0.4745
## F-statistic: 356.7 on 3 and 1179 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -24.6789 203.0175 304.5261
##
## 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
```

Step 3: identify lags for myts

```
par(mfrow=c(1,2), mar=c(5,4,3,3))
acf(myts)
pacf(myts)
```





Step 4: train the model with auto.arima for myts

```
fit_myts = auto.arima(myts, max.p=10, max.q=10, ic="aicc",
                       seasonal=FALSE, stationary=TRUE, lambda=NULL,
                       stepwise=FALSE, approximation=FALSE
                       )
summary(fit_myts)
```

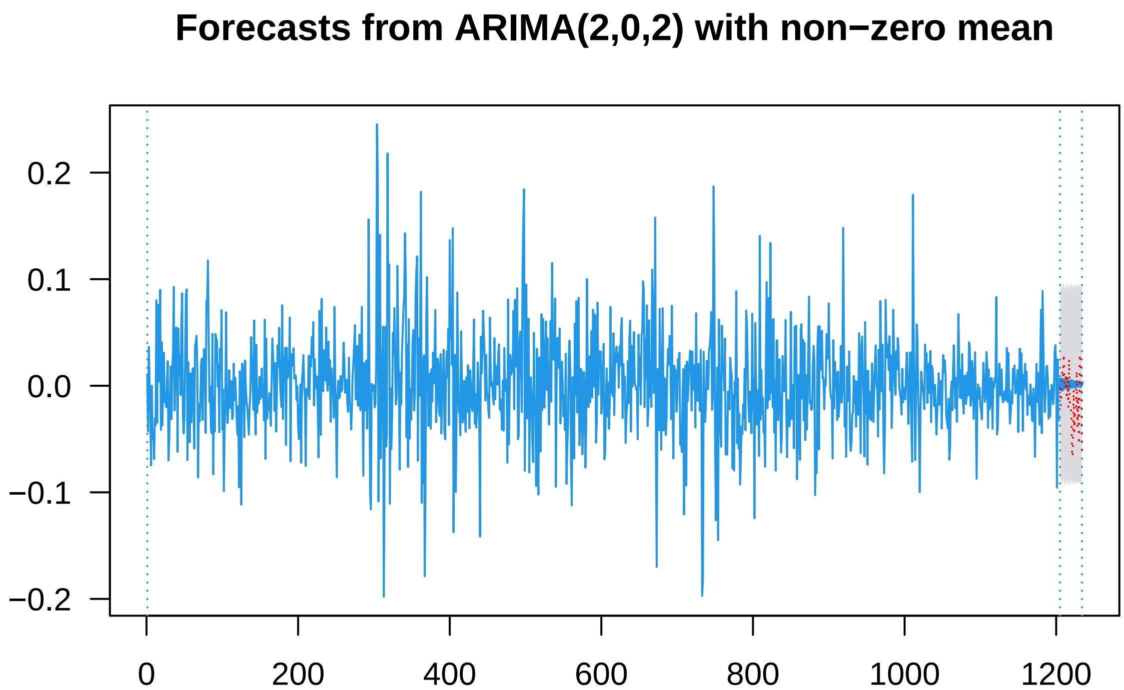
```
## Series: myts
## ARIMA(2,0,2) with zero mean
##
## Coefficients:
##           ar1      ar2     ma1     ma2
##       -0.5106  -0.9139  0.5392  0.8835
## s.e.   0.0539   0.0546  0.0584  0.0674
##
## sigma^2 = 0.002204:  log likelihood = 1976.3
## AIC=-3942.59   AICc=-3942.54   BIC=-3917.12
##
## Training set error measures:
##                       ME       RMSE       MAE  MPE MAPE      MASE       ACF1
## Training set 0.001399314 0.04686746 0.0343572 -Inf  Inf 0.7112457 0.02665648
```

Step 5: fit the log return, i.e. myts

```
fit_myts = arima(myts, c(2,0,2))
summary(fit_myts)
```

```
##
## Call:
## arima(x = myts, order = c(2, 0, 2))
##
## Coefficients:
##           ar1      ar2     ma1     ma2  intercept
##       -0.5103  -0.9137  0.5389  0.8831     0.0014
## s.e.   0.0541   0.0547  0.0587  0.0676     0.0013
##
## sigma^2 estimated as 0.002195:  log likelihood = 1976.83,  aic = -3941.67
##
## Training set error measures:
##                         ME       RMSE        MAE  MPE MAPE      MASE       ACF1
## Training set -1.913423e-06 0.04684652 0.03441971 -Inf  Inf 0.7125396 0.02674928
```

```
forecast_myts = forecast(fit_myts, h=30, level=0.95)
plot(forecast_myts, col=4, las=1)
abline(v=c(1, 1205, 1234), lty="dotted", col=4)
lines(1205:1234, MRNAtimeseries[1205:1234], lty="dotted", col="red")
```




```
# red is observation and blue is prediction
```

Step 6: fit the closing price, i.e. myts

```
#MRNAtimeseries_1 <- ts(log_closing, frequency = 1)
#plot(JJtimeseries_1, col=4, las=1, main = "JJ_stock log closing price")
#abline(v=c(1, 1227, 1255), lty="dotted", col=4)
```

```
#data_len = length(JJtimeseries_1)
#myts_1 = subset(JJtimeseries_1, subset=rep(c(TRUE, FALSE), times=c(1227,30)))

#fit_myts_1 = arima(myts_1, c(4,1,1))
#summary(fit_myts_1)
```

```
#forecast_myts1 = forecast(fit_myts_1, h=30, level=0.95)
#print(forecast_myts1)
#plot(forecast_myts1, col=4, las=1)
#abline(v=c(1, 1227, 1255), lty="dotted", col=4)
#lines(1227:1255, JJtimeseries_1[1227:1255], lty="dotted", col="red")
# red is observation and blue is prediction
```