ARIMA Practice

Anyi Fu

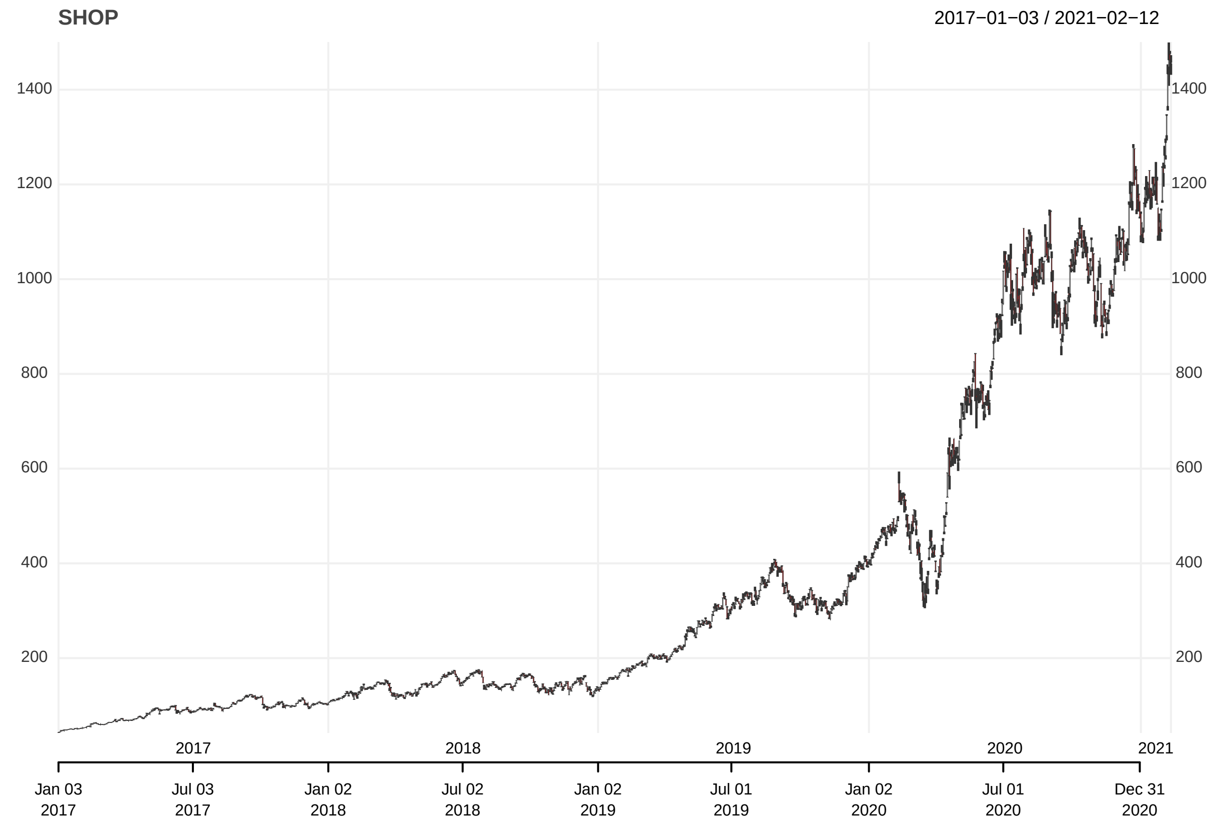
09/02/2021

Explore Data

```
summary(SHOP)
```

```
##      Index              SHOP.Open        SHOP.High         SHOP.Low
##  Min.   :2017-01-03   Min.   :  43.26   Min.   :  43.46   Min.   :  42.13
##  1st Qu.:2018-01-11   1st Qu.: 118.84   1st Qu.: 120.81   1st Qu.: 115.92
##  Median :2019-01-24   Median : 169.90   Median : 173.38   Median : 167.05
##  Mean   :2019-01-23   Mean   : 352.55   Mean   : 360.40   Mean   : 344.31
##  3rd Qu.:2020-02-04   3rd Qu.: 423.20   3rd Qu.: 435.81   3rd Qu.: 412.26
##  Max.   :2021-02-12   Max.   :1475.63   Max.   :1499.75   Max.   :1438.03
##    SHOP.Close       SHOP.Volume        SHOP.Adjusted
##  Min.   :  42.82   Min.   :  403700   Min.   :  42.82
##  1st Qu.: 118.64   1st Qu.: 1195900   1st Qu.: 118.64
##  Median : 171.18   Median : 1669100   Median : 171.18
##  Mean   : 353.03   Mean   : 2029440   Mean   : 353.03
##  3rd Qu.: 421.79   3rd Qu.: 2404225   3rd Qu.: 421.79
##  Max.   :1463.31   Max.   :20895900   Max.   :1463.31
```

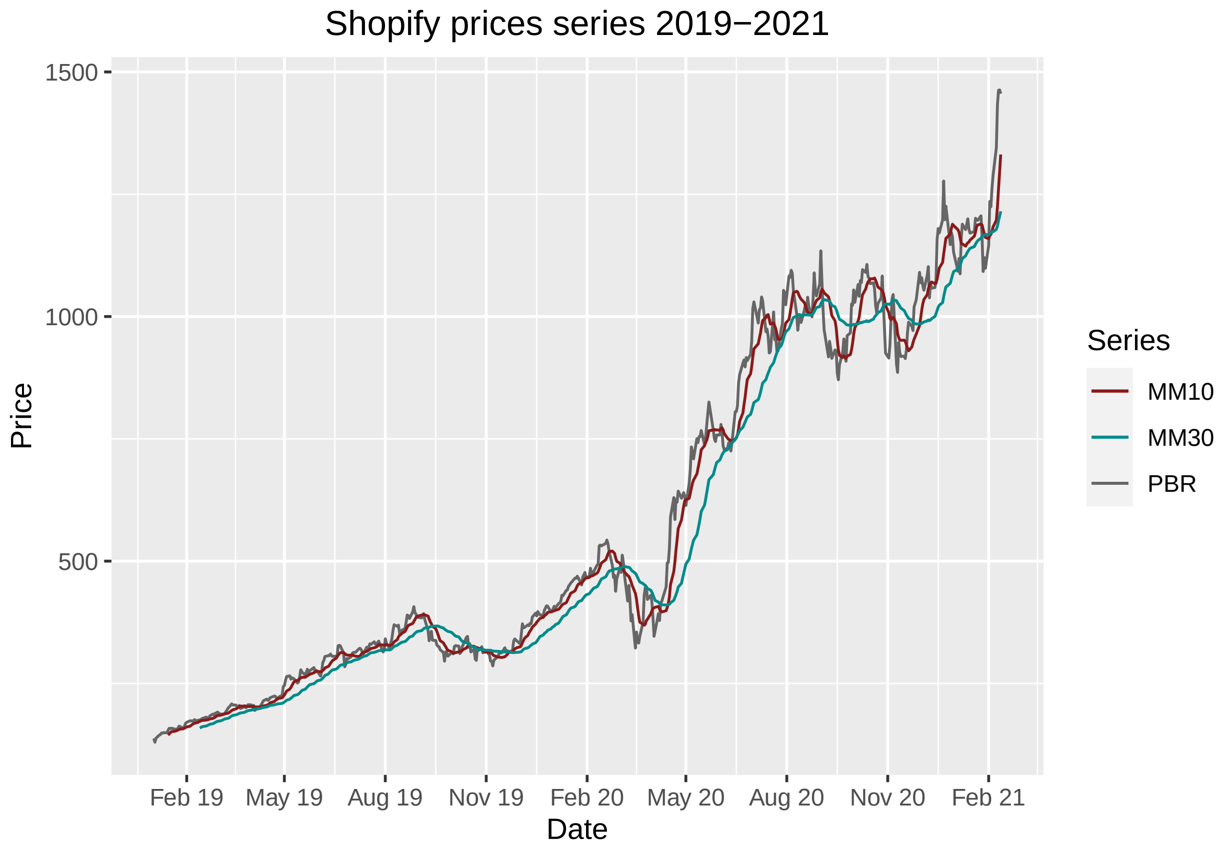
```
p1 = chart_Series(SHOP)

p1
```




```
#The data is clearly non-stationary, hence we will find the log returns of it
```

```
##MA

SHOP <- subset(SHOP_Ori, index(SHOP_Ori) >= "2019-01-01")
##two MA using 10 and 30 days of windows,
SHOP_mm10 <- rollmean(SHOP[,6], 10, fill = list(NA, NULL, NA), align = "right")
SHOP_mm30 <- rollmean(SHOP[,6], 30, fill = list(NA, NULL, NA), align = "right")


ggplot(SHOP, aes(x = index(SHOP))) +
  geom_line(aes(y = SHOP[,6], color = "PBR")) + ggtitle("Shopify prices series 2019-2021") +
  geom_line(aes(y = SHOP_mm10, color = "MM10")) +
  geom_line(aes(y = SHOP_mm30, color = "MM30")) + xlab("Date") + ylab("Price") +
  theme(plot.title = element_text(hjust = 0.5), panel.border = element_blank()) +
  scale_x_date(date_labels = "%b %y", date_breaks = "3 months") +
  scale_colour_manual("Series", values=c("PBR"="gray40", "MM10"="firebrick4", "MM30"="darkcyan"))
```

```
## Don't know how to automatically pick scale for object of type xts/zoo. Defaulting to continuous.

## Warning: Removed 9 row(s) containing missing values (geom_path).

## Warning: Removed 29 row(s) containing missing values (geom_path).
```

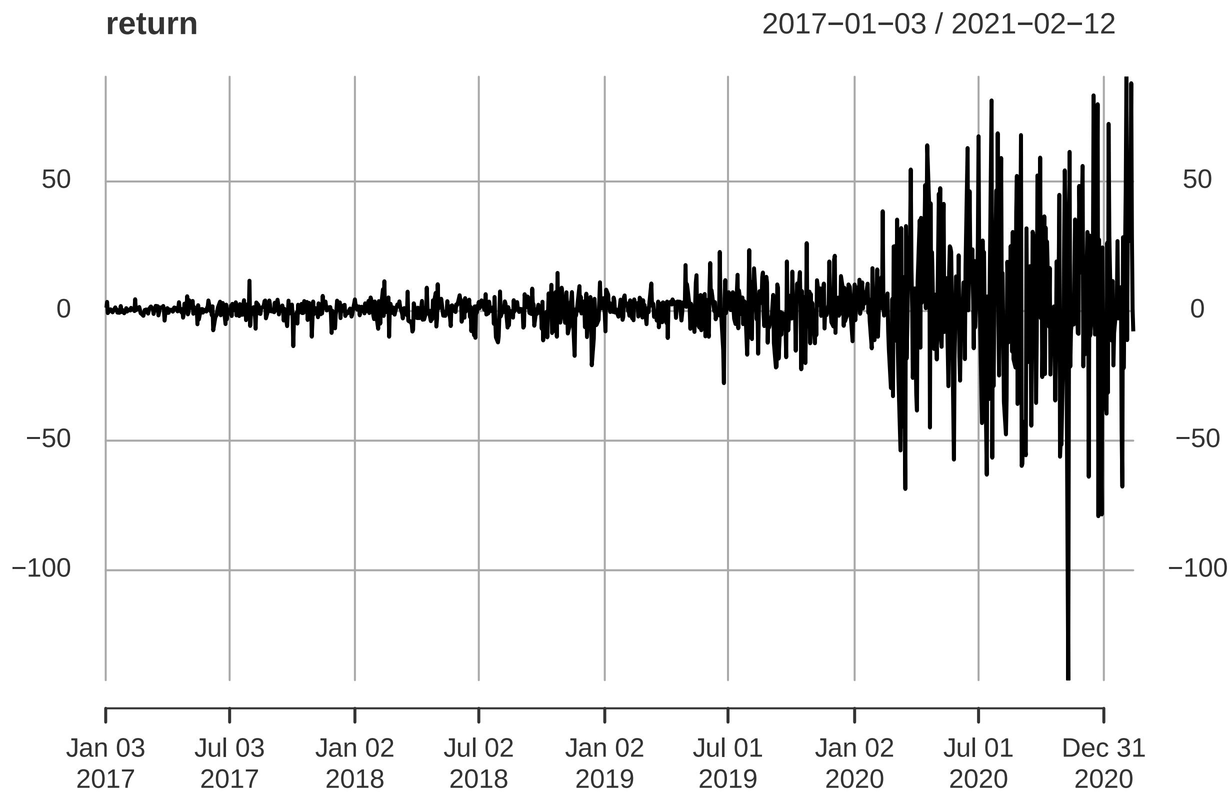



```
myTheme<-chart_theme()
myTheme$col$up.col<-'darkgreen'
myTheme$col$dn.col<-'darkred'
myTheme$col$dn.border <- 'black'
myTheme$col$up.border <- 'black'
myTheme$rylab <- FALSE
myTheme$col$grid <- "lightgrey"

p2 = chart_Series(SHOP['2020-12/2021-02'], theme = myTheme)
SHOP_Ori <- SHOP
p2
```


```
SHOP %>%
    chartSeries(TA='addBBands();
                    addBBands(draw="p");
                    addVo();
                    addMACD()',
                subset='2020/2021',
                theme="white"
                )
```

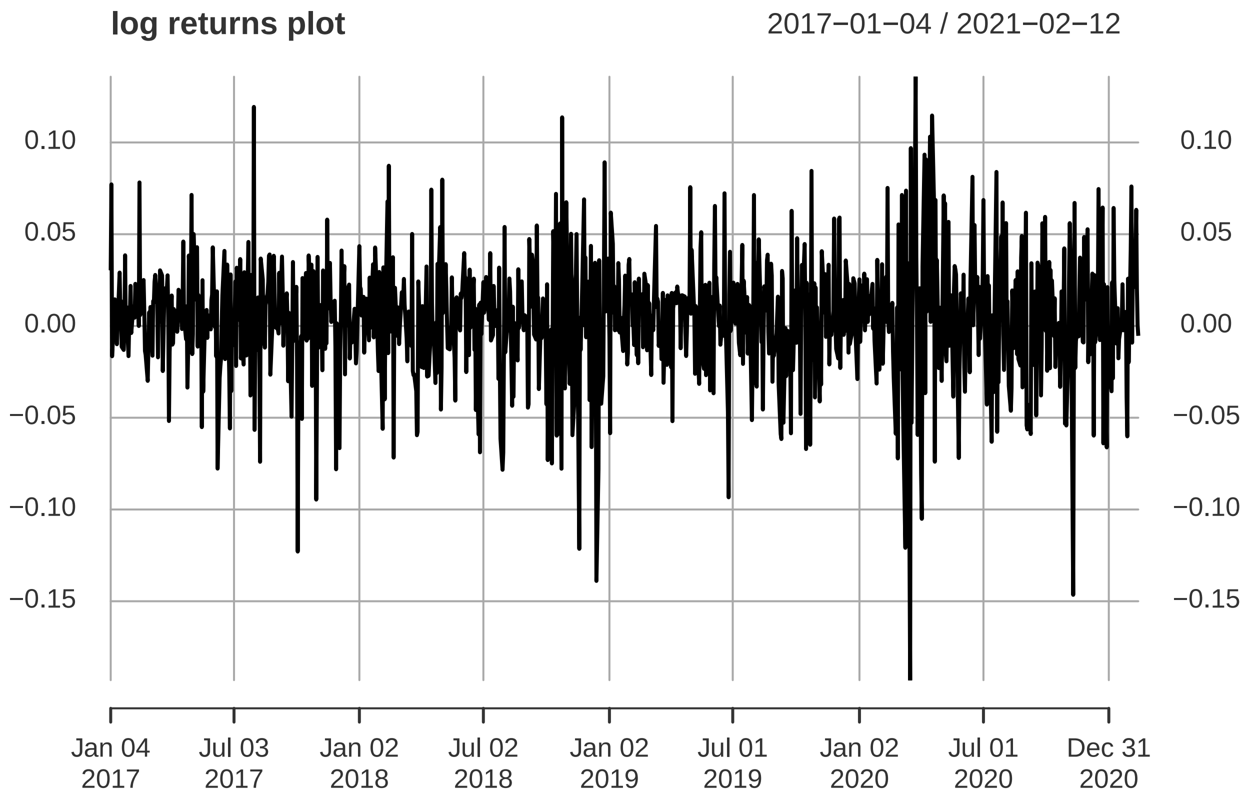
Return and log Return

```
return = diff(adj_prices,lag=1)
plot(return)
```




```r
# The series is obviously not stationary
```

```r
# Compute the log returns for the stock
logR = diff(log(adj_prices),lag=1)
logR = logR[!is.na(logR)]
# Plot log returns
plot(logR,type='l', main='log returns plot')
```




```r
summary(logR)
```

```
##      Index             SHOP.Adjusted
##  Min.   :2017-01-04   Min.   :-0.193000
##  1st Qu.:2018-01-14   1st Qu.:-0.013119
##  Median :2019-01-25   Median : 0.004968
##  Mean   :2019-01-23   Mean   : 0.003407
##  3rd Qu.:2020-02-04   3rd Qu.: 0.022101
##  Max.   :2021-02-12   Max.   : 0.135820
```

```r
skewness(logR)
```

```
## [1] -0.4345547
## attr(,"method")
## [1] "moment"
```

```r
kurtosis(logR)
```

```
## [1] 2.788449
## attr(,"method")
## [1] "excess"
```

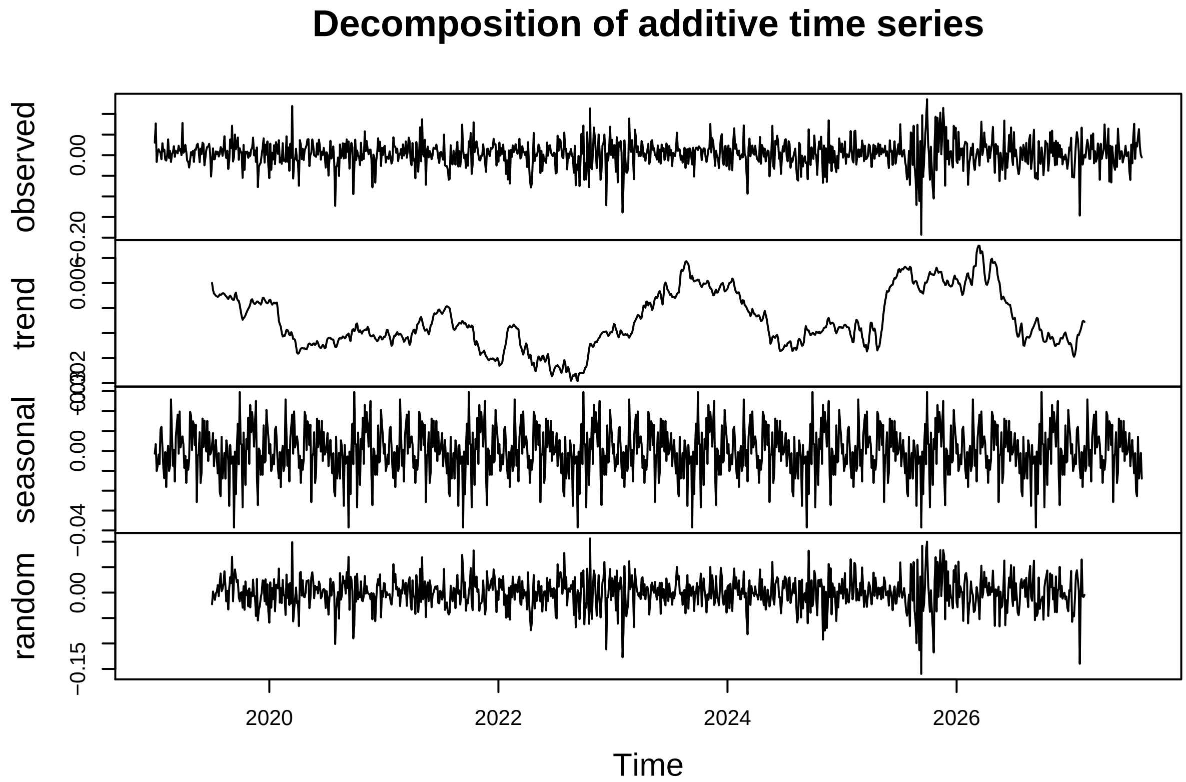
```r
#skewness is -0.4351852 which implies a slightly negative skewness
#kurtosis is 5.799651 which is much higher than 3 since it is a log function
```

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

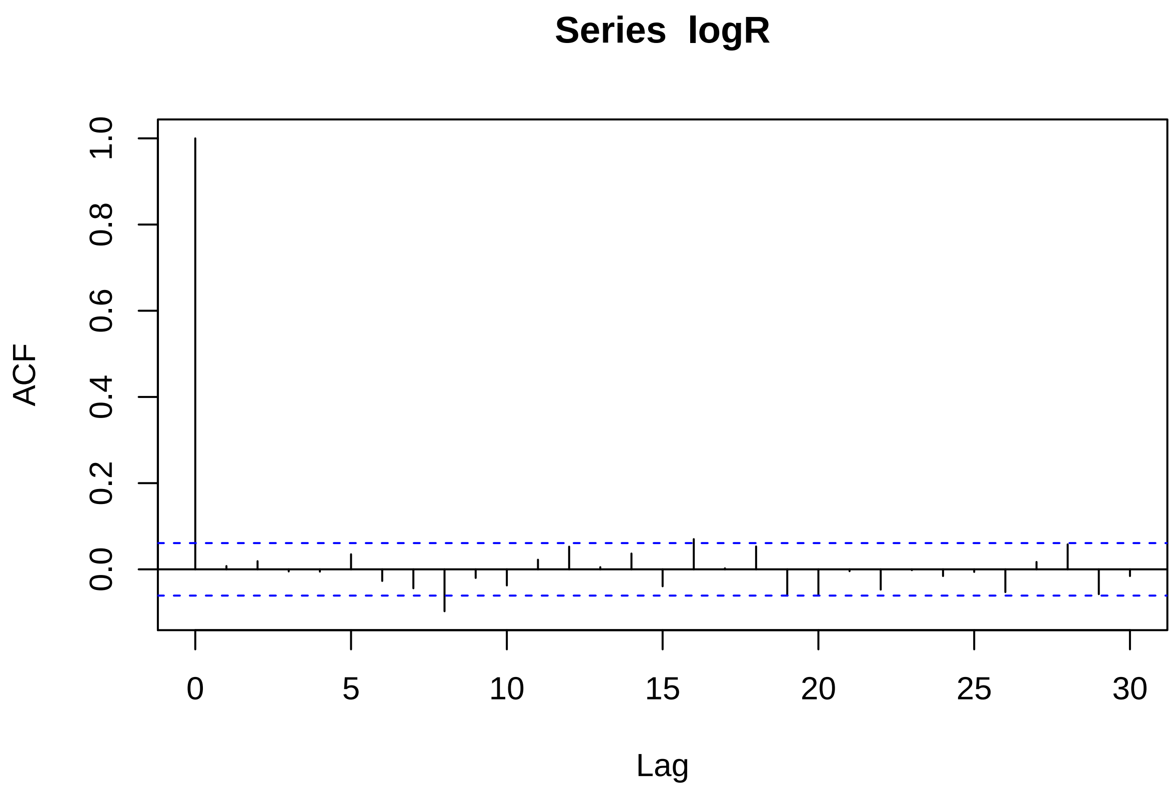
```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.194703 -0.016286  0.001515  0.018636  0.130803
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.357e-03  2.176e-03   1.083    0.279
## z.lag.1     -9.731e-01  4.432e-02 -21.958   <2e-16 ***
## tt           1.580e-06  3.603e-06   0.439    0.661
## z.diff.lag  -1.952e-02  3.146e-02  -0.621    0.535
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03358 on 1010 degrees of freedom
## Multiple R-squared:  0.4965, Adjusted R-squared:  0.495
## F-statistic:   332 on 3 and 1010 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -21.9578 160.7161 241.0737
##
## 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
```

```r
logR_mm10 <- rollmean(logR, 10, fill = list(NA, NULL, NA), align = "right")
logR_mm30 <- rollmean(logR, 30, fill = list(NA, NULL, NA), align = "right")
## weights for moving avg
```

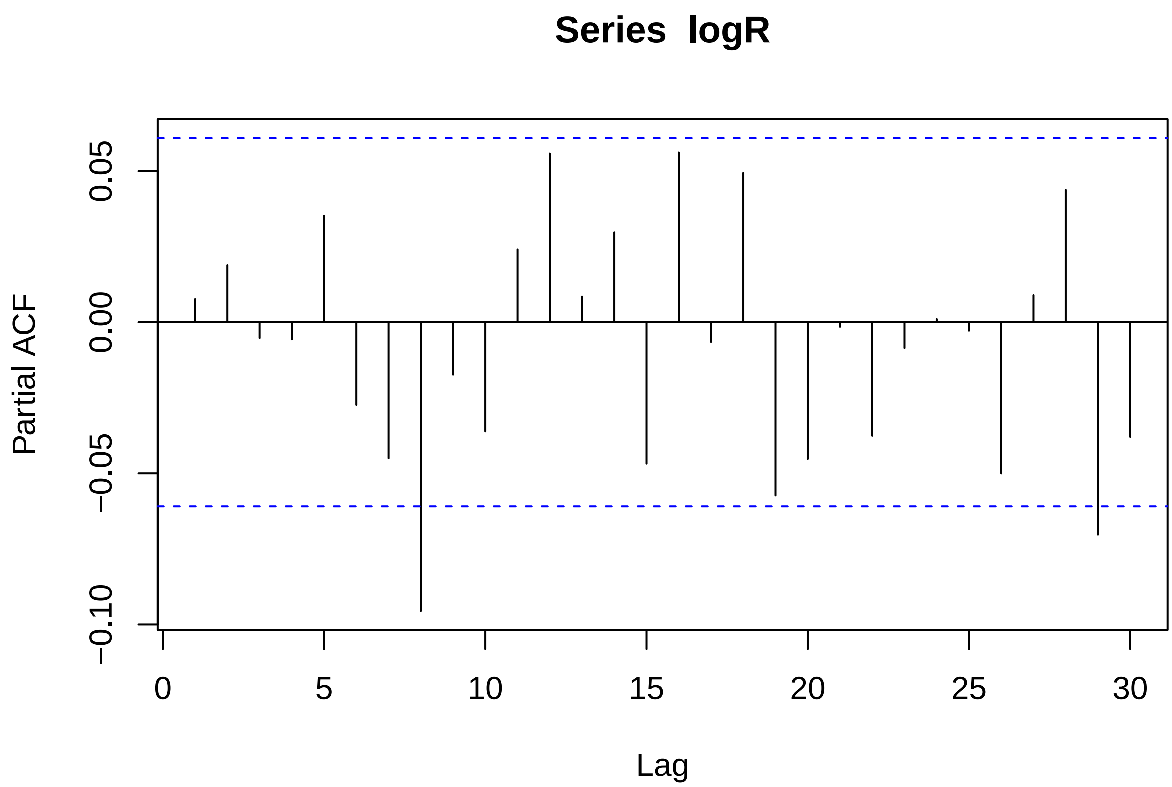
```r
#Decomposition

logR_ts <- ts(logR,frequency = 120 , start = 2019 )
logR_de <- decompose(logR_ts)
plot(logR_de)
```



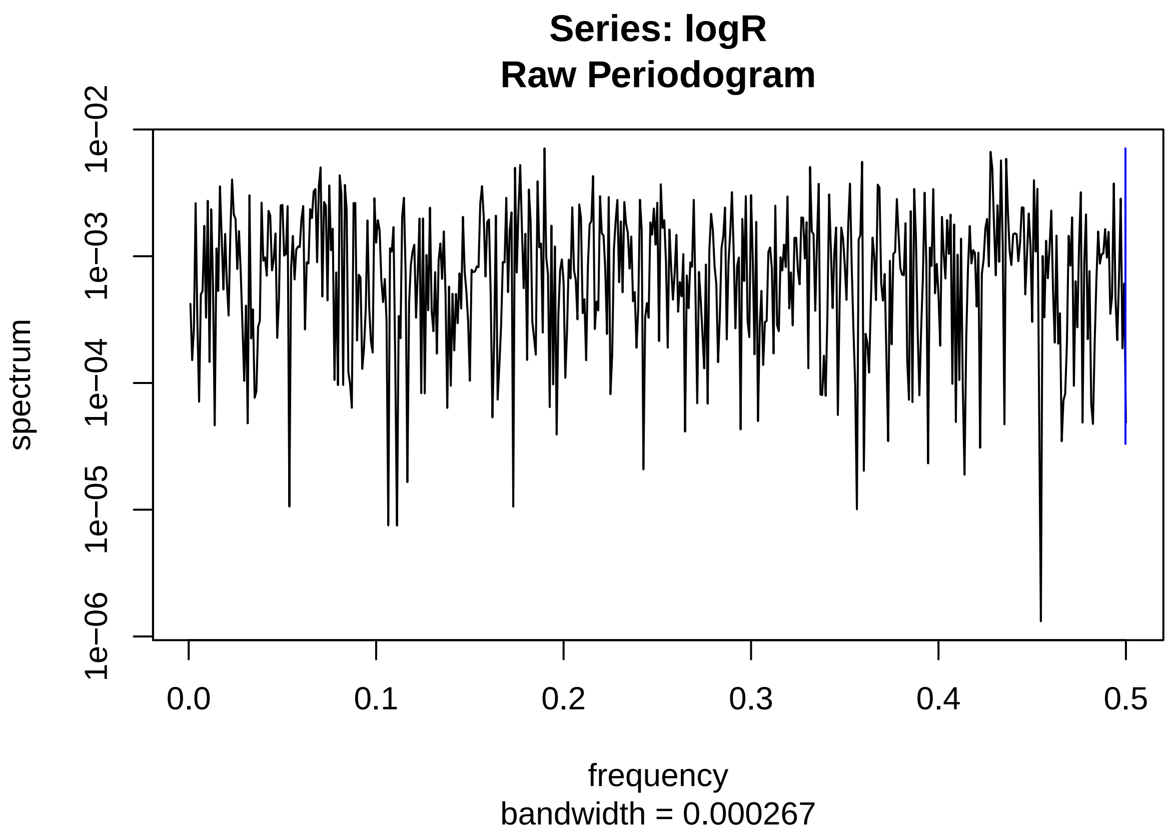

```
##Check for the seasonality
acf(logR)
```



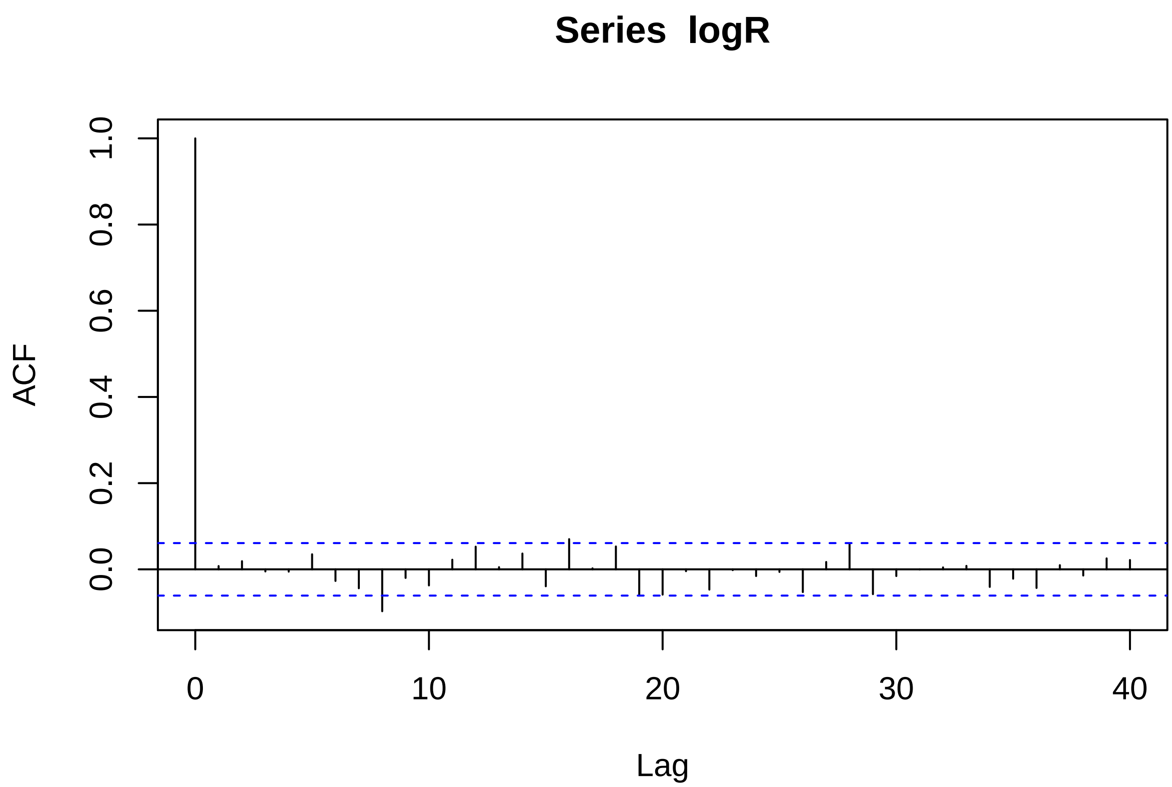

```
pacf(logR)
```



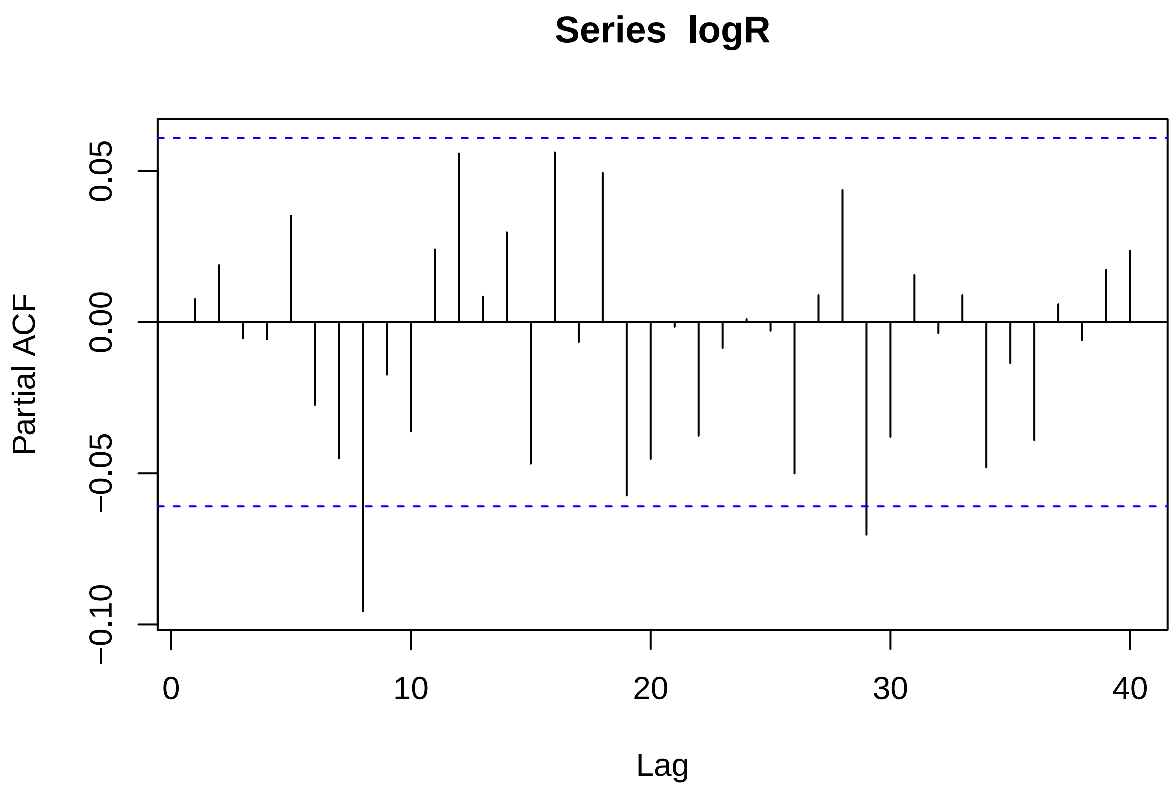

```
spec.pgram(logR)
```



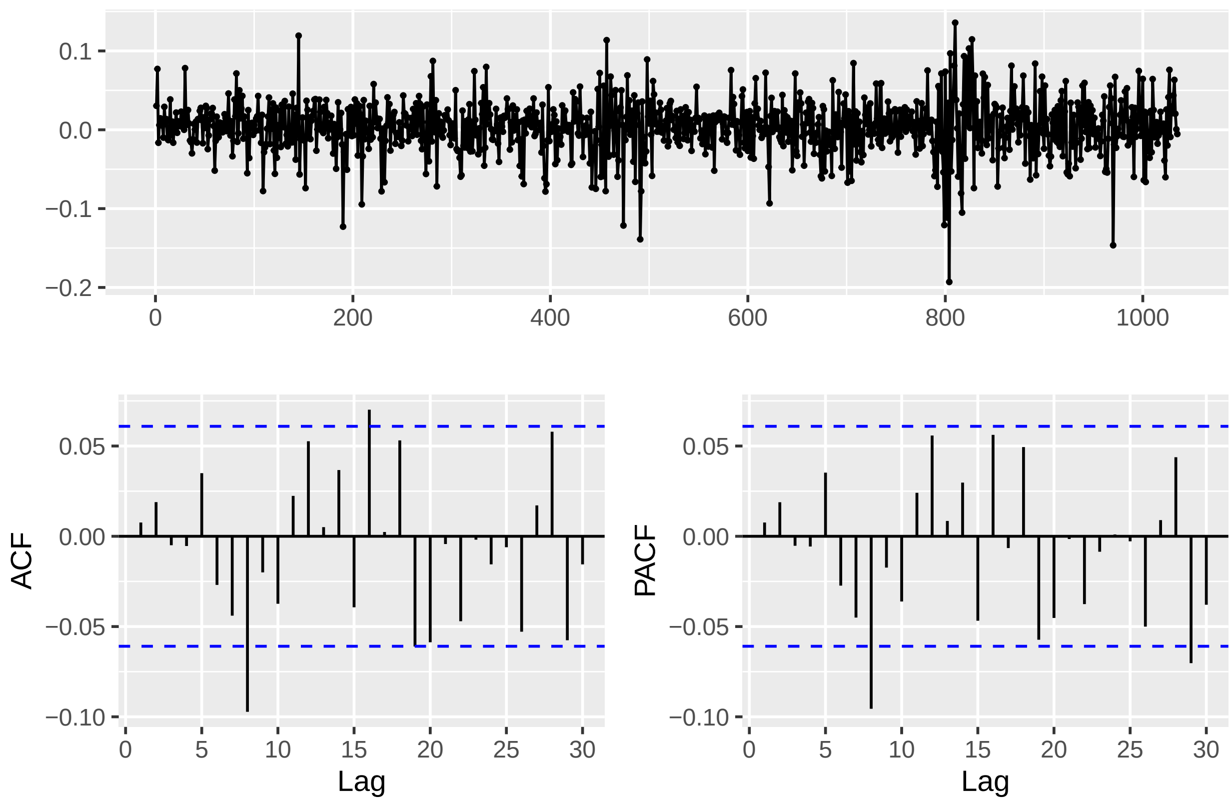

```
acf(logR,lag.max = 40)
```




```
pacf(logR,lag.max = 40)
```




```
logR %>% ggtsdisplay(main="")
```




```
## For AR models, the ACF will dampen exponentially and the PACF plot will be used to identify the orde

##The PACF shown below is suggestive of an AR(18) model. So an initial candidate model is an ##ARIMA(18

##We fit an ARIMA(18,1,0) model along with variations including ARIMA(4,1,0), ARIMA(2,1,0), ARIMA(3,1,1
```

```
#the Augmented Dickey-Fuller test, the Phillips-Perron unit root test and the KPSS test for stationarit

ndiffs(logR)
```

```
## [1] 0
```

```
adf.test(logR); pp.test(logR); kpss.test(logR)
```

```
## Warning in adf.test(logR): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  logR
## Dickey-Fuller = -10.745, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary
```

```
## Warning in pp.test(logR): p-value smaller than printed p-value
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  logR
## Dickey-Fuller Z(alpha) = -1034, Truncation lag parameter = 7, p-value =
## 0.01
## alternative hypothesis: stationary
```

```
## Warning in kpss.test(logR): p-value greater than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  logR
## KPSS Level = 0.061001, Truncation lag parameter = 7, p-value = 0.1
```

```
# The null hypothesis is rejected for ADF and Unit Root test
# For KPSS test, the null hypothesis of stationarity around a trend is not rejected since the p-value i
# Hence we can conclude that the series is stationary
```

Fit ARIMA Model

```
#AIC Table

aicc_table = function(dataset,P,Q){
  table = matrix(NA,(P+1),(Q+1))
  for (p in 0:P){
    for (q in 0:Q){
      table[p+1,q+1] = Arima(dataset,order=c(p,0,q))$aicc
    }
  }
  dimnames(table) = list(paste("<b> AR",0:P,"</b>",sep=""),paste("MA",0:Q,sep=""))
  table
}
```

```
logR_aicc_table = aicc_table(logR,10,10)
require(knitr)
```

```
## Loading required package: knitr
```

```
kable(logR_aicc_table,digits=2)
```

	MA0	MA1	MA2	MA3	MA4	MA5	MA6	MA7	MA8	MA9	MA10
AR0	-4099.30	-4097.35	-4095.72	-4093.74	-4091.76	-4091.23	-4089.73	-4089.85	-4096.92	-4095.16	-4093.96
AR1	-4097.35	-4095.44	-4093.55	-4094.70	-4092.66	-4089.49	-4093.36	-4093.90	-4095.36	-4093.33	-4091.07
AR2	-4095.71	-4093.69	-4095.40	-4093.82	-4091.85	-4095.50	-4094.58	-4096.60	-4093.44	-4091.40	-4098.53

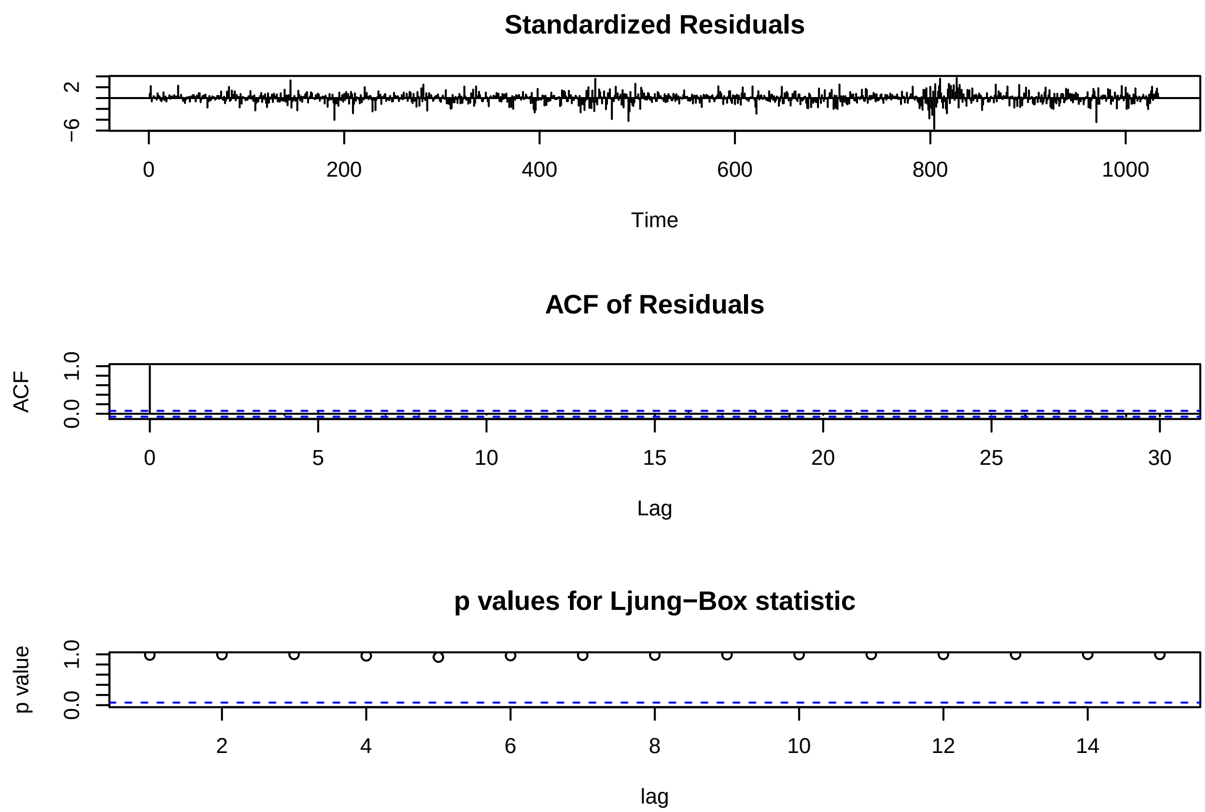
	MA0	MA1	MA2	MA3	MA4	MA5	MA6	MA7	MA8	MA9	MA10
AR3	-4093.72	-4091.69	-4093.83	-4091.80	-4090.13	-4094.33	-4095.87	-4093.83	-4091.72	-4092.28	-4096.59
AR4	-4091.73	-4089.70	-4091.86	-4089.67	-4098.30	-4092.10	-4093.98	-4091.80	-4101.61	-4100.88	-4094.49
AR5	-4090.99	-4089.73	-4093.15	-4092.17	-4096.96	-4090.75	-4088.76	-4090.16	-4099.71	-4099.60	-4097.21
AR6	-4089.73	-4093.18	-4098.45	-4092.84	-4094.79	-4096.63	-4099.09	-4103.85	-4099.00	-4097.48	-4093.56
AR7	-4089.82	-4093.53	-4096.64	-4094.86	-4092.80	-4090.66	-4102.78	-4097.10	-4097.24	-4098.91	-4099.36
AR8	-4097.35	-4095.89	-4094.01	-4093.76	-4103.38	-4101.33	-4099.72	-4100.38	-4099.50	-4098.42	-4100.91
AR9	-4095.63	-4093.85	-4092.05	-4093.68	-4101.52	-4100.37	-4098.85	-4095.32	-4099.06	-4102.62	-4097.72
AR10	-4094.98	-4093.06	-4099.93	-4097.98	-4095.99	-4099.54	-4102.89	-4100.81	-4100.32	-4098.70	-4096.10

```
## The lowest one is at AR6 & MA7, so try ARIMA(6,0,7)

##The AIC works as such: Some models, such as ARIMA(3,1,3), may offer better fit than ARIMA(2,1,3), but
```

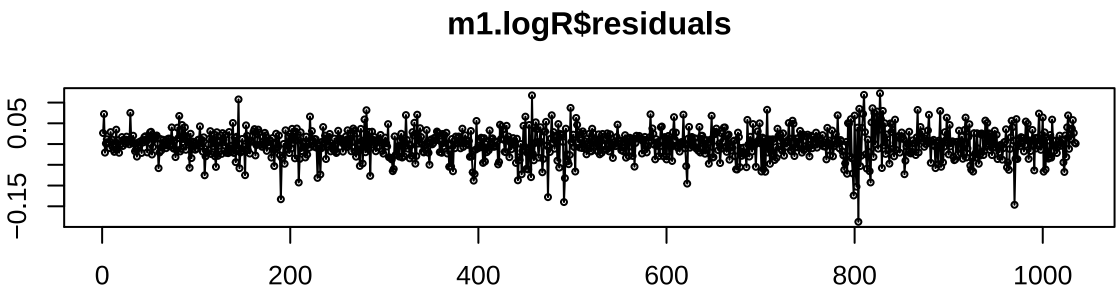
```
m1.logR=Arima(logR,order=c(6,0,7))
summary(m1.logR)
```

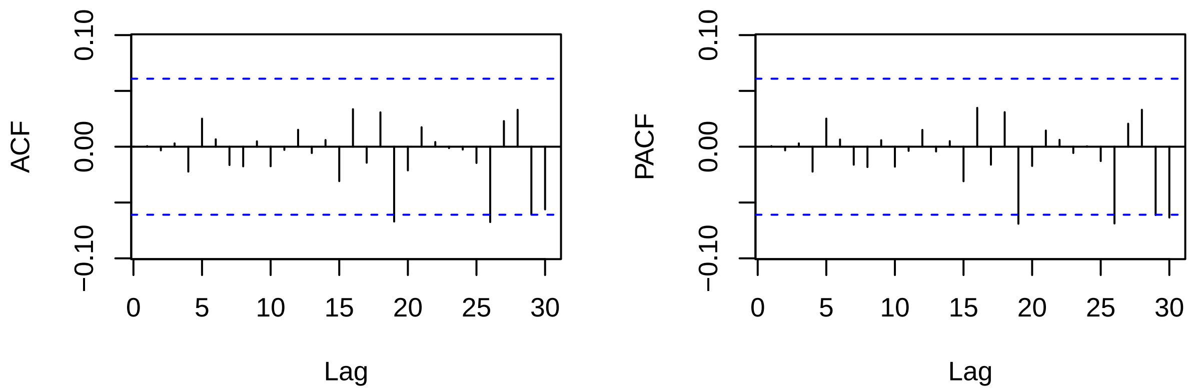
```
## Series: logR
## ARIMA(6,0,7) with non-zero mean
##
## Coefficients:
##          ar1    ar2      ar3     ar4     ar5      ar6      ma1      ma2     ma3
##       1.0872  0.294  -1.3250  0.1632  1.0986  -0.9002  -1.0847  -0.2847  1.3117
## s.e.  0.0556  0.035   0.0583  0.0451  0.0435   0.0596   0.0638   0.0512  0.0647
##           ma4      ma5     ma6     ma7    mean
##       -0.1360  -1.1039  0.8562  0.0081  0.0033
## s.e.   0.0681   0.0503  0.0805  0.0340  0.0010
##
## sigma^2 estimated as 0.001092:  log likelihood=2067.16
## AIC=-4104.32   AICc=-4103.85   BIC=-4030.19
##
## Training set error measures:
##                       ME       RMSE        MAE  MPE MAPE      MASE         ACF1
## Training set 0.000122528 0.03282064 0.02393867 -Inf  Inf 0.6861385 0.0005626716
```

```
#non-seasonal
tsdiag(m1.logR,gof=15,omit.initial=FALSE)
```




```r
#plot of residuals for diagnostic analysis: White Noise, etc.
tsdisplay(m1.logR$residuals)
```

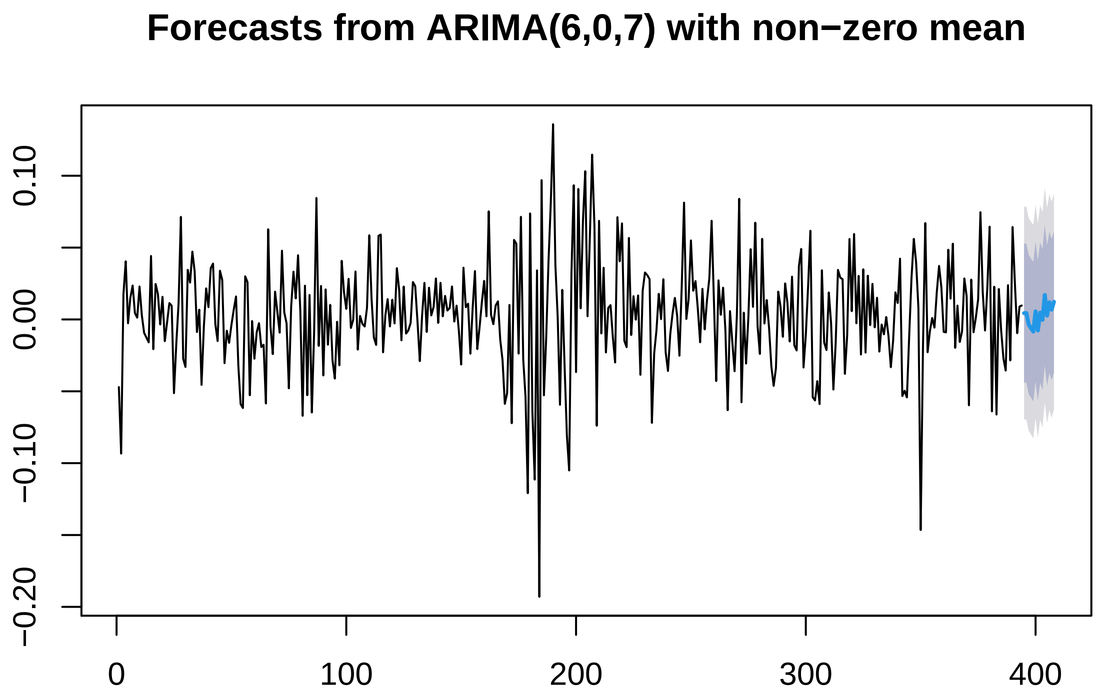





```
# From acf: no significant autocorrelation - hence white noise series
```

Further Exploration - 14 Days

```
## Here we choose training data : testing data = .95 : .05

logR1 <- logR[(0.6*length(logR)): length(logR)]
logR_train <- logR1[1:((0.95)*length(logR1))]
logR_test <- logR1[(0.95*length(logR1)):length(logR1)]
arima_train <- Arima(logR_train, order = c(6,0,7))
pred <- predict(arima_train, n.ahead = (length(logR1)-(0.95*length(logR1)) + 1))$pred
forecast <- forecast(arima_train, h = 14)
plot(forecast)
```




```
##The heavy gray bar and light gray bar separately represent the 99% and 95% confidence interval for th

accuracy(pred,logR_test)
```

```
##                     ME       RMSE        MAE      MPE     MAPE
## Test set 0.007828905 0.03397488 0.02654998 51.98492 170.5449
```

```
### I chose logR[(0.43*length(logR)): length(logR)] this portion to be the dataset which can return a r
### MAE and RMSE are two of the most common metrices to measure accuracy for continuous variable
### Still need to be interpreted: which metric should we use?
### ( FYI RMSE is used by the sample I referred)
```

```
##Trying to convert log-returns to actual prices


## Add new dates
addDate = function(lastDate, num){
  dayList <- vector(mode = "list")
  dayList[1] = format(lastDate, "%Y-%m-%d")

  for (n in 1:num){
    if (n == 1) {
      newDate = as.Date(paste(dayList[n])) + 1
   }
    else {
```

```
      newDate = as.Date.numeric(as.numeric(dayList[n])) + 1
    }
   if (weekdays(newDate) == "Saturday") {
     dayList[n+1] = newDate + 2
   }
   else if (weekdays(newDate) == "Sunday") {
     dayList[n+1] = newDate + 1
    }
   else {
      dayList[n+1] = newDate
   }
  }
   as.Date.numeric(as.numeric(dayList[2:length(dayList)]))
}
```

```
## Find actual prices
actPrice = function(lastPrice, logReturns){
  pList <- vector(mode = "list", length = length(logReturns) + 1)
  pList[1] = as.numeric(lastPrice)

  for (n in 1:length(logReturns)){
    newP = exp(log(as.numeric(pList[n])) + as.numeric(logReturns[n]))
    pList[n+1] = newP
  }
   pList[2:length(pList)]
}
```

```
lastAdj = SHOP$SHOP.Adjusted[length(SHOP$SHOP.Adjusted)]
lastMax =SHOP$SHOP.High[length(SHOP$SHOP.Adjusted)]
lastMin = SHOP$SHOP.Low[length(SHOP$SHOP.Adjusted)]

pList_Adj = actPrice(lastAdj, forecast$mean )
pList_Max = actPrice(lastMax, forecast$upper[,1] ) #here we chose Hi 80%
pList_Min = actPrice(lastMin, forecast$lower[,1] ) # and Lo 80%

SHOP_prices <- SHOP
SHOP_prices$SHOP.Close = NULL
SHOP_prices$SHOP.Open = NULL
SHOP_prices$SHOP.Volume = NULL

lastDate = index(adj_prices)[length(adj_prices)]
days_pred = addDate(lastDate, 14)


df_ini <- data.frame(Date = days_pred,
                 Name = rep(c("SHOP.High", "SHOP.Low","SHOP.Adjusted"), each = 14),
                 X = rnorm(42))
zdf <- read.zoo(file = df_ini,split = "Name")
new_prices = rbind(SHOP_prices, zdf)
new_prices$SHOP.Adjusted[535:548] <- as.numeric(pList_Adj)
new_prices$SHOP.High[535:548] <- as.numeric(pList_Max)
new_prices$SHOP.Low[535:548] <- as.numeric(pList_Min)
```

```r
tail(new_prices)
```

```
##            SHOP.High SHOP.Low SHOP.Adjusted
## 2021-02-25  2265.856 918.0984      1445.738
## 2021-02-26  2420.310 889.2368      1470.527
## 2021-03-01  2548.530 849.0346      1474.472
## 2021-03-02  2708.930 818.1074      1492.220
## 2021-03-03  2864.771 784.1955      1502.401
## 2021-03-04  3046.376 755.8448      1521.027
```

```r
fig <- plot(new_prices['2020-12/2021-03'], col = c('palegreen3','salmon','orange'),
            main = "Upcoming 7 Days Price Range Prediction for SHOP")
addLegend("topleft",
              legend.names=c("Max. Price", "Min. Price", "Adj. Price"),
              col=c('palegreen3','salmon','orange'),
              lty=c(1,1,1),
              lwd=c(2,2,2),
              bg="white")
```