ADA.R

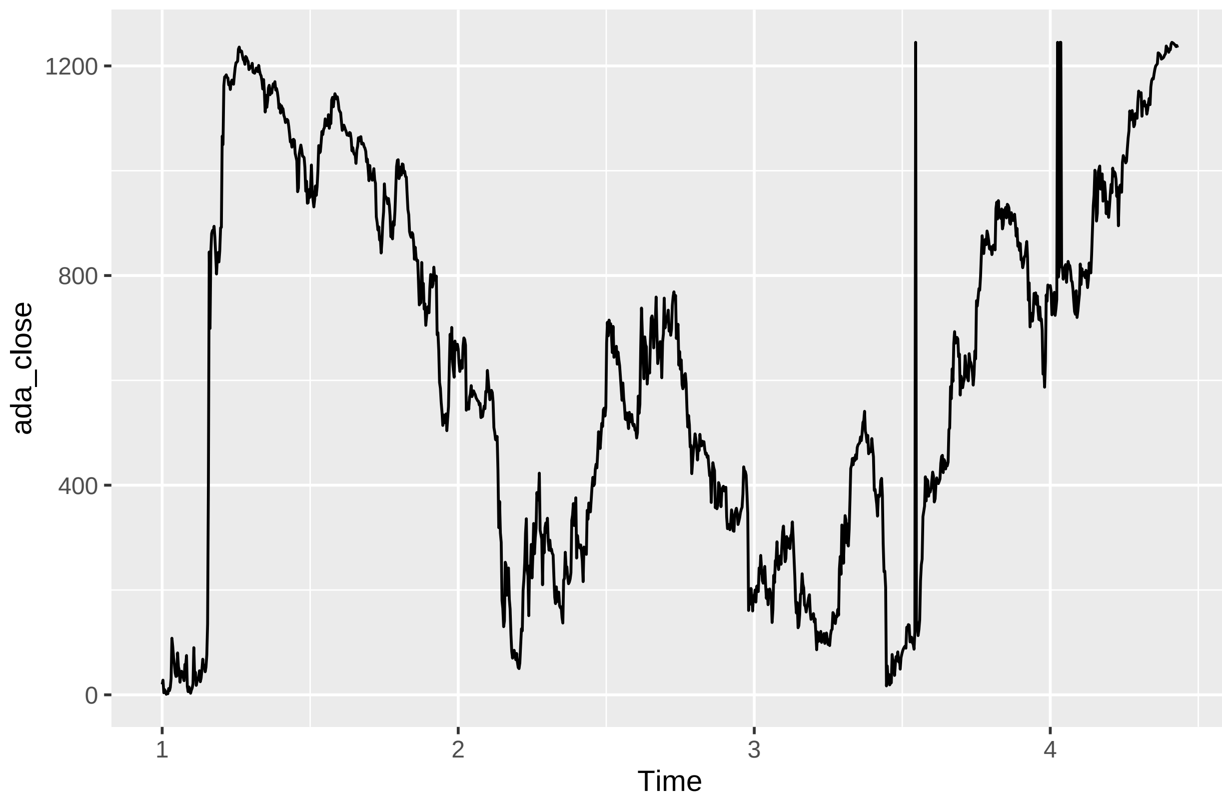
romeoleon

2021-03-06

```r
library(ggplot2)
library(ggfortify)
library(readxl)
library(forecast)
```

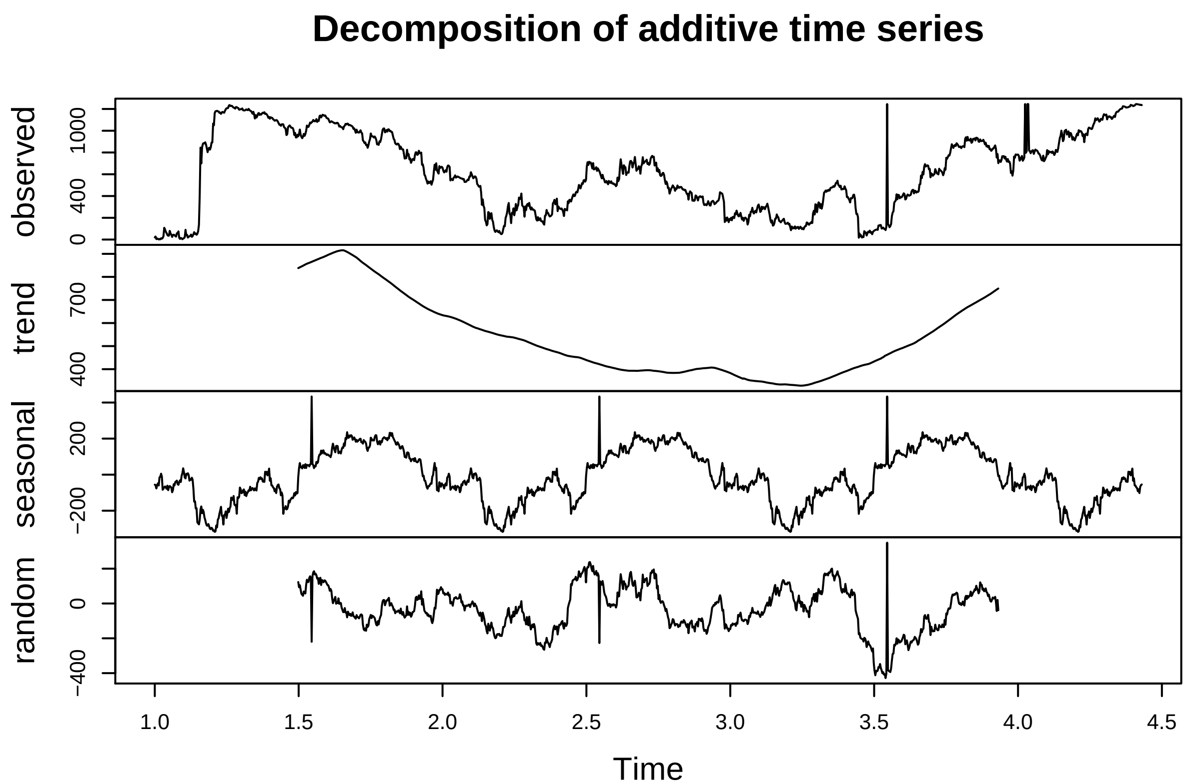
```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

## Registered S3 methods overwritten by 'forecast':
##   method                 from
##   autoplot.Arima         ggfortify
##   autoplot.acf           ggfortify
##   autoplot.ar            ggfortify
##   autoplot.bats          ggfortify
##   autoplot.decomposed.ts ggfortify
##   autoplot.ets           ggfortify
##   autoplot.forecast      ggfortify
##   autoplot.stl           ggfortify
##   autoplot.ts            ggfortify
##   fitted.ar              ggfortify
##   fortify.ts             ggfortify
##   residuals.ar           ggfortify
```

```r
ada <- read.csv('/Users/romeoleon/Desktop/Python & R/Stock prediction/ADA-USD (1).csv')

#Convert to time series
ada_ts <- ts(ada,frequency = 365)
ada_close <- ada_ts[,'Close']

####BELOW THE INTERMEDIATE STEPS BEFORE AUTO ARIMA####
#Plot data
autoplot(ada_close)
```




```r
#Plot seasonality
ggseasonplot(ada_close, season.labels = NULL,xlab='year')
```

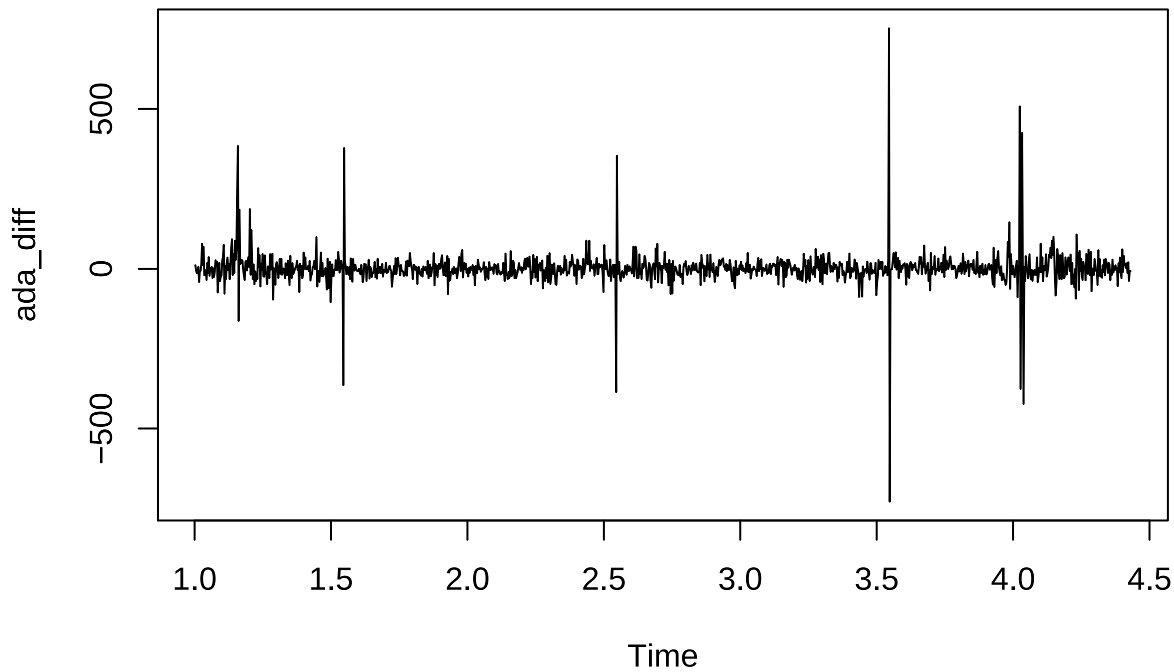

```
#Decompose the data
decompose_ada <- decompose(ada_close)
plot(decompose_ada)
```



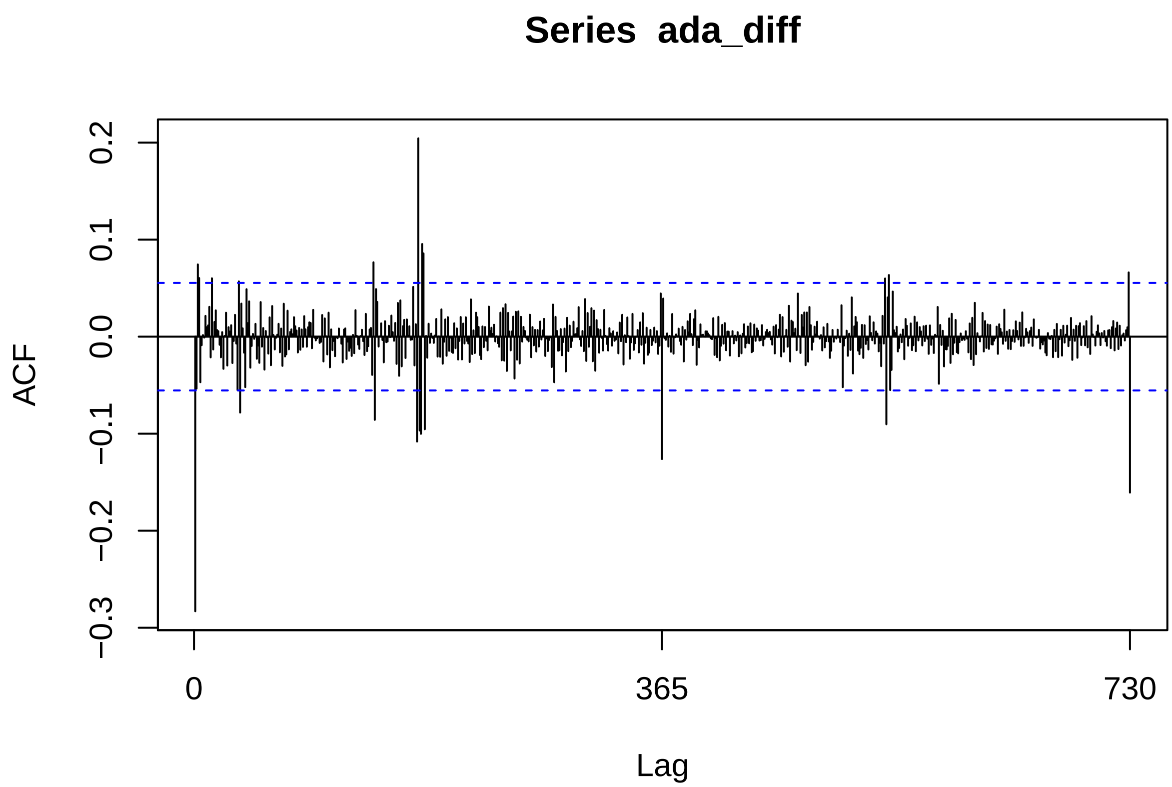

```r
#Select seasonality
ada_seasonal <- decompose_ada$seasonal

#Make data stationary
ada_nos <- ada_close - ada_seasonal

#Differencing the data
ndiffs(ada_nos)
```

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

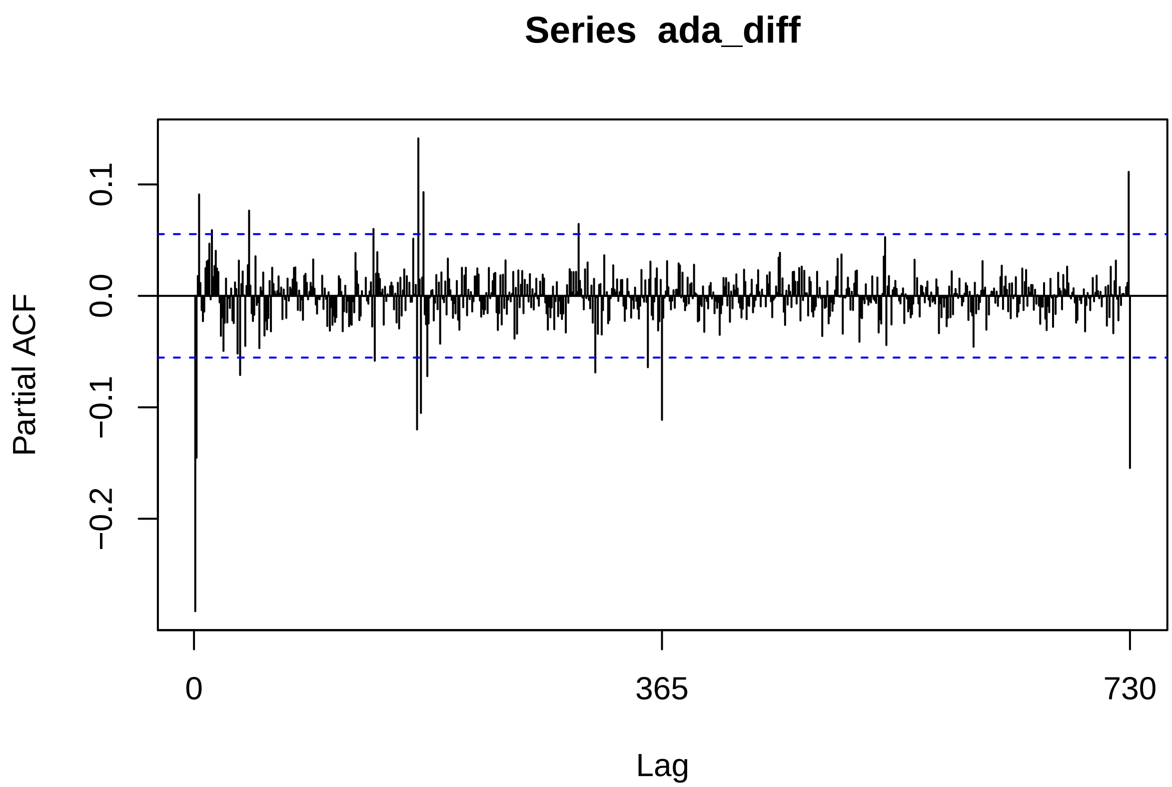
```r
ada_diff <- diff(ada_nos)
plot(ada_diff)
```




```r
#plot ACF and PACF
Acf(ada_diff,plot=TRUE)
```




```
Pacf(ada_diff,plot=TRUE)
```



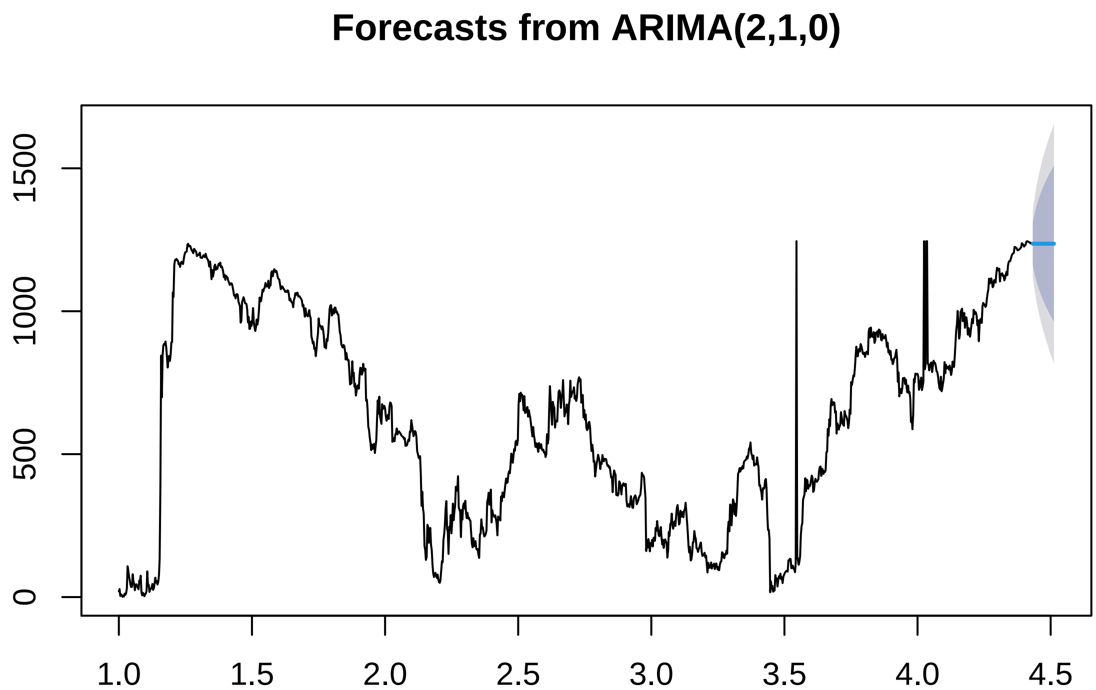

```
##############################

#Do all step automatically
arima_train <- auto.arima(ada_close[1:1002],stationary = FALSE,seasonal = TRUE)
accuracy(arima_train)
```

```
##                     ME     RMSE    MAE       MPE     MAPE     MASE        ACF1
## Training set 0.8837463 57.65667 25.009 -4.973976 12.84513 1.037765 -0.01379517
```

```
arima_test <- Arima(ada_close[1002:1253],model=arima_train)
accuracy(arima_test)
```

```
##                    ME     RMSE      MAE       MPE     MAPE     MASE        ACF1
## Training set 3.725329 58.55377 25.72923 0.1648827 2.875957 1.015096 -0.04505717
```

```
#Create model with all data as training
arima_final <- auto.arima(ada_close,stationary = FALSE,seasonal = TRUE) #Check if AIC is better with fu
ada_forecast <- forecast(arima_final,h=30)
plot(ada_forecast)
```




```r
summary(ada_forecast)
```

```
##
## Forecast method: ARIMA(2,1,0)
##
## Model Information:
## Series: ada_close
## ARIMA(2,1,0)
##
## Coefficients:
##           ar1      ar2
##       -0.3528  -0.1607
## s.e.   0.0279   0.0279
##
## sigma^2 estimated as 3342:  log likelihood=-6855.11
## AIC=13716.23   AICc=13716.25   BIC=13731.62
##
## Error measures:
##                    ME   RMSE      MAE       MPE     MAPE       MASE
## Training set 1.468577 57.739 25.43847 -3.875866 10.90877 0.05416463
##                      ACF1
## Training set -0.005180515
##
## Forecasts:
##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
```

```
## 4.432877       1236.090 1162.0055 1310.174 1122.7877 1349.392
## 4.435616       1236.348 1148.1024 1324.594 1101.3879 1371.309
## 4.438356       1236.082 1136.9062 1335.258 1084.4058 1387.758
## 4.441096       1236.134 1124.8828 1347.386 1065.9898 1406.279
## 4.443836       1236.159 1114.5845 1357.733 1050.2270 1422.090
## 4.446575       1236.142 1105.1816 1367.102 1035.8555 1436.428
## 4.449315       1236.144 1096.3073 1375.980 1022.2823 1450.005
## 4.452055       1236.146 1087.9856 1384.306 1009.5543 1462.737
## 4.454795       1236.145 1080.1141 1392.175  997.5165 1474.773
## 4.457534       1236.145 1072.6156 1399.674  986.0485 1486.241
## 4.460274       1236.145 1065.4470 1406.843  975.0850 1497.205
## 4.463014       1236.145 1058.5681 1413.722  964.5646 1507.725
## 4.465753       1236.145 1051.9456 1420.344  954.4364 1517.853
## 4.468493       1236.145 1045.5531 1426.737  944.6599 1527.630
## 4.471233       1236.145 1039.3681 1432.921  935.2009 1537.089
## 4.473973       1236.145 1033.3718 1438.918  926.0302 1546.259
## 4.476712       1236.145 1027.5477 1444.742  917.1231 1555.167
## 4.479452       1236.145 1021.8819 1450.408  908.4580 1563.832
## 4.482192       1236.145 1016.3621 1455.928  900.0162 1572.273
## 4.484932       1236.145 1010.9776 1461.312  891.7812 1580.508
## 4.487671       1236.145 1005.7188 1466.571  883.7387 1588.551
## 4.490411       1236.145 1000.5775 1471.712  875.8756 1596.414
## 4.493151       1236.145  995.5459 1476.744  868.1806 1604.109
## 4.495890       1236.145  990.6175 1481.672  860.6432 1611.646
## 4.498630       1236.145  985.7860 1486.504  853.2541 1619.036
## 4.501370       1236.145  981.0461 1491.244  846.0050 1626.285
## 4.504110       1236.145  976.3926 1495.897  838.8881 1633.402
## 4.506849       1236.145  971.8210 1500.469  831.8965 1640.393
## 4.509589       1236.145  967.3272 1504.962  825.0238 1647.266
## 4.512329       1236.145  962.9073 1509.382  818.2641 1654.026
```