Forecasting using GARCH and ARIMA implementation

Tanishq Chauhan

3/23/2020

Introduction

In our analysis, we will study the data set from JP Morgan, S&P500, and the National Home Price Index (NHPI). For the first two data sets, we will do some basic arithmetic calculations and implement a two-variable regression as well. The latter then will be compared to a similar calculation done using MS Excel. With the NHPI dataset we will forecast it using an ARMA model, plot the results and try to find any irregularities in its trend. We will then attempt to smoothen the time-series process. This will make it easier to decompose the process and analyze for cycles, trends, and seasonality. This process will be tested for stationarity using the Augmented Dickey-Fuller (ADF) Test. Depending on whether the null hypothesis for the test is rejected or not rejected, we will proceed. If the null hypothesis is not rejected and the process shows non-stationarity, we will model the process as an ARIMA non-stationary model. Then we will conduct the ADF test on this ARIMA process. After a successful ADF test about the stationarity of this model, the Auto Correlation Function (ACF) and the Partial Auto Correlation Function (PACF) will be implemented to determine the values of the order of the autoregressive model (p), and the order of the moving average model (q). Finally we will forecast the NPHI dataset and, use the ACF and the PACF for examining model residuals.

For project reference purposes, we will be using Tsay(2015)[1] for theory and Ivo Dinov's (2020)[4] online notes for coding.

Section I: Basic Statistics

Calculation in R

We have downloaded JP Morgan stock historical prices from Yahoo Finance with the following characteristics:

1. Period: February 1, 2018 – December 30, 2018
2. Frequency: Daily
3. Price considered in the analysis: Close price adjusted for dividends and splits

###Using this data and R as the programming language, calculate the following: 1. Average stock value 2. Stock volatility 3. Daily stock return

```
JPMorganData <- read.csv("JPM.csv",sep=',')
JPMorganAdjClose = JPMorganData[["Adj.Close"]]
averageJPMReturns = mean(JPMorganAdjClose)
rows = length(JPMorganAdjClose)
JPMReturns <- log(JPMorganAdjClose[2:rows]/JPMorganAdjClose[1:(rows-1)])
stdDev = sd(JPMReturns)
nBusinessDays = 252
```

```
volatility = stdDev * sqrt(nBusinessDays)
cat("Average stock value = ", averageJPMReturns, "\n")
```

```
## Average stock value =  105.6712
```

```
cat("Stock volatility = ", volatility, "\n")
```

```
## Stock volatility =  0.2288891
```

```
cat("Daily JPMReturns\n")
```

```
## Daily JPMReturns
```

```
print(JPMReturns)
```

```
##   [1] -2.241070e-02 -4.914030e-02  2.996933e-02  6.756138e-03 -4.521739e-02
##   [6]  1.982453e-02  1.533092e-02  6.156051e-03  2.286208e-02  4.164246e-03
##  [11] -7.211529e-03  2.615456e-04  4.175774e-03 -1.824707e-03  2.006159e-02
##  [16]  1.236901e-02 -1.194269e-02 -1.597563e-02 -1.808460e-02 -9.703152e-04
##  [21]  1.523806e-02  8.688225e-04 -3.740974e-03  8.715148e-05  2.835476e-02
##  [26] -3.224384e-03 -1.205605e-02 -1.124589e-02  2.519792e-03  1.733970e-03
##  [31] -7.914070e-03  9.600042e-04  8.718660e-04 -4.264305e-02 -2.710338e-02
##  [36]  3.037228e-02 -1.959062e-02 -1.572662e-03  1.807634e-02 -1.946626e-02
##  [41]  1.362955e-02  1.506918e-02  1.304502e-02 -2.525361e-02  1.193690e-02
##  [46]  1.893202e-02 -1.694111e-02  2.455583e-02 -2.745290e-02 -8.163382e-04
##  [51]  0.000000e+00 -8.108301e-03  2.171644e-02 -2.240232e-03 -4.856140e-03
##  [56] -4.698652e-03 -3.811300e-03  9.995892e-04 -6.378107e-03 -5.683486e-03
##  [61]  0.000000e+00 -7.937240e-03 -6.320903e-03  1.103544e-02  8.631859e-03
##  [66]  1.470345e-02  2.156969e-02  7.729343e-03 -3.769397e-03  3.512135e-04
##  [71] -7.667749e-03  2.738924e-03 -3.358239e-03 -1.633314e-02  9.136613e-03
##  [76]  7.638934e-03 -4.611972e-03 -1.126404e-02 -5.137698e-03 -4.368403e-02
##  [81]  2.258817e-02 -1.244435e-02  1.290580e-02  4.610874e-04 -5.640630e-03
##  [86]  2.309921e-02  3.979017e-03  2.793972e-03 -2.523183e-03 -5.791386e-03
##  [91] -1.998543e-03 -1.779869e-02 -1.204109e-03  2.591689e-03 -5.840716e-03
##  [96] -9.303656e-05 -2.789271e-04 -1.650600e-02 -9.119611e-03  6.678941e-04
## [101] -1.556972e-02  1.623709e-02 -6.981424e-03  8.409809e-03 -1.408804e-02
## [106]  6.480594e-03  3.272699e-03  3.047460e-02 -6.171084e-03 -2.159499e-03
## [111]  4.314344e-03 -4.596404e-03  3.890972e-02 -7.237864e-04  9.278099e-03
## [116] -1.481369e-02  1.256960e-02  1.843089e-02  7.032880e-03  8.982893e-03
## [121] -2.869209e-03  1.022183e-02  6.014788e-03 -1.536633e-02  6.157580e-03
## [126]  4.227596e-03  8.060439e-03  2.562864e-04  3.664636e-03  2.039639e-03
## [131] -7.755728e-03 -9.887762e-03 -1.602684e-02  6.651001e-03 -8.320688e-03
## [136]  9.366755e-03  0.000000e+00 -1.307875e-03  6.088627e-03 -3.039712e-03
## [141] -2.089647e-03 -4.358565e-04  1.754648e-02 -4.895846e-03 -3.277241e-03
## [146] -4.936196e-03 -5.309673e-03  4.962401e-03 -4.875143e-03 -4.285136e-03
## [151]  1.926266e-03 -5.350262e-03  6.312038e-03 -1.186785e-02  3.883405e-03
## [156] -1.761707e-04  2.991258e-03  4.032570e-03  2.863249e-02  8.550339e-03
## [161] -6.596750e-03 -9.634833e-03 -2.831299e-03 -1.184052e-02 -4.356596e-03
## [166] -1.477850e-02  5.831845e-03  4.132424e-03  9.344741e-03  8.975525e-03
## [171] -5.654808e-03  6.088642e-03 -6.961518e-03 -2.699384e-02 -3.042143e-02
## [176] -1.097266e-02 -5.719957e-03  2.121400e-02  1.107822e-02 -1.596961e-02
## [181] -1.666544e-03 -1.446799e-02 -1.049116e-02 -1.879784e-02  1.508558e-02
## [186] -1.382779e-02  1.373241e-02  1.749028e-02  2.151017e-02 -3.668534e-04
## [191] -5.520832e-03  6.529596e-03  4.664267e-03  1.700787e-02  8.040673e-03
## [196] -9.746569e-03 -2.125033e-02  5.857039e-03 -2.083788e-02  2.520831e-02
## [201] -7.271121e-04  7.608113e-03 -2.170834e-02 -7.496897e-03 -9.239811e-03
```

```
## [206]  2.417790e-02  4.201253e-03  1.105791e-02 -7.963870e-03  1.021479e-02
## [211]  9.398954e-03 -4.566330e-02 -1.920787e-02 -1.822765e-02 -1.886194e-02
## [216] -9.815350e-03  6.455134e-03  9.895493e-04 -8.242094e-03 -1.284507e-02
## [221] -4.758249e-03 -1.276643e-02 -8.671474e-03 -2.392303e-02 -2.179238e-02
## [226]  4.062215e-02  1.119190e-02 -2.166359e-03
```

Section II: Linear Regression

Implementing a two-variable regression in R, based on the following

1. Explained variable: JP Morgan stock (adjusted close price)
2. Explanatory variable: S&P500
3. Period: February 1, 2018 – December 30, 2018
4. Frequency: Daily

```
SP500data <- read.csv("S&P500.csv")
lm(JPMorganAdjClose ~SP500data$Adj.Close)
```

```
##
## Call:
## lm(formula = JPMorganAdjClose ~ SP500data$Adj.Close)
##
## Coefficients:
##         (Intercept)  SP500data$Adj.Close
##            13.55492              0.03358
```

Section III: Univariate Time Series

Forecast using the ARMA model and testing using Augmented Dickey-Fuller test.

We Download the following data:

1. Datasource:https://fred.stlouisfed.org/series/CSUSHPISA
2. Period considered in theanalysis: January 1987 – latest data
3. Frequency: monthly data

With this data, do the following using R: ### 1.Forecast S&P/Case-Shiller U.S. National Home Price Index using an ARMA model.

The first thing we do is plot the series. We start by loading the housing price dataset.

```
CSUSHPISA_Data <- read.csv("CSUSHPISA.csv")

library(forecast)
```

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

```
library(xts)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

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

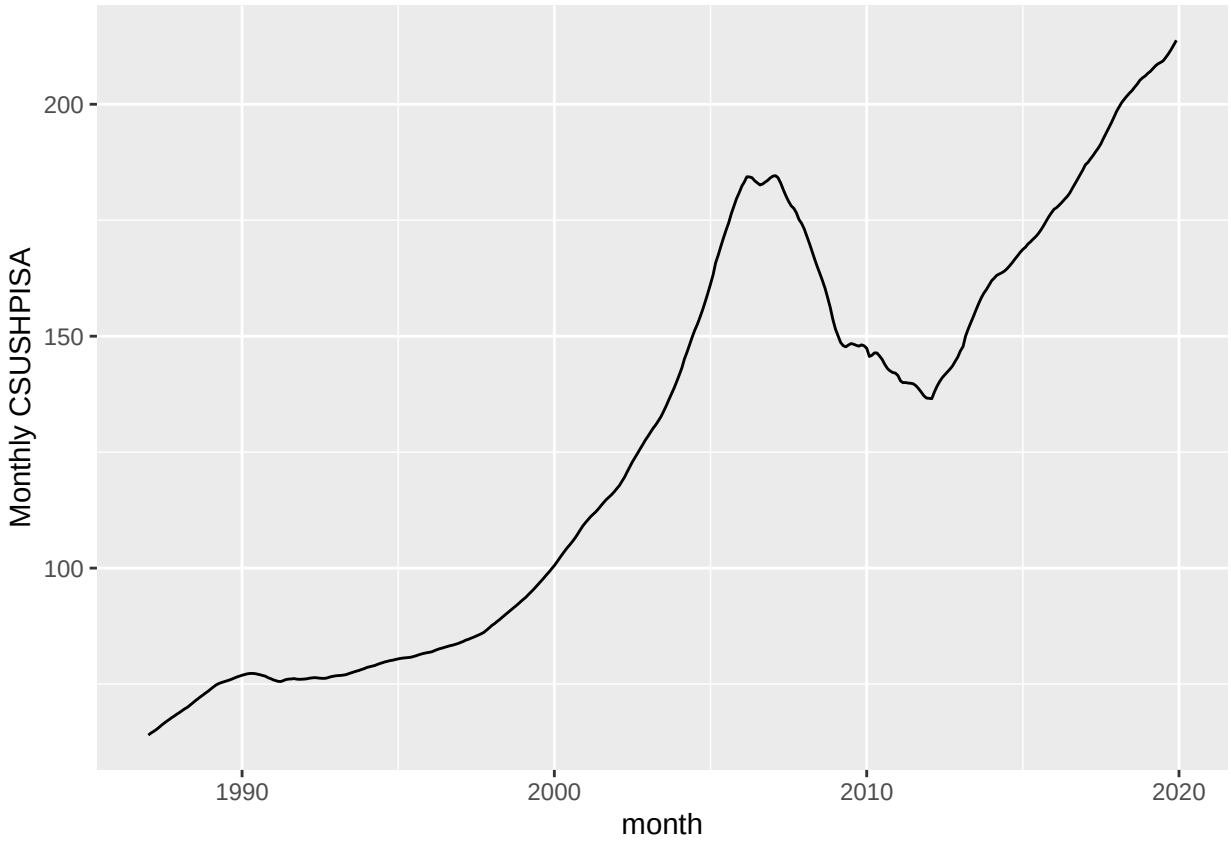
```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::first()  masks xts::first()
## x dplyr::lag()    masks stats::lag()
## x dplyr::last()   masks xts::last()
```

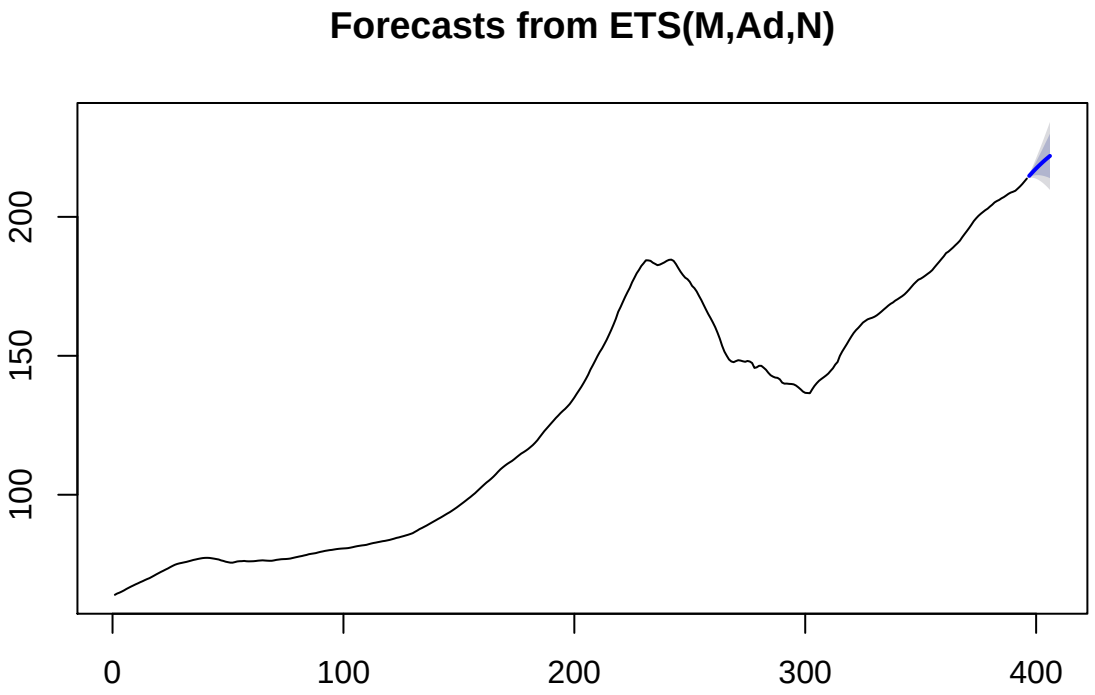
```
library(stats)
library(tseries)

CSUSHPISA_Data$DATE = as.Date(CSUSHPISA_Data$DATE)
# generally plotting home price index
ggplot(CSUSHPISA_Data, aes(DATE, CSUSHPISA)) + geom_line() + scale_x_date('month')  + ylab("Monthly CSUS
            xlab("")
```



```r
summary(CSUSHPISA_Data)
```

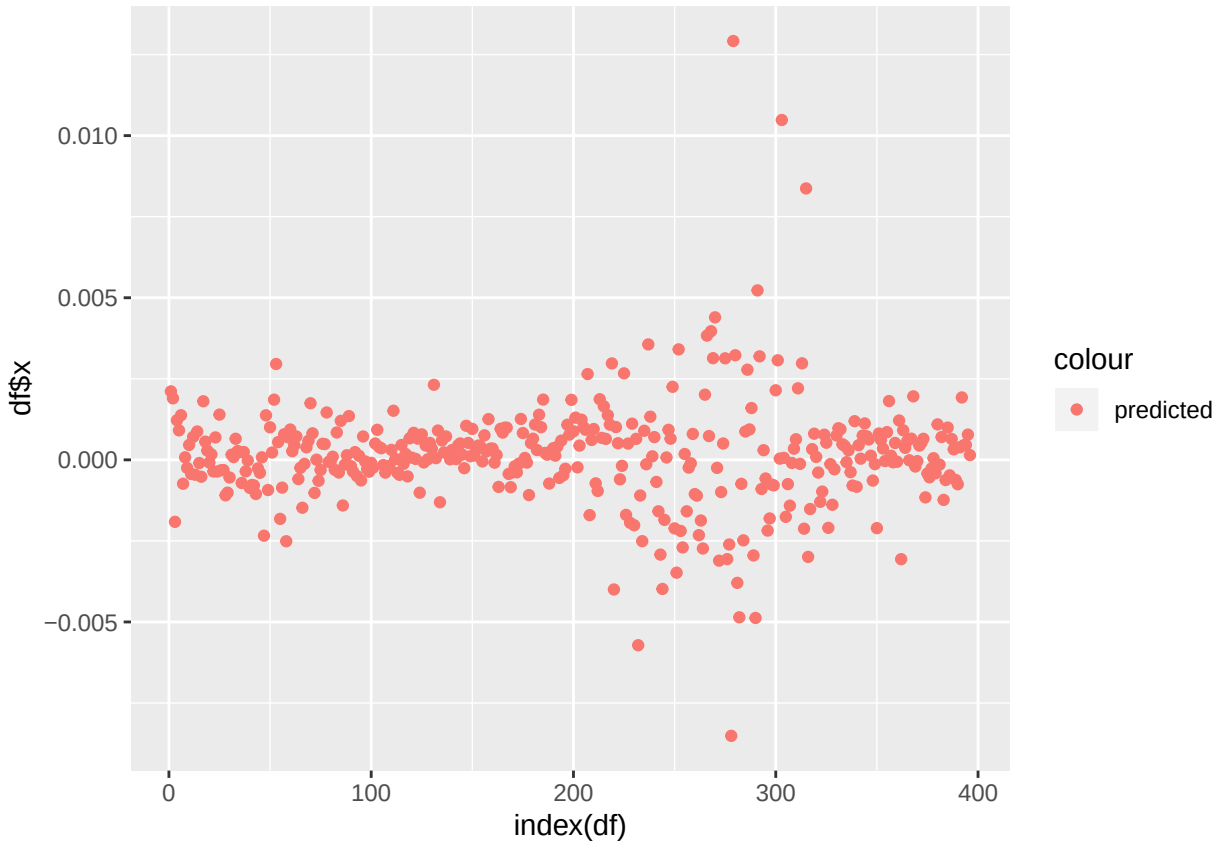
```
##       DATE              CSUSHPISA
##  Min.   :1987-01-01   Min.   : 63.99
##  1st Qu.:1995-03-24   1st Qu.: 80.66
##  Median :2003-06-16   Median :133.21
##  Mean   :2003-06-16   Mean   :127.51
##  3rd Qu.:2011-09-08   3rd Qu.:166.84
##  Max.   :2019-12-01   Max.   :213.79
```

```r
plot(forecast(CSUSHPISA_Data$CSUSHPISA))
```

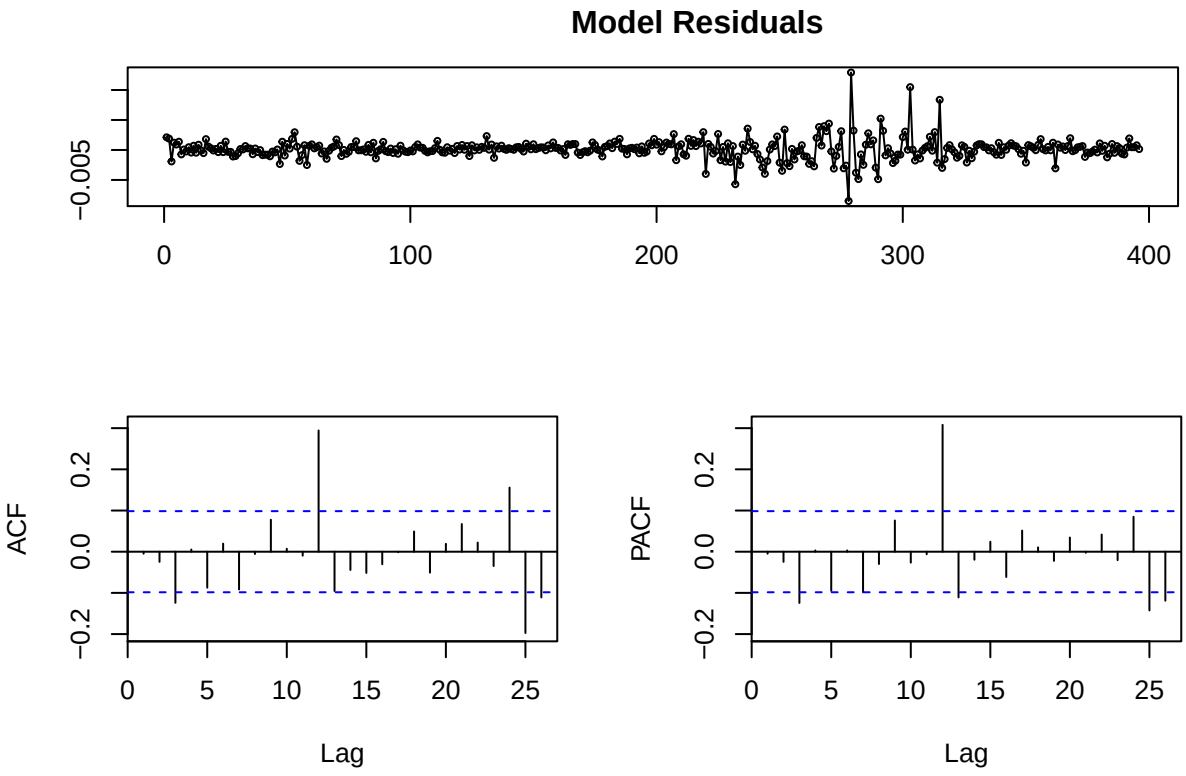



```r
df = as.data.frame(forecast(CSUSHPISA_Data$CSUSHPISA)$residuals)
ggplot(df, aes(index(df))) +
    geom_point(aes(y = df$x, color="predicted"))
```

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




```
tsdisplay(residuals(forecast(CSUSHPISA_Data$CSUSHPISA)), main='
          Model Residuals')
```



2.Implementing the Augmented Dickey-Fuller Test for checking the existence of a unitroot in Case-Shiller Index series

We conduct a preliminary ADF test on the home price series.

```
#conducting an ADF test
adf_test <- adf.test(CSUSHPISA_Data$CSUSHPISA,alternative = 'stationary')
print(adf_test)
```

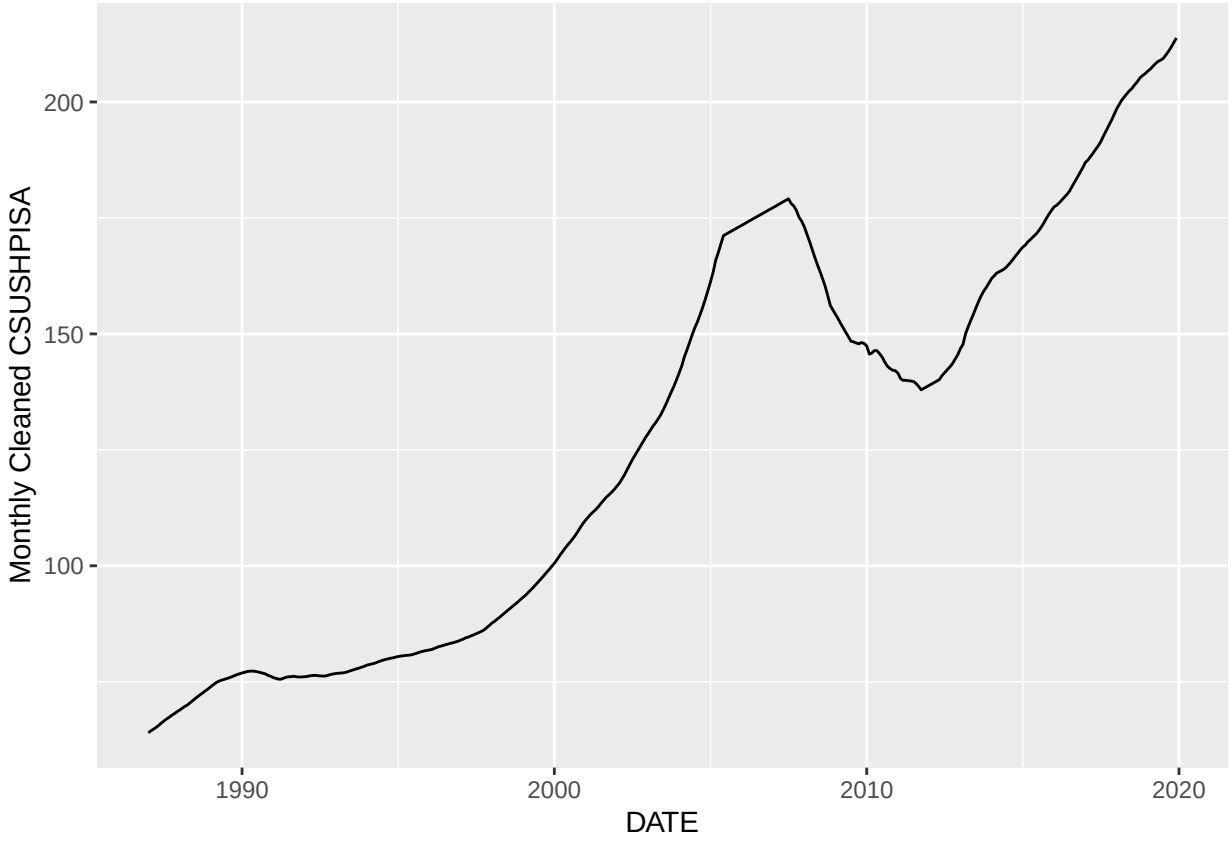
```
##
##  Augmented Dickey-Fuller Test
##
## data:  CSUSHPISA_Data$CSUSHPISA
## Dickey-Fuller = -3.1079, Lag order = 7, p-value = 0.1093
## alternative hypothesis: stationary
```

First let us have a brief primer on the Augmented Dickey-Fuller (ADF) test. ADF is a test for the stationarity of a given series (a unit-root test for stationarity). A time-series with a unit root exhibits unpredictable pattern. The null hypothesis is that there is a unit root, i.e. a random walk with a drift (suggesting nonstationarity). The alternative hypothesis varies with the model being tested.

Conducting a preleminary ADF test on the home price index datasets, the p-value is clearly greater than 0.05 (which is to what we have set our alpha level). Hence, we can conclude that the price series is nonstationary and a unit root exists. Thus we should use Integrated ARMA model i.e. ARIMA `(p,d,q)` model, and we cannot apply an ARMA`(p,q)` model to this series. However, we can use the `tsclean()` function to identify and replace outliers. This is done using series smoothing and decomposition. Doing so will input missing values in the series and replace outliers. We do so to successfully be able to run the series through an ADF test and show stationarity.
ts() command to create a time series object to pass to tsclean().

```
# create time series object using ts() to pass on to tsclean()
CSUSHPISA_ts = ts(CSUSHPISA_Data[, c('CSUSHPISA')])

CSUSHPISA_Data$clean_CSUSHPISA = tsclean(CSUSHPISA_ts)

CSUSHPISA_Data <- CSUSHPISA_Data[,c(1,3)]

ggplot() +  geom_line(data = CSUSHPISA_Data, aes(x = DATE, y = clean_CSUSHPISA)) +
  ylab('Monthly Cleaned CSUSHPISA')
```

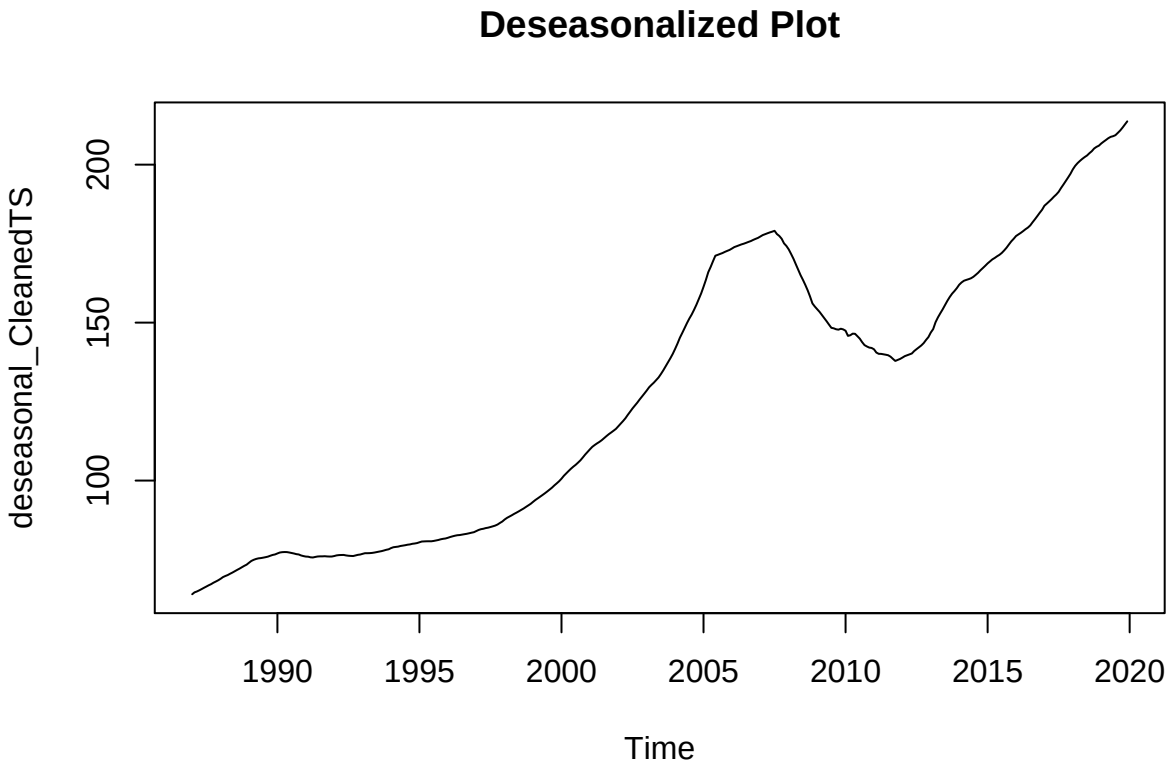
```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```



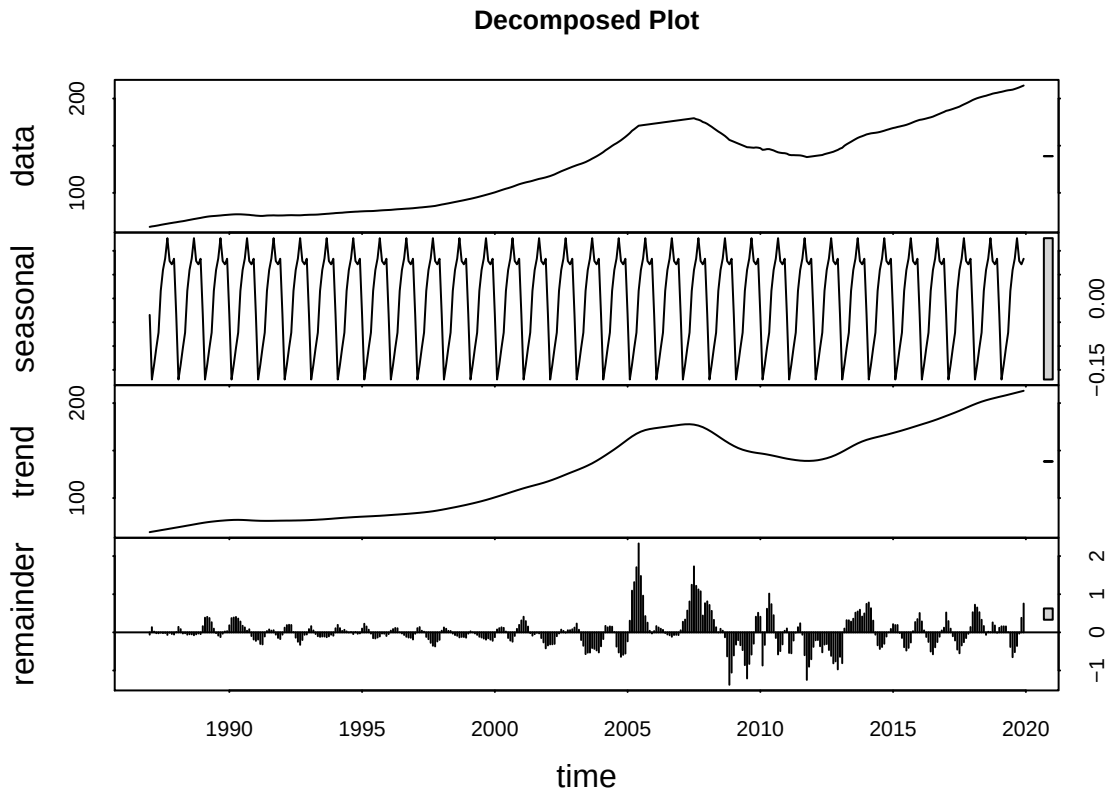
Let us deconstruct the above series to analyze its seasonaliy and trend.

```
cleanedTimeSeries = ts(CSUSHPISA_Data$clean_CSUSHPISA ,start = c(1987,1),end =
                c(2019,12), frequency=12)
```

```
# De-seasonalize the series
decomp = stl(cleanedTimeSeries, s.window="periodic")
deseasonal_CleanedTS <- forecast::seasadj(decomp)
plot(deseasonal_CleanedTS, main = "Deseasonalized Plot")
```

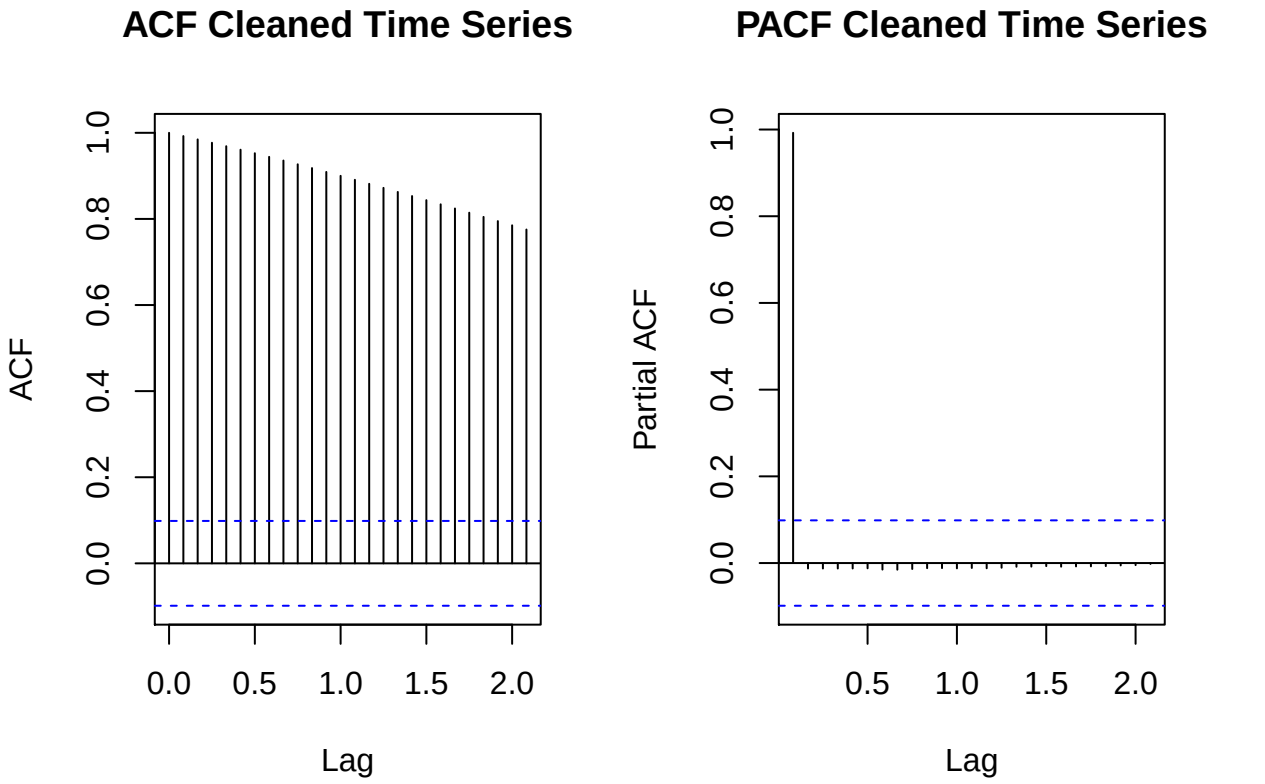



```
plot(decomp, main="Decomposed Plot")
```



Conducting ACF and PACF test on the cleaned price series

```r
par(mfrow = c(1,2))
acf(cleanedTimeSeries, main = "ACF Cleaned Time Series")
pacf(cleanedTimeSeries, main = "PACF Cleaned Time Series")
```



We perform a simple ADF test on the cleaned home price index.

```r
adf.test(cleanedTimeSeries, alternative = "stationary")
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  cleanedTimeSeries
## Dickey-Fuller = -3.4433, Lag order = 7, p-value = 0.04816
## alternative hypothesis: stationary
```

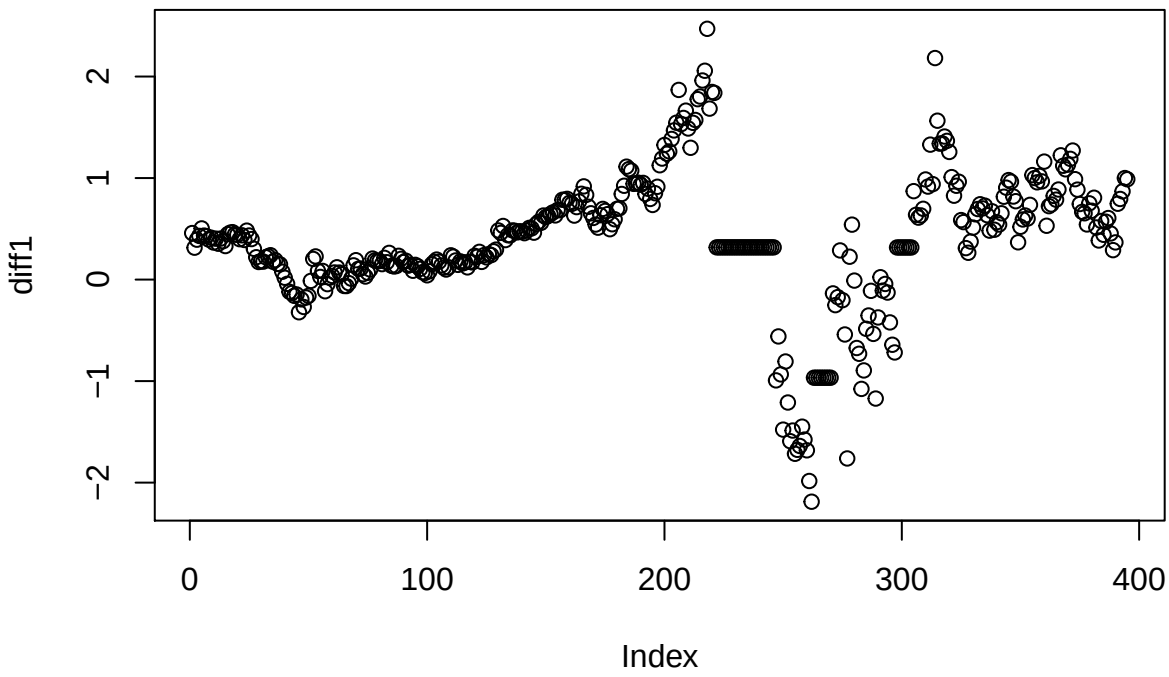
Taking the alpha level as 0.05, from the above results we can see that the p-value is less than 0.05, but it is very close. With alpha level of 0.01 we would not be able to reject null hypothesis of unit root. We will keep using the cleaned time series, but will using the method of differencing.Referring to the method of differencing, as suggested in R. S. Tsay(2015)[1], we will try to transform this nonstationary series into a stationary one by considering its change series. The change series follows a stationary and an invertible ARMA(p,q) model.

3.Implement an ARIMA(p,d,q) model. Determine p, d, q using Information Criterion or Box-Jenkins methodology. Comment the results

It has been determined that the home price index series is nonstationary, so we apply differencing and test again.

```r
totalRows = nrow(CSUSHPISA_Data) # total no. of rows
names(CSUSHPISA_Data) <- c('DATE','CSUSHPISA')
```

```r
diff1 <- CSUSHPISA_Data[2:totalRows,"CSUSHPISA"] - CSUSHPISA_Data[1:(totalRows-1),"CSUSHPISA"]
#plot first difference
plot(diff1)
```

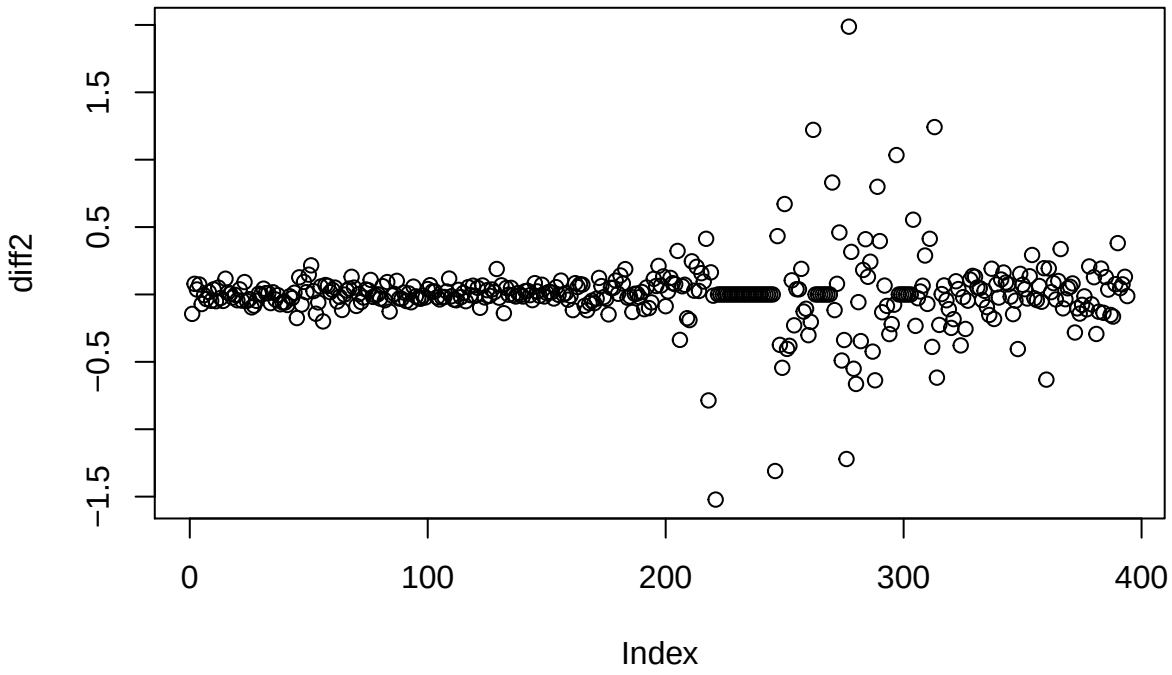



```r
#ADF test for stationarity of first difference n (suggesting nonstationarity)
adf.test(diff1,alternative = "stationary")
```

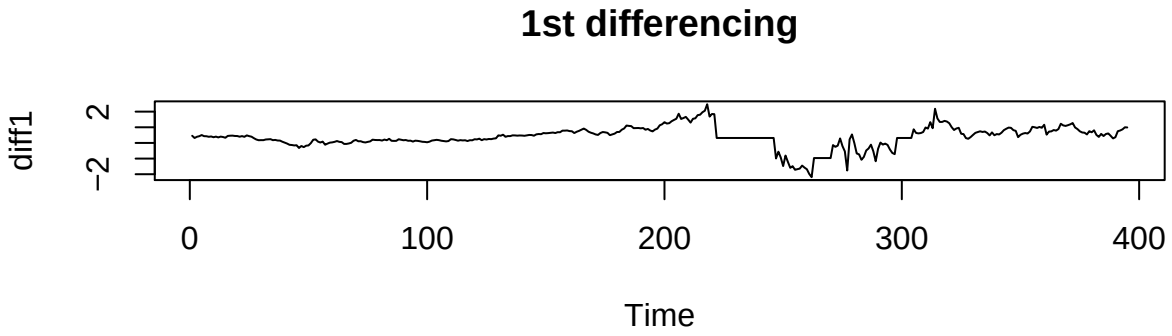
```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff1
## Dickey-Fuller = -2.5954, Lag order = 7, p-value = 0.3258
## alternative hypothesis: stationary
```

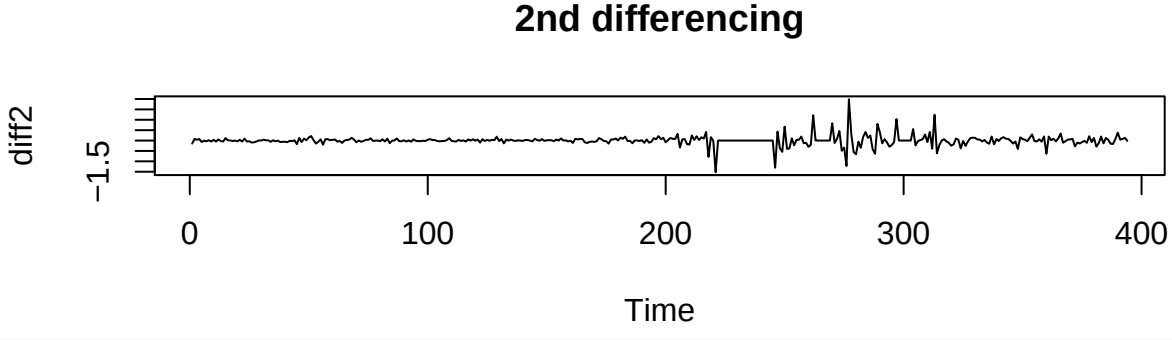
The p-value from the ADF test is greater than 0.05, hence we cannot reject the null hypothesis. Thus, the test on first difference indicated that the series is non-stationary.

```r
#repeat differencing procedure for second difference
lengthOfDiff1<-length(diff1)
#applying the differencing method as suggested by R.S.Tsay(2015)
diff2<-diff1[2:lengthOfDiff1]-diff1[1:lengthOfDiff1-1]
plot(diff2)
```




```
par(mfrow = c(2,1))
plot.ts(diff1, main="1st differencing")
plot.ts(diff2, main="2nd differencing")
```





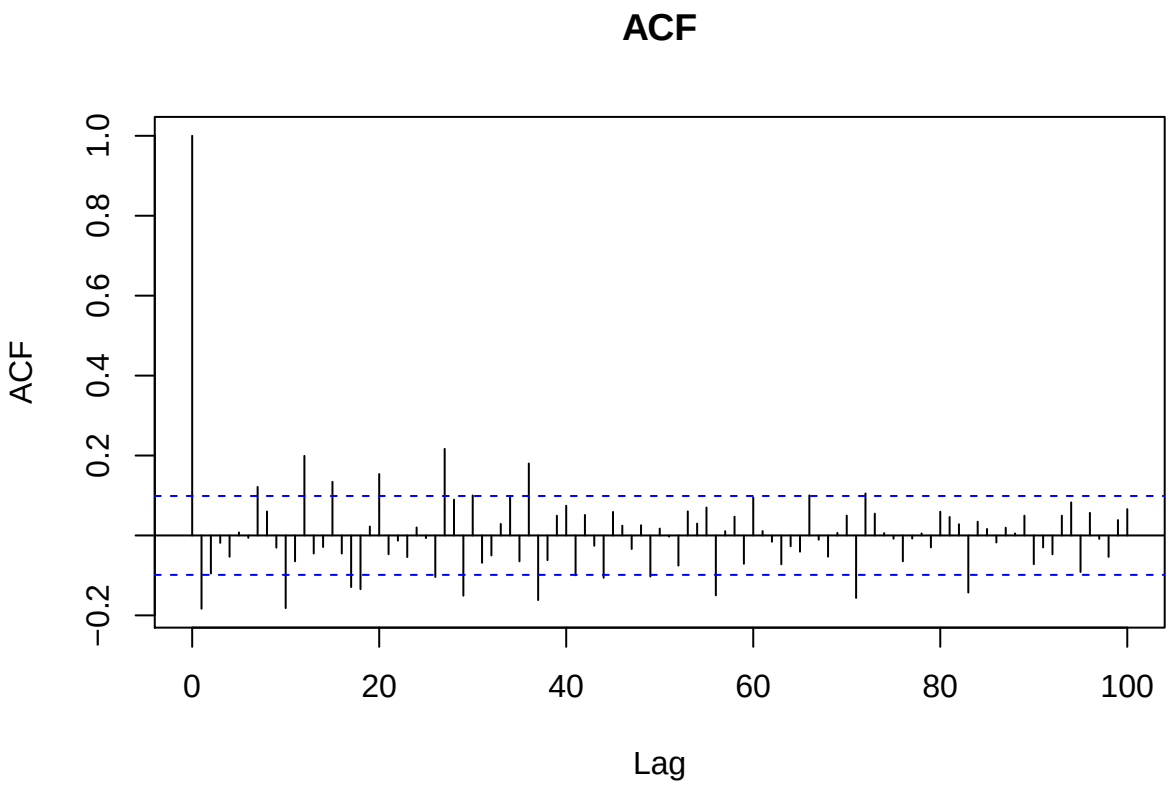

```
#perform an ADF test on the second difference
adf.test(diff2)
```

```
## Warning in adf.test(diff2): p-value smaller than printed p-value
```

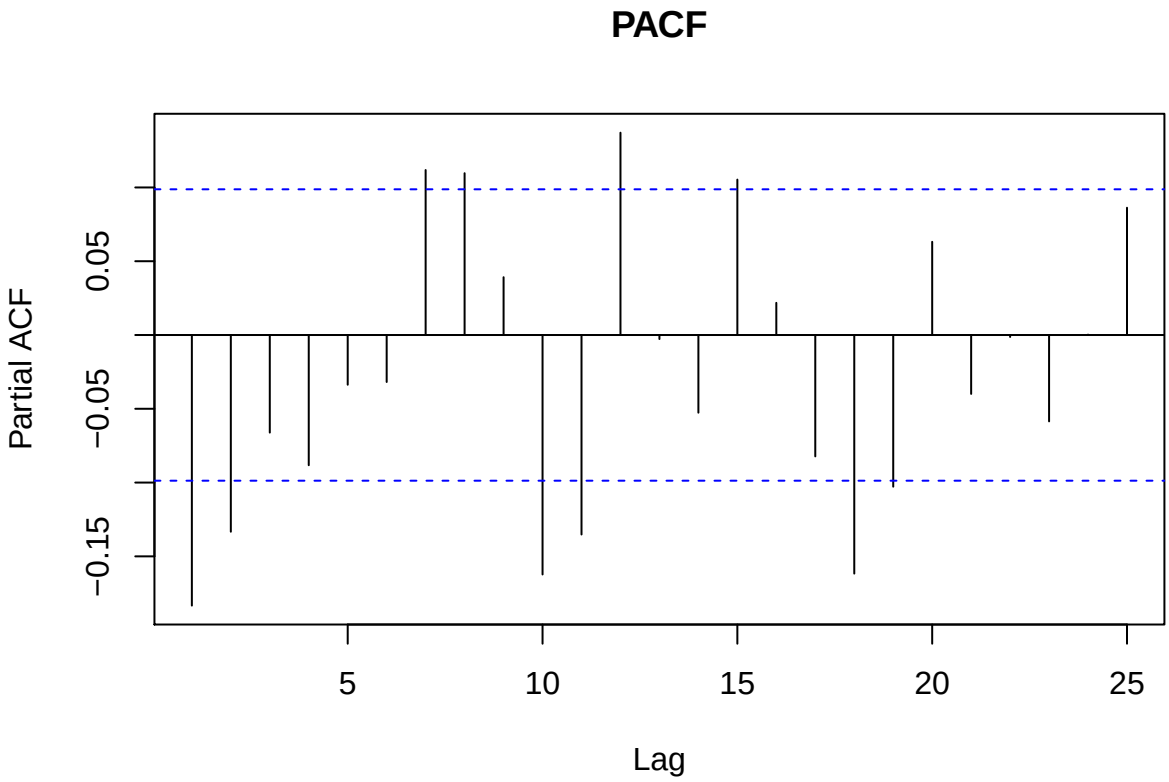
```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff2
## Dickey-Fuller = -6.3748, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

The p-value from the ADF test is less than our alpha level of 0.05. Thus, we reject the null hypothesis and conclude that with second-order differencing the process is stationary. We use the Auto Correlation Function (ACF) `acf` and the Partial Auto Correlation Function (PACF) `pacf` to determine `p` and `q`(`d` is 2 because we had to difference twice to make it stationary).

```
#par(mfrow = c(1,2))
acf(diff2, lag.max = 100, main = "ACF")
```




```
pacf(diff2, main = "PACF")
```



Since we have applied differencing twice, `d = 2`. The `auto.arima()` function can help us to find the optimal estimates for the remaining parameters of the ARIMA model, `p` and `q`.

```
auto.arima(CSUSHPISA_Data$CSUSHPISA)
```

```
## Series: CSUSHPISA_Data$CSUSHPISA
## ARIMA(0,2,2)
##
## Coefficients:
##           ma1      ma2
##       -0.2262  -0.1223
## s.e.   0.0498   0.0500
##
## sigma^2 estimated as 0.05963:  log likelihood=-2.58
## AIC=11.15   AICc=11.21   BIC=23.08
```

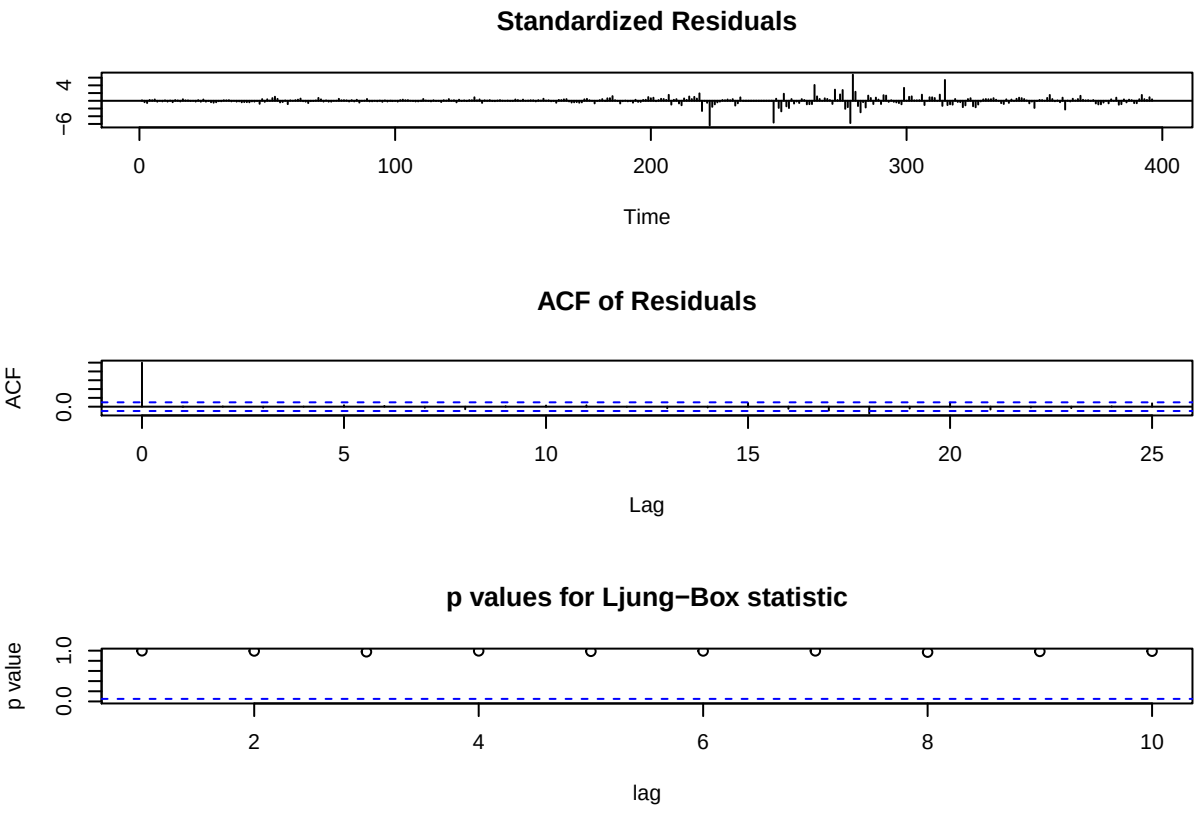
As the plot shown, there is strike mostly in first 12 lags. As the plot above shown, acf pattern showing expoential decays with damped sine wave pattern. Therefore, it can be inferred that ARIMA(12,2,0) model fits the data well. Now we try with:

```
p <- 12
d <- 2
q <- 0
ARIMA_Model <- arima(CSUSHPISA_Data$CSUSHPISA, order=c(p,d,q), method = "ML")
summary(ARIMA_Model)
```

```
##
## Call:
## arima(x = CSUSHPISA_Data$CSUSHPISA, order = c(p, d, q), method = "ML")
##
## Coefficients:
```

```
##           ar1      ar2      ar3      ar4      ar5      ar6     ar7     ar8
##       -0.2365  -0.1192  -0.0414  -0.0706  -0.0278  -0.0005  0.1200  0.0926
## s.e.   0.0499   0.0510   0.0505   0.0505   0.0507   0.0501  0.0501  0.0504
##           ar9     ar10     ar11    ar12
##       -0.0137  -0.1724  -0.0991  0.1344
## s.e.   0.0505   0.0504   0.0507  0.0495
##
## sigma^2 estimated as 0.05379:  log likelihood = 16.2,  aic = -6.39
##
## Training set error measures:
##                       ME      RMSE       MAE         MPE      MAPE      MASE
## Training set 0.001769052 0.2313884 0.1253396 0.002795406 0.0928395 0.2076474
##                      ACF1
## Training set 0.0001977013
```

```
tsdiag(ARIMA_Model)
```

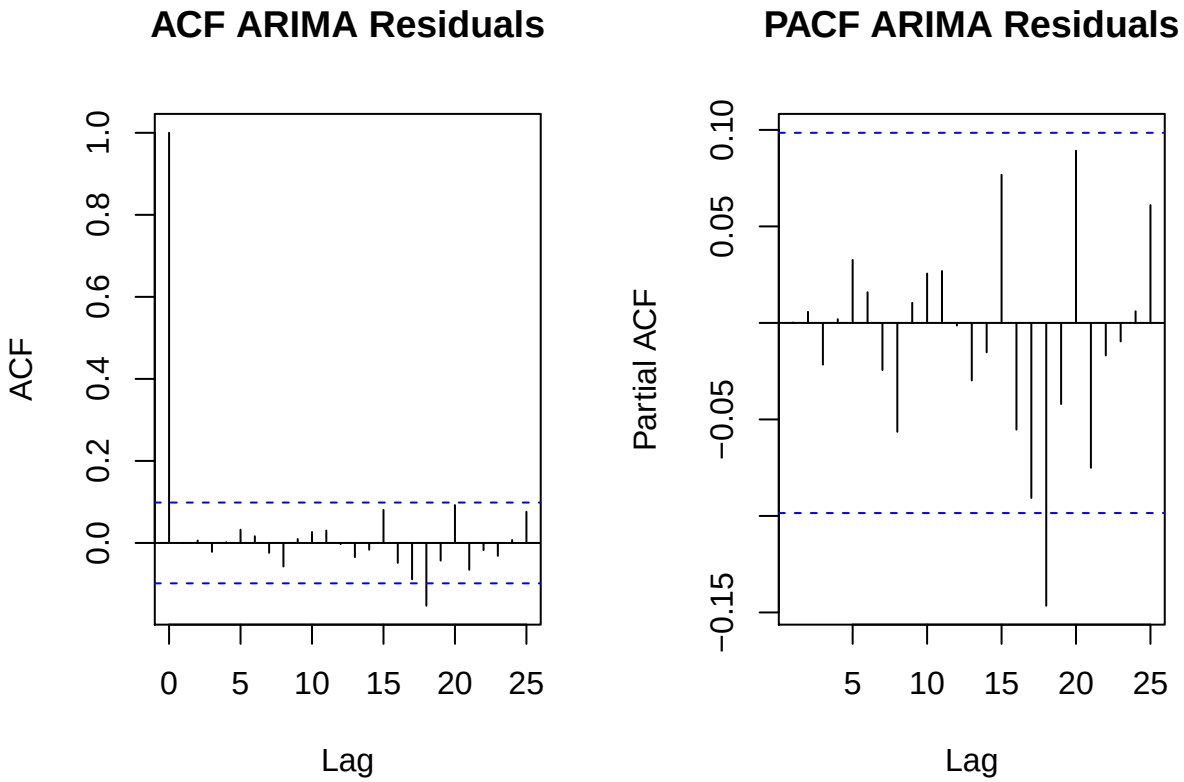



```
coef(ARIMA_Model)
```

```
##           ar1           ar2           ar3           ar4           ar5
## -0.2364594281 -0.1192106567 -0.0414380648 -0.0706101736 -0.0278279373
##           ar6           ar7           ar8           ar9          ar10
## -0.0004742658  0.1200175759  0.0926435482 -0.0136579410 -0.1724002815
##          ar11          ar12
## -0.0990611823  0.1343835093
```

From the summary, we can see σ^2 (variance) and aic are quite small, which is a good sign suggesting the model fits to data. To check if the residual of ARIMA(12,2,0) model is pure white noise.

```
par(mfrow = c(1,2))
acf(ARIMA_Model$residuals, main = "ACF ARIMA Residuals")
pacf(ARIMA_Model$residuals, main = "PACF ARIMA Residuals")
```



As shown in partial autocorrelation graph, we do not see significant strike except lag 18. So, we can perform Box-Pierce test on this lag.

```
Box.test(ARIMA_Model$residuals, lag = 18)
```

```
##
##  Box-Pierce test
##
## data:  ARIMA_Model$residuals
## X-squared = 19.4, df = 18, p-value = 0.3676
```

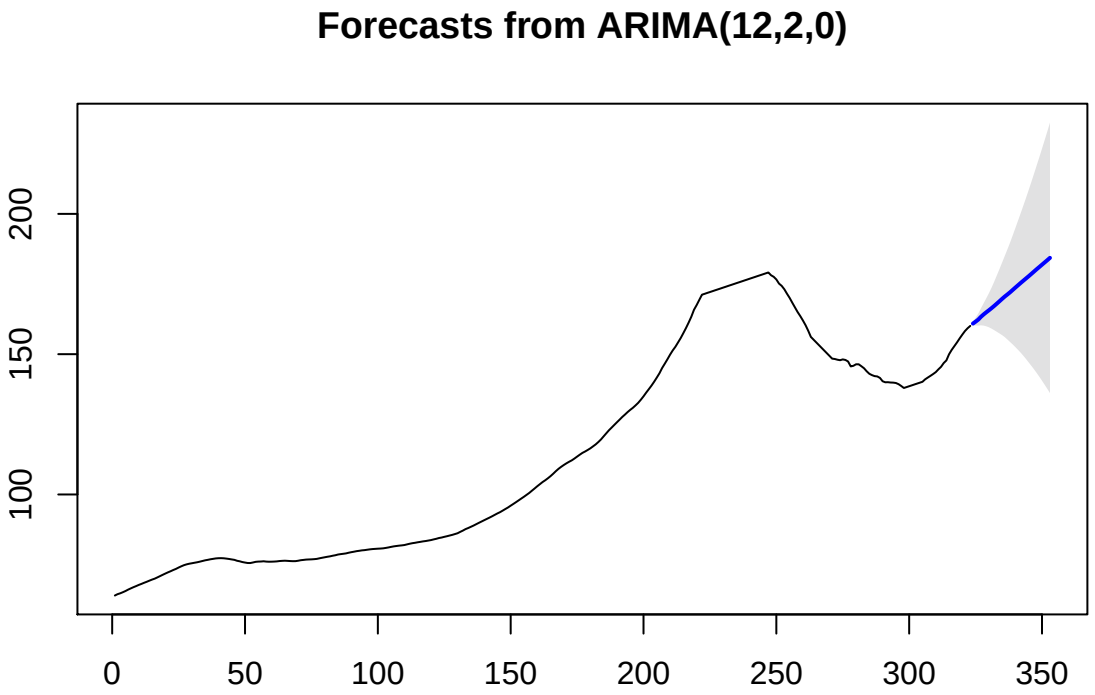
Its p-value is very high, so we fail to reject null hypothesis that errors are white noise. Therefore, the residuals are proved to be white noise.

4.Forecast the future evolution of Case-Shiller Index using the ARMA model. Test model using in-sample forecasts

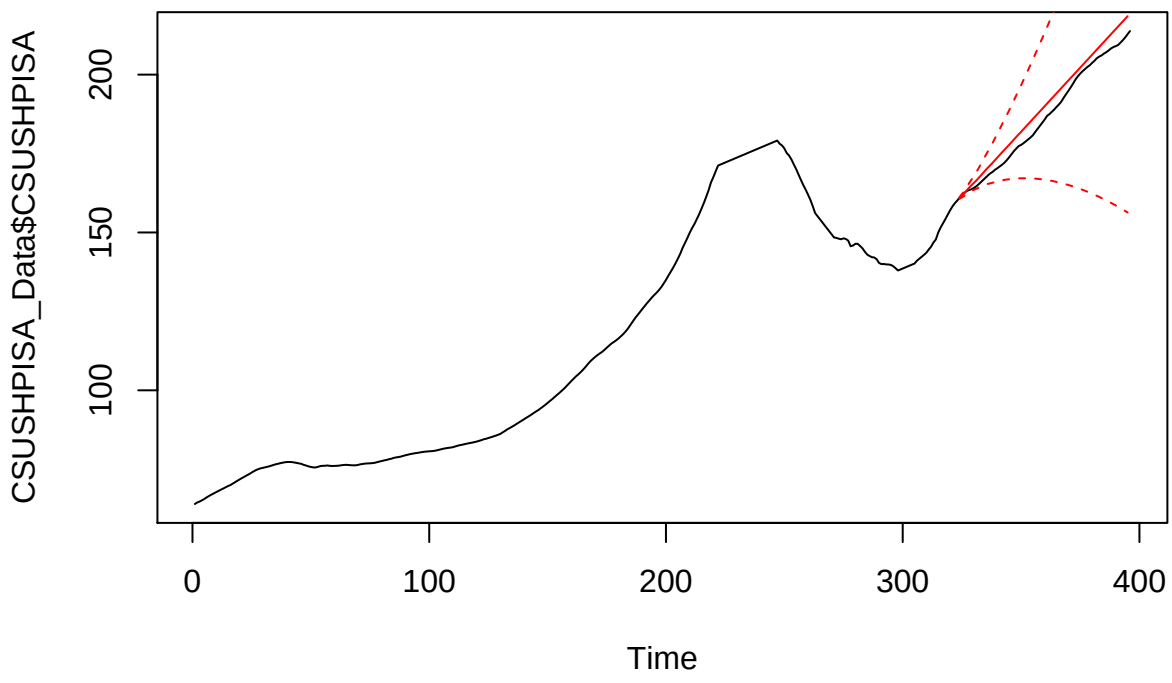
First we will forecast the evolution of Case-Shiller Index using. We can specify forecast horizon h periods ahead for predictions to be made, and use the fitted model to generate those predictions

```
dataLength = length(CSUSHPISA_Data$CSUSHPISA)
ARIMA_PredictModel <- arima(window(CSUSHPISA_Data$CSUSHPISA,1,dataLength-73),
                      order=c(p,d,q), method = "ML")
```

```
Forecast <- forecast(ARIMA_PredictModel, h=30, level=c(99.5))
plot(Forecast)
```



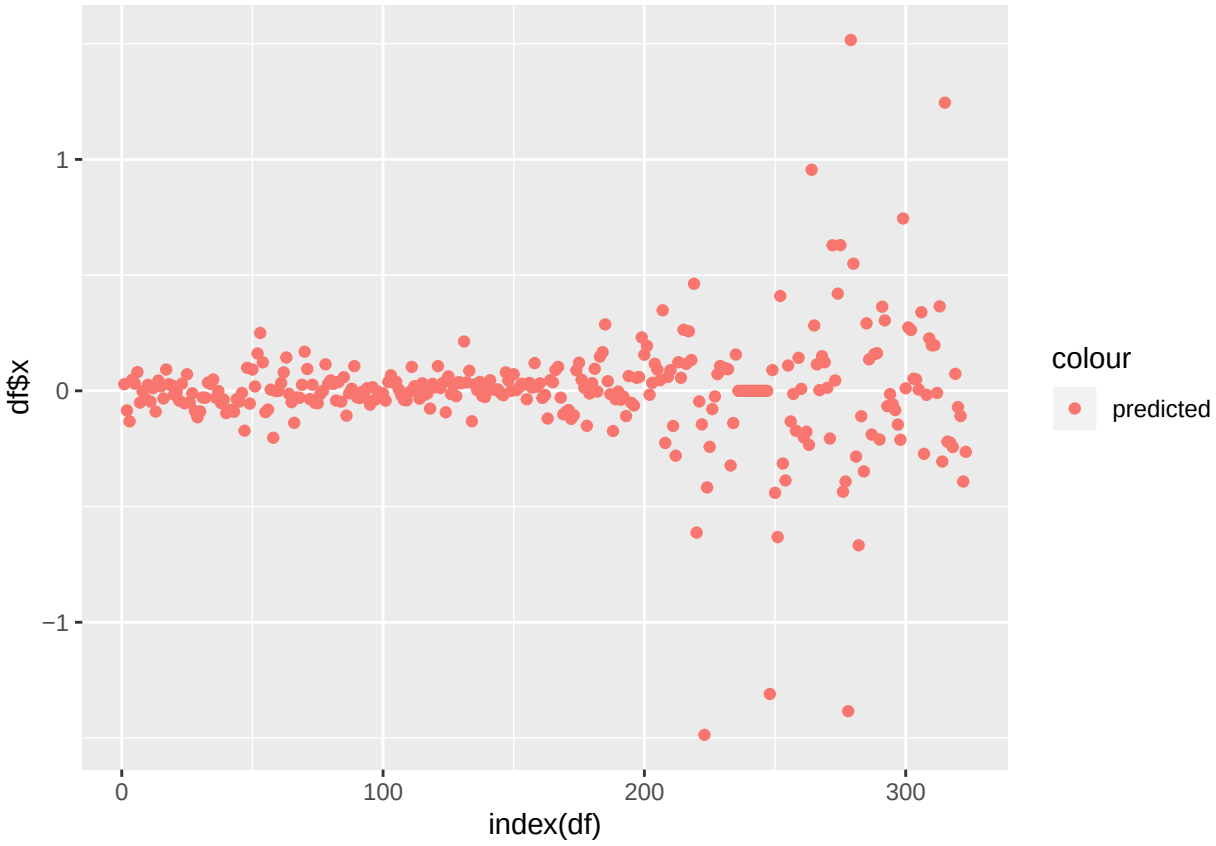

```
ARIMA_forecast <- predict(ARIMA_PredictModel, n.ahead=72, se.fit=TRUE)
plot(CSUSHPISA_Data$CSUSHPISA) + lines(ARIMA_forecast$pred, col="red") + lines(ARIMA_forecast$pred+1*AR
      lty="dashed") + lines(ARIMA_forecast$pred-1*ARIMA_forecast$se, col="red",
      lty="dashed")
```




```
## integer(0)
```

```
df = as.data.frame(ARIMA_PredictModel$residuals)
ggplot(df, aes(index(df))) +
    geom_point(aes(y = df$x, color="predicted"))
```

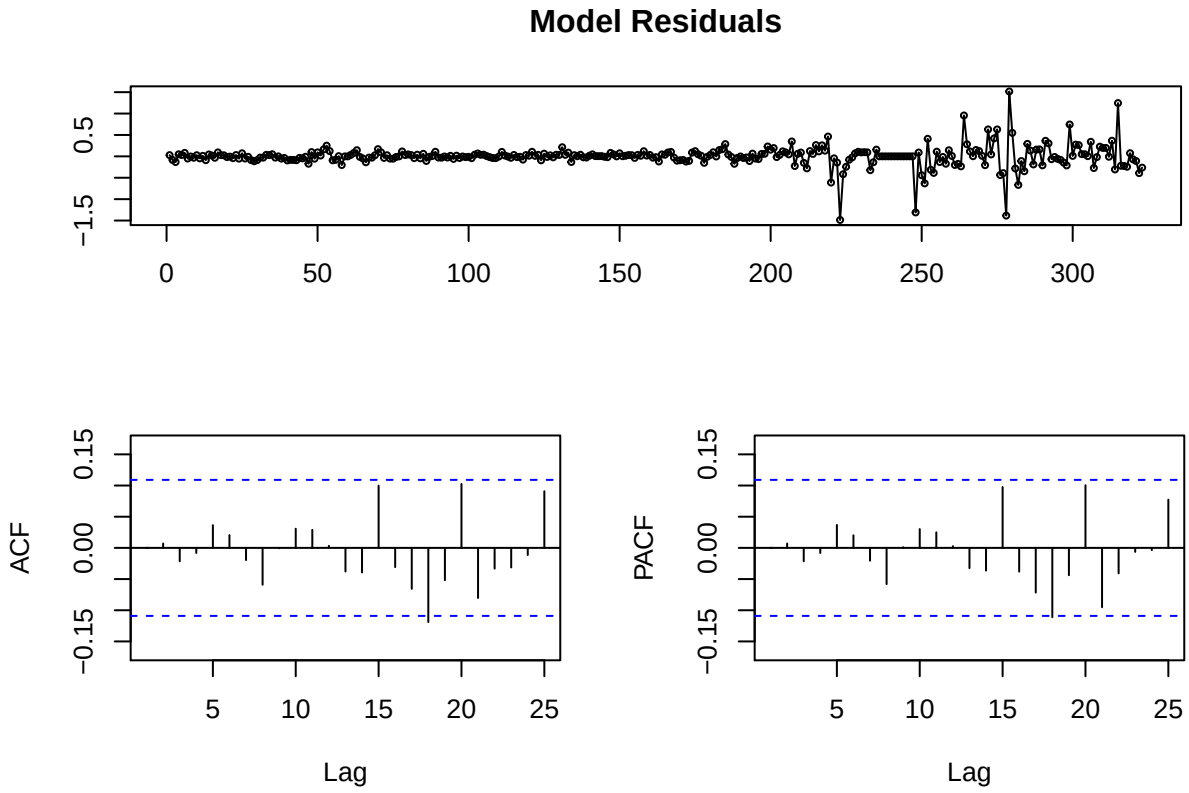
```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```



The model prediction scatters after period 200 and does not converge.

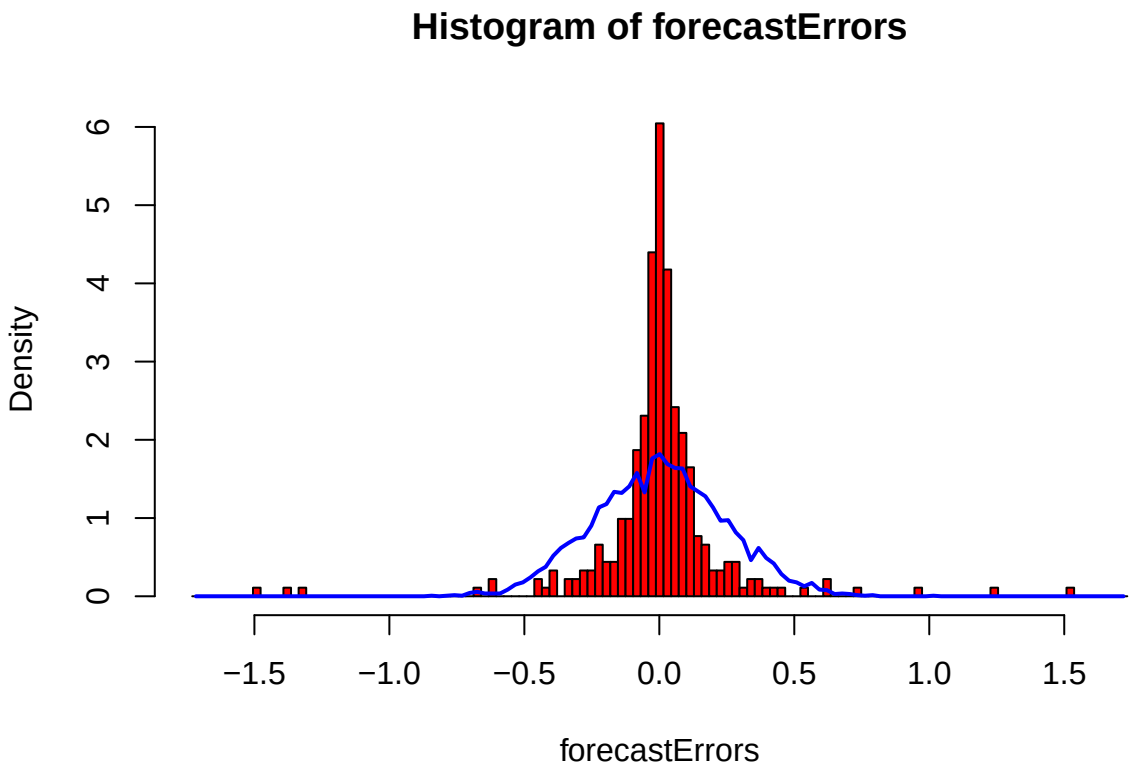
To evaluate the model, we can start by examining ACF and PACF plots for model residuals. If model order parameters and structure are correctly specified, we would expect no significant autocorrelations present.

```
tsdisplay(residuals(ARIMA_PredictModel), main='Model Residuals')
```



As we see in the graph, most of the test data are within one standard deviation from our prediction suggesting that ARIMA(12,2,0) is accurate.

```
displayForecastErrors <- function(forecastErrors)
  {
     # Generate a histogram of the Forecast Errors
     binsize <- IQR(forecastErrors)/4
     sd   <- sd(forecastErrors)
     min  <- min(forecastErrors) - sd
     max  <- max(forecastErrors) + sd

     # Generate 5K normal(0,sd) RVs
     norm <- rnorm(5000, mean=0, sd=sd)
     min2 <- min(norm)
     max2 <- max(norm)
     if (min2 < min) { min <- min2 }
     if (max2 > max) { max <- max2 }

     # Plot histogram of the forecast errors
     bins <- seq(min, max, binsize)
     hist(forecastErrors, col="red", freq=FALSE, breaks=bins)

     myHist <- hist(norm, plot=FALSE, breaks=bins)

     # Overlay the Blue normal curve on top of forecastErrors histogram
     points(myHist$mids, myHist$density, type="l", col="blue", lwd=2)
}

displayForecastErrors(residuals(ARIMA_PredictModel))
```



Types of exogenous variables that can improve forecasts.

Our aim throughout the exercise was to determine how does the aggregate level of housing prices change. We need to include some exogenous variables which help us in determining the changes. Though there are a number of variables which are important, we choose the the lending rates set by the federal reserves, and the LIBOR (London Inter-bank Offered Rate). The reason for this is these rates were important for determining the risks involved in the sub-prime mortgage market (during the 2007-10) crisis. The defaults started skyrocketing after the Federal Reserve increased the lending rates, and since the adjustable mortgage rates were higher than that of LIBOR (and LIBOR itself was influenced by the Federal rates), the adjustable rates on mortgages increased beyond what the home-owners could afford. Thus, leading to the crisis and hence the reason for our preference with respect to the context of our question.

References

1. Ruey S. Tsay. (2015). Analysis of Financial Time Series (3rd Edition). Wiley Series in Probability and Statistics.
2. Robert Nau. Statistical forecasting: notes on regression and time series analysis, ARIMA models for time series forecasting. Fuqua School of Business Duke University. Available at: http://people.duke.edu/~rnau/411home.htm
3. J. Berglund. (2007). Determinants and Forecasting of House Prices. Department of Economics, Uppsala University. Available at: https://pdfs.semanticscholar.org/8dbe/ef3c862ef9d4ba2ff6281d149aa563921e9f.pdf
4. Ivo Dinov. (2020). Data Science and Predictive Analytics (UMich HS650), Bilongitudnal Analysis. University of Michigan, Statistics Online Computational Resource (SOCR). Available at: http://www.socr.umich.edu/people/dinov/courses/DSPA_notes/18_BigLongitudinalDataAnalysis.html

5. R. J. Hyndman. (2018). Forecasting: Principles and Practice. Monash University. Available at: https://otexts.com/fpp2/