META monthly data analysis

Lynnstacy Kegeshi

2022-12-09

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

## Loading required package: timechange

##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

## Loading required package: zoo

##
## Attaching package: 'zoo'

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

##
## Attaching package: 'xts'

## The following objects are masked from 'package:dplyr':
##
##     first, last

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

```
##
## Attaching package: 'FinTS'

## The following object is masked from 'package:forecast':
##
##     Acf

## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0     v purrr   0.3.5
## v tibble  3.1.8     v stringr 1.4.1
## v tidyr   1.2.1     v forcats 0.5.2
## v readr   2.1.3
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x lubridate::as.difftime() masks base::as.difftime()
## x lubridate::date()        masks base::date()
## x dplyr::filter()          masks stats::filter()
## x xts::first()             masks dplyr::first()
## x lubridate::intersect()   masks base::intersect()
## x dplyr::lag()             masks stats::lag()
## x xts::last()              masks dplyr::last()
## x lubridate::setdiff()     masks base::setdiff()
## x lubridate::union()       masks base::union()
```

Preliminary Analysis

We shall be using the META monthly stock price data in our time series analysis.The read.csv function reads the data in as a data frame, and assign the data frame to a variable. Then we check the structure of the data to see the data types in our data frame

```
metam<-read.csv("META monthly historical prices data.csv", header = TRUE,  sep = ",")
str(metam)
```

```
## 'data.frame':    120 obs. of  7 variables:
##  $ Date     : chr  "2012-06-01" "2012-07-01" "2012-08-01" "2012-09-01" ...
##  $ Open     : num  28.9 31.2 21.5 18.1 22.1 ...
##  $ High     : num  33.5 32.9 22.5 23.4 24.2 ...
##  $ Low      : num  25.5 21.6 18 17.5 18.8 ...
##  $ Close    : num  31.1 21.7 18.1 21.7 21.1 ...
##  $ Adj.Close: num  31.1 21.7 18.1 21.7 21.1 ...
##  $ Volume   : int  667910500 520189700 1151944900 1058643700 1100938300 1527490200 1191832200 167585
```

The date variable is a character. We need to change it to a date format in before plotting our time series

```
metam$New_date=ymd(metam$Date)
str(metam$New_date)
```

```
##  Date[1:120], format: "2012-06-01" "2012-07-01" "2012-08-01" "2012-09-01" "2012-10-01" ...
```

This command creates a new column New_Date and the data type in this column is in date format.

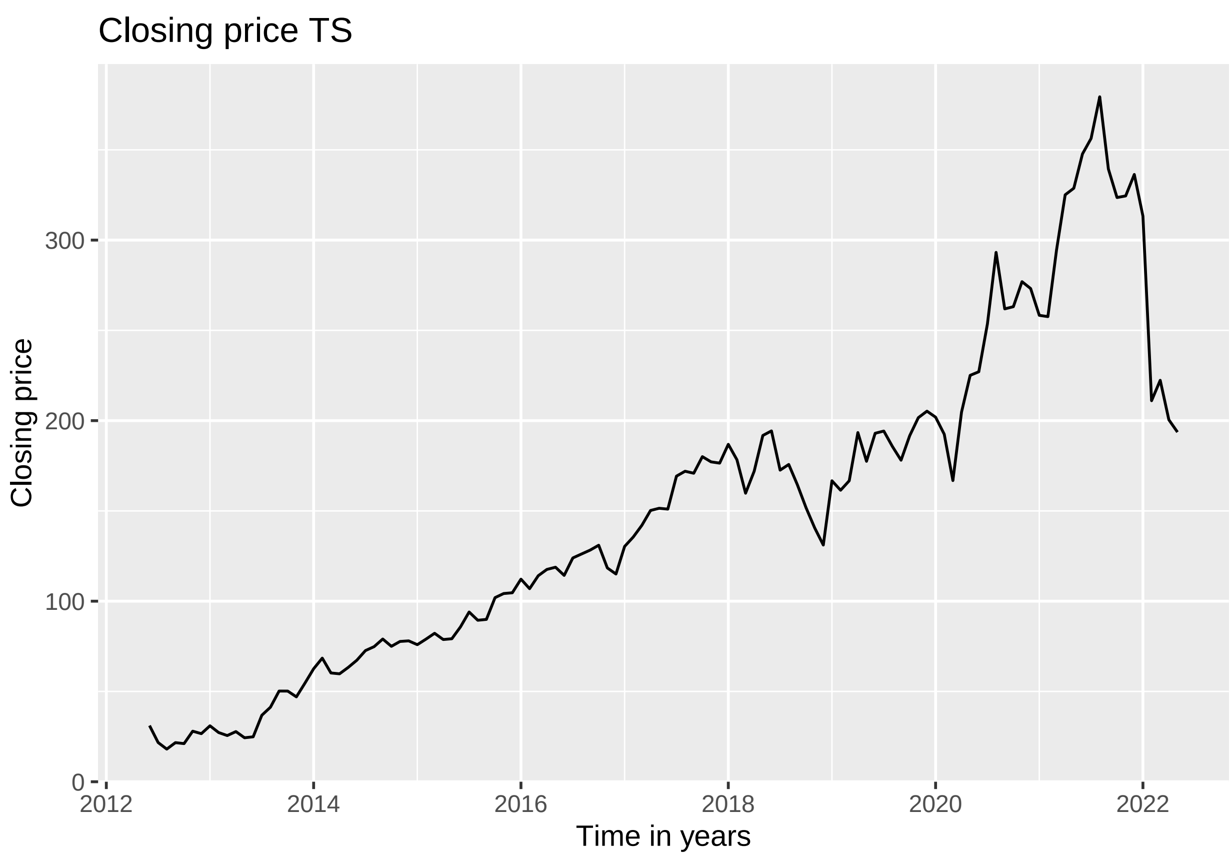
Plotting the time series

We chose to use the Closing price column setting the frequency to 12, monthly.

```
colnames(metam)
```

```
## [1] "Date"      "Open"      "High"      "Low"       "Close"     "Adj.Close"
## [7] "Volume"    "New_date"
```

```
closingts<-ts(metam$Close,start=c(2012,6),end=c(2022, 5), frequency = 12)
autoplot(closingts,main = "Closing price TS", xlab = "Time in years", ylab="Closing price" )
```



The graph above is a time series of the closing price from 2012 to 2022

Box plot of the closing price

In order to compare the distribution of the closing price between the months we used a boxplot We first needed to extract the months from the data

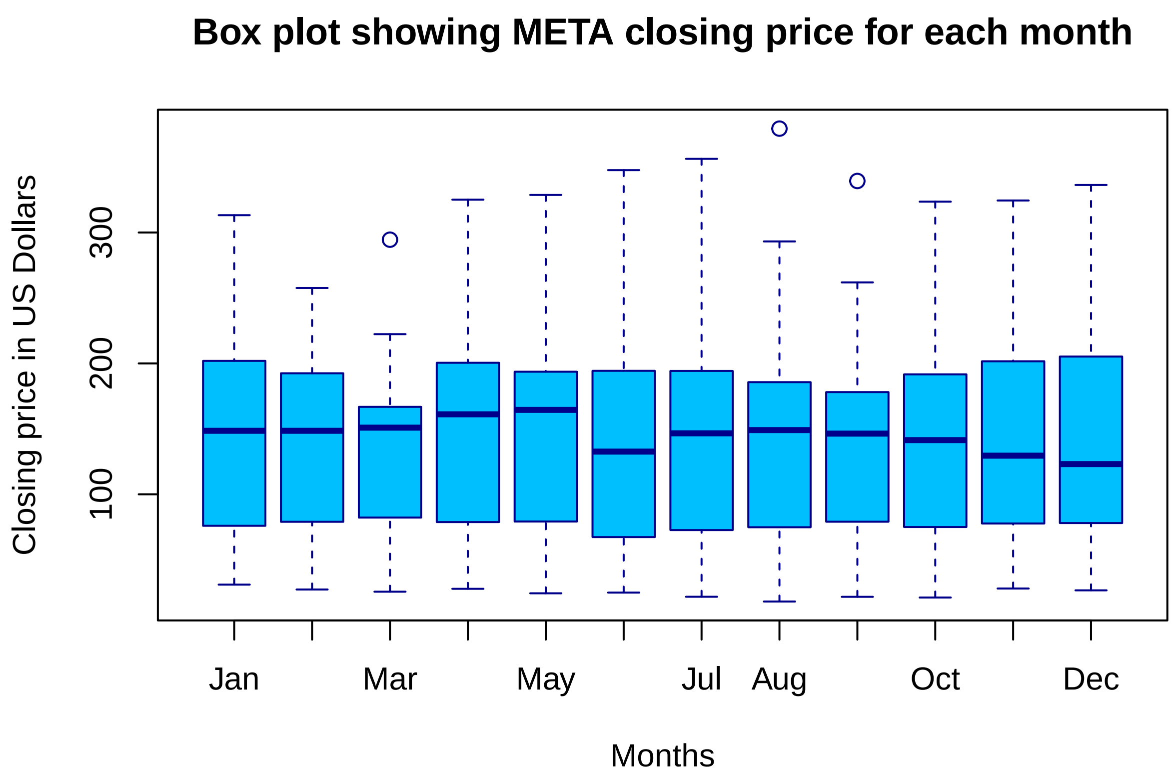
```
metam$monthhh=month(ymd(metam$New_date), label = TRUE)
metam$monthhh
```

```
##   [1] Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov
##  [19] Dec Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar Apr May
```

```
##  [37] Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov
##  [55] Dec Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar Apr May
##  [73] Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov
##  [91] Dec Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar Apr May
## [109] Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar Apr May
## 12 Levels: Jan < Feb < Mar < Apr < May < Jun < Jul < Aug < Sep < ... < Dec
```

After that we plot it

```
boxplot(Close~monthhh,
        data=metam,
        main="Box plot showing META closing price for each month",
        xlab="Months",
        ylab="Closing price in US Dollars",
        col="deepskyblue",
        border="darkblue")
```



##Decomposing the time series

```
closing_dec<-decompose(closingts, "multiplicative")
closing_dec
```

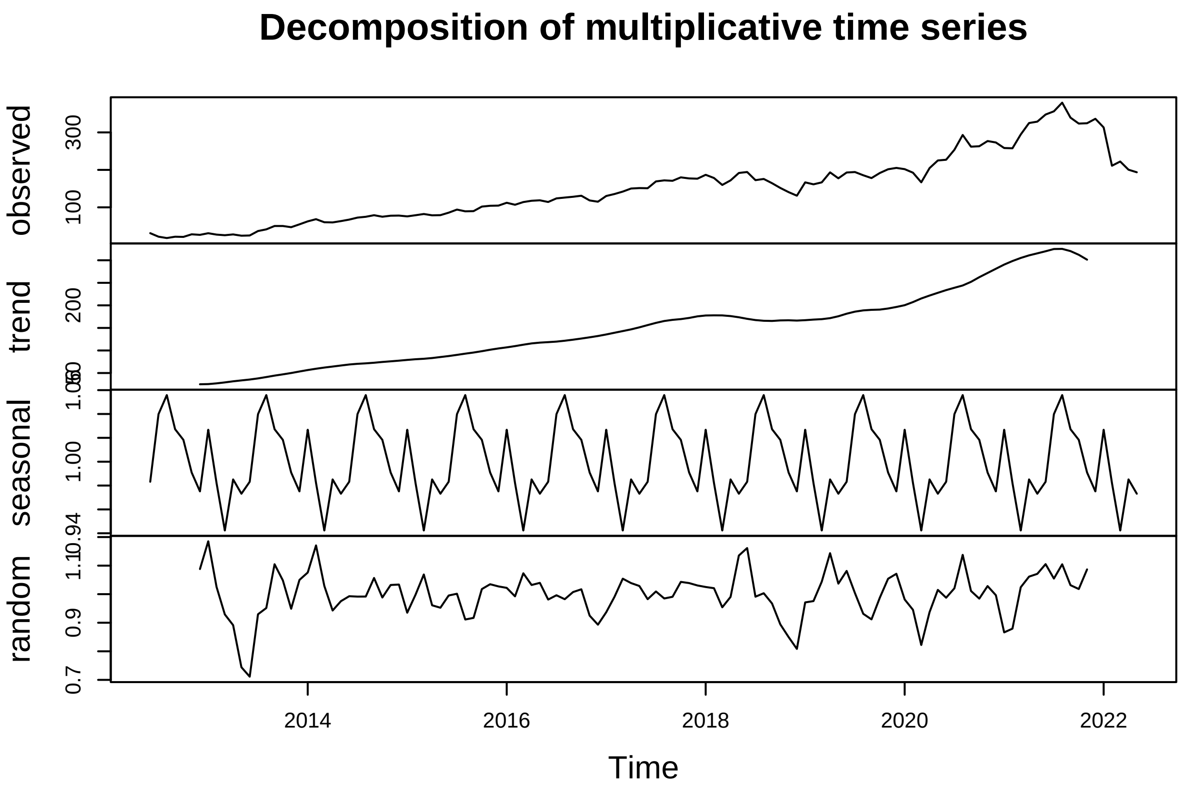
```
## $x
##          Jan     Feb     Mar     Apr     May     Jun     Jul     Aug     Sep     Oct
## 2012                                          31.10   21.71   18.06   21.66   21.11
```

```
## 2013  30.98  27.25  25.58  27.77  24.35  24.88  36.80  41.29  50.23  50.21
## 2014  62.57  68.46  60.24  59.78  63.30  67.29  72.65  74.82  79.04  74.99
## 2015  75.91  78.97  82.22  78.77  79.19  85.77  94.01  89.43  89.90 101.97
## 2016 112.21 106.92 114.10 117.58 118.81 114.28 123.94 126.12 128.27 130.99
## 2017 130.32 135.54 142.05 150.25 151.46 150.98 169.25 171.97 170.87 180.06
## 2018 186.89 178.32 159.79 172.00 191.78 194.32 172.58 175.73 164.46 151.79
## 2019 166.69 161.45 166.69 193.40 177.47 193.00 194.23 185.67 178.08 191.65
## 2020 201.91 192.47 166.80 204.71 225.09 227.07 253.67 293.20 261.90 263.11
## 2021 258.33 257.62 294.53 325.08 328.73 347.71 356.30 379.38 339.39 323.57
## 2022 313.26 211.03 222.36 200.47 193.64
##         Nov    Dec
## 2012  28.00  26.62
## 2013  47.01  54.65
## 2014  77.70  78.02
## 2015 104.24 104.66
## 2016 118.42 115.05
## 2017 177.18 176.46
## 2018 140.61 131.09
## 2019 201.64 205.25
## 2020 276.97 273.16
## 2021 324.46 336.35
## 2022
##
## $seasonal
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2012                                                   0.9831909 1.0398485
## 2013 1.0267779 0.9821273 0.9423434 0.9851433 0.9731599 0.9831909 1.0398485
## 2014 1.0267779 0.9821273 0.9423434 0.9851433 0.9731599 0.9831909 1.0398485
## 2015 1.0267779 0.9821273 0.9423434 0.9851433 0.9731599 0.9831909 1.0398485
## 2016 1.0267779 0.9821273 0.9423434 0.9851433 0.9731599 0.9831909 1.0398485
## 2017 1.0267779 0.9821273 0.9423434 0.9851433 0.9731599 0.9831909 1.0398485
## 2018 1.0267779 0.9821273 0.9423434 0.9851433 0.9731599 0.9831909 1.0398485
## 2019 1.0267779 0.9821273 0.9423434 0.9851433 0.9731599 0.9831909 1.0398485
## 2020 1.0267779 0.9821273 0.9423434 0.9851433 0.9731599 0.9831909 1.0398485
## 2021 1.0267779 0.9821273 0.9423434 0.9851433 0.9731599 0.9831909 1.0398485
## 2022 1.0267779 0.9821273 0.9423434 0.9851433 0.9731599
##            Aug       Sep       Oct       Nov       Dec
## 2012 1.0558361 1.0272349 1.0183039 0.9909262 0.9751077
## 2013 1.0558361 1.0272349 1.0183039 0.9909262 0.9751077
## 2014 1.0558361 1.0272349 1.0183039 0.9909262 0.9751077
## 2015 1.0558361 1.0272349 1.0183039 0.9909262 0.9751077
## 2016 1.0558361 1.0272349 1.0183039 0.9909262 0.9751077
## 2017 1.0558361 1.0272349 1.0183039 0.9909262 0.9751077
## 2018 1.0558361 1.0272349 1.0183039 0.9909262 0.9751077
## 2019 1.0558361 1.0272349 1.0183039 0.9909262 0.9751077
## 2020 1.0558361 1.0272349 1.0183039 0.9909262 0.9751077
## 2021 1.0558361 1.0272349 1.0183039 0.9909262 0.9751077
## 2022
##
## $trend
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2012                                                          NA        NA
## 2013  25.45958  27.05625  29.21458  31.61750  33.62208  35.58208  38.06625
## 2014  56.64625  59.53708  62.13458  64.36750  66.67875  68.93125  70.46083
```

```
## 2015  79.06083  80.55958  81.62083  83.19750  85.42750  87.64333  90.26583
## 2016 106.92292 109.69875 112.82625 115.63417 117.43417 118.45792 119.64542
## 2017 135.50375 139.30208 142.98750 146.80708 151.30000 156.30708 161.22292
## 2018 177.54625 177.84167 177.73125 176.28625 173.58458 170.17042 167.43833
## 2019 167.14875 168.46500 169.44667 171.67500 175.87875 181.51167 186.06917
## 2020 200.35750 207.31458 215.28750 221.75750 227.87375 233.84208 239.02250
## 2021 290.44375 298.31083 305.13041 310.87833 315.37625 319.98792 324.90958
## 2022        NA        NA        NA        NA        NA
##            Aug       Sep       Oct       Nov       Dec
## 2012        NA        NA        NA        NA  25.09000
## 2013  41.09958  44.26083  47.03875  49.99542  53.38542
## 2014  71.45458  72.80833  74.51542  75.96875  77.40083
## 2015  92.94292  95.43583  98.38125 101.64917 104.48792
## 2016 121.59250 123.94958 126.47542 129.19708 132.08667
## 2017 165.36250 167.88417 169.52958 172.11583 175.60167
## 2018 165.89375 165.47833 166.65750 166.95292 166.30167
## 2019 188.82917 190.12625 190.60208 193.05750 196.46125
## 2020 244.08792 252.12458 262.46208 271.79583 281.14083
## 2021 325.25708 320.30875 312.10958 301.28875        NA
## 2022
##
## $random
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2012                                                          NA        NA
## 2013 1.1850962 1.0254893 0.9291624 0.8915567 0.7442008 0.7111826 0.9296889
## 2014 1.0757676 1.1707968 1.0288271 0.9427355 0.9755107 0.9928795 0.9915572
## 2015 0.9351066 0.9981071 1.0689743 0.9610614 0.9525515 0.9953565 1.0015683
## 2016 1.0220785 0.9924062 1.0731645 1.0321620 1.0396192 0.9812243 0.9961973
## 2017 0.9366628 0.9906997 1.0542266 1.0388864 1.0286671 0.9824329 1.0095593
## 2018 1.0251751 1.0209366 0.9540620 0.9903999 1.1352930 1.1614369 0.9912096
## 2019 0.9712475 0.9757994 1.0439203 1.1435364 1.0368773 1.0814711 1.0038569
## 2020 0.9814670 0.9452907 0.8221821 0.9370470 1.0150272 0.9876413 1.0206110
## 2021 0.8662361 0.8793115 1.0243181 1.0614520 1.0710905 1.1052124 1.0545889
## 2022        NA        NA        NA        NA        NA
##            Aug       Sep       Oct       Nov       Dec
## 2012        NA        NA        NA        NA 1.0880649
## 2013 0.9515048 1.1047750 1.0482311 0.9488963 1.0498202
## 2014 0.9917247 1.0568080 0.9882795 1.0321545 1.0337313
## 2015 0.9113190 0.9170193 1.0178474 1.0348783 1.0272167
## 2016 0.9823826 1.0074193 1.0170789 0.9249772 0.8932541
## 2017 0.9849613 0.9908007 1.0430241 1.0388493 1.0305405
## 2018 1.0032736 0.9674965 0.8944187 0.8499256 0.8083889
## 2019 0.9312712 0.9118078 0.9874242 1.0540197 1.0714050
## 2020 1.1376828 1.0112314 0.9844493 1.0283682 0.9964157
## 2021 1.1047174 1.0314793 1.0180843 1.0867682        NA
## 2022
##
## $figure
##  [1] 0.9831909 1.0398485 1.0558361 1.0272349 1.0183039 0.9909262 0.9751077
##  [8] 1.0267779 0.9821273 0.9423434 0.9851433 0.9731599
##
## $type
## [1] "multiplicative"
##
```

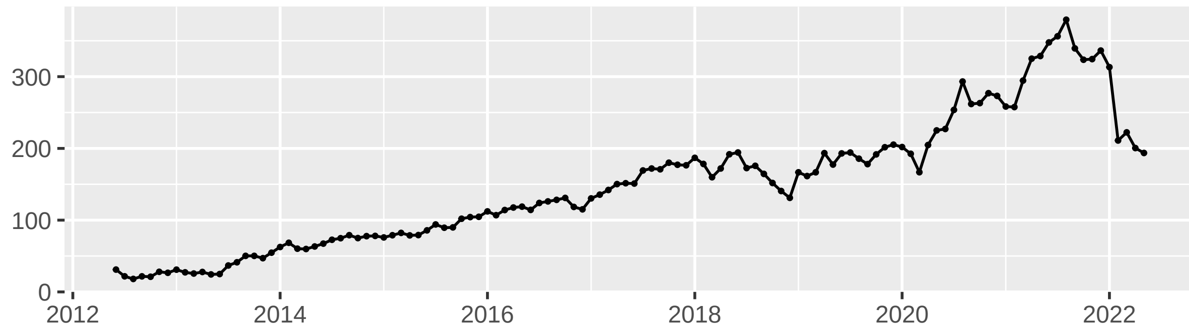
```
## attr(,"class")
## [1] "decomposed.ts"
```

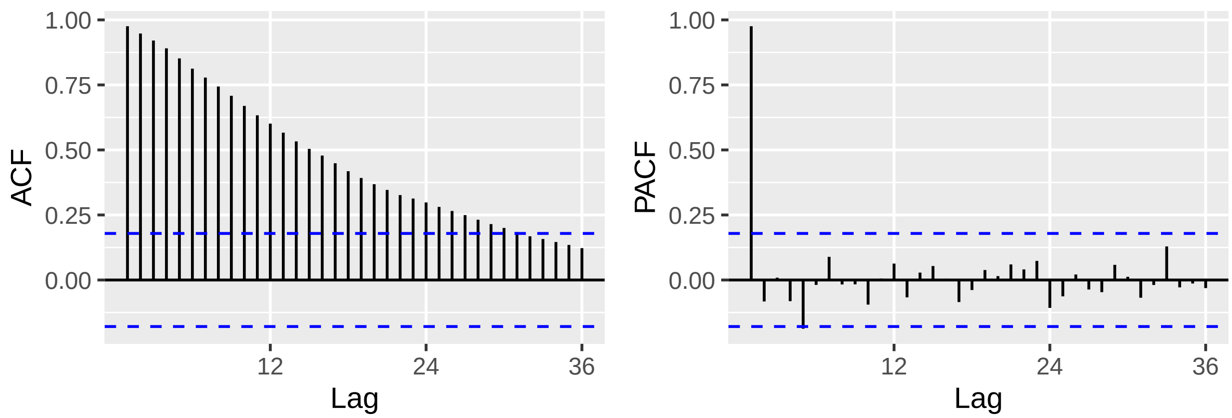
```
plot(closing_dec)
```



The time series has an upward trend. The variability in the data is also increasing over time. ### Plot the original ACF AND PACF

```
#Plot the original data ACF and PACF
ggtsdisplay(closingts)
```





The ACF plot shows slow decay of lag to 0 indicating an AR model. The PACF plot suggests AR model of the order 1 AR(1) as PACF number is close to 0 after lag 1.

Test for stationarity

```
adf.test(closingts)
```

ADF test

```
##
##  Augmented Dickey-Fuller Test
##
## data:  closingts
## Dickey-Fuller = -3.268, Lag order = 4, p-value = 0.07992
## alternative hypothesis: stationary
```

pvalue>0.05: 0.07992>0.05, Fail to reject the null hypothesis and conclude that the time series is non stationary

```
kpss.test(closingts)
```

KPSS test

```
## Warning in kpss.test(closingts): p-value smaller than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  closingts
## KPSS Level = 2.2203, Truncation lag parameter = 4, p-value = 0.01
```

p-value is 0.01. $0.01<0.05$, Reject null hypothesis and conclude that the time series is non stationary.

Make the data stationary

We need to remove the upward trend of the data and the variability in order to stationarize this time series.

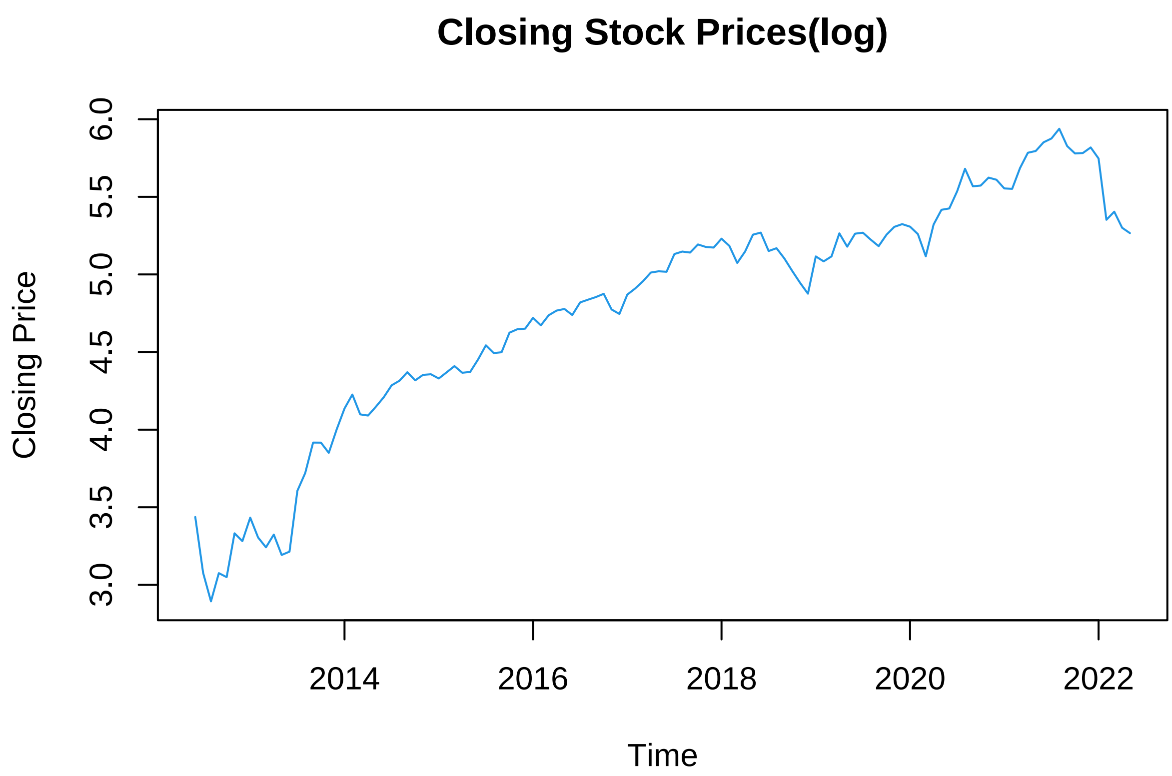
Removing Variability Using Logarithmic Transformation

Since the data shows changing variance,log transformation of the data using the log() function will stabilize the data. The resulting series will be a linear time series. check how many times we need to differentiate the data in order to make it stationary

```
closingts_linear<-log(closingts)
```

Plotting the resulting series

```
plot.ts(closingts_linear, main="Closing Stock Prices(log)", ylab="Closing Price", col=4)
```



Log transformation of the original time series has enabled more "balanced" variance observed throughout the time series, and in a way coerced the data into stationarity.

Removing Linear Trend

Check how many times we need to differentiate the log transformed time series

```
ndiffs(closingts_linear)
```

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

We will now perform the first difference of the log transformed series to make it stationary

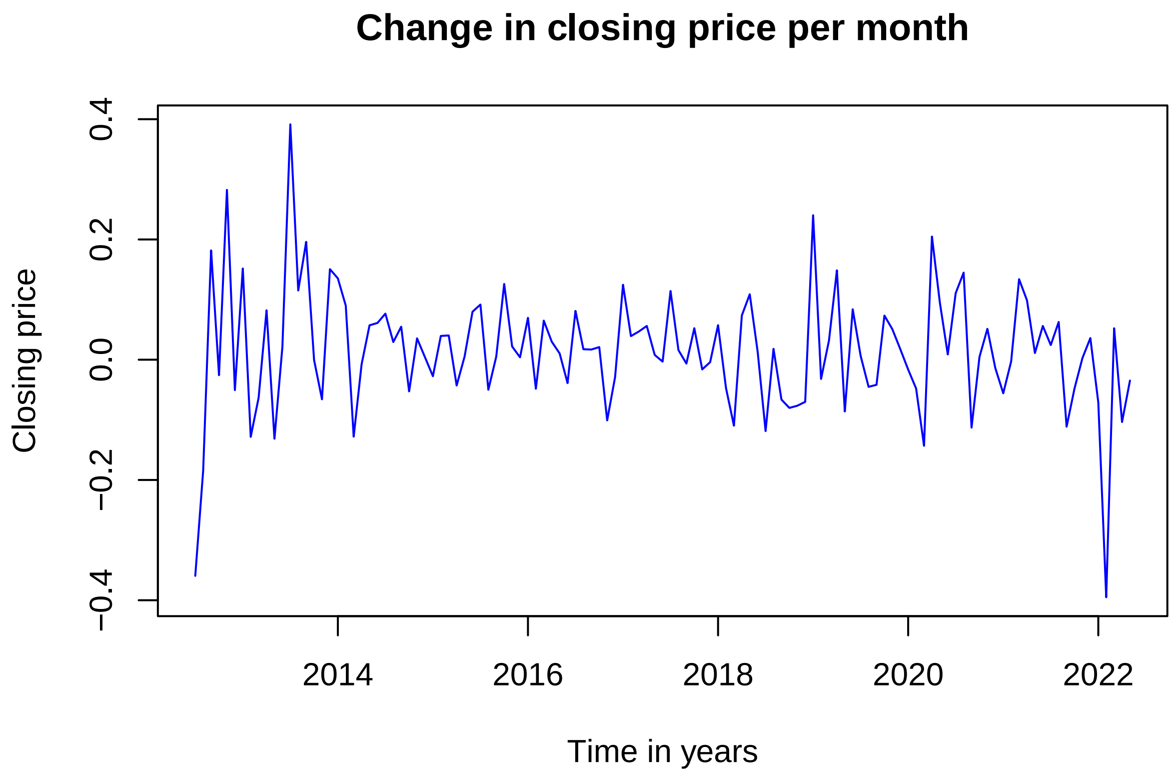
Differentiate it and plot the time series, ACF AND PACF plot

```
diffclose<-diff(closingts_linear)
diffclose
```

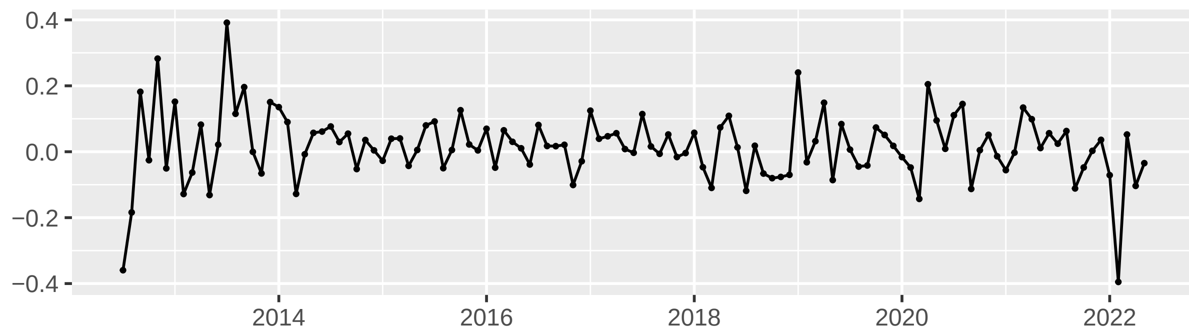
```
##                Jan           Feb           Mar           Apr           May
## 2012
## 2013  0.1516789845 -0.1282883139 -0.0632427250  0.0821455052 -0.1314244518
## 2014  0.1353366788  0.0899636885 -0.1279130018 -0.0076654766  0.0572141732
## 2015 -0.0274166849  0.0395195654  0.0403305475 -0.0428664180  0.0053178778
## 2016  0.0696550676 -0.0482912342  0.0649943668  0.0300437306  0.0104065917
## 2017  0.1246261755  0.0392737301  0.0469123814  0.0561214440  0.0080210365
```

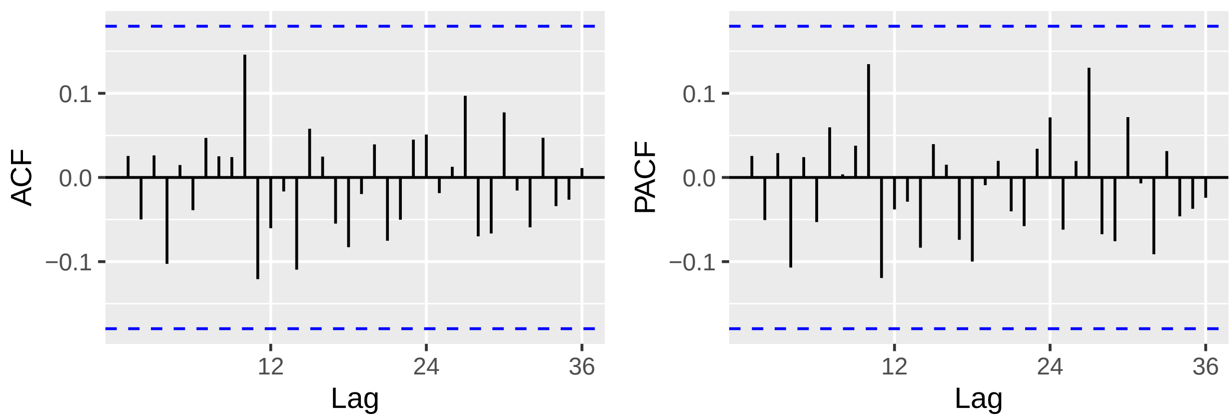
```
## 2018  0.0574259417 -0.0469404748 -0.1097193190  0.0736340675  0.1088543997
## 2019  0.2402517323 -0.0319403333  0.0319403333  0.1486247400 -0.0859589658
## 2020 -0.0164066750 -0.0478817708 -0.1431447947  0.2047988697  0.0949059268
## 2021 -0.0558199317 -0.0027521755  0.1338952505  0.0986904038  0.0111655127
## 2022 -0.0711187537 -0.3950332498  0.0522973959 -0.1036330800 -0.0346638547
##                Jun           Jul           Aug           Sep           Oct
## 2012               -0.3594348814 -0.1840734452  0.1817677490 -0.0257203323
## 2013  0.0215323781  0.3914335903  0.1151225461  0.1959920936 -0.0003982676
## 2014  0.0611263387  0.0766417655  0.0294318114  0.0548688372 -0.0525993214
## 2015  0.0798192065  0.0917319208 -0.0499449845  0.0052417674  0.1259806985
## 2016 -0.0388739930  0.0811460251  0.0174362648  0.0169035900  0.0209835741
## 2017 -0.0031742587  0.1142295606  0.0159431370 -0.0064170446  0.0523871134
## 2018  0.0131574442 -0.1186454120  0.0180877948 -0.0662812813 -0.0801694816
## 2019  0.0838886027  0.0063528143 -0.0450721096 -0.0417379949  0.0734378618
## 2020  0.0087580664  0.1107758310  0.1448208091 -0.1128922823  0.0046094101
## 2021  0.0561319945  0.0244042658  0.0627653155 -0.1113884217 -0.0477344373
## 2022
##                Nov           Dec
## 2012  0.2824576009 -0.0505416596
## 2013 -0.0658538876  0.1505889487
## 2014  0.0355004741  0.0041099469
## 2015  0.0220172514  0.0040211255
## 2016 -0.1008834124 -0.0288707677
## 2017 -0.0161239921 -0.0040718640
## 2018 -0.0765078328 -0.0701060280
## 2019  0.0508131637  0.0177448241
## 2020  0.0513370620 -0.0138514819
## 2021  0.0027467388  0.0359900681
## 2022
```

```
plot.ts(diffclose, main="Change in closing price per month",xlab="Time in years", ylab="Closing price",
```




```
ggtsdisplay(diffclose)
```





based on our limited analysis, let’s say we will go with p=0, d=0, and q=0.

Our suggested model is then ARIMA(0,0,0).

Now check number of differences in the differenced time series for confirmation that the time series has been detrended.

```
ndiffs(diffclose)
```

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

0 indicates we can’t differentiate the time series any further.

Seasonality and trend

```
isSeasonal(diffclose, test="wo")
```

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

The time series shows no evidence of seasonality.

```
adf.test(diffclose)
```

ADF test

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diffclose
## Dickey-Fuller = -4.8555, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

p-value is 0.01. 0.01<0.05, we thus reject the null hypothesis that the time series is non stationary and conclude that the time series is stationary.

```
kpss.test(diffclose)
```

KPSS test

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

```
##
##  KPSS Test for Level Stationarity
##
## data:  diffclose
## KPSS Level = 0.23105, Truncation lag parameter = 4, p-value = 0.1
```

0.1>0.05 fail to reject null hypothesis and conclude that the time series is stationary

ARIMA

ARIMA models cannot handle any type of non-stationarity and the time series is now stationary so we can proceed. To find the best ARIMA model to fit our time series, we;

```
ARIMAfitmeta <- auto.arima(diffclose, trace=TRUE,
                       ic= c("bic"), #c("aicc","aic","bic"),
)
```

```
##
##  ARIMA(2,0,2)(1,0,1)[12] with non-zero mean : -168.1334
##  ARIMA(0,0,0)            with non-zero mean : -194.1324
##  ARIMA(1,0,0)(1,0,0)[12] with non-zero mean : -185.2795
##  ARIMA(0,0,1)(0,0,1)[12] with non-zero mean : -185.1887
##  ARIMA(0,0,0)            with zero mean     : -196.2823
##  ARIMA(0,0,0)(1,0,0)[12] with non-zero mean : -190.0366
```

```
##  ARIMA(0,0,0)(0,0,1)[12] with non-zero mean : -189.9338
##  ARIMA(0,0,0)(1,0,1)[12] with non-zero mean : -185.8804
##  ARIMA(1,0,0)            with non-zero mean : -189.4403
##  ARIMA(0,0,1)            with non-zero mean : -189.451
##  ARIMA(1,0,1)            with non-zero mean : -185.7659
##
##  Best model: ARIMA(0,0,0)            with zero mean
```

```
ARIMAfitmeta
```

```
## Series: diffclose
## ARIMA(0,0,0) with zero mean
##
## sigma^2 = 0.01081:  log likelihood = 100.53
## AIC=-199.06   AICc=-199.03   BIC=-196.28
```

The best model is thus ARIMA(0,0,0)

Next we fit the model as ARIMA(0, 0, 0).

```
diffclose_ar<- arima(diffclose, order=c(0,0,0))
diffclose_ar
```

```
##
## Call:
## arima(x = diffclose, order = c(0, 0, 0))
##
## Coefficients:
##       intercept
##          0.0154
## s.e.     0.0094
##
## sigma^2 estimated as 0.01057:  log likelihood = 101.85,  aic = -199.69
```

```
BIC(diffclose_ar)
```

```
## [1] -194.1324
```