bitcoin.R

Alexandros

2023-01-22

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.1.2
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.4.0      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.0
## v readr   1.4.0      v forcats 0.5.1
```

```
## Warning: package 'ggplot2' was built under R version 4.1.3
```

```
## Warning: package 'tibble' was built under R version 4.1.3
```

```
## Warning: package 'tidyr' was built under R version 4.1.3
```

```
## Warning: package 'dplyr' was built under R version 4.1.3
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
library(ggpubr)
```

```
## Warning: package 'ggpubr' was built under R version 4.1.3
```

```
library(tsibble)
```

```
## Warning: package 'tsibble' was built under R version 4.1.3
```

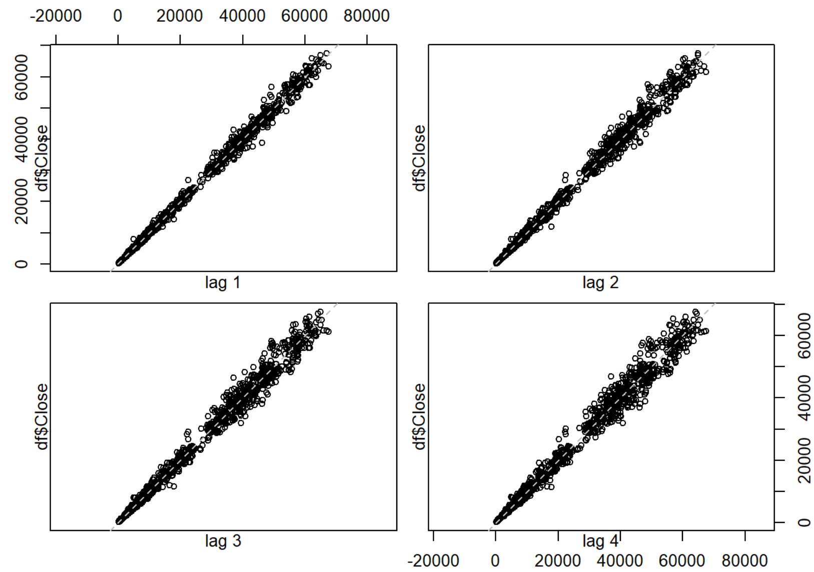
```
##
## Attaching package: 'tsibble'
```

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

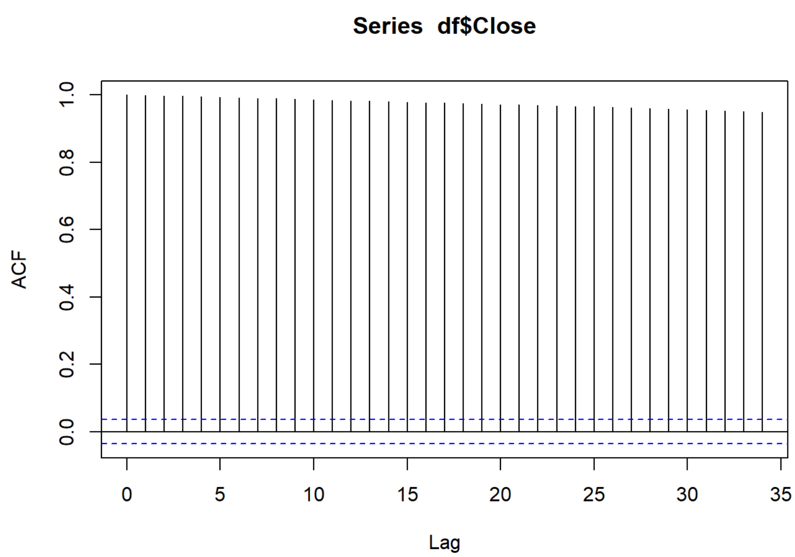
```
df=read.csv("BTC-USD.csv")

df %>% is.na() %>% sum
```

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

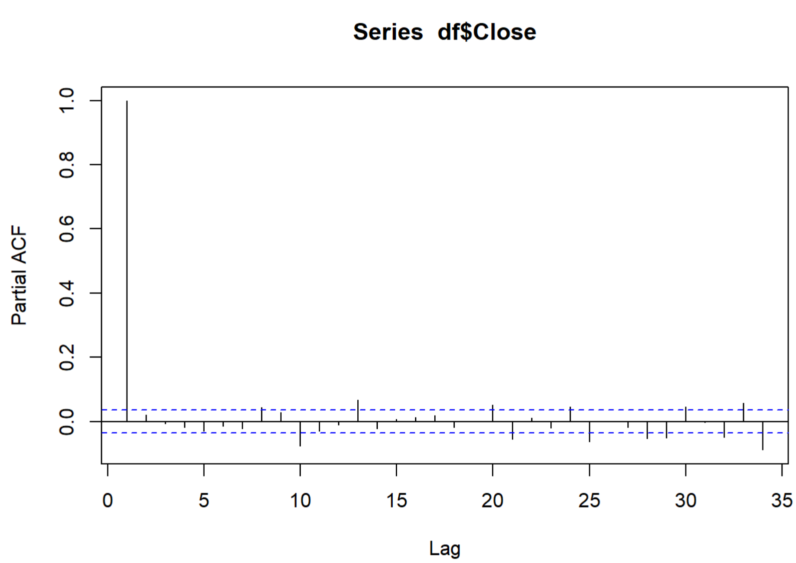
```
### Format date column as date
df$Date=df$Date %>% as.Date(format="%Y-%m-%d")

df=df %>% as_tsibble(index=Date)

##### Autocorrelation and partial autocorrelation

lag.plot(df$Close,lags=4)
```




```
acf(df$Close)
```



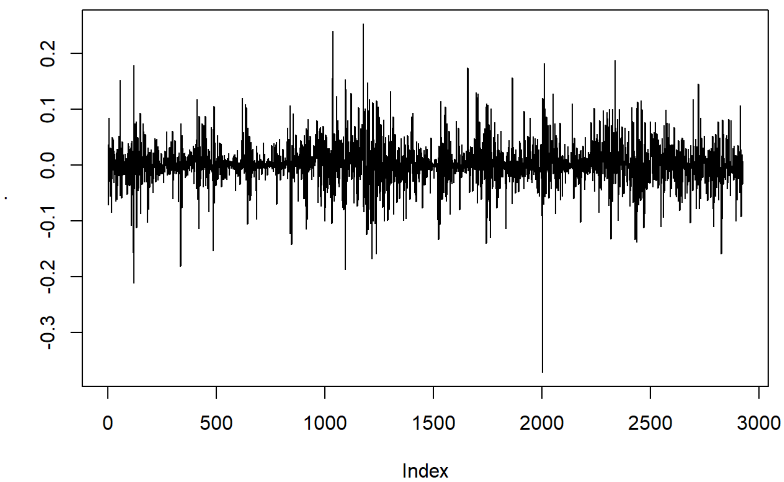

```
pacf(df$Close)
```



```
pacf(df$Close,plot=FALSE)
```

```
##
## Partial autocorrelations of series 'df$Close', by lag
##
##      1      2      3      4      5      6      7      8      9     10     11
##  0.999  0.021 -0.006 -0.019 -0.029 -0.014 -0.022  0.044  0.029 -0.077 -0.029
##     12     13     14     15     16     17     18     19     20     21     22
## -0.010  0.067 -0.022  0.008  0.014  0.018 -0.019  0.002  0.051 -0.056  0.012
##     23     24     25     26     27     28     29     30     31     32     33
## -0.020  0.045 -0.063 -0.001 -0.019 -0.053 -0.051  0.045 -0.002 -0.049  0.058
##     34
## -0.088
```

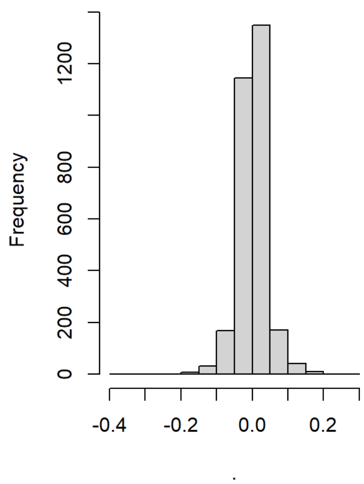
```
######### Cast into terms of growth rates

dfperc=df %>% mutate_at(vars(2,3,4,5,6,7),.funs=function(x) (x-lag(x))/lag(x))

dfperc$Close %>% plot(type="l")
```

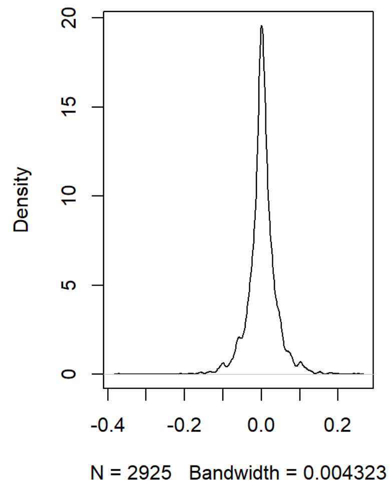



```
### seems like white noise with periods of increased volatility

##### Distirbution of percentace closing price change
dfperc$Close %>% na.omit %>% summary %>%
  append(sd(dfperc$Close,na.rm=T))
```

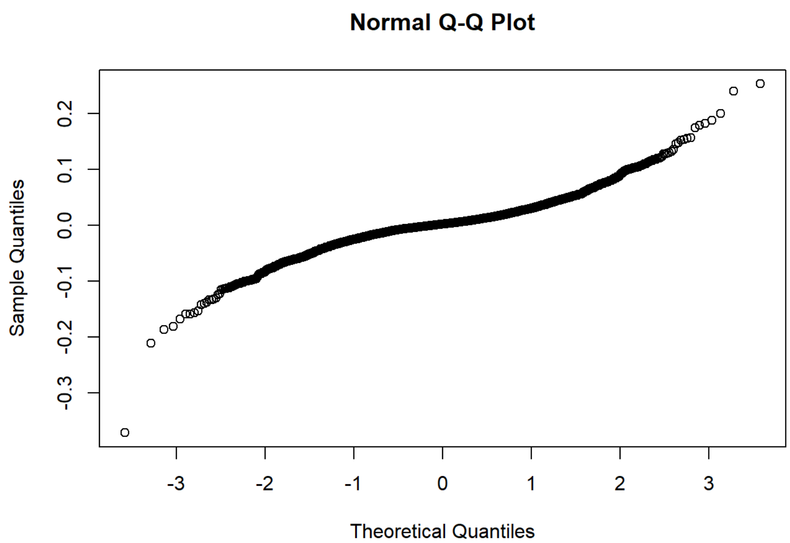
```
##         Min.      1st Qu.       Median         Mean      3rd Qu.         Max.
## -0.371695386 -0.013779656  0.001922469  0.002027929  0.017980108  0.252471694
##
##  0.038677277
```

```
par(mfrow=c(1,2))
dfperc$Close %>% na.omit %>% hist(main="")
dfperc$Close %>% na.omit %>% density %>% plot(main="")
```






```
par(mfrow=c(1,1))
qqnorm(y=dfperc$Close)
```




```
e1071::kurtosis(dfperc$Close,na.rm=T)
```

```
## [1] 7.030796
```

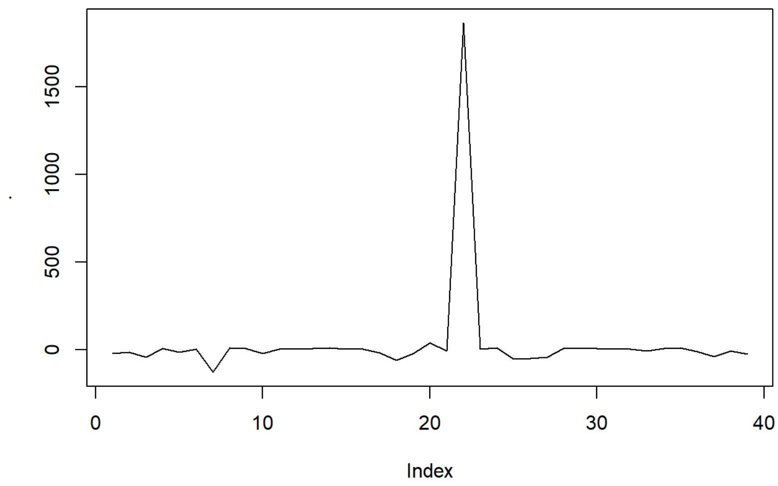
```
#Overall the distirbution of daily returns is leptokurtic, as shown by the density plot and the estimated kurtosis


#####Distirbution of daily returns by period
#find a suitable period split based on available data
dfperc$Close %>% length # Observations are length of series - 1 , due to taking differences
```

```
## [1] 2926
```

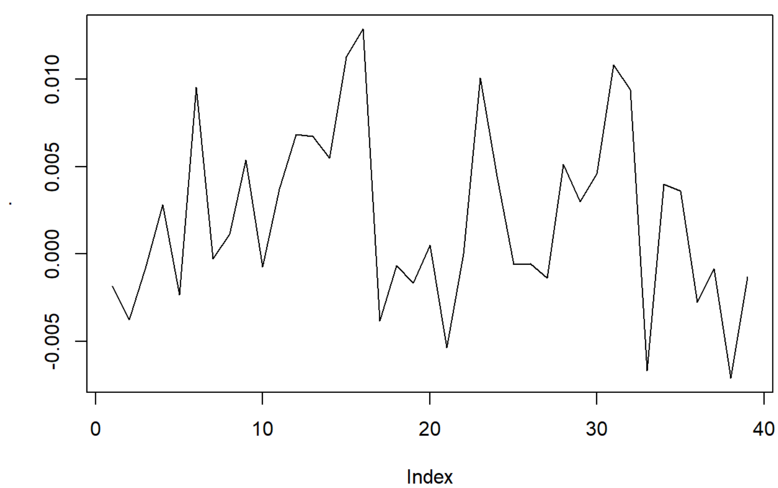
```
df_periods=dfperc %>% na.omit
df_periods$period=rep(c(1:39),each=75)

Price_periods=df_periods %>% as.tibble %>% group_by(period) %>%
  summarize(mean=mean(Close),std=sd(Close))
```

```
## Warning: `as.tibble()` was deprecated in tibble 2.0.0.
## i Please use `as_tibble()` instead.
## i The signature and semantics have changed, see `?as_tibble`.
```

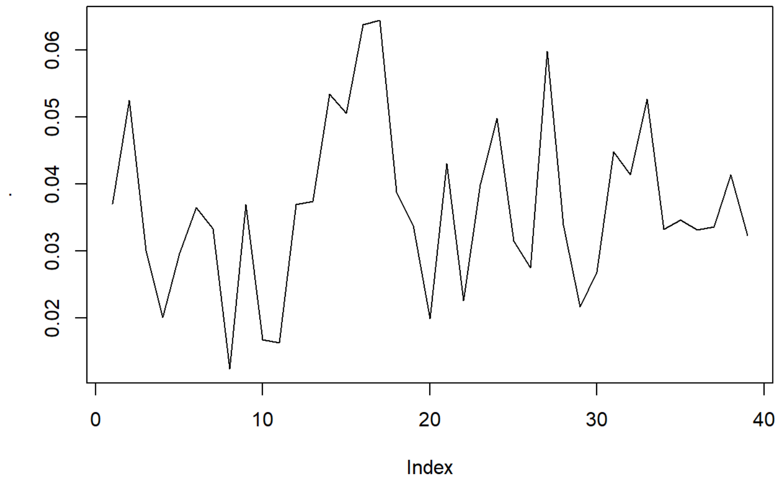
```
Price_periods$cv=Price_periods$std/Price_periods$mean  #estimate coefficient of variation for each period

Price_periods$cv %>% plot(type="l")
```




```
Price_periods$mean %>% plot(type="l")
```




```
Price_periods$std %>% plot(type="l")
```



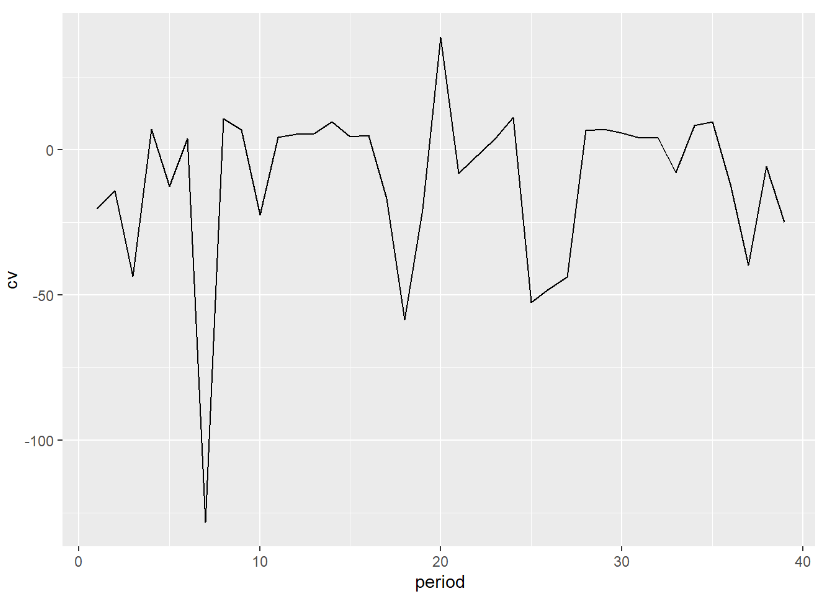

```
#There is an outlier in the coefficient of variation
Price_periods[which(Price_periods$cv==max(Price_periods$cv)),]
```

```
## # A tibble: 1 x 4
##   period      mean    std    cv
##    <int>     <dbl>  <dbl> <dbl>
## 1     22 0.0000121 0.0225 1865.
```

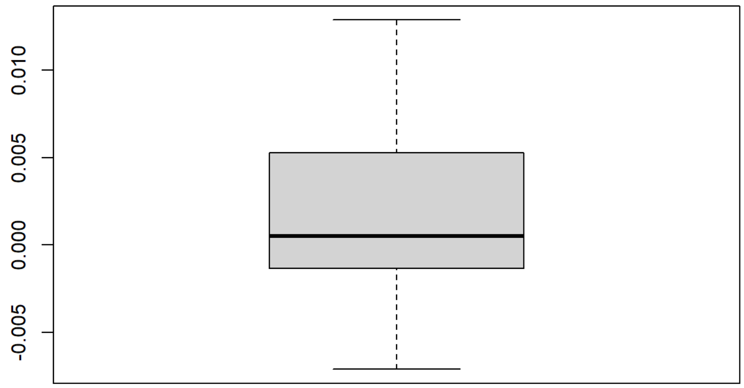
```
Price_periods %>% arrange(desc(cv))
```

```
## # A tibble: 39 x 4
##    period      mean    std      cv
##     <int>     <dbl>  <dbl>   <dbl>
##  1     22 0.0000121 0.0225 1865.
##  2     20 0.000513  0.0199   38.7
##  3     24 0.00447   0.0498   11.1
##  4      8 0.00115   0.0123   10.8
##  5     14 0.00550   0.0534    9.71
##  6     35 0.00361   0.0346    9.59
##  7     34 0.00400   0.0332    8.29
##  8     29 0.00301   0.0216    7.18
##  9      4 0.00282   0.0201    7.11
## 10      9 0.00539   0.0369    6.85
## # ... with 29 more rows
```

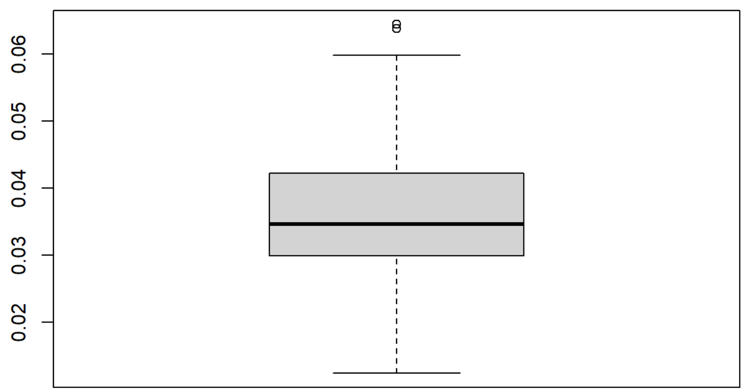
```
#Lets plot the coefficient of variation boxplot without the aforementioned outlier
ggplot(data=Price_periods %>% filter(period!=22))+
  geom_line(aes(x=period,y=cv))
```



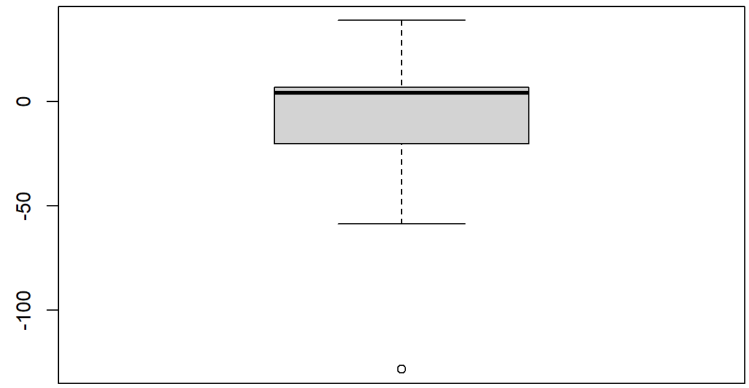
```
Price_periods$mean %>% boxplot
```



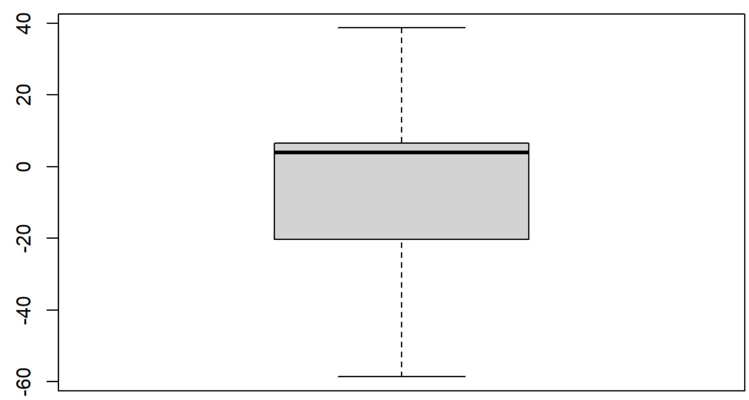

```
Price_periods$std %>% boxplot
```




```
Price_periods[which(Price_periods$cv!=max(Price_periods$cv)),"cv"] %>% boxplot
```

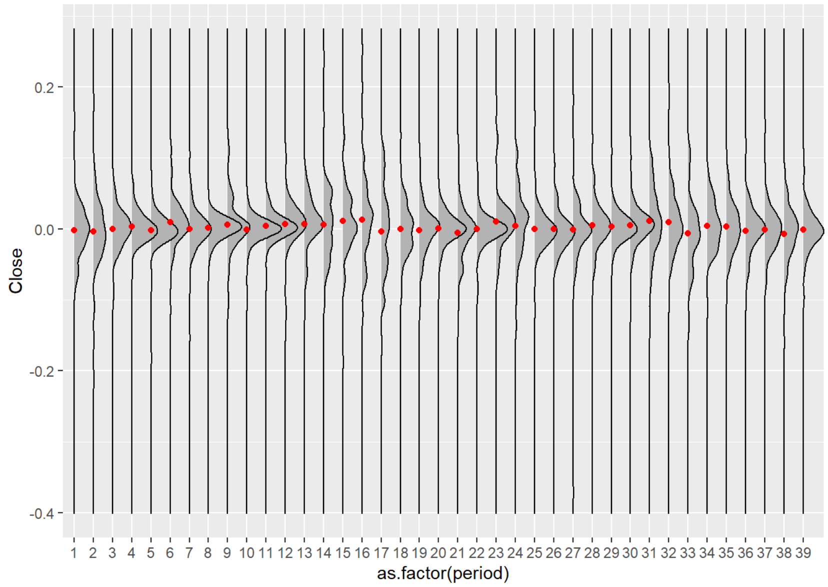



```
Price_periods[which(!(Price_periods$period %in% c(7,22))),"cv"] %>% boxplot
```



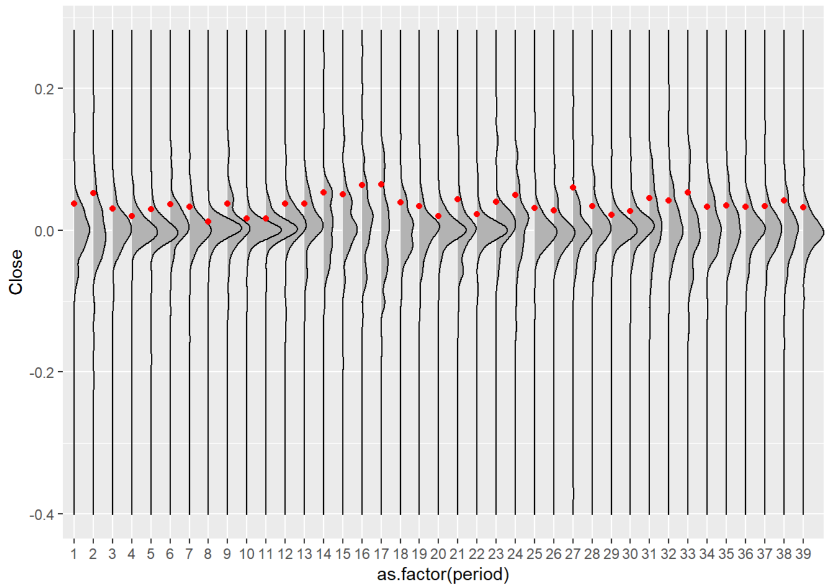

```
#Ridgelines  over time

ggplot(df_periods, aes(x =Close, y =as.factor(period))) +
  ggridges::geom_density_ridges()+
  geom_point(data=Price_periods,aes(x=mean,y=period),col="red")+coord_flip()
```

```
## Picking joint bandwidth of 0.0099
```



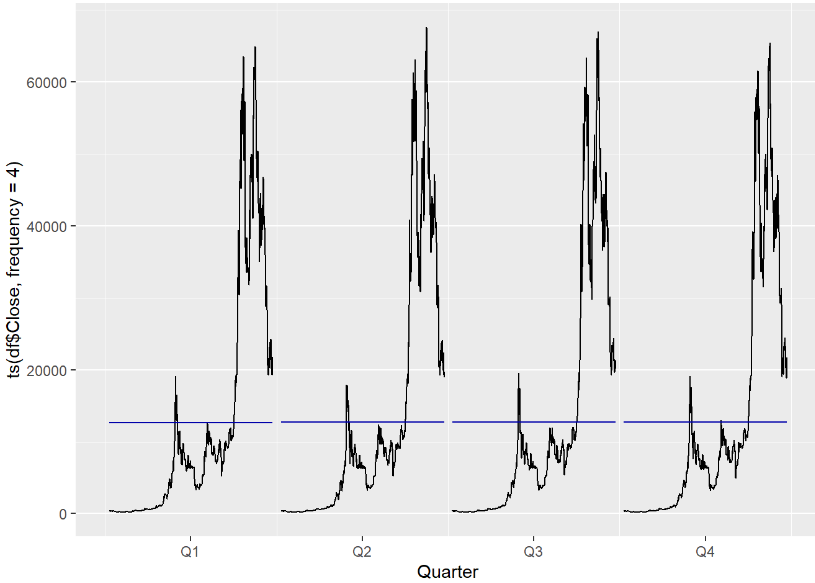

```
ggplot(df_periods, aes(x =Close, y =as.factor(period))) +
  ggridges::geom_density_ridges()+
  geom_point(data=Price_periods,aes(x=std,y=period),col="red")+coord_flip()
```

```
## Picking joint bandwidth of 0.0099
```

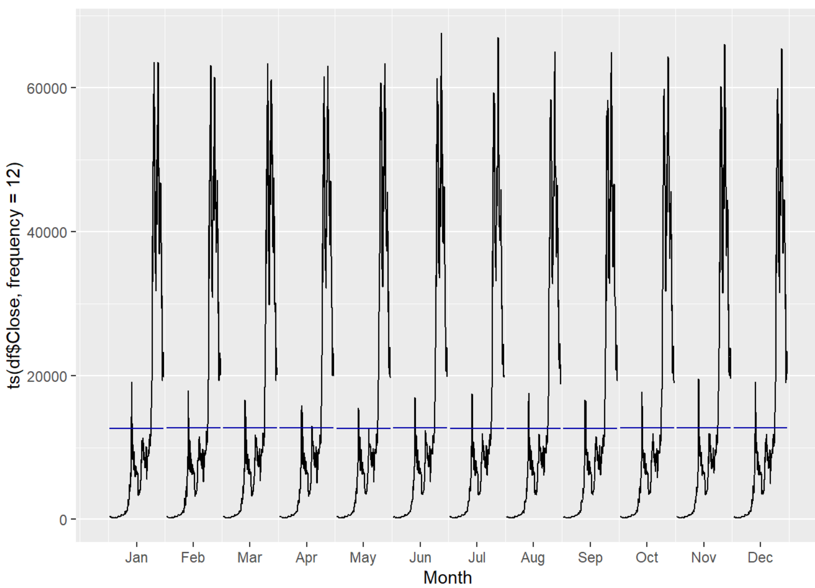



```
##### Seasonal adjustment

forecast::ggsubseriesplot(ts(df$Close,frequency=4)) #Check quarterly seasonality
```

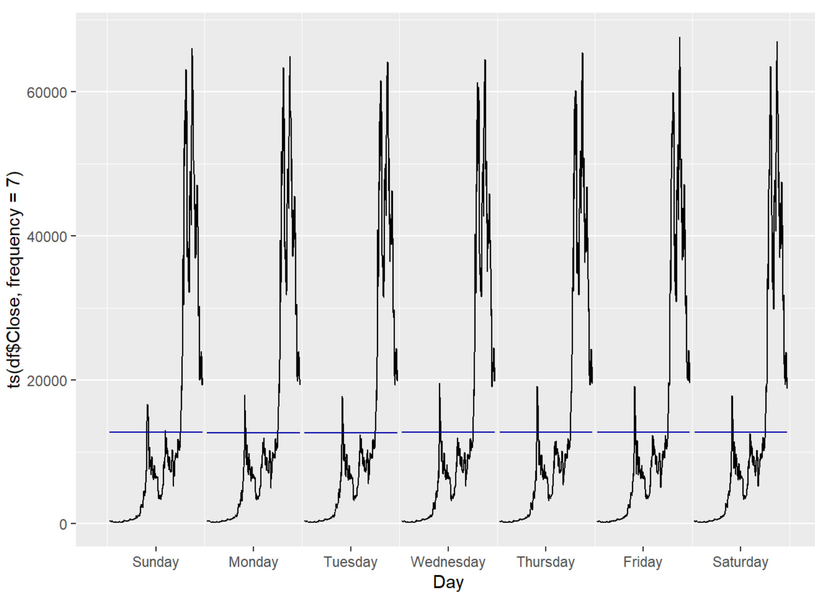
```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```



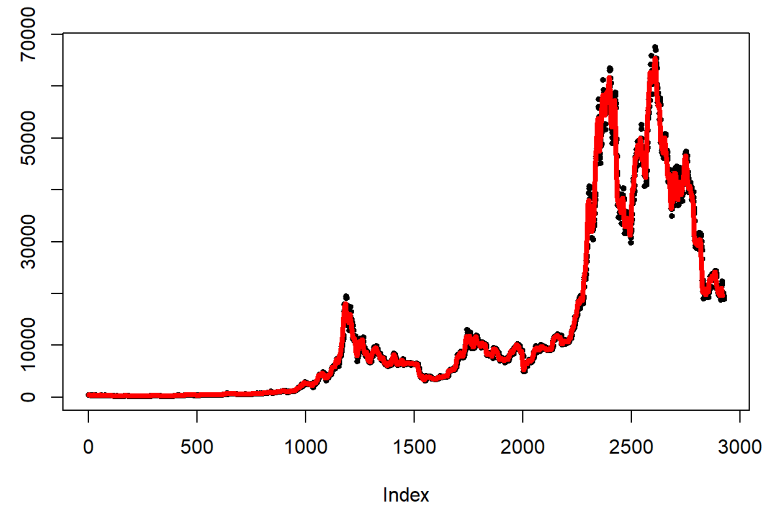

```
forecast::ggsubseriesplot(ts(df$Close,frequency=12)) #check monthly seasonality
```



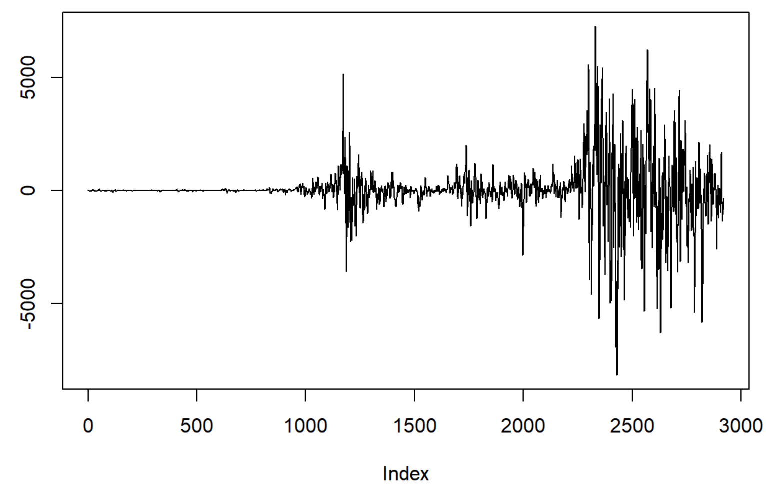

```
forecast::ggsubseriesplot(ts(df$Close,frequency=7)) #check daily seasonality
```



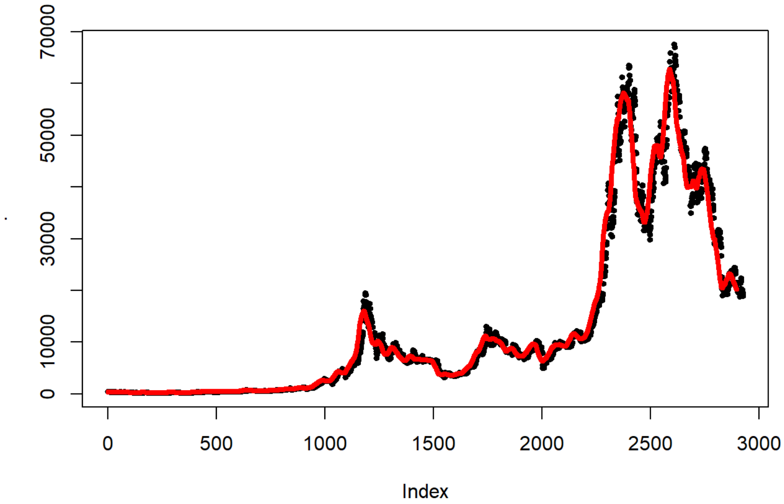
```
###Weekly rolling average
df$Close %>% plot(type="p",pch=21,bg="black",cex=0.6)
lines(zoo::rollmean(df$Close,k=7),col="red",lwd=4)
```



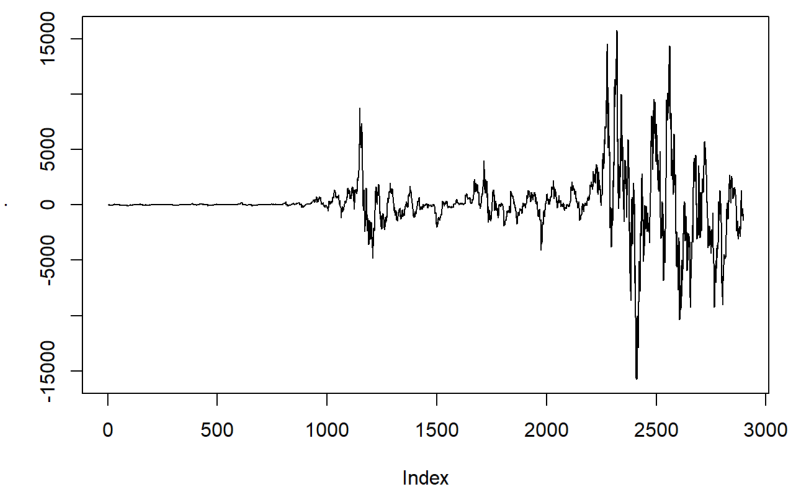

```
(
  df$Close[7:length(df$Close)] -zoo::rollmean(df$Close,k=7)
) %>% plot(type="l")  #plot after removing 7day moving average
```



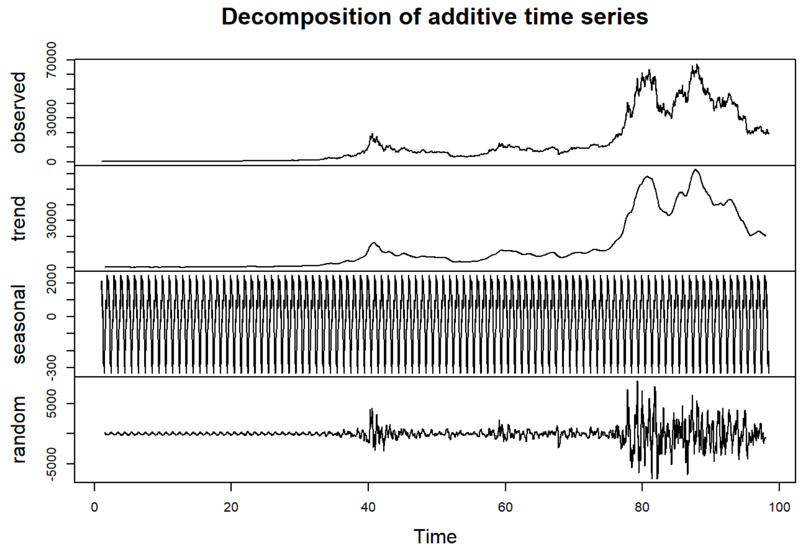

```
###Monthly rolling average

df$Close %>% plot(type="p",pch=21,bg="black",cex=0.6)
lines(zoo::rollmean(df$Close,k=30),col="red",lwd=4)
```



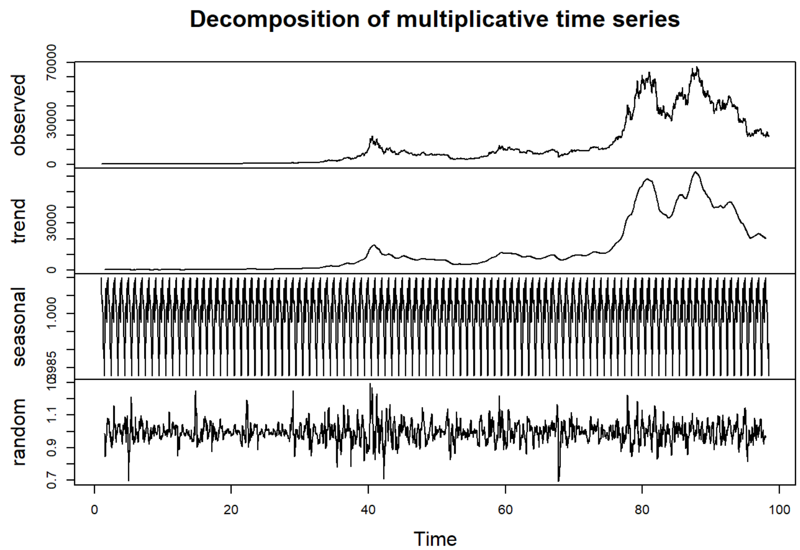

```
(
  df$Close[30:length(df$Close)] -zoo::rollmean(df$Close,k=30)
) %>% plot(type="l") #plot after removing 30day moving average
```



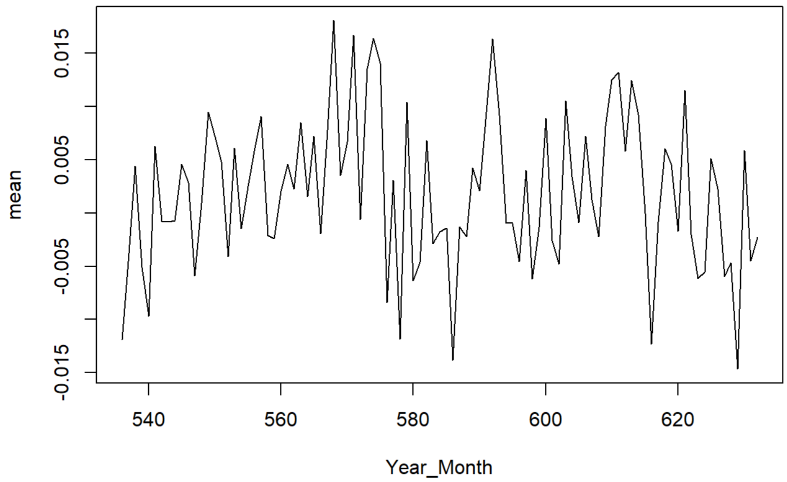

```
decadd=ts(df$Close,frequency = 30) %>% decompose (type="additive")
decmult=ts(df$Close,frequency = 30) %>% decompose (type="multiplicative")

decadd %>% plot
```



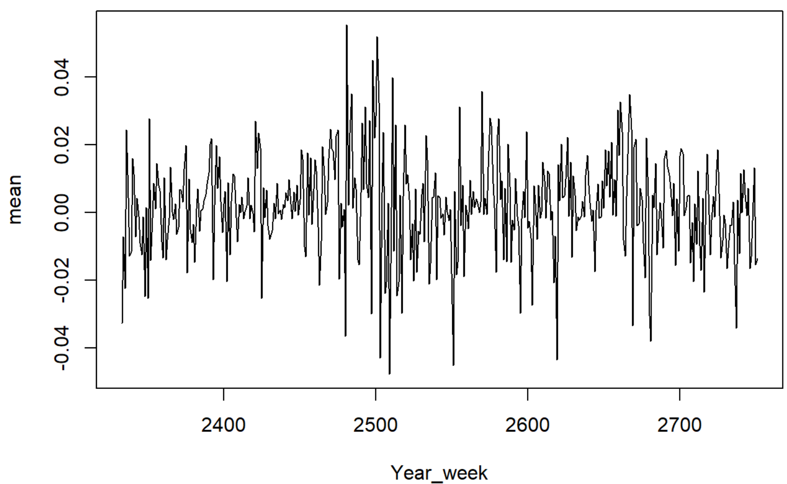

```
decmult %>% plot
```




```
##### Lower frequency plots

dfperc %>% na.omit %>% index_by(Year_Month = ~ yearmonth(.)) %>% summarize(mean=mean(Close)) %>% plot(type="l")
```



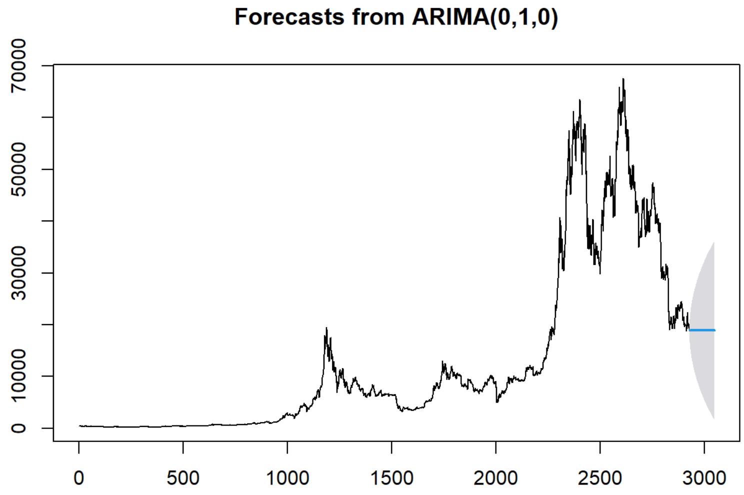

```
dfperc %>%na.omit %>%  index_by(Year_week = ~ yearweek(.)) %>%  summarize(mean=mean(Close)) %>% plot(type="l")
```




```
##### Fit an ARIMA model
model=forecast::auto.arima(df$Close)

model
```

```
## Series: df$Close
## ARIMA(0,1,0)
##
## sigma^2 estimated as 645877:  log likelihood=-23716.25
## AIC=47434.51   AICc=47434.51   BIC=47440.49
```

```
#random walk

forecast::forecast(model, level=c(95), h=10*12) %>% plot
```




```
##### tidymodels

library(tidymodels)
```

```
## Warning: package 'tidymodels' was built under R version 4.1.3
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.0.0 --
```

```
## v broom        1.0.2     v rsample      1.1.1
## v dials        1.1.0     v tune         1.0.1
## v infer        1.0.4     v workflows    1.1.2
## v modeldata    1.0.1     v workflowsets 1.0.0
## v parsnip      1.0.3     v yardstick    1.1.0
## v recipes      1.0.4
```

```
## Warning: package 'broom' was built under R version 4.1.3
```

```
## Warning: package 'dials' was built under R version 4.1.3
```

```
## Warning: package 'scales' was built under R version 4.1.3
```

```
## Warning: package 'infer' was built under R version 4.1.3
```

```
## Warning: package 'modeldata' was built under R version 4.1.3
```

```
## Warning: package 'parsnip' was built under R version 4.1.3
```

```
## Warning: package 'recipes' was built under R version 4.1.3
```

```
## Warning: package 'rsample' was built under R version 4.1.3
```

```
## Warning: package 'tune' was built under R version 4.1.3
```

```
## Warning: package 'workflows' was built under R version 4.1.3
```

```
## Warning: package 'workflowsets' was built under R version 4.1.3
```

```
## Warning: package 'yardstick' was built under R version 4.1.3
```

```
## -- Conflicts ---------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Use suppressPackageStartupMessages() to eliminate package startup messages
```

```
library(modeltime)
```

```
## Warning: package 'modeltime' was built under R version 4.1.3
```

```
library(rsample)
library(timetk)
```

```
## Warning: package 'timetk' was built under R version 4.1.3
```

```
tsplits=time_series_split(data=df %>% as.tibble(),date_var=Date,assess=30,cumulative=TRUE)

###visualize train_test split

tsplits %>%  tk_time_series_cv_plan() %>%
  plot_time_series_cv_plan(Date,Close,.interactive = FALSE)
```


```
#list of possible models https://www.tidymodels.org/find/parsnip/

model_fit_naive=naive_reg() %>%
  set_engine("naive") %>%
  fit(Close~Date,training(tsplits))

show_engines("exp_smoothing")
```

```
## # A tibble: 4 x 2
##   engine    mode
##   <chr>     <chr>
## 1 ets       regression
## 2 croston   regression
## 3 theta     regression
## 4 smooth_es regression
```

```
model_fit_ets = exp_smoothing() %>%
  set_engine("ets") %>%
  fit(Close ~ Date, training(tsplits))
```

```
## frequency = 7 observations per 1 week
```

```
model_fit_arima = arima_reg() %>%
  set_engine("auto_arima") %>%
  fit(Close ~ Date, training(tsplits))
```

```
## frequency = 7 observations per 1 week
```

```
model_fit_prophet=prophet_reg() %>%
  set_engine("prophet", quarterly.seasonality = TRUE) %>%
  fit(Close ~ Date, training(tsplits))
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

```
#####Comparison with model time
library(modeltime)

model_table =modeltime_table(
  model_fit_naive,
  model_fit_arima,
  model_fit_ets,
  model_fit_prophet
)

model_table
```

```
## # Modeltime Table
## # A tibble: 4 x 3
##   .model_id .model   .model_desc
##       <int> <list>   <chr>
## 1         1 <fit[+]> NAIVE
## 2         2 <fit[+]> ARIMA(0,1,0)(0,0,1)[7]
## 3         3 <fit[+]> ETS(M,A,N)
## 4         4 <fit[+]> PROPHET
```

```
#Calibration

calibration_table=model_table %>%
  modeltime_calibrate(testing(tsplits))

calibration_table
```

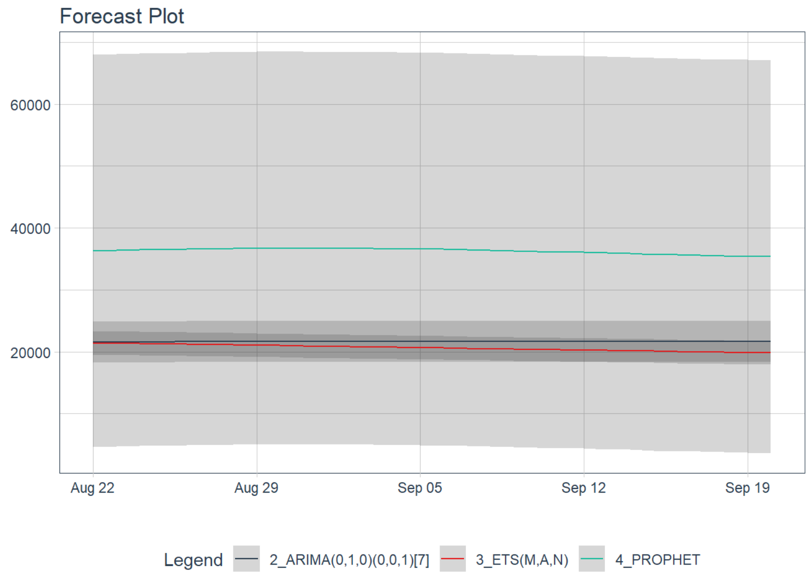
```
## # Modeltime Table
## # A tibble: 4 x 5
##   .model_id .model   .model_desc            .type .calibration_data
##       <int> <list>   <chr>                  <chr> <list>
## 1         1 <fit[+]> NAIVE                  Test  <tibble [30 x 4]>
## 2         2 <fit[+]> ARIMA(0,1,0)(0,0,1)[7] Test  <tibble [30 x 4]>
## 3         3 <fit[+]> ETS(M,A,N)             Test  <tibble [30 x 4]>
## 4         4 <fit[+]> PROPHET                Test  <tibble [30 x 4]>
```

```
#forecasting


calibration_table %>%
  modeltime_forecast(actual_data = df) %>%
  plot_modeltime_forecast(.interactive = FALSE)
```

```
## Using '.calibration_data' to forecast.
```

```
## Error: Column `Date` (index) must not contain `NA`.
```

```
## Warning: Unknown or uninitialised column: `.key`.
```




```
#See accuracy of each model

calibration_table %>%
  modeltime_accuracy() %>%
  table_modeltime_accuracy(.interactive = FALSE)
```

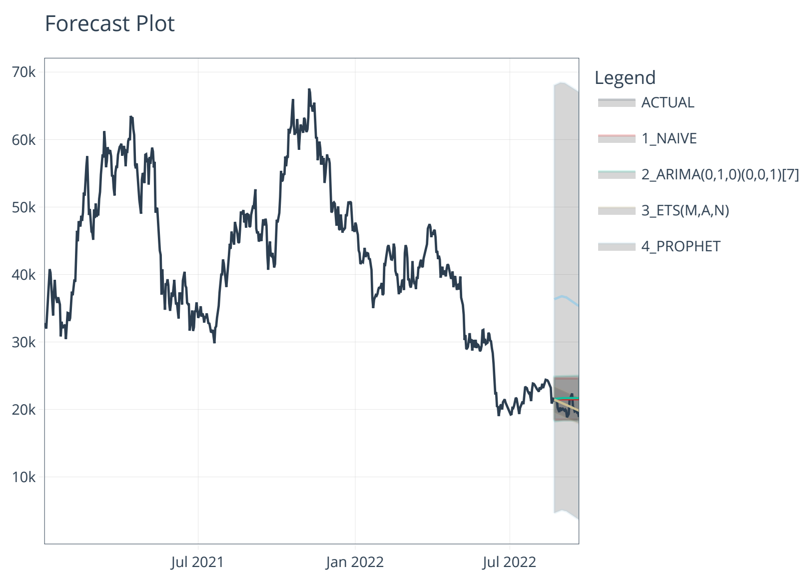
```
## Warning: A correlation computation is required, but `estimate` is constant
## and has 0 standard deviation, resulting in a divide by 0 error. `NA` will be
## returned.
```

Accuracy Table

.model_id	.model_desc	.type	mae	mape	mase	smape	rmse	rsq
1	NAIVE	Test	1340.97	6.79	2.84	6.49	1549.67	NA
2	ARIMA(0,1,0)(0,0,1)[7]	Test	1441.84	7.30	3.06	6.96	1671.48	0.24
3	ETS(M,A,N)	Test	800.40	3.96	1.70	3.92	962.19	0.09
4	PROPHET	Test	16005.60	79.27	33.93	56.65	16034.62	0.01

```
calibration_table %>%
  modeltime_forecast(actual_data = df %>% as.tibble) %>%filter(.index>"2021-01-01") %>%
  plot_modeltime_forecast(.interactive = T)
```

```
## Using '.calibration_data' to forecast.
```



```
#####Machine Learning models
df
```

```
## # A tsibble: 2,926 x 7 [1D]
##    Date        Open  High   Low Close Adj.Close   Volume
##    <date>     <dbl> <dbl> <dbl> <dbl>     <dbl>    <dbl>
##  1 2014-09-17  466.  468.  452.  457.      457. 21056800
##  2 2014-09-18  457.  457.  413.  424.      424. 34483200
##  3 2014-09-19  424.  428.  385.  395.      395. 37919700
##  4 2014-09-20  395.  423.  390.  409.      409. 36863600
##  5 2014-09-21  408.  412.  393.  399.      399. 26580100
##  6 2014-09-22  399.  407.  397.  402.      402. 24127600
##  7 2014-09-23  402.  442.  396.  436.      436. 45099500
##  8 2014-09-24  436.  436.  421.  423.      423. 30627700
##  9 2014-09-25  423.  424.  409.  412.      412. 26814400
## 10 2014-09-26  411.  415.  400.  404.      404. 21460800
## # ... with 2,916 more rows
```

```
training(tsplits)
```

```
## # A tibble: 2,896 x 7
##    Date        Open  High   Low Close Adj.Close   Volume
##    <date>     <dbl> <dbl> <dbl> <dbl>     <dbl>    <dbl>
##  1 2014-09-17  466.  468.  452.  457.      457. 21056800
##  2 2014-09-18  457.  457.  413.  424.      424. 34483200
##  3 2014-09-19  424.  428.  385.  395.      395. 37919700
##  4 2014-09-20  395.  423.  390.  409.      409. 36863600
##  5 2014-09-21  408.  412.  393.  399.      399. 26580100
##  6 2014-09-22  399.  407.  397.  402.      402. 24127600
##  7 2014-09-23  402.  442.  396.  436.      436. 45099500
##  8 2014-09-24  436.  436.  421.  423.      423. 30627700
##  9 2014-09-25  423.  424.  409.  412.      412. 26814400
## 10 2014-09-26  411.  415.  400.  404.      404. 21460800
## # ... with 2,886 more rows
```

```
###Preprocess
rspec=recipe(Close ~ Date, training(tsplits)) %>%
  step_timeseries_signature(Date) %>% step_fourier(Date,period=365,K=6) %>%
  step_rm(contains("am.pm"), contains("hour"), contains("minute"),
          contains("second"), contains("xts"),contains("lbl"))

rspec%>% prep() %>% juice() %>% names
```

```
##  [1] "Date"           "Close"          "Date_index.num" "Date_year"
##  [5] "Date_year.iso"  "Date_half"      "Date_quarter"   "Date_month"
##  [9] "Date_day"       "Date_wday"      "Date_mday"      "Date_qday"
## [13] "Date_yday"      "Date_mweek"     "Date_week"      "Date_week.iso"
## [17] "Date_week2"     "Date_week3"     "Date_week4"     "Date_mday7"
## [21] "Date_sin365_K1" "Date_cos365_K1" "Date_sin365_K2" "Date_cos365_K2"
## [25] "Date_sin365_K3" "Date_cos365_K3" "Date_sin365_K4" "Date_cos365_K4"
## [29] "Date_sin365_K5" "Date_cos365_K5" "Date_sin365_K6" "Date_cos365_K6"
```

```
#Elastic NET model

model_spec_glmnet=linear_reg(penalty = 0.01, mixture = 0.5) %>%
  set_engine("glmnet")

workflow_fit_glmnet <- workflow() %>%
  add_model(model_spec_glmnet) %>%
  add_recipe(rspec %>% step_rm(Date)) %>%
  fit(training(tsplits))

model_spec_prophet_boost <- prophet_boost() %>%
  set_engine("prophet_xgboost", quarterly.seasonality = TRUE)

workflow_fit_prophet_boost <- workflow() %>%
  add_model(model_spec_prophet_boost) %>%
  add_recipe(rspec) %>%
  fit(training(tsplits))
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

```
## [15:40:57] WARNING: amalgamation/../src/learner.cc:627:
## Parameters: { "quarterly_seasonality" } might not be used.
##
##   This could be a false alarm, with some parameters getting used by language bindings but
##   then being mistakenly passed down to XGBoost core, or some parameter actually being used
##   but getting flagged wrongly here. Please open an issue if you find any such cases.
```

```
##### Assesing ml models

model_table =modeltime_table(
  workflow_fit_glmnet,
  workflow_fit_prophet_boost
)

calibration_table <- model_table %>%
  modeltime_calibrate(testing(tsplits))
calibration_table
```

```
## # Modeltime Table
## # A tibble: 2 x 5
##   .model_id .model     .model_desc               .type .calibration_data
##       <int> <list>     <chr>                     <chr> <list>
## 1         1 <workflow> GLMNET                    Test  <tibble [30 x 4]>
## 2         2 <workflow> PROPHET W/ XGBOOST ERRORS Test  <tibble [30 x 4]>
```

```
calibration_table %>%
  modeltime_accuracy() %>%
  table_modeltime_accuracy(.interactive = FALSE)
```

Accuracy Table

.model_id	.model_desc	.type	mae	mape	mase	smape	rmse	rsq
1	GLMNET	Test	13926.43	69.03	29.52	51.19	13963.21	0.29
2	PROPHET W/ XGBOOST ERRORS	Test	2378.16	11.98	5.04	11.13	2668.36	0.12

```
calibration_table %>%
  modeltime_forecast(actual_data = df) %>%filter(.index>"2021-01-01") %>%
  plot_modeltime_forecast(.interactive = T)
```

```
## Using '.calibration_data' to forecast.
```


```
#Overall the ets model seems to give us better predictions, followerd by prophet with xg boos
```