# R Notebook

Load the required Libraries

```r
library(quantmod)
```

```
## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

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

## Loading required package: TTR

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

## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```r
library(tseries)
library(timeSeries)
```

```
## Loading required package: timeDate

##
## Attaching package: 'timeSeries'

## The following object is masked from 'package:zoo':
##
##     time<-
```

```r
library(forecast)
```

```
## Registered S3 methods overwritten by 'forecast':
##   method             from
##   fitted.fracdiff    fracdiff
##   residuals.fracdiff fracdiff
```

```r
library(xts)
```

Scrape the data for Tech Mahindra company from Yahoo Finance

```r
tech_stock_df <- getSymbols('TECHM.NS', from='2015-01-01', to='2018-05-01',auto.assign = FALSE)
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
```

```
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.

## Warning: TECHM.NS contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.

## Warning: 'indexClass<-' is deprecated.
## Use 'tclass<-' instead.
## See help("Deprecated") and help("xts-deprecated").
```

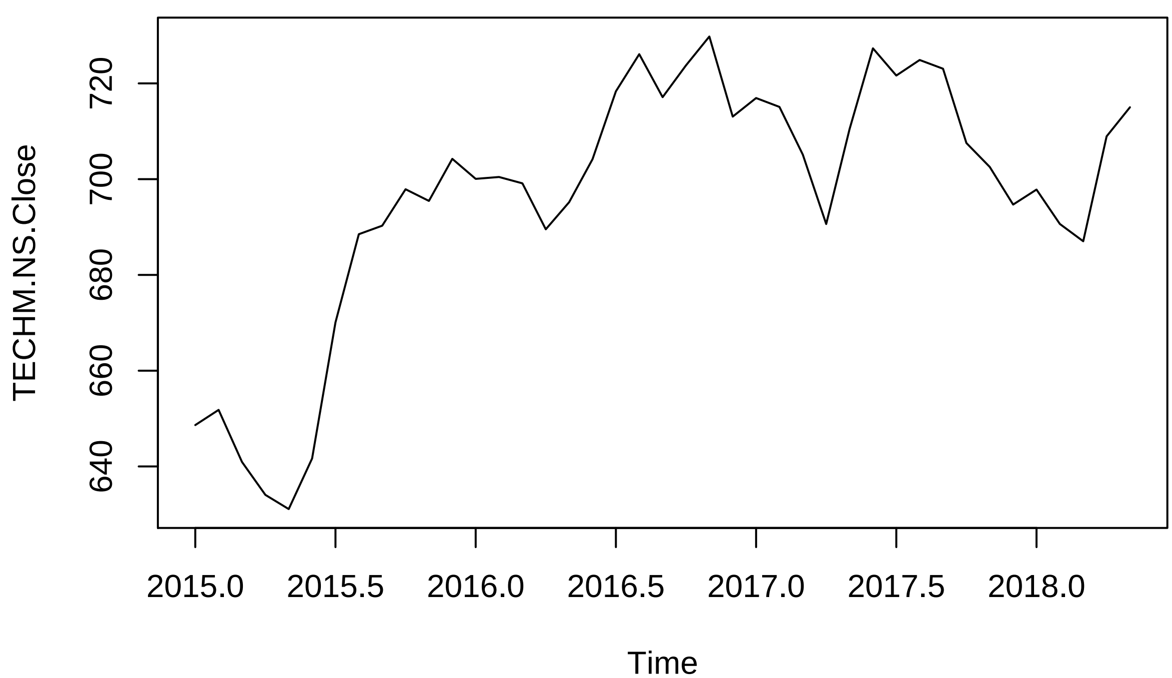
```
tech_stock_df <- na.omit(tech_stock_df)
```

Select the Closed price column and create its data frame

```
tech_stock_closed_price_df <- tech_stock_df[,4]
```

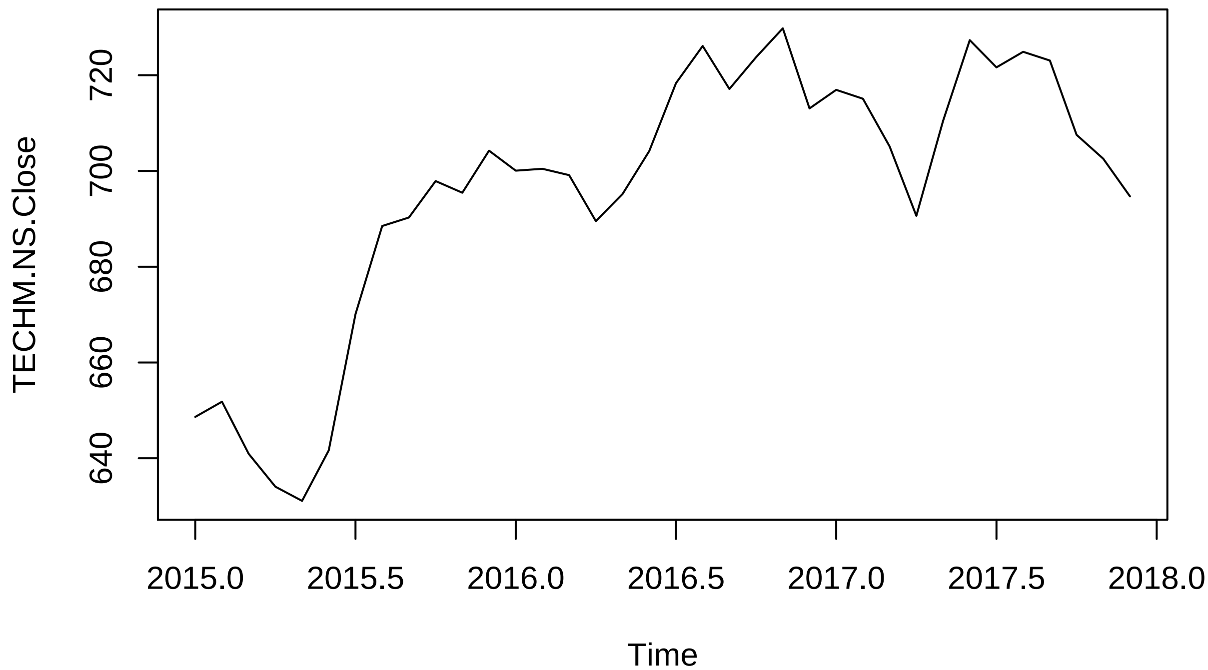
Convert this data into time series format

```
ts_stock_data <- ts(tech_stock_closed_price_df, start = c(2015,1),end = c(2018,5), frequency = 12 )

plot(ts_stock_data)
```



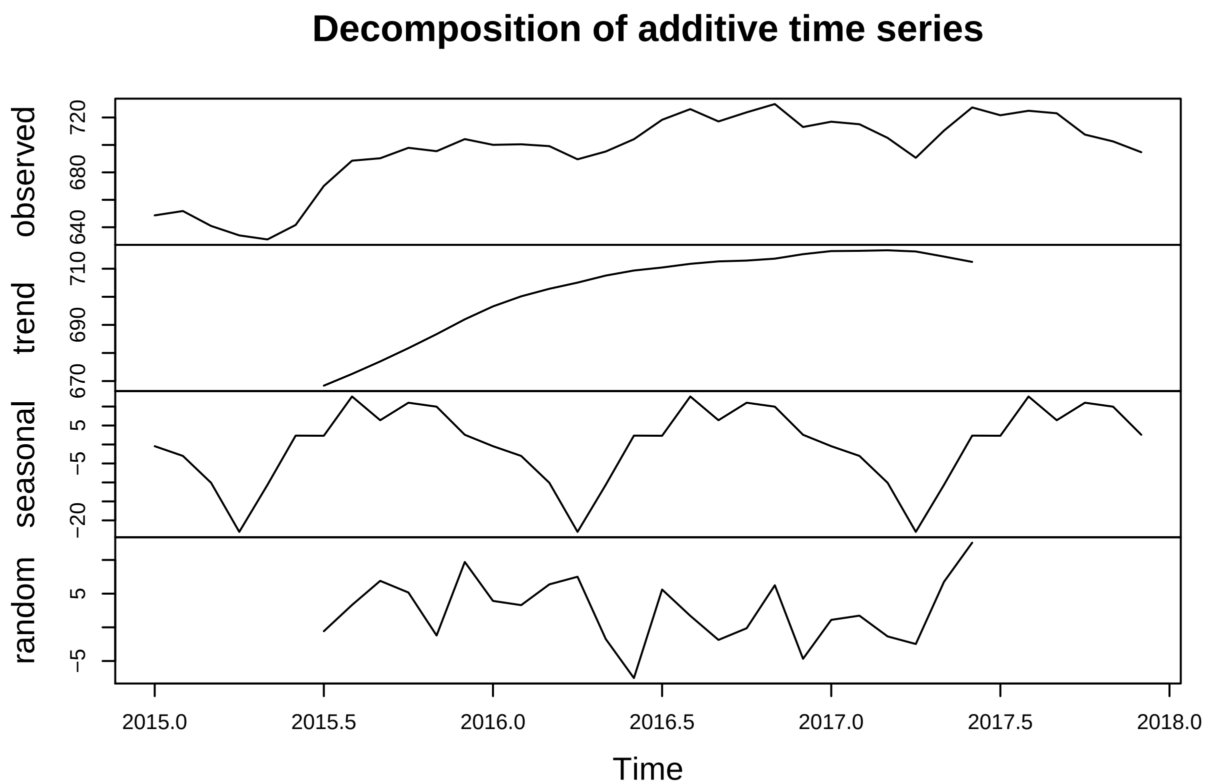

divide data into training and test data set

```
ts_stock_data_training = ts(tech_stock_closed_price_df, start=c(2015, 1), end=c(2017, 12), freq=12)
plot(ts_stock_data_training)
```




Exploratory Data Analysis

```
components.ts_stock_data_training = decompose(ts_stock_data_training)
plot(components.ts_stock_data_training)
```

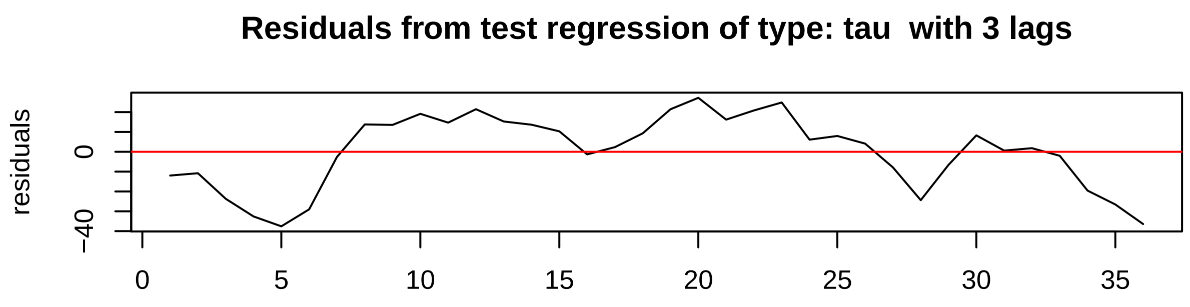


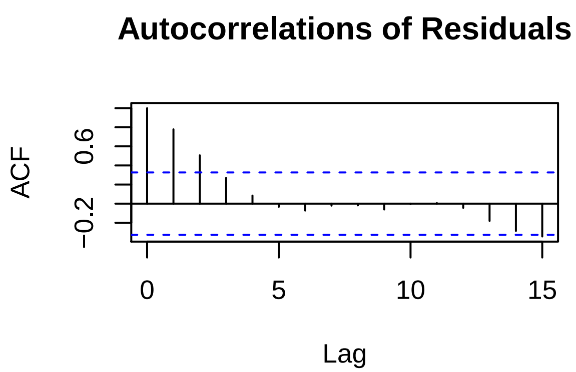

to achieve the stationarity perform Unit root test which finds out first difference or regression that should be used on trending data in order to make it stationary.

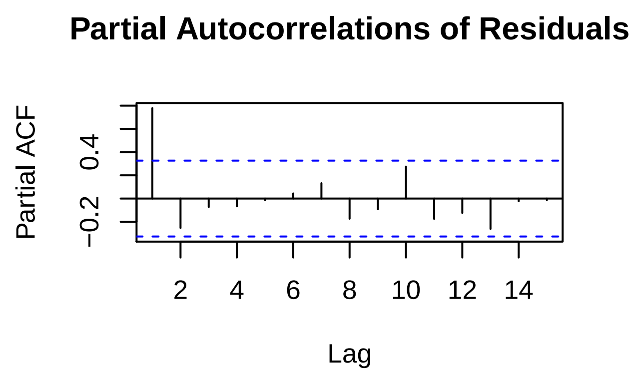
```
# install.packages("fUnitRoots")
library(fUnitRoots)
```

```
## Loading required package: fBasics

##
## Attaching package: 'fBasics'

## The following object is masked from 'package:TTR':
##
##     volatility
```

```
urkpssTest(ts_stock_data_training, type = c("tau"), lags = c("short"),use.lag = NULL, doplot = TRUE )
```

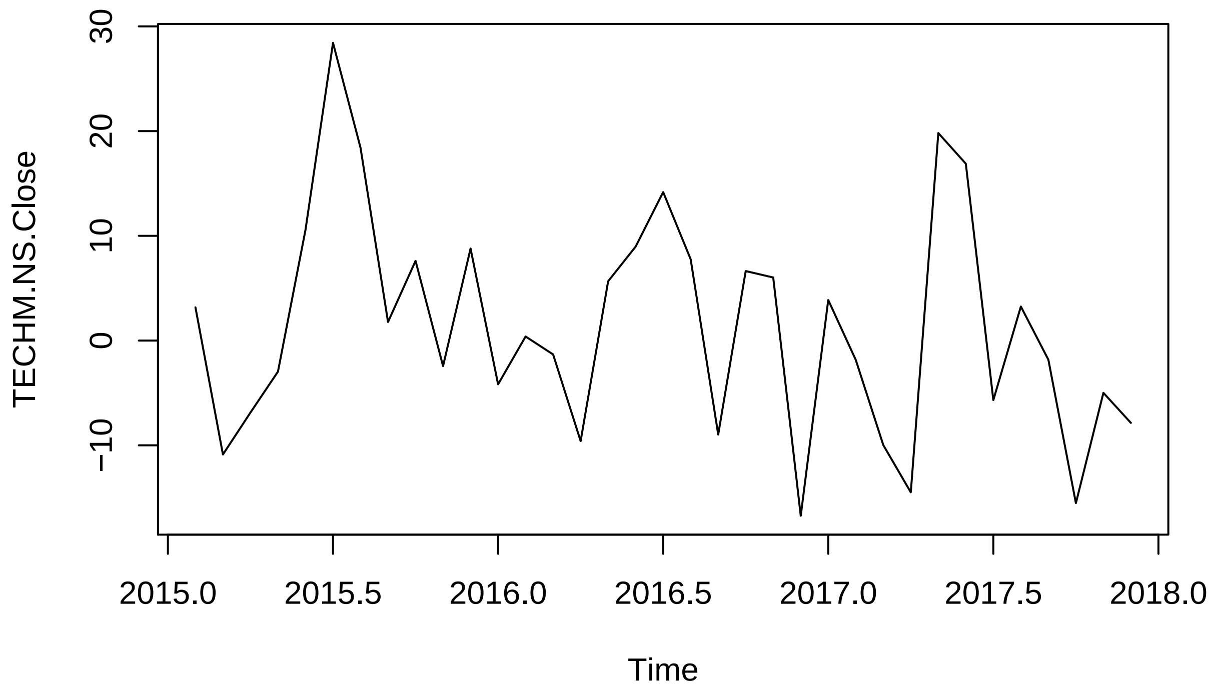


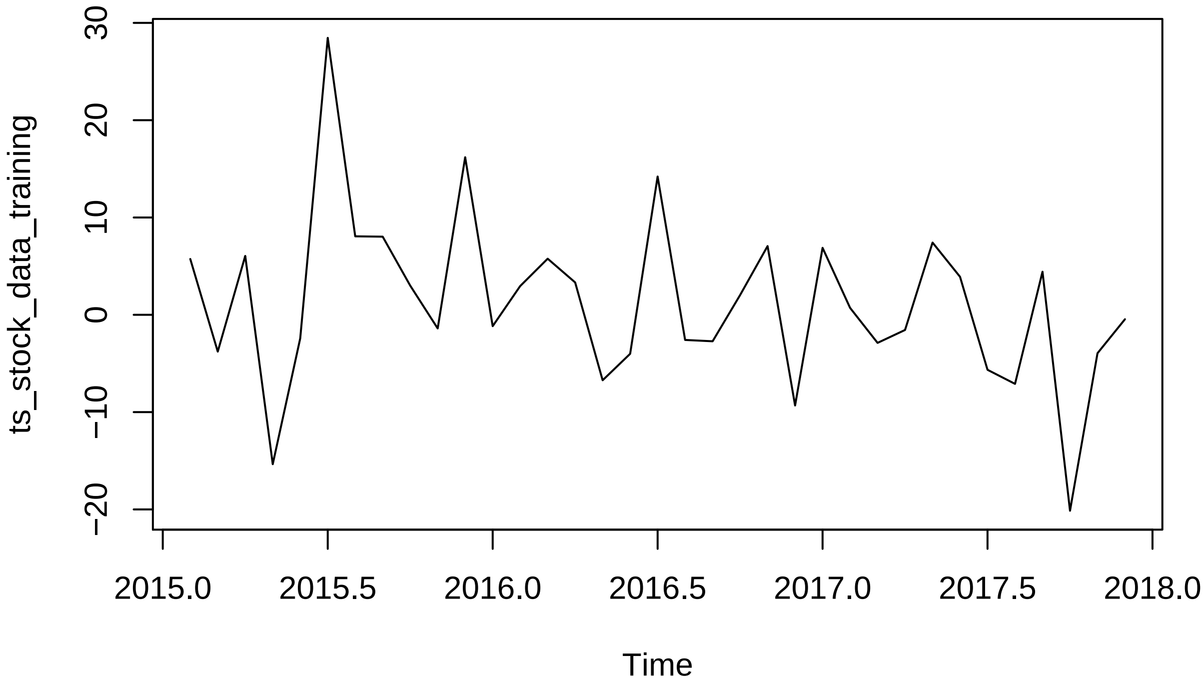

```
##
## Title:
##  KPSS Unit Root Test
##
## Test Results:
##   NA
##
## Description:
##  Wed Feb 12 16:01:36 2020 by user: gadka
```

```
ts_stock_data_stationary = diff(ts_stock_data_training, differences = 1)

plot(ts_stock_data_stationary)
```




Now remove Seasonality from the data

```
ts_stock_data_seasonally_adjusted <- ts_stock_data_training- components.ts_stock_data_training$seasonal
ts_stock_data_stationary <- diff(ts_stock_data_seasonally_adjusted, differences=1)
plot(ts_stock_data_stationary)
```

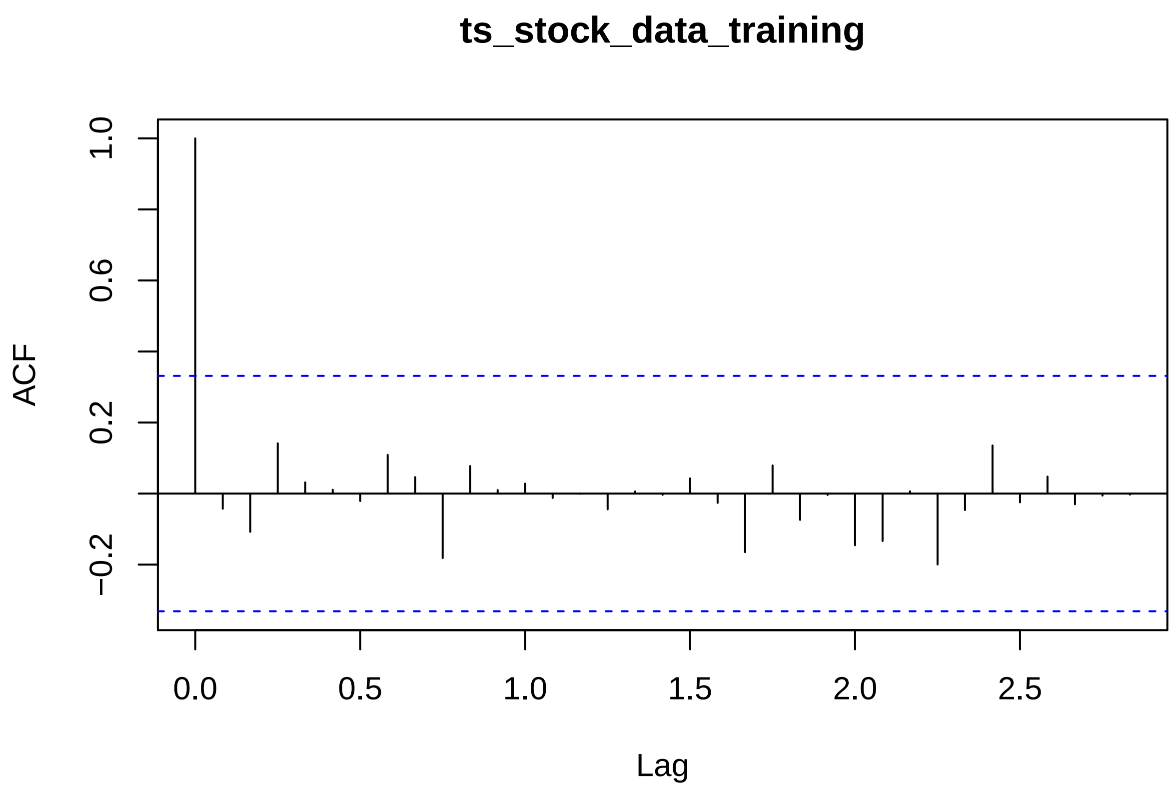



Model Estimation :

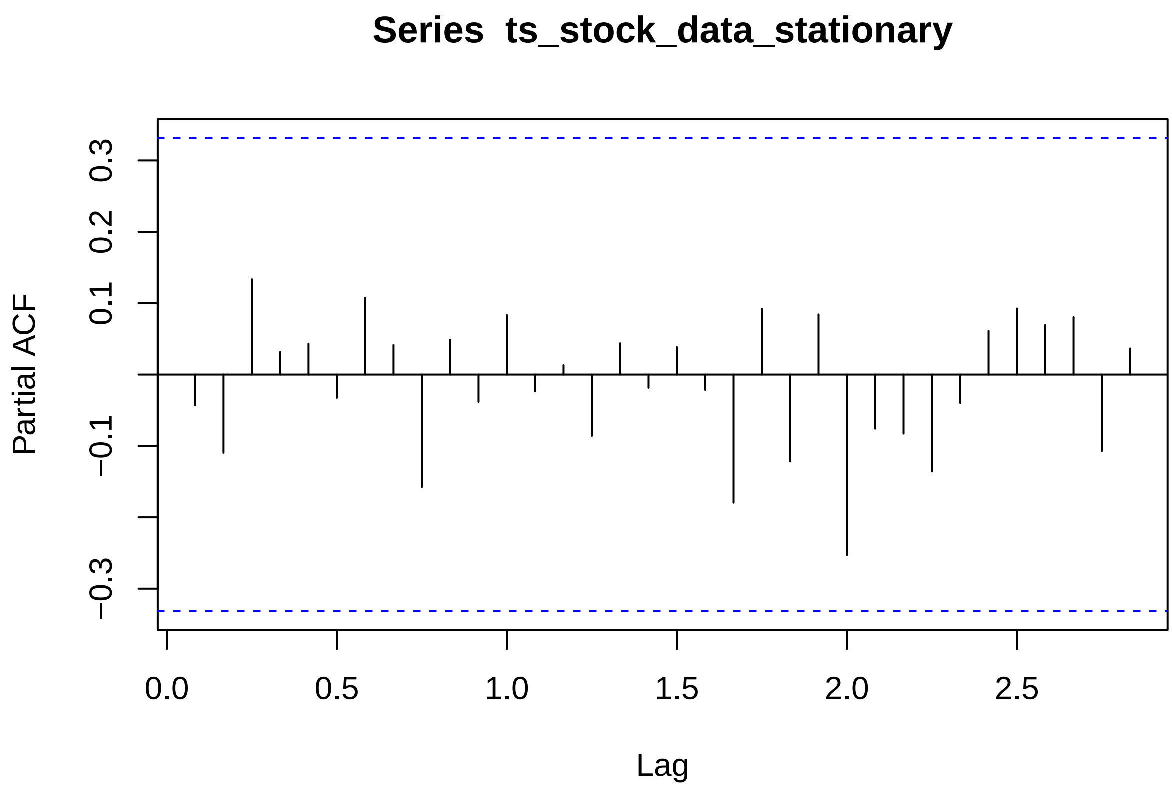
ACF(Auto Correlation Function) and PACF (Partial autoCorrelation Function) help to understand the correlation component of different data points at different time lags. Thus it helps to determine the order in which we are going to create ARIMA model.

To elaborate, we can say a given time series is a stationary one when its mean, variance and autocorrelation remains constant over time.

```
acf(ts_stock_data_stationary, lag.max=34)
```




```
pacf(ts_stock_data_stationary, lag.max=34)
```




using MA(1) that means p=0,q=1 and d=1 build the model

```
fit_model = Arima(ts_stock_data_training, order = c(0,1,1), include.drift = TRUE)
summary(fit_model)
```

```
## Series: ts_stock_data_training
## ARIMA(0,1,1) with drift
##
## Coefficients:
##          ma1   drift
##       0.3697  1.2804
## s.e.  0.1971  2.2890
##
## sigma^2 estimated as 105.3:  log likelihood=-130.2
## AIC=266.39   AICc=267.17   BIC=271.06
##
## Training set error measures:
##                       ME     RMSE      MAE          MPE     MAPE     MASE
## Training set -0.02356761 9.824054 8.152603 -0.004761122 1.170404 0.291452
##                     ACF1
## Training set -0.04217156
```

diagnosis measures

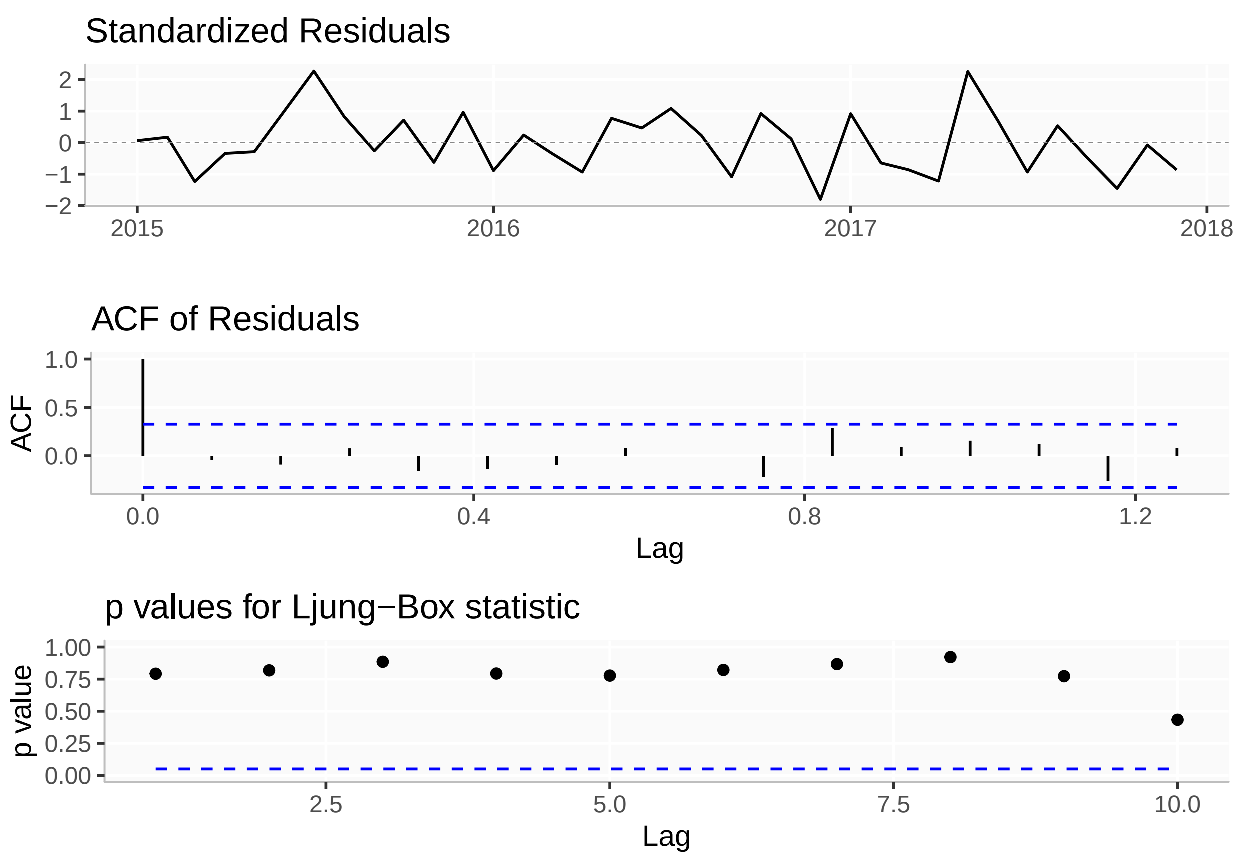
```
confint(fit_model)
```

```
##              2.5 %    97.5 %
## ma1   -0.01666639 0.7560526
## drift -3.20602927 5.7668073
```

```r
# install.packages("plotly")
library(ggfortify)
```

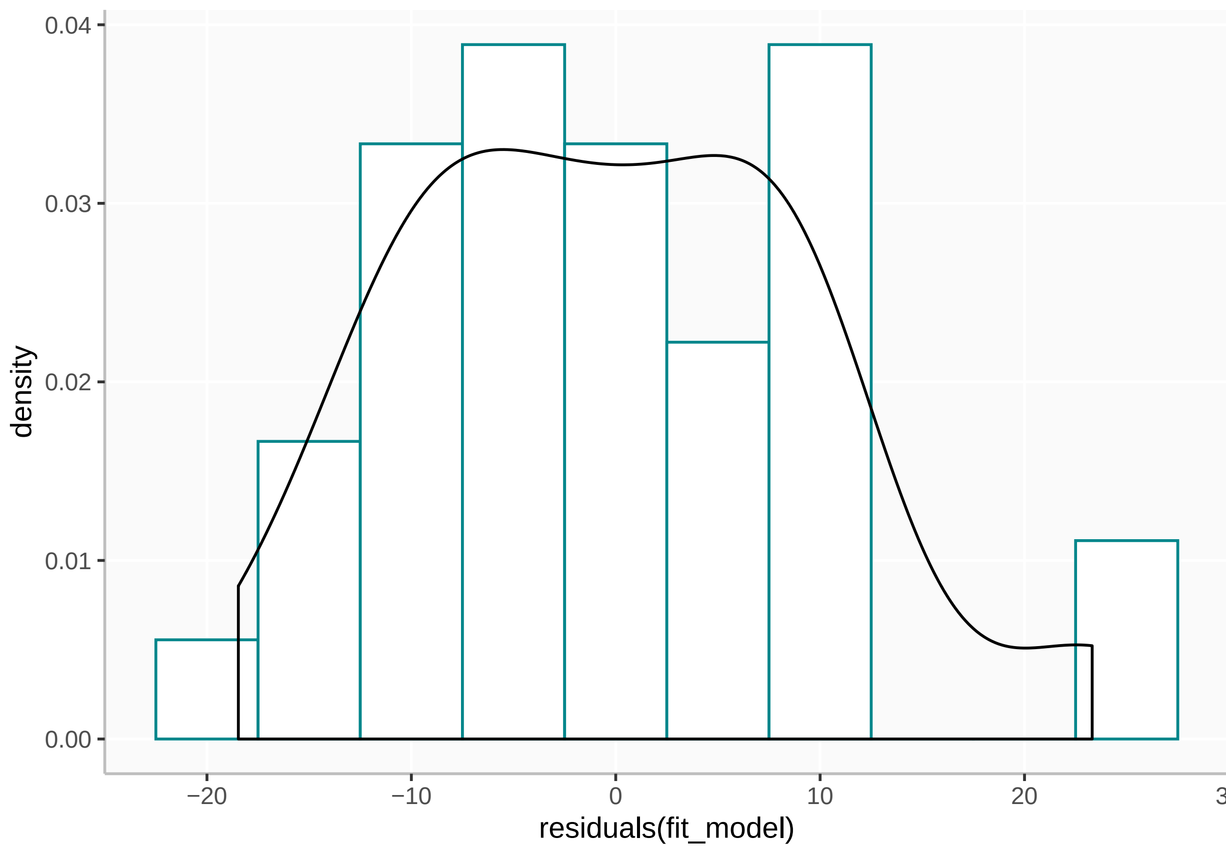
```
## Loading required package: ggplot2

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

```r
# ggfortify::ggtsdiag()
ggtsdiag(fit_model) +
  theme(panel.background = element_rect(fill = "gray98"),
        panel.grid.minor = element_blank(),
        axis.line.y = element_line(colour="gray"),
        axis.line.x = element_line(colour="gray"))
```



```r
ggplot(data=fit_model, aes(residuals(fit_model))) +
  geom_histogram(aes(y =..density..),
                 binwidth = 5,
                 col="turquoise4", fill="white") +
  geom_density(col=1) +
  theme(panel.background = element_rect(fill = "gray98"),
        panel.grid.minor = element_blank(),
        axis.line   = element_line(colour="gray"),
        axis.line.x = element_line(colour="gray"))
```

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



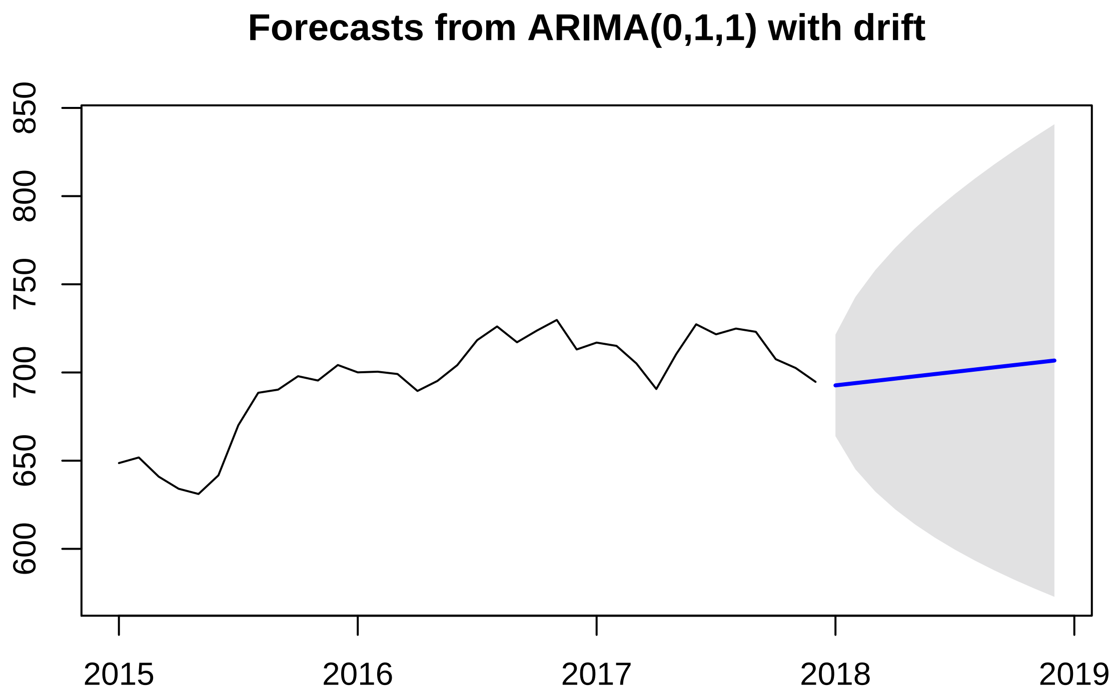

Create testing data set

```r
ts_stock_data_testing = ts(tech_stock_closed_price_df, start=c(2018, 1), freq=12)
nrow(ts_stock_data_testing)
```

```
## [1] 821
```

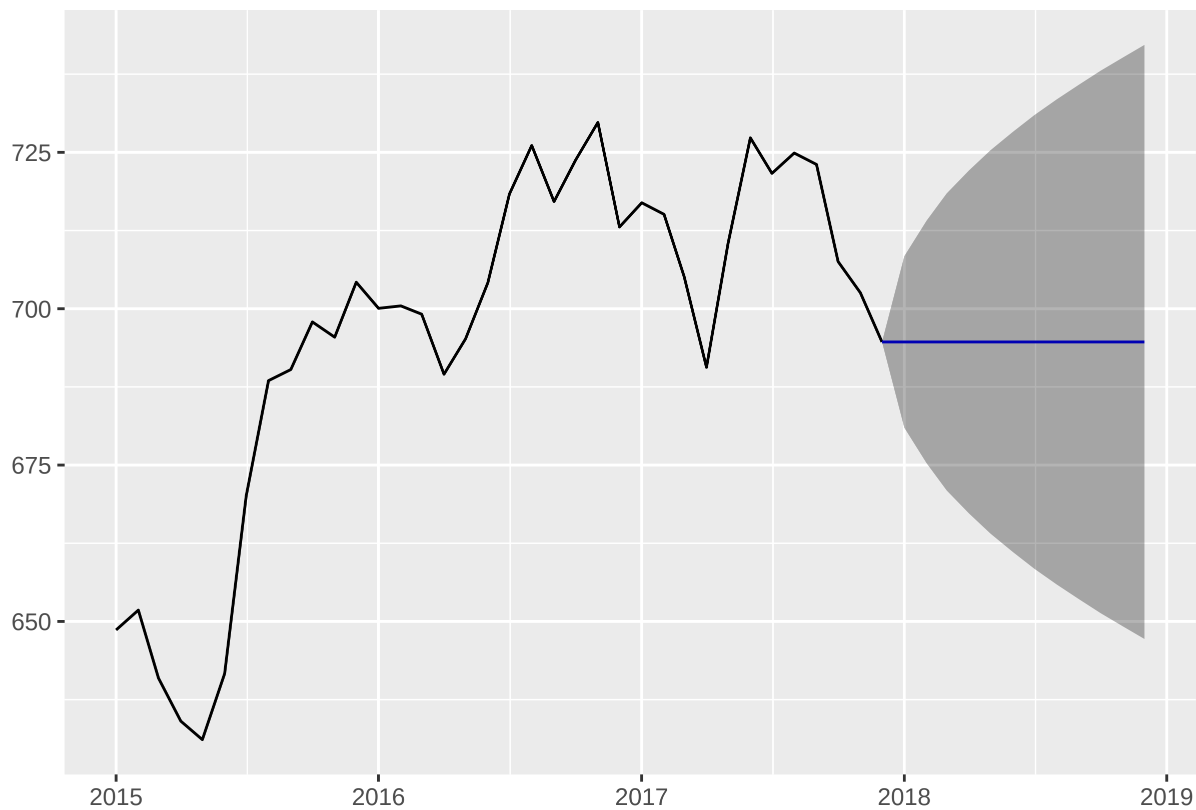
predict values for test data

```r
# library(TSPred)
pred_val = forecast(fit_model, h=12, level=c(99.5))
plot(pred_val)
```

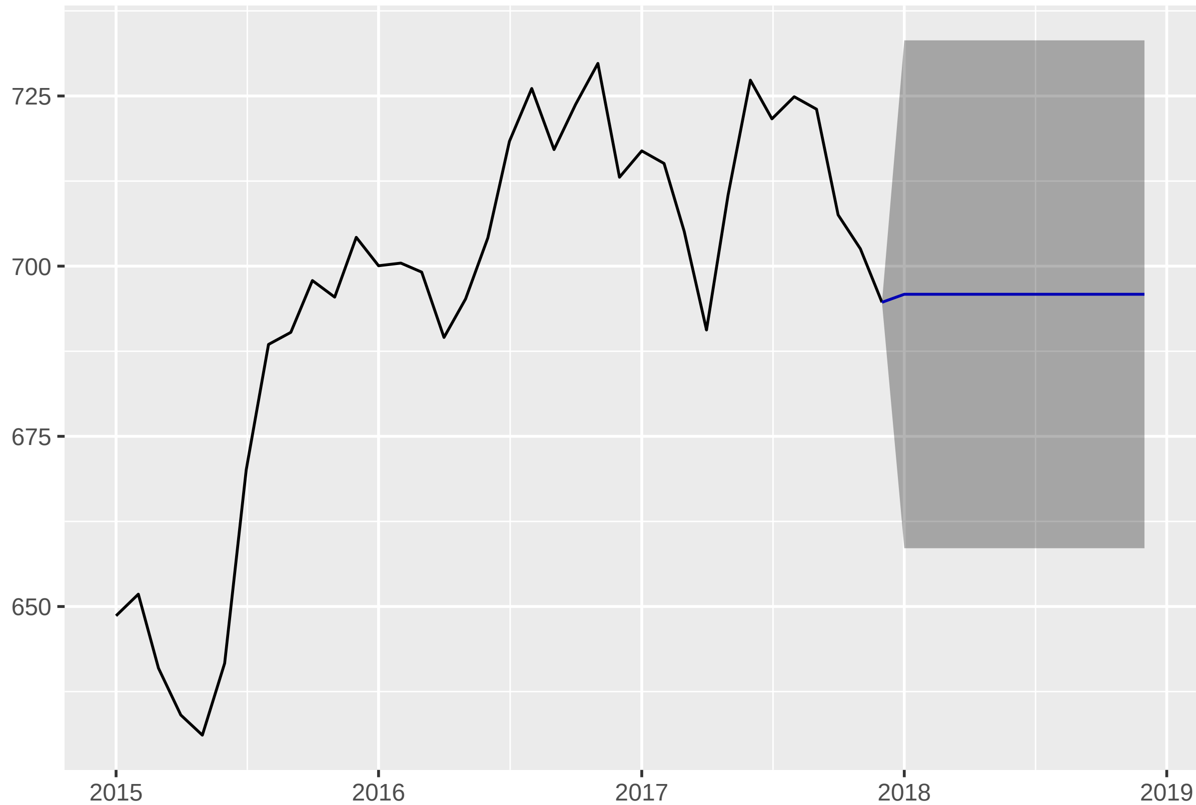



```
# plotarimapred(ts_stock_data_testing,fit.arima = pred_val, xlim = c(2018, 2019) )
```

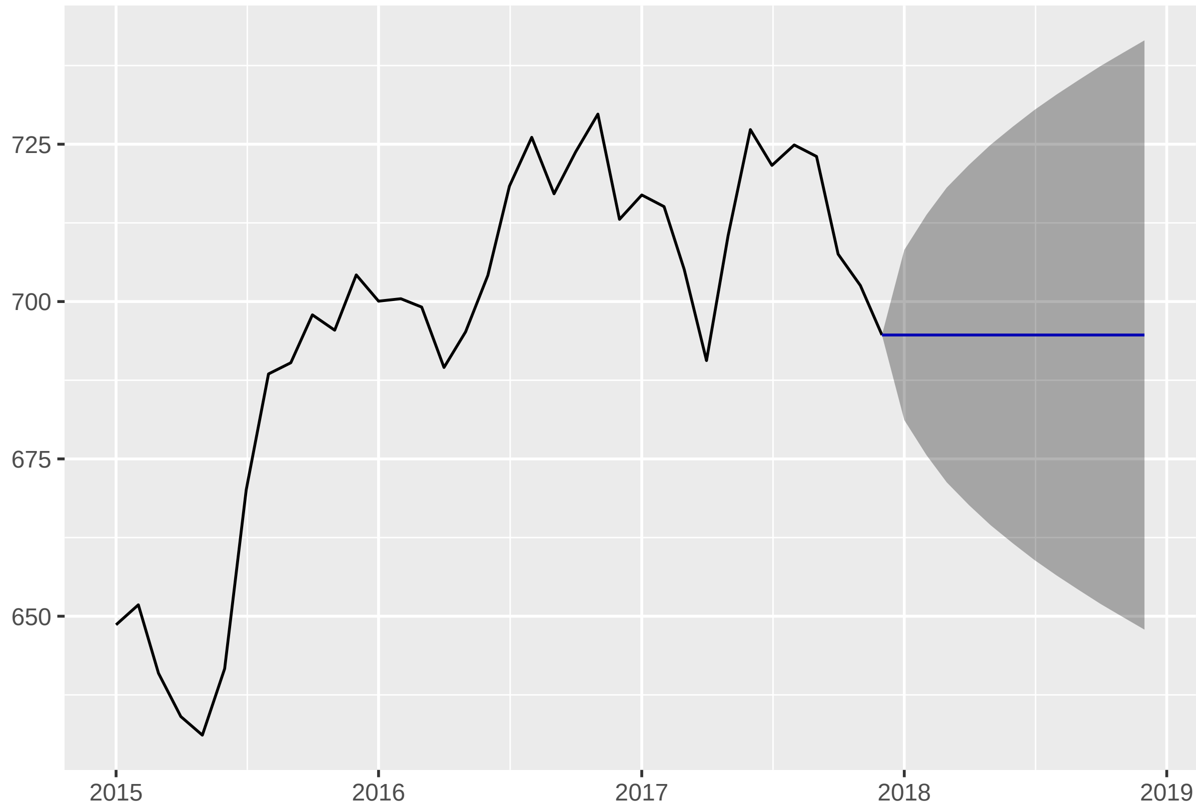
```
# other forecasting methods

fit_ets <- forecast(ets(ts_stock_data_training), h = 12)
autoplot(fit_ets)
```



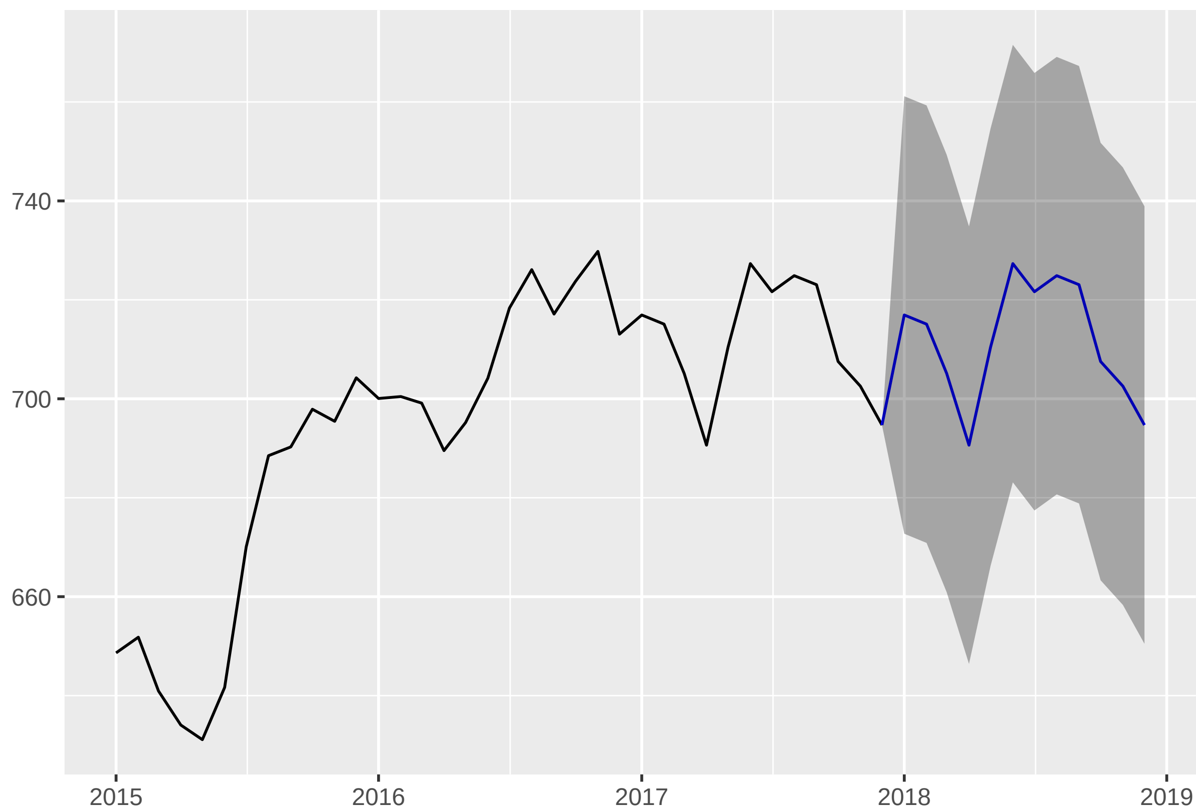

```
fit_meanf <- meanf(ts_stock_data_training, h = 12)
autoplot(fit_meanf)
```




```
fit_naive <- naive(ts_stock_data_training, h = 12)
autoplot(fit_naive)
```




```
fit_snaive <- snaive(ts_stock_data_training, h = 12)
autoplot(fit_snaive)
```




```r
print("Arima Model Accuracy")
```

```
## [1] "Arima Model Accuracy"
```

```r
round(accuracy(pred_val, ts_stock_data_testing),3)
```

```
##                    ME   RMSE    MAE    MPE  MAPE  MASE   ACF1 Theil's U
## Training set   -0.024  9.824  8.153 -0.005 1.170 0.291 -0.042        NA
## Test set      -33.513 40.536 33.513 -5.174 5.174 1.198  0.797     3.403
```

```r
print("Exponential Smoothing Forecast Accuracy")
```

```
## [1] "Exponential Smoothing Forecast Accuracy"
```

```r
round(accuracy(fit_ets, ts_stock_data_testing),3)
```

```
##                    ME   RMSE    MAE    MPE  MAPE  MASE  ACF1 Theil's U
## Training set    1.279 10.399  8.288  0.179 1.192 0.296 0.253        NA
## Test set      -28.468 38.885 30.719 -4.438 4.758 1.098 0.802     3.232
```

```r
print("Mean Forecast Accuracy")
```

```
## [1] "Mean Forecast Accuracy"
```

```r
round(accuracy(fit_meanf, ts_stock_data_testing),3)
```

```
##                    ME   RMSE    MAE    MPE  MAPE  MASE  ACF1 Theil's U
## Training set    0.000 27.777 21.088 -0.167 3.100 0.754 0.890        NA
## Test set      -29.647 39.756 31.378 -4.615 4.861 1.122 0.802     3.303
```

```r
print("Naive Forecast Accuracy")
```

```
## [1] "Naive Forecast Accuracy"
```

```r
round(accuracy(fit_naive, ts_stock_data_testing),3)
```

```
##                   ME   RMSE    MAE    MPE  MAPE  MASE  ACF1 Theil's U
## Training set   1.316 10.546  8.524  0.184 1.226 0.305 0.253        NA
## Test set     -28.467 38.884 30.719 -4.437 4.758 1.098 0.802     3.232
```

```r
print("Seasonal Naive Forecasr Accuracy")
```

```
## [1] "Seasonal Naive Forecasr Accuracy"
```

```r
round(accuracy(fit_snaive, ts_stock_data_testing),3)
```

```
##                   ME   RMSE    MAE    MPE  MAPE  MASE  ACF1 Theil's U
## Training set  22.718 34.503 27.972  3.206 3.955 1.000 0.832        NA
## Test set     -45.437 53.937 47.028 -6.989 7.215 1.681 0.684     4.427
```