Time Series Analysis & Forecasting of Bitcoin Price Prediction

```
library(fpp)

## Loading required package: forecast

## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo

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

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

## Loading required package: fma

## Loading required package: expsmooth

## Loading required package: lmtest

## Loading required package: zoo

##
## Attaching package: 'zoo'

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

## Loading required package: tseries

library(fpp2)

## Loading required package: ggplot2

##
## Attaching package: 'fpp2'

## The following objects are masked from 'package:fpp':
##
##     ausair, ausbeer, austa, austourists, debitcards, departures,
##     elecequip, euretail, guinearice, oil, sunspotarea, usmelec

library(ggplot2)
library(quantmod)
```

```
## Loading required package: xts

## Loading required package: TTR

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

#Setting the start date and end date
from_date <- as.Date("2015-01-04")
from_date

## [1] "2015-01-04"

to_date <- as.Date("2019-01-11")
to_date

## [1] "2019-01-11"
```

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
#Using lapply function
lapply(from_date, class)

## [[1]]
## [1] "Date"

lapply(to_date, class)

## [[1]]
## [1] "Date"
```

Including Plots

You can also embed plots, for example:

```
#Web crawling from Yahoo finance
getSymbols("BTC-USD", src = "yahoo", from = from_date, to = to_date)

## '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.

## [1] "BTC-USD"

View(`BTC-USD`)
head(`BTC-USD`)

##            BTC-USD.Open BTC-USD.High BTC-USD.Low BTC-USD.Close BTC-
USD.Volume
## 2015-01-04      281.146      287.230     257.612       264.195
55629100
## 2015-01-05      265.084      278.341     265.084       274.474
43962800
## 2015-01-06      274.611      287.553     272.696       286.189
23245700
## 2015-01-07      286.077      298.754     283.079       294.337
24866800
## 2015-01-08      294.135      294.135     282.175       283.349
19982500
## 2015-01-09      282.383      291.114     280.533       290.408
18718600
##            BTC-USD.Adjusted
## 2015-01-04          264.195
## 2015-01-05          274.474
## 2015-01-06          286.189
## 2015-01-07          294.337
## 2015-01-08          283.349
## 2015-01-09          290.408

summary(`BTC-USD`)

##      Index              BTC-USD.Open      BTC-USD.High      BTC-USD.Low
##  Min.   :2015-01-04   Min.   :  176.9   Min.   :  211.7   Min.   :  171.5
##  1st Qu.:2016-01-06   1st Qu.:  403.7   1st Qu.:  411.9   1st Qu.:  391.8
##  Median :2017-01-07   Median :  903.5   Median :  919.3   Median :  887.0
##  Mean   :2017-01-07   Mean   : 3112.7   Mean   : 3206.2   Mean   : 3007.5
##  3rd Qu.:2018-01-09   3rd Qu.: 6235.0   3rd Qu.: 6349.2   3rd Qu.: 6103.3
##  Max.   :2019-01-11   Max.   :19475.8   Max.   :20089.0   Max.   :18974.1
##  BTC-USD.Close     BTC-USD.Volume      BTC-USD.Adjusted
##  Min.   :  178.1   Min.   :1.060e+07   Min.   :  178.1
##  1st Qu.:  407.2   1st Qu.:5.071e+07   1st Qu.:  407.2
##  Median :  907.6   Median :1.534e+08   Median :  907.6
##  Mean   : 3114.7   Mean   :2.168e+09   Mean   : 3114.7
##  3rd Qu.: 6228.8   3rd Qu.:3.889e+09   3rd Qu.: 6228.8
##  Max.   :19497.4   Max.   :2.384e+10   Max.   :19497.4

#Converting to timeseries data
data_ts <- ts(`BTC-USD`,start=c(2015,1),end=c(2019,01), frequency = 12)
data_ts
```

```
##          BTC-USD.Open BTC-USD.High BTC-USD.Low BTC-USD.Close BTC-
USD.Volume
## Jan 2015      281.146      287.230     257.612       264.195
55629100
## Feb 2015      265.084      278.341     265.084       274.474
43962800
## Mar 2015      274.611      287.553     272.696       286.189
23245700
## Apr 2015      286.077      298.754     283.079       294.337
24866800
## May 2015      294.135      294.135     282.175       283.349
19982500
## Jun 2015      282.383      291.114     280.533       290.408
18718600
## Jul 2015      287.303      288.127     273.966       274.796
15264300
## Aug 2015      274.608      279.638     265.039       265.660
18200800
## Sep 2015      266.146      272.203     265.200       267.796
18880300
## Oct 2015      267.394      268.277     219.906       225.861
72843904
## Nov 2015      223.894      223.894     171.510       178.103
97638704
## Dec 2015      176.897      229.067     176.897       209.844
81773504
## Jan 2016      209.070      221.591     199.771       208.097
38421000
## Feb 2016      207.834      211.731     194.875       199.260
23469700
## Mar 2016      200.050      218.695     194.506       210.339
30085100
## Apr 2016      211.471      216.728     207.318       214.861
18658300
## May 2016      212.907      215.241     205.153       211.315
24051100
## Jun 2016      211.378      227.788     211.212       226.897
29924600
## Jul 2016      227.322      237.019     226.434       233.406
33544600
## Aug 2016      233.517      234.845     225.196       232.879
24621700
## Sep 2016      232.700      248.210     230.022       247.847
24782500
## Oct 2016      247.352      255.074     243.890       253.718
33582700
## Nov 2016      254.079      309.384     254.079       273.473
106794000
## Dec 2016      273.167      275.480     250.653       263.475
44399000
```

```
## Jan 2017       263.351      266.535      227.046       233.915
44352200
## Feb 2017       233.348      238.706      220.712       233.513
32213400
## Mar 2017       232.772      242.851      225.839       226.425
26605200
## Apr 2017       226.441      233.504      216.309       217.464
23348200
## May 2017       216.867      231.574      212.015       226.972
29128500
## Jun 2017       226.491      242.175      222.659       238.229
30612100
## Jul 2017       237.454      245.957      224.483       227.268
40783700
## Aug 2017       227.511      230.058      221.113       226.853
26594300
## Sep 2017       227.665      239.405      214.725       217.111
22516400
## Oct 2017       216.923      230.510      216.232       222.266
24435300
## Nov 2017       222.633      230.299      222.607       227.754
21604200
## Dec 2017       227.693      229.438      221.077       223.412
17145200
## Jan 2018       223.389      223.977      217.019       220.110
27791300
## Feb 2018       220.282      221.807      215.332       219.839
21115100
## Mar 2018       219.732      223.406      218.074       219.185
17201900
## Apr 2018       219.208      222.199      217.614       221.764
15206200
## May 2018       221.969      240.259      221.262       235.427
42744400
## Jun 2018       235.528      259.808      235.528       257.321
49732500
## Jul 2018       257.507      265.611      227.684       234.825
56552400
## Aug 2018       234.825      239.521      229.022       233.843
28153700
## Sep 2018       233.422      245.775      232.314       243.610
27363100
## Oct 2018       243.780      244.251      232.340       236.326
25200800
## Nov 2018       236.410      242.672      235.592       240.283
18270500
## Dec 2018       240.251      247.101      239.299       243.779
23876700
## Jan 2019       243.752      255.320      243.184       244.534
12284200
```

```
##          BTC-USD.Adjusted
## Jan 2015          264.195
## Feb 2015          274.474
## Mar 2015          286.189
## Apr 2015          294.337
## May 2015          283.349
## Jun 2015          290.408
## Jul 2015          274.796
## Aug 2015          265.660
## Sep 2015          267.796
## Oct 2015          225.861
## Nov 2015          178.103
## Dec 2015          209.844
## Jan 2016          208.097
## Feb 2016          199.260
## Mar 2016          210.339
## Apr 2016          214.861
## May 2016          211.315
## Jun 2016          226.897
## Jul 2016          233.406
## Aug 2016          232.879
## Sep 2016          247.847
## Oct 2016          253.718
## Nov 2016          273.473
## Dec 2016          263.475
## Jan 2017          233.915
## Feb 2017          233.513
## Mar 2017          226.425
## Apr 2017          217.464
## May 2017          226.972
## Jun 2017          238.229
## Jul 2017          227.268
## Aug 2017          226.853
## Sep 2017          217.111
## Oct 2017          222.266
## Nov 2017          227.754
## Dec 2017          223.412
## Jan 2018          220.110
## Feb 2018          219.839
## Mar 2018          219.185
## Apr 2018          221.764
## May 2018          235.427
## Jun 2018          257.321
## Jul 2018          234.825
## Aug 2018          233.843
## Sep 2018          243.610
## Oct 2018          236.326
## Nov 2018          240.283
## Dec 2018          243.779
## Jan 2019          244.534
```

```
## attr(,".indexCLASS")
## [1] Date
## attr(,"tclass")
## [1] Date
## attr(,".indexTZ")
## [1] UTC
## attr(,"tzone")
## [1] UTC
## attr(,"src")
## [1] yahoo
## attr(,"updated")
## [1] 2020-04-24 19:06:26 EDT
## attr(,"index")
##  [1] 1420329600 1420416000 1420502400 1420588800 1420675200 1420761600
##  [7] 1420848000 1420934400 1421020800 1421107200 1421193600 1421280000
## [13] 1421366400 1421452800 1421539200 1421625600 1421712000 1421798400
## [19] 1421884800 1421971200 1422057600 1422144000 1422230400 1422316800
## [25] 1422403200 1422489600 1422576000 1422662400 1422748800 1422835200
## [31] 1422921600 1423008000 1423094400 1423180800 1423267200 1423353600
## [37] 1423440000 1423526400 1423612800 1423699200 1423785600 1423872000
## [43] 1423958400 1424044800 1424131200 1424217600 1424304000 1424390400
## [49] 1424476800
## attr(,"index")attr(,"tzone")
## [1] UTC
## attr(,"index")attr(,"tclass")
## [1] Date
```

```r
#Selecting only BTC-USD.Open column for further analysis
data=data_ts[,1]
data
```

```
##          Jan     Feb     Mar     Apr     May     Jun     Jul     Aug
Sep
## 2015 281.146 265.084 274.611 286.077 294.135 282.383 287.303 274.608
266.146
## 2016 209.070 207.834 200.050 211.471 212.907 211.378 227.322 233.517
232.700
## 2017 263.351 233.348 232.772 226.441 216.867 226.491 237.454 227.511
227.665
## 2018 223.389 220.282 219.732 219.208 221.969 235.528 257.507 234.825
233.422
## 2019 243.752
##          Oct     Nov     Dec
## 2015 267.394 223.894 176.897
## 2016 247.352 254.079 273.167
## 2017 216.923 222.633 227.693
## 2018 243.780 236.410 240.251
## 2019
```

```r
View(data)
```

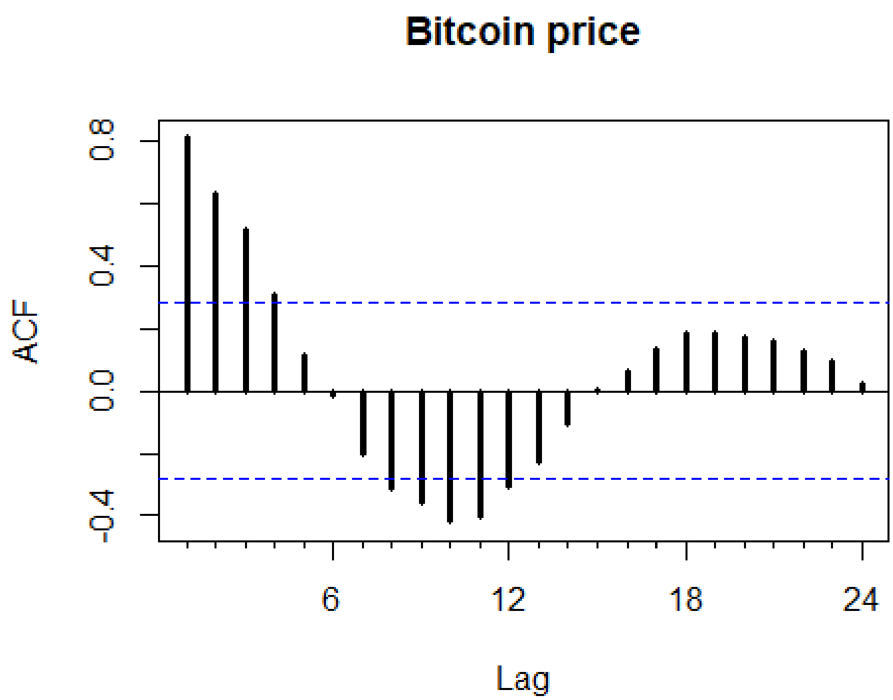
```r
#Calculating training data
training_data = window(data,start=c(2015,1), end=c(2018,1))
training_data
```

```
##          Jan     Feb     Mar     Apr     May     Jun     Jul     Aug
Sep
## 2015 281.146 265.084 274.611 286.077 294.135 282.383 287.303 274.608
266.146
## 2016 209.070 207.834 200.050 211.471 212.907 211.378 227.322 233.517
232.700
## 2017 263.351 233.348 232.772 226.441 216.867 226.491 237.454 227.511
227.665
## 2018 223.389
##          Oct     Nov     Dec
## 2015 267.394 223.894 176.897
## 2016 247.352 254.079 273.167
## 2017 216.923 222.633 227.693
## 2018
```

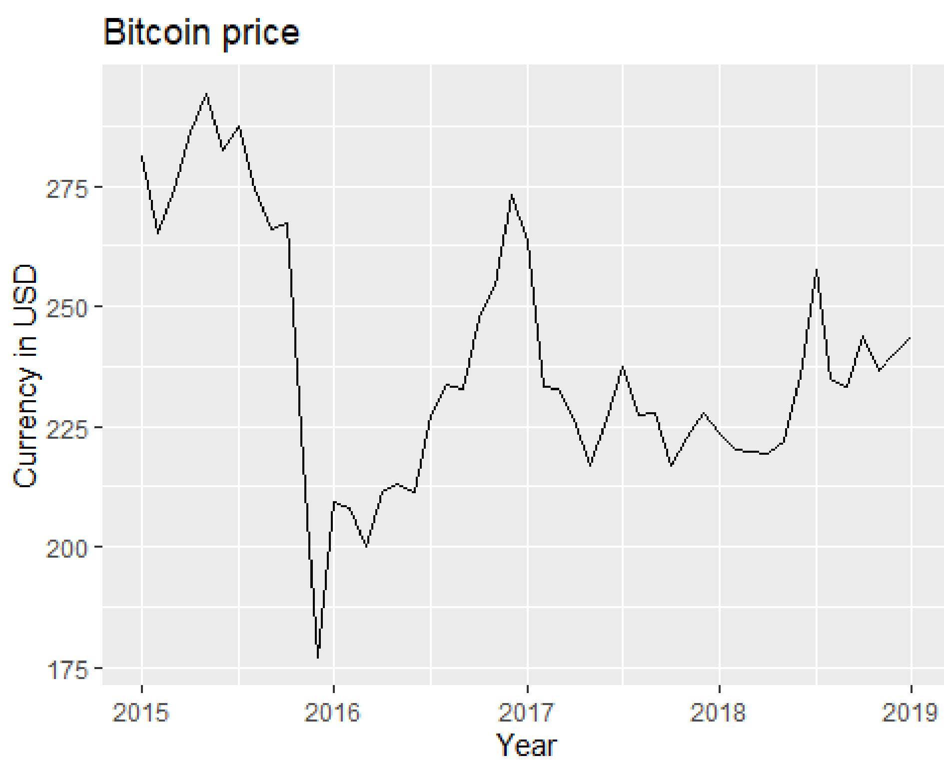
```r
#Calculating testing data
testing_data = window(data,start=c(2018,1), end=c(2019,1))
testing_data
```

```
##          Jan     Feb     Mar     Apr     May     Jun     Jul     Aug
Sep
## 2018 223.389 220.282 219.732 219.208 221.969 235.528 257.507 234.825
233.422
## 2019 243.752
##          Oct     Nov     Dec
## 2018 243.780 236.410 240.251
## 2019
```

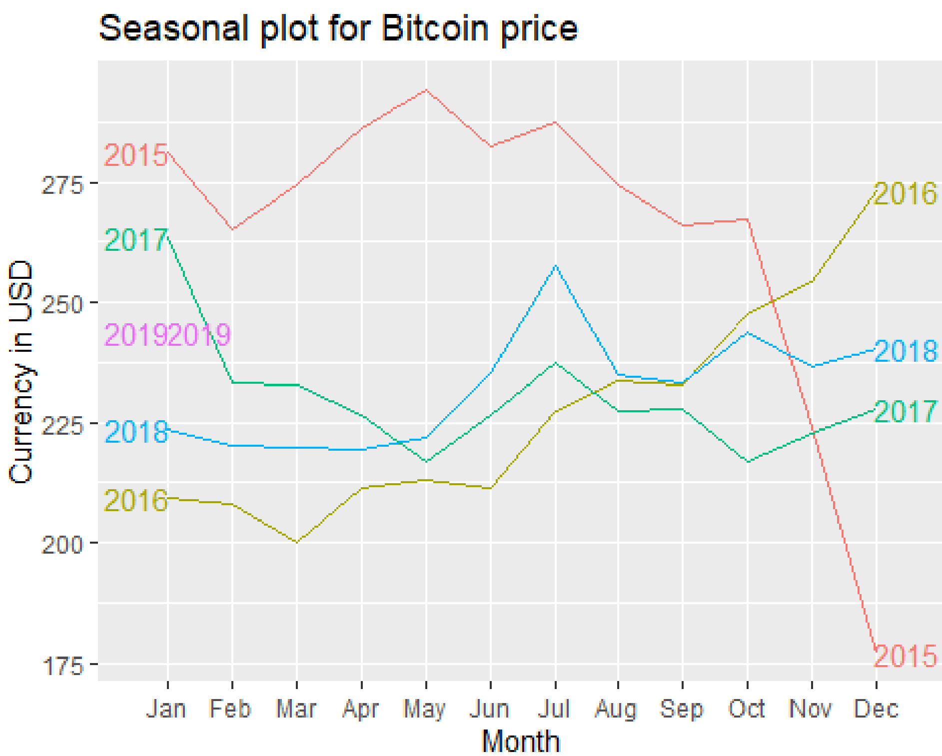
```r
#Autocorelation Function
#Here, lwd indicates the width of the lines.
Acf(data, lwd=3,main="Bitcoin price")
```



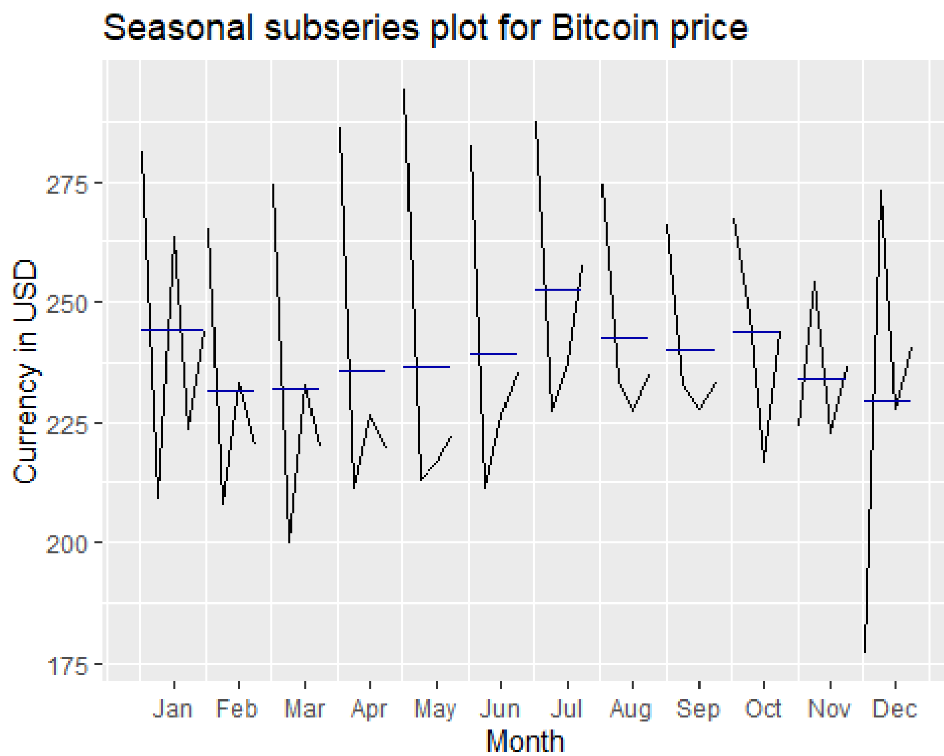

```
#DataPlot
autoplot(data) + ggtitle("Bitcoin price") + xlab("Year") + ylab("Currency in
USD")
```



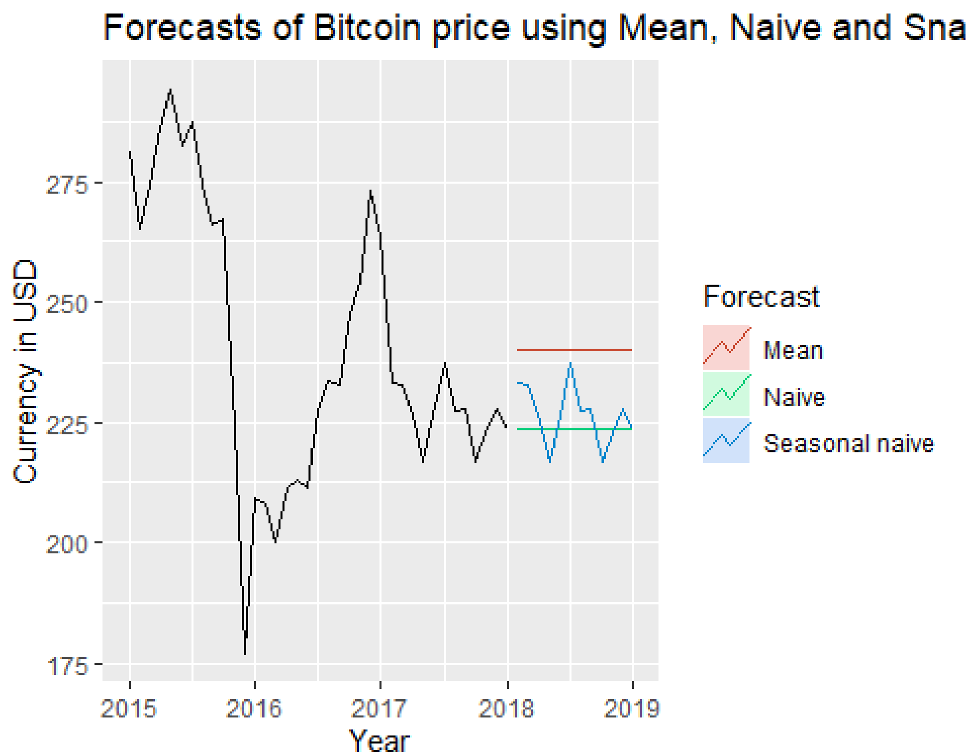
```
#Seasonal plot
ggseasonplot(data, year.labels=TRUE, year.labels.left=TRUE) +
ggtitle("Seasonal plot for Bitcoin price") + ylab("Currency in USD")
```




```
#Seasonal subseries plot
ggsubseriesplot(data) + ggtitle("Seasonal subseries plot for Bitcoin price")
+ ylab("Currency in USD")
```




```r
#Ploting Mean, Naive, and Seasonal Naive
mean_fit <- meanf(training_data,h=12)
naive_fit <- naive(training_data,h=12)
snaive_fit <- snaive(training_data,h=12)
autoplot(training_data) +
  autolayer(meanf(training_data, h=12),
            series="Mean", PI=FALSE) +
  autolayer(naive(training_data, h=12),
            series="Naive", PI=FALSE) +
  autolayer(snaive(training_data, h=12),
            series="Seasonal naive", PI=FALSE) +
  ggtitle("Forecasts of Bitcoin price using Mean, Naive and Snaive") +
  xlab("Year") + ylab("Currency in USD") +
  guides(colour=guide_legend(title="Forecast"))
```

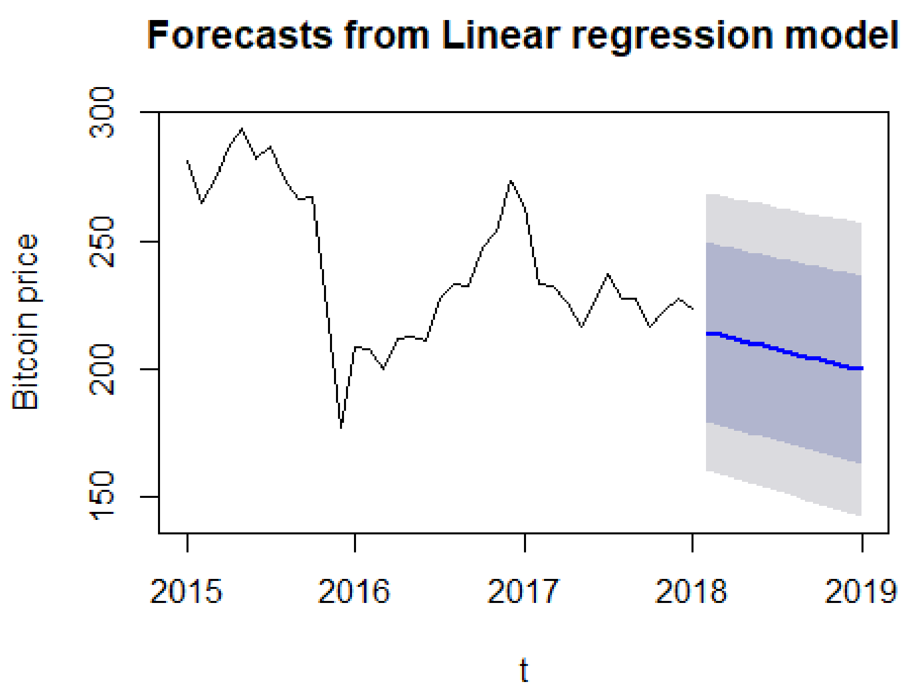



```r
#Performing linear regression
lreg <- tslm(training_data ~ trend)
tslm_fit=forecast(lreg, h=12)
summary(tslm_fit)
```

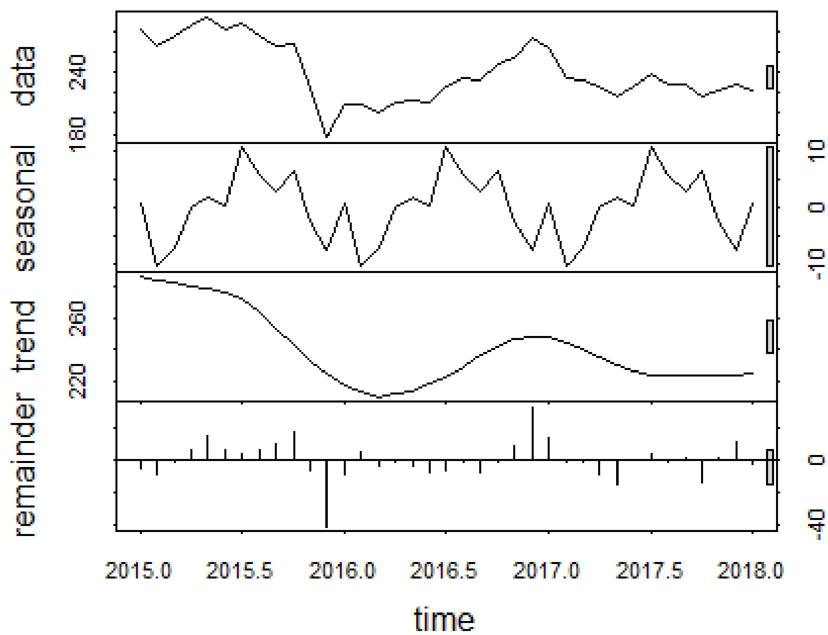
```
##
## Forecast method: Linear regression model
##
## Model Information:
##
## Call:
## tslm(formula = training_data ~ trend)
##
## Coefficients:
## (Intercept)        trend
##     265.637       -1.345
##
##
## Error measures:
##                        ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 3.073121e-15 24.67793 19.05638 -1.115835 8.253339 0.4639818
##                   ACF1
## Training set 0.7986267
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Feb 2018       214.5289 179.5697 249.4880 160.1955 268.8622
```

```
## Mar 2018       213.1839 178.0798 248.2880 158.6252 267.7426
## Apr 2018       211.8390 176.5831 247.0948 157.0444 266.6335
## May 2018       210.4940 175.0797 245.9084 155.4532 265.5348
## Jun 2018       209.1491 173.5697 244.7285 153.8517 264.4464
## Jul 2018       207.8041 172.0531 243.5551 152.2401 263.3682
## Aug 2018       206.4592 170.5302 242.3882 150.6185 262.2999
## Sep 2018       205.1142 169.0009 241.2276 148.9870 261.2415
## Oct 2018       203.7693 167.4653 240.0733 147.3459 260.1927
## Nov 2018       202.4244 165.9237 238.9251 145.6952 259.1536
## Dec 2018       201.0794 164.3759 237.7829 144.0351 258.1238
## Jan 2019       199.7345 162.8223 236.6467 142.3657 257.1032
```

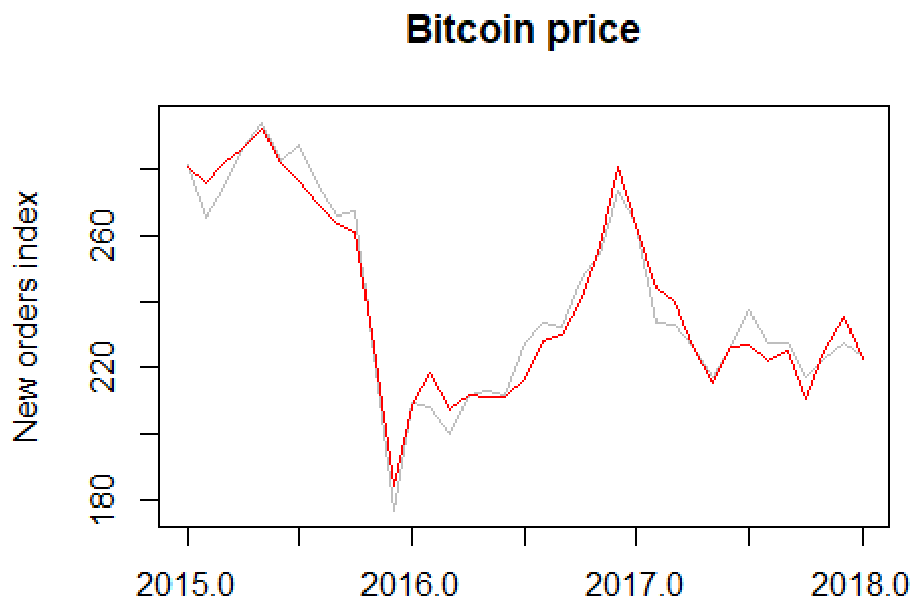
```
plot(tslm_fit, ylab="Bitcoin price",
     xlab="t")
```



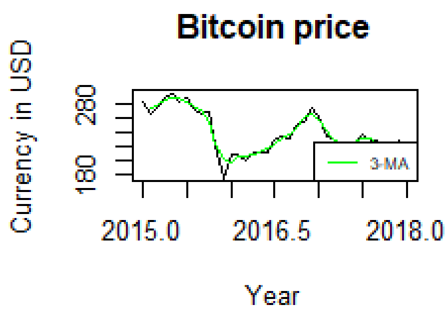

```
#Performing STL decomposition
stl_decomp <- stl(training_data, t.window=12, s.window="periodic")
plot(stl_decomp)
```

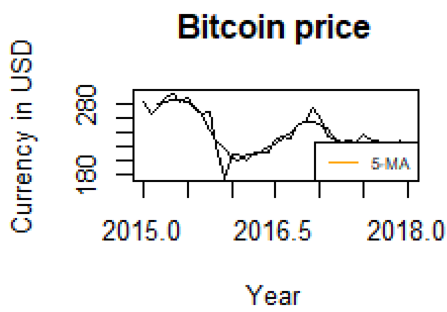


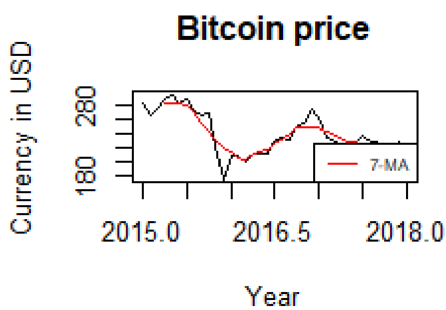

```r
  #Demonstrating seasonally adjusted data
plot(training_data, col="grey",
     main="Bitcoin price",
     xlab="", ylab="New orders index")
lines(seasadj(stl_decomp),col="red",ylab="Seasonally adjusted")
```

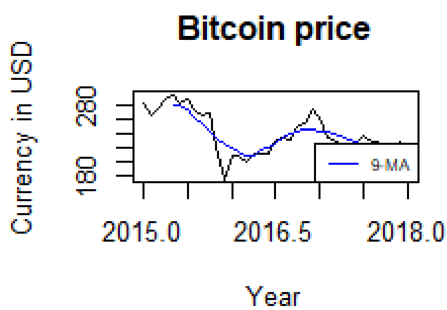



```r
#Performing Moving Average
par(mfrow=c(2,2))

plot(training_data, main="Bitcoin price",
     ylab="Currency in USD", xlab="Year")
lines(ma(training_data,3),col="green")
legend("bottomright",lty=1,col="green",cex=0.6,legend=c("3-MA"))
plot(training_data, main="Bitcoin price",
     ylab="Currency in USD", xlab="Year")
lines(ma(training_data,5),orange="orange")

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "orange" is not a
## graphical parameter

legend("bottomright",lty=1,col="orange",cex=0.6,legend=c("5-MA"))
plot(training_data, main="Bitcoin price",
     ylab="Currency in USD", xlab="Year")
lines(ma(training_data,7),col="red")
legend("bottomright",lty=1,col="red",cex=0.6,legend=c("7-MA"))
plot(training_data, main="Bitcoin price",
     ylab="Currency in USD", xlab="Year")
lines(ma(training_data,9),col="blue")
legend("bottomright",lty=1,col="blue",cex=0.6,legend=c("9-MA"))
```



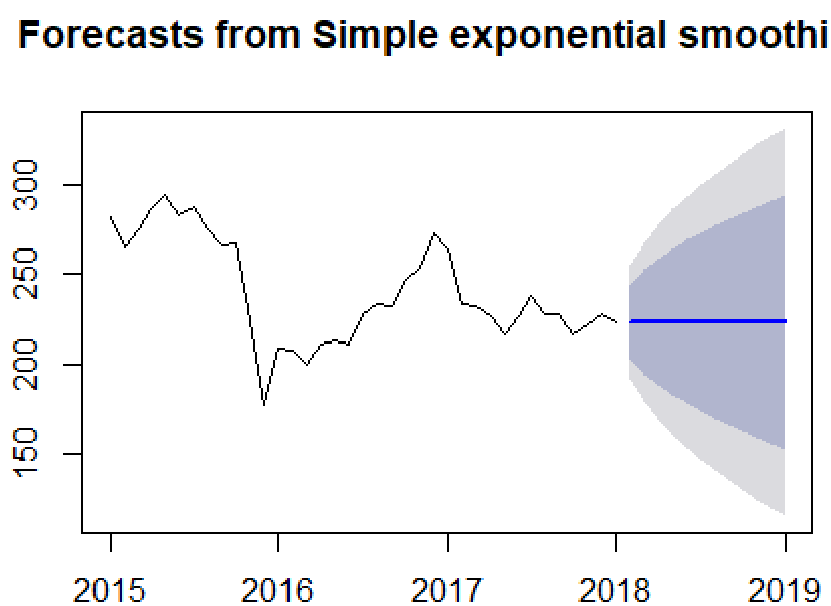






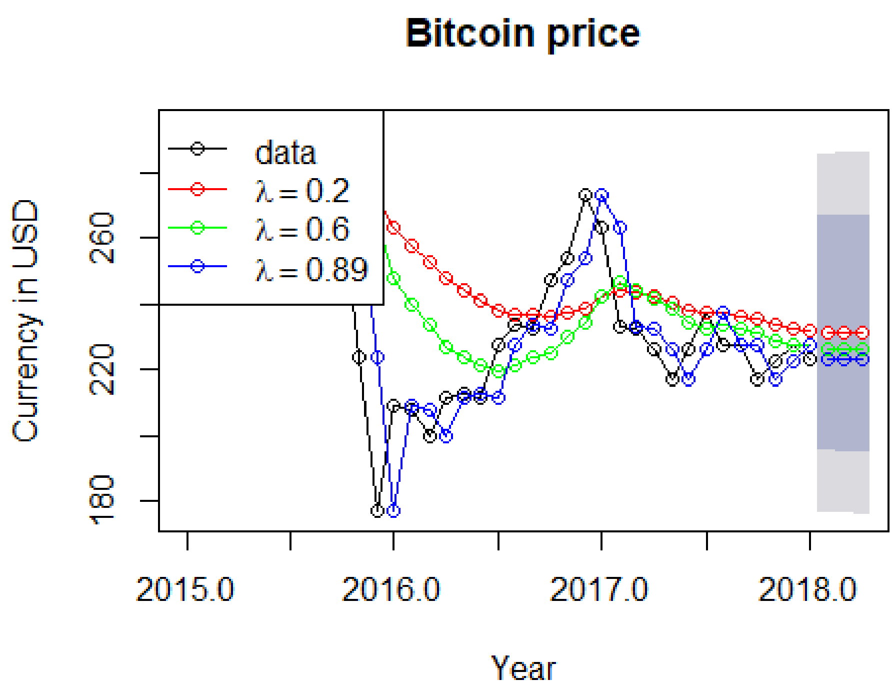

```r
#Simple Exponential Smoothing technique
ses_fit <- ses(training_data, h = 12)
ses_fit <- forecast(ses_fit)
summary(ses_fit)

##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = training_data, h = 12)
##
##   Smoothing parameters:
##     alpha = 0.9999
##
##   Initial states:
##     l = 281.1389
##
##   sigma:  15.8789
##
##      AIC     AICc      BIC
## 342.1574 342.8846 346.9901
##
## Error measures:
##                        ME      RMSE       MAE         MPE      MAPE       MASE
```

```
## Training set -1.560952 15.44381 10.98632 -0.8666531 4.802155 0.2674931
##                      ACF1
## Training set 0.06163641
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Feb 2018       223.3894 203.0398 243.7391 192.2673 254.5116
## Mar 2018       223.3894 194.6121 252.1668 179.3783 267.4006
## Apr 2018       223.3894 188.1451 258.6337 169.4879 277.2910
## May 2018       223.3894 182.6932 264.0857 161.1498 285.6290
## Jun 2018       223.3894 177.8898 268.8890 153.8038 292.9751
## Jul 2018       223.3894 173.5473 273.2316 147.1625 299.6164
## Aug 2018       223.3894 169.5539 277.2250 141.0551 305.7238
## Sep 2018       223.3894 165.8369 280.9420 135.3705 311.4084
## Oct 2018       223.3894 162.3459 284.4330 130.0313 316.7475
## Nov 2018       223.3894 159.0439 287.7349 124.9815 321.7974
## Dec 2018       223.3894 155.9034 290.8755 120.1784 326.6005
## Jan 2019       223.3894 152.9026 293.8763 115.5891 331.1898

plot(ses_fit)
```




```
fit1 <-ses(training_data, alpha=0.1, initial="simple", h=3)
fit2 <-ses(training_data, alpha=0.2, initial="simple", h=3)
fit3 <-ses(training_data, h=3)
plot(fit1,main="Bitcoin price", ylab="Currency in USD", xlab="Year",
fcol="white", type="o")
lines(fitted(fit1), col="red", type="o")
```

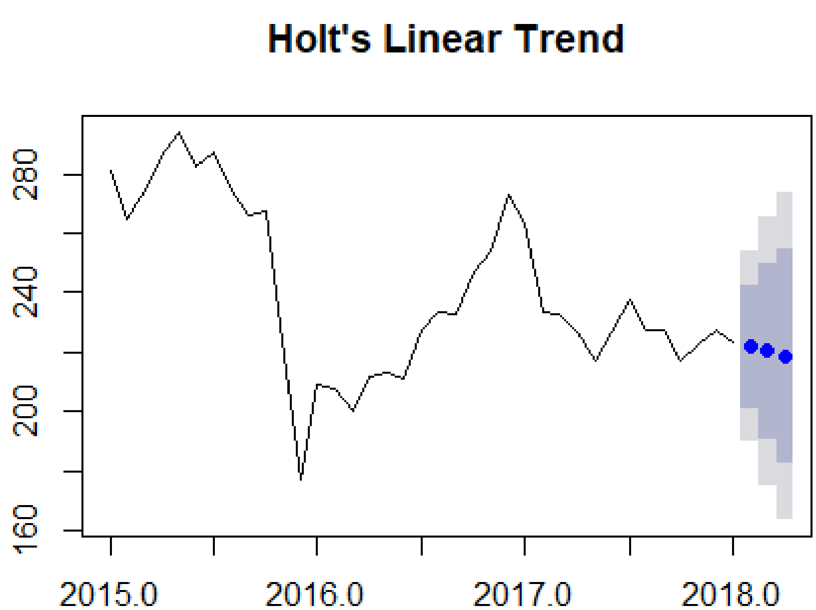
```
lines(fitted(fit2), col="green", type="o")
lines(fitted(fit3), col="blue", type="o")
lines(fit1$mean, col="red", type="o")
lines(fit2$mean, col="green", type="o")
lines(fit3$mean, col="blue", type="o")
legend("topleft",lty=1, col=c(1,"red","green","blue"),
       c("data", expression(lambda == 0.2), expression(lambda == 0.6),
         expression(lambda == 0.89)),pch=1)
```



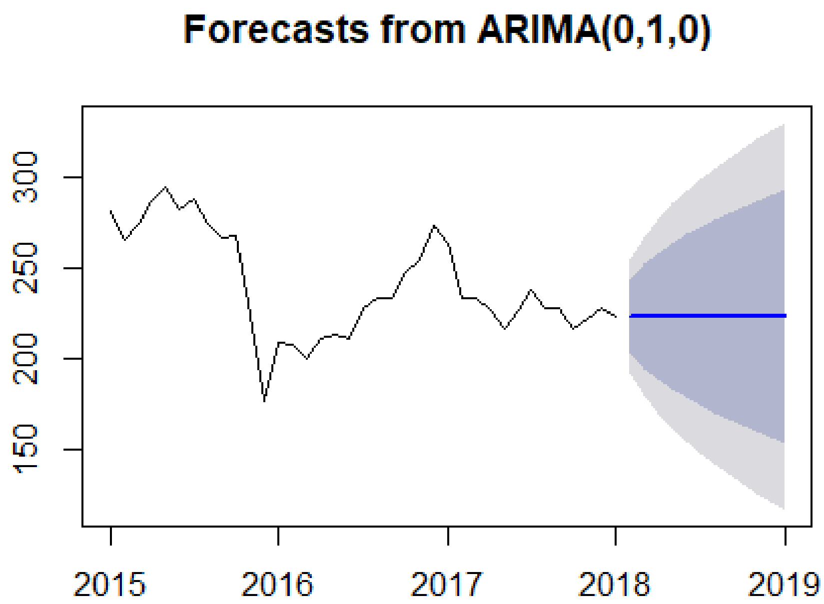

```
#Holt's Linear trend
#This method has two smoothing techniques.
#SES is not reliable here.
hlin_fit <- holt(training_data, h=3)
hlin_fit <- forecast(hlin_fit)
summary(hlin_fit)

##
## Forecast method: Holt's method
##
## Model Information:
## Holt's method
##
## Call:
##  holt(y = training_data, h = 3)
##
##   Smoothing parameters:
##     alpha = 0.9999
```

```
##     beta  = 1e-04
##
##   Initial states:
##     l = 282.8126
##     b = -1.5514
##
##   sigma:  16.2679
##
##      AIC     AICc      BIC
## 345.7709 347.7064 353.8255
##
## Error measures:
##                       ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.05330743 15.36337 11.10116 -0.2266501 4.833905 0.2702894
##                   ACF1
## Training set 0.0644396
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Feb 2018       221.8377 200.9896 242.6858 189.9533 253.7221
## Mar 2018       220.2861 190.8024 249.7697 175.1947 265.3774
## Apr 2018       218.7344 182.6233 254.8456 163.5072 273.9617
```

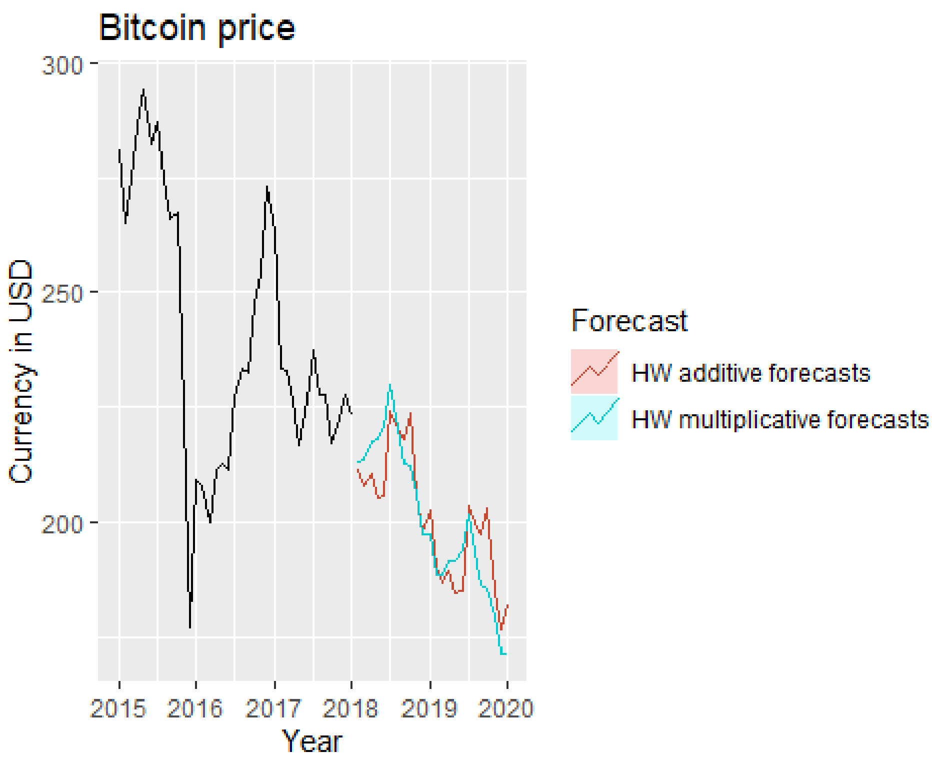
```r
plot(hlin_fit, main = "Holt's Linear Trend")
lines(training_data)
```



```
#Performing Auto Arima
#Using auto.arima() to predict values automatially
acc.arima <- auto.arima(training_data)
arima_fit <- forecast(acc.arima, h=12)
summary(arima_fit)

##
## Forecast method: ARIMA(0,1,0)
##
## Model Information:
## Series: training_data
## ARIMA(0,1,0)
##
## sigma^2 estimated as 245.1:  log likelihood=-150.11
## AIC=302.23   AICc=302.35   BIC=303.81
##
## Error measures:
##                     ME     RMSE      MAE        MPE    MAPE     MASE
ACF1
## Training set -1.553401 15.44377 10.99379 -0.8639343 4.80481 0.267675
0.06109573
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Feb 2018        223.389 203.3240 243.4540 192.7023 254.0758
## Mar 2018        223.389 195.0128 251.7652 179.9914 266.7866
## Apr 2018        223.389 188.6354 258.1426 170.2380 276.5400
## May 2018        223.389 183.2590 263.5190 162.0155 284.7625
## Jun 2018        223.389 178.5223 268.2557 154.7713 292.0067
## Jul 2018        223.389 174.2400 272.5380 148.2221 298.5559
## Aug 2018        223.389 170.3020 276.4760 142.1995 304.5785
## Sep 2018        223.389 166.6366 280.1414 136.5938 310.1843
## Oct 2018        223.389 163.1940 283.5840 131.3287 315.4493
## Nov 2018        223.389 159.9379 286.8401 126.3490 320.4291
## Dec 2018        223.389 156.8410 289.9371 121.6126 325.1655
## Jan 2019        223.389 153.8818 292.8962 117.0870 329.6910

plot(arima_fit)
```




```r
#Holt's Winter Additive and Multiplicative technique
#These two variations differ in nature of the seasnal component
add_1 <- hw(training_data,seasonal="additive")
add_1 <- forecast(add_1)
mul_2 <- hw(training_data,seasonal="multiplicative")
mul_2 <- forecast(mul_2)
autoplot(training_data) +
  autolayer(add_1, series="HW additive forecasts", PI=FALSE) +
  autolayer(mul_2, series="HW multiplicative forecasts", PI=FALSE) +
  ggtitle("Bitcoin price") +
  xlab("Year") +
  ylab("Currency in USD") +
  guides(colour=guide_legend(title="Forecast"))
```




```
#Comparison between the models developed
acc.mean=accuracy(mean_fit,testing_data)
acc.naive=accuracy(naive_fit,testing_data)
acc.snaive=accuracy(snaive_fit,testing_data)
acc.linear=accuracy(tslm_fit,testing_data)
acc.ses=accuracy(ses_fit, testing_data)
acc.holt=accuracy(hlin_fit, testing_data)
acc.multi=accuracy(add_1, testing_data)
acc.add=accuracy(mul_2, testing_data)
acc.arima=accuracy(arima_fit, testing_data)


acc.table<-rbind(acc.mean, acc.naive, acc.snaive, acc.linear, acc.ses,
acc.holt, acc.add, acc.multi, acc.arima)
acc.table

##                         ME       RMSE        MAE         MPE       MAPE
## Training set  7.678730e-15 28.5519148 24.6311915 -1.42169812 10.3110861
## Test set     -6.193954e+00 12.9328447 10.3537629 -2.88794597  4.5310197
## Training set -1.604361e+00 15.6567258 11.2913597 -0.89071025  4.9354991
## Test set      1.049982e+01 15.4641422 12.5606650  4.26620461  5.2031777
## Training set -1.841964e+01 48.6222867 41.0714001 -9.26140034 18.2295610
## Test set      7.289915e+00 14.3555908 12.8464177  2.89697480  5.4245755
## Training set  3.073121e-15 24.6779293 19.0563801 -1.11583483  8.2533390
## Test set      2.675717e+01 30.6348944 26.7571719 11.16520606 11.1652061
## Training set -1.560952e+00 15.4438085 10.9863162 -0.86665314  4.8021550
## Test set      1.049939e+01 15.4638497 12.5605214  4.26602003  5.2031235
```

```
## Training set -5.330743e-02 15.3633705 11.1011628 -0.22665006  4.8339046
## Test set     -5.453956e-01  0.9918585  0.8610944 -0.24744851  0.3914664
## Training set  1.059059e+00 16.2645585 12.6171094  0.23430291  5.4376718
## Test set      2.056815e+01 25.2082828 20.5681485  8.57569636  8.5756964
## Training set  2.064677e-01 14.3075741 11.4617054 -0.05708893  4.9335631
## Test set      2.274654e+01 25.5441214 22.7465427  9.56932662  9.5693266
## Training set -1.553401e+00 15.4437684 10.9937864 -0.86393430  4.8048099
## Test set      1.049982e+01 15.4641422 12.5606650  4.26620461  5.2031777
##                    MASE          ACF1 Theil's U
## Training set 0.59971638  8.211432e-01        NA
## Test set     0.25209179  4.625381e-01 1.1643232
## Training set 0.27492025  6.425778e-02        NA
## Test set     0.30582510  4.625381e-01 1.4679311
## Training set 1.00000000  8.421614e-01        NA
## Test set     0.31278256  5.228566e-01 1.3275514
## Training set 0.46398175  7.986267e-01        NA
## Test set     0.65147942  5.876783e-01 2.9003945
## Training set 0.26749310  6.163641e-02        NA
## Test set     0.30582160  4.625381e-01 1.4679031
## Training set 0.27028937  6.443960e-02        NA
## Test set     0.02096579 -3.648256e-05 0.9593251
## Training set 0.30719940  2.477611e-01        NA
## Test set     0.50079005  6.703702e-01 2.3756071
## Training set 0.27906780  1.489308e-01        NA
## Test set     0.55382925  5.562002e-01 2.4277667
## Training set 0.26767498  6.109573e-02        NA
## Test set     0.30582510  4.625381e-01 1.4679311
```

```
row.names(acc.table)<-c('Mean training','Mean test', 'Naive training', 'Naive
test', 'Seasonal. Naive training', 'Seasonal. Naive test' ,'Linear training',
'Linear test','ses training', 'ses test',"Holt's Linear training", "Holt's
Linear test", 'Add training', 'Add test','Multi training', 'Multi
test','ARIMA training', 'ARIMA test')

#Tabular format
#Overall comarison between our different accuracy models to determine the
best out of all
acc.table<-as.data.frame(acc.table)
acc.table
```

```
##                                    ME       RMSE        MAE         MPE
## Mean training             7.678730e-15 28.5519148 24.6311915 -1.42169812
## Mean test                -6.193954e+00 12.9328447 10.3537629 -2.88794597
## Naive training           -1.604361e+00 15.6567258 11.2913597 -0.89071025
## Naive test                1.049982e+01 15.4641422 12.5606650  4.26620461
## Seasonal. Naive training -1.841964e+01 48.6222867 41.0714001 -9.26140034
## Seasonal. Naive test      7.289915e+00 14.3555908 12.8464177  2.89697480
## Linear training           3.073121e-15 24.6779293 19.0563801 -1.11583483
## Linear test               2.675717e+01 30.6348944 26.7571719 11.16520606
## ses training             -1.560952e+00 15.4438085 10.9863162 -0.86665314
```

```
## ses test                   1.049939e+01 15.4638497 12.5605214  4.26602003
## Holt's Linear training    -5.330743e-02 15.3633705 11.1011628 -0.22665006
## Holt's Linear test        -5.453956e-01  0.9918585  0.8610944 -0.24744851
## Add training               1.059059e+00 16.2645585 12.6171094  0.23430291
## Add test                   2.056815e+01 25.2082828 20.5681485  8.57569636
## Multi training             2.064677e-01 14.3075741 11.4617054 -0.05708893
## Multi test                 2.274654e+01 25.5441214 22.7465427  9.56932662
## ARIMA training            -1.553401e+00 15.4437684 10.9937864 -0.86393430
## ARIMA test                 1.049982e+01 15.4641422 12.5606650  4.26620461
##                                  MAPE       MASE          ACF1 Theil's U
## Mean training            10.3110861 0.59971638  8.211432e-01        NA
## Mean test                 4.5310197 0.25209179  4.625381e-01 1.1643232
## Naive training            4.9354991 0.27492025  6.425778e-02        NA
## Naive test                5.2031777 0.30582510  4.625381e-01 1.4679311
## Seasonal. Naive training 18.2295610 1.00000000  8.421614e-01        NA
## Seasonal. Naive test      5.4245755 0.31278256  5.228566e-01 1.3275514
## Linear training           8.2533390 0.46398175  7.986267e-01        NA
## Linear test              11.1652061 0.65147942  5.876783e-01 2.9003945
## ses training              4.8021550 0.26749310  6.163641e-02        NA
## ses test                  5.2031235 0.30582160  4.625381e-01 1.4679031
## Holt's Linear training    4.8339046 0.27028937  6.443960e-02        NA
## Holt's Linear test        0.3914664 0.02096579 -3.648256e-05 0.9593251
## Add training              5.4376718 0.30719940  2.477611e-01        NA
## Add test                  8.5756964 0.50079005  6.703702e-01 2.3756071
## Multi training            4.9335631 0.27906780  1.489308e-01        NA
## Multi test                9.5693266 0.55382925  5.562002e-01 2.4277667
## ARIMA training            4.8048099 0.26767498  6.109573e-02        NA
## ARIMA test                5.2031777 0.30582510  4.625381e-01 1.4679311
```