Time series Analysis models to predict next slowdown/recession in US Economy

EXECUTIVE SUMMARY

This Project Focuses on Developing Time Series Analysis models to predict next slowdown/recession in US Economy based on naive, Exponential smoothening, ARIMA Methods and Neural Network.

Understanding S&P 500 data, Unemployment Data, Yield Curve and House Price Index is highly-relevent in Understanding the health of US Economy. We will see Later that these features are highly correlated (Statistically).

For this project we leverage the horsepower of R-studio and deliver, where appropriate, gorgeous intractive data visualization using ggplot and plotly

Load Packages

```r
library(tidyr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date
```

```r
library(dplyr)
```

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

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

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

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

```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.5.2
```

```r
library(markdown)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.5.2

## Loading required package: lattice
```

```r
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.5.2
```

```
## corrplot 0.84 loaded
```

```r
library(plotly)
```

```
## Warning: package 'plotly' was built under R version 3.5.3
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```r
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 3.5.2
```

```r
library(tseries)
```

```
## Warning: package 'tseries' was built under R version 3.5.2
```

Loading all Data Files

We Got all our Data from "Federal Reserve Economic Data" except S&P 500 Data which we got from yahoo.com

Link to this website https://fred.stlouisfed.org/

```r
stockprice <- read.csv("SP500.csv")
unemployment <- read.csv("Unrate.csv")
yieldcurve <- read.csv("Yieldcurve.csv")
ushpi <- read.csv("Usahouseprice.csv")
```

Basic Summary Statistics:

Summary Statistics for S&P Data:

SP 500 data has min-value of 735.1 with mean of 1729.3 and median of 1559.3, It doesn't have missing values

Summary Statistics for US House price Index Data:

HPI has min value of 306.5 with mean of 352.1 and median value of 346.8, It Doesn't have missing values

Summary Statistics for US Unemployment Rate Data:

Unemployment Rate data has min-value of 3.7 with mean of 6.46 and median value of 5.9, It Doesn't have missing values

Summary Statistics for US Yield Curve Data:

Yield Curve data has min-value of -0.130 with mean of 1.518 and median of 1.55, It Doesn't have missing values

```r
summary(stockprice)
```

```
##          DATE         SP500
##  2007-01-01:  1   Min.   : 735.1
##  2007-02-01:  1   1st Qu.:1290.7
##  2007-03-01:  1   Median :1559.3
##  2007-04-01:  1   Mean   :1729.3
##  2007-05-01:  1   3rd Qu.:2100.1
##  2007-06-01:  1   Max.   :2914.0
##  (Other)   :142
```

```r
summary(ushpi)
```

```
##          DATE       USSTHPI
##  2007-01-01: 1   Min.   :306.5
##  2007-04-01: 1   1st Qu.:323.4
##  2007-07-01: 1   Median :346.8
##  2007-10-01: 1   Mean   :352.1
##  2008-01-01: 1   3rd Qu.:374.8
##  2008-04-01: 1   Max.   :432.1
##  (Other)   :42
```

```r
summary(unemployment)
```

```
##          DATE         UNRATE
##  2007-01-01:  1   Min.   : 3.700
##  2007-02-01:  1   1st Qu.: 4.700
##  2007-03-01:  1   Median : 5.900
##  2007-04-01:  1   Mean   : 6.465
##  2007-05-01:  1   3rd Qu.: 8.250
##  2007-06-01:  1   Max.   :10.000
##  (Other)   :141
```

```r
summary(yieldcurve)
```

```
##          DATE         T10Y2Y
##  2007-01-06:  1   Min.   :-0.130
```

```
##  2007-02-06:  1   1st Qu.: 0.950
##  2007-03-06:  1   Median : 1.550
##  2007-04-06:  1   Mean   : 1.518
##  2007-05-06:  1   3rd Qu.: 2.195
##  2007-06-06:  1   Max.   : 2.900
##  (Other)   :141
```

Seeing the First Few values of Data to get sense of what is there

```r
head(stockprice, 20)
```

```
##          DATE   SP500
## 1  2007-01-01 1438.24
## 2  2007-02-01 1406.82
## 3  2007-03-01 1420.86
## 4  2007-04-01 1482.37
## 5  2007-05-01 1530.62
## 6  2007-06-01 1503.35
## 7  2007-07-01 1455.27
## 8  2007-08-01 1473.99
## 9  2007-09-01 1526.75
## 10 2007-10-01 1549.38
## 11 2007-11-01 1481.14
## 12 2007-12-01 1468.36
## 13 2008-01-01 1378.55
## 14 2008-02-01 1330.63
## 15 2008-03-01 1322.70
## 16 2008-04-01 1385.59
## 17 2008-05-01 1400.38
## 18 2008-06-01 1280.00
## 19 2008-07-01 1267.38
## 20 2008-08-01 1282.83
```

```r
head(unemployment,20)
```

```
##          DATE UNRATE
## 1  2007-01-01    4.6
## 2  2007-02-01    4.5
## 3  2007-03-01    4.4
## 4  2007-04-01    4.5
## 5  2007-05-01    4.4
## 6  2007-06-01    4.6
## 7  2007-07-01    4.7
## 8  2007-08-01    4.6
## 9  2007-09-01    4.7
## 10 2007-10-01    4.7
## 11 2007-11-01    4.7
## 12 2007-12-01    5.0
## 13 2008-01-01    5.0
## 14 2008-02-01    4.9
## 15 2008-03-01    5.1
## 16 2008-04-01    5.0
## 17 2008-05-01    5.4
## 18 2008-06-01    5.6
## 19 2008-07-01    5.8
```

```
## 20 2008-08-01    6.1
```

```r
head(yieldcurve,20)
```

```
##          DATE T10Y2Y
## 1  2007-01-06  -0.11
## 2  2007-02-06  -0.13
## 3  2007-03-06  -0.05
## 4  2007-04-06   0.01
## 5  2007-05-06  -0.11
## 6  2007-06-06   0.00
## 7  2007-07-06   0.20
## 8  2007-08-06   0.26
## 9  2007-09-06   0.43
## 10 2007-10-06   0.57
## 11 2007-11-06   0.68
## 12 2007-12-06   0.99
## 13 2008-01-06   1.10
## 14 2008-02-06   1.65
## 15 2008-03-06   2.09
## 16 2008-04-06   1.62
## 17 2008-05-06   1.55
## 18 2008-06-06   1.54
## 19 2008-07-06   1.48
## 20 2008-08-06   1.50
```

```r
head(ushpi,20)
```

```
##          DATE USSTHPI
## 1  2007-01-01  378.22
## 2  2007-04-01  377.96
## 3  2007-07-01  373.73
## 4  2007-10-01  372.56
## 5  2008-01-01  369.86
## 6  2008-04-01  360.54
## 7  2008-07-01  349.20
## 8  2008-10-01  345.99
## 9  2009-01-01  348.53
## 10 2009-04-01  339.32
## 11 2009-07-01  330.35
## 12 2009-10-01  327.89
## 13 2010-01-01  323.96
## 14 2010-04-01  321.06
## 15 2010-07-01  324.10
## 16 2010-10-01  321.75
## 17 2011-01-01  312.90
## 18 2011-04-01  307.43
## 19 2011-07-01  309.73
## 20 2011-10-01  311.09
```

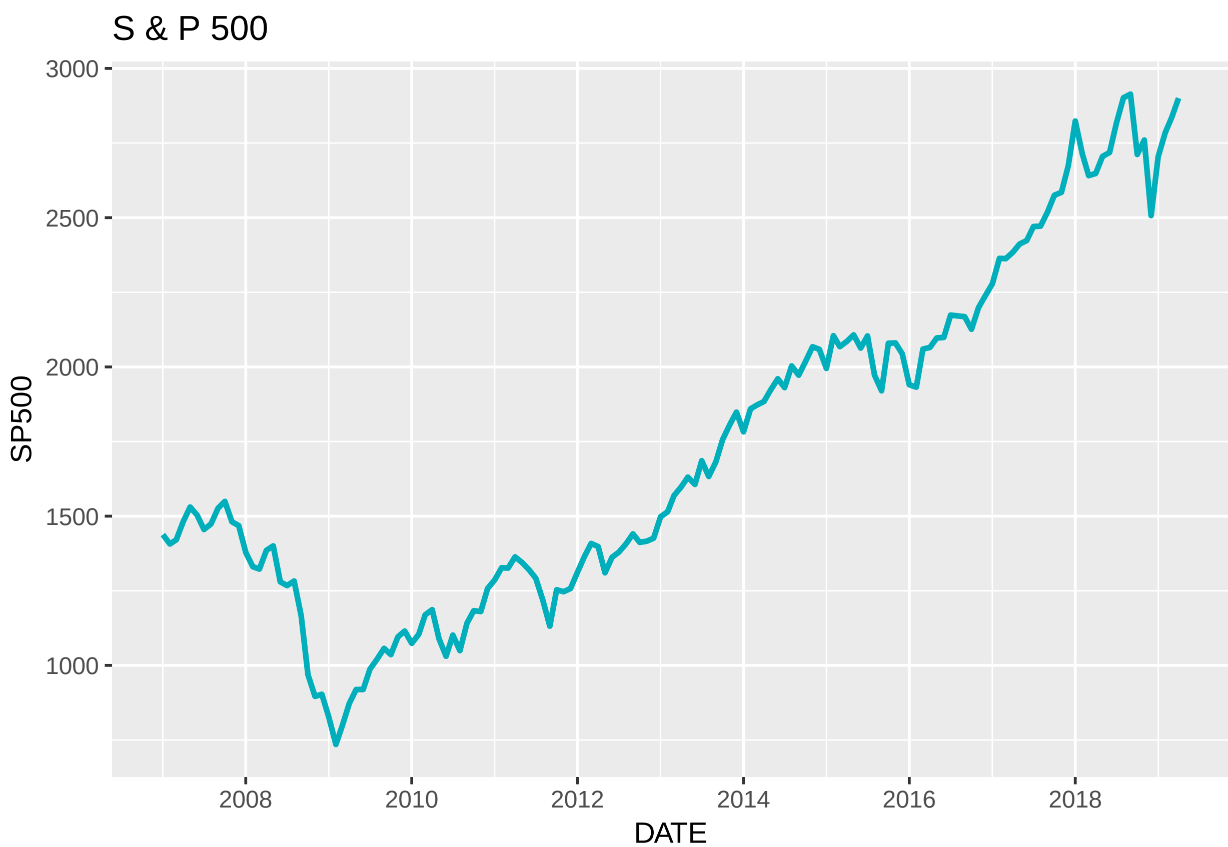
Ploting Basic Graphs Using ggplot

S&P 500

US Stock Market crashed after 2008 Recession. S&P 500 graph have trend and Random components

We see ups and downs in the data but not at regular interval of time so seasonal components might be missing.

```
stockprice$DATE <- as.Date(stockprice$DATE)
stockprice$SP500 <- as.numeric(stockprice$SP500)
# Base plot with date axis
p <- ggplot(data = stockprice, aes(x = DATE, y = SP500)) + ggtitle("S & P 500") +
     geom_line(color = "#00AFBB", size = 1)
# Set axis limits c(min, max)
min <- as.Date("2007-01-01")
max <- NA
p + scale_x_date(limits = c(min, max))
```



US Unemployment Rate

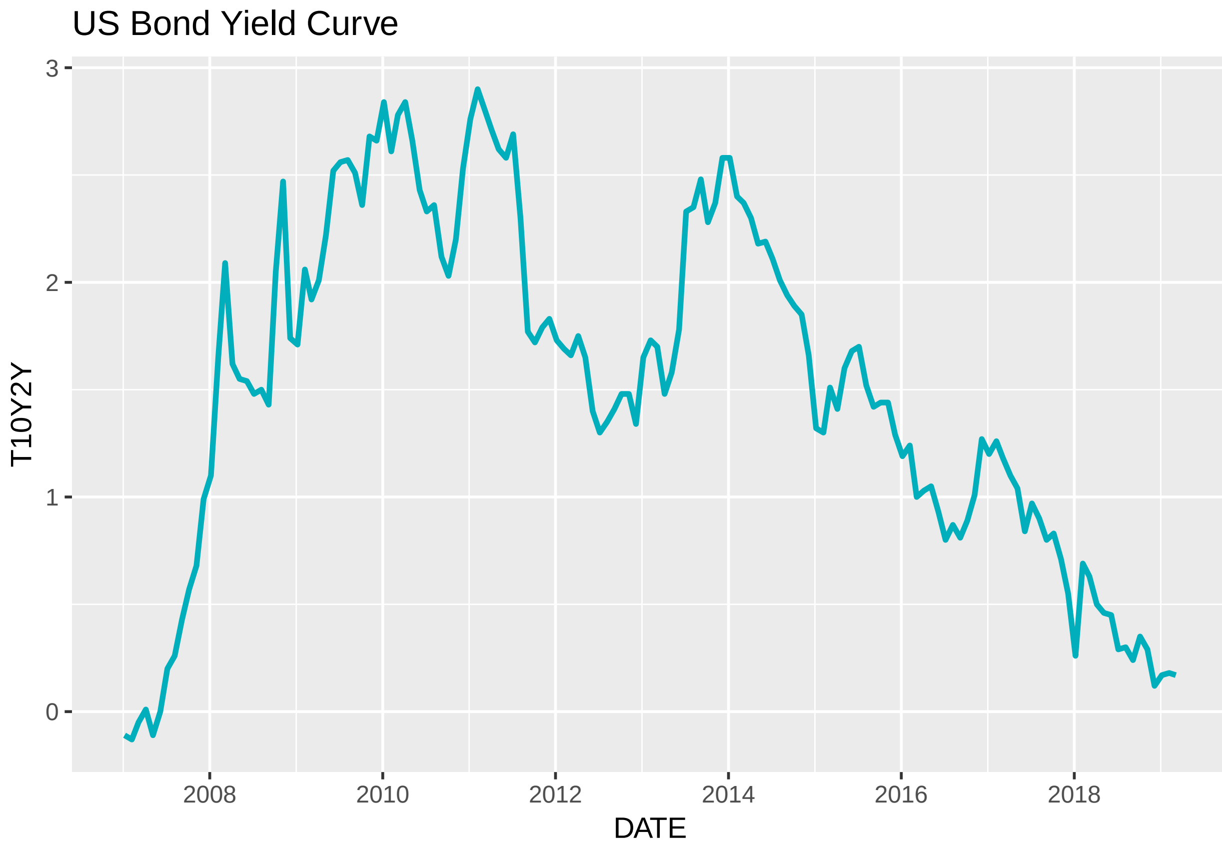
We see before the 2008 recession, unemployment rate was very low and after that, unemployment rate increased to its peak. From there it has reduced and touching the lowest unemployment rate of decade.We see trend and randomness in the data but the seasonal componenent seems to be missing from the data.

```
unemployment$DATE <- as.Date(unemployment$DATE)
unemployment$UNRATE<-as.numeric(unemployment$UNRATE)
ggplot(data = unemployment, aes(x = DATE, y = UNRATE)) + ggtitle("US Unemployment Rate") +
     geom_line(color = "#00AFBB", size = 1)
```

US Bond Yield Curve

US Bond yield curve inverted before 2008 recession, this was and has been very accurate predictor as its a investor sentiments predictor through treasury yield. Here also, we see trent and Random component in the data with seasonal component might be missing from the data.

```
yieldcurve$DATE <- as.Date(yieldcurve$DATE)
yieldcurve$T10Y2Y<-as.numeric(yieldcurve$T10Y2Y)
ggplot(data = yieldcurve, aes(x = DATE, y = T10Y2Y)) + ggtitle("US Bond Yield Curve") +
     geom_line(color = "#00AFBB", size = 1)
```

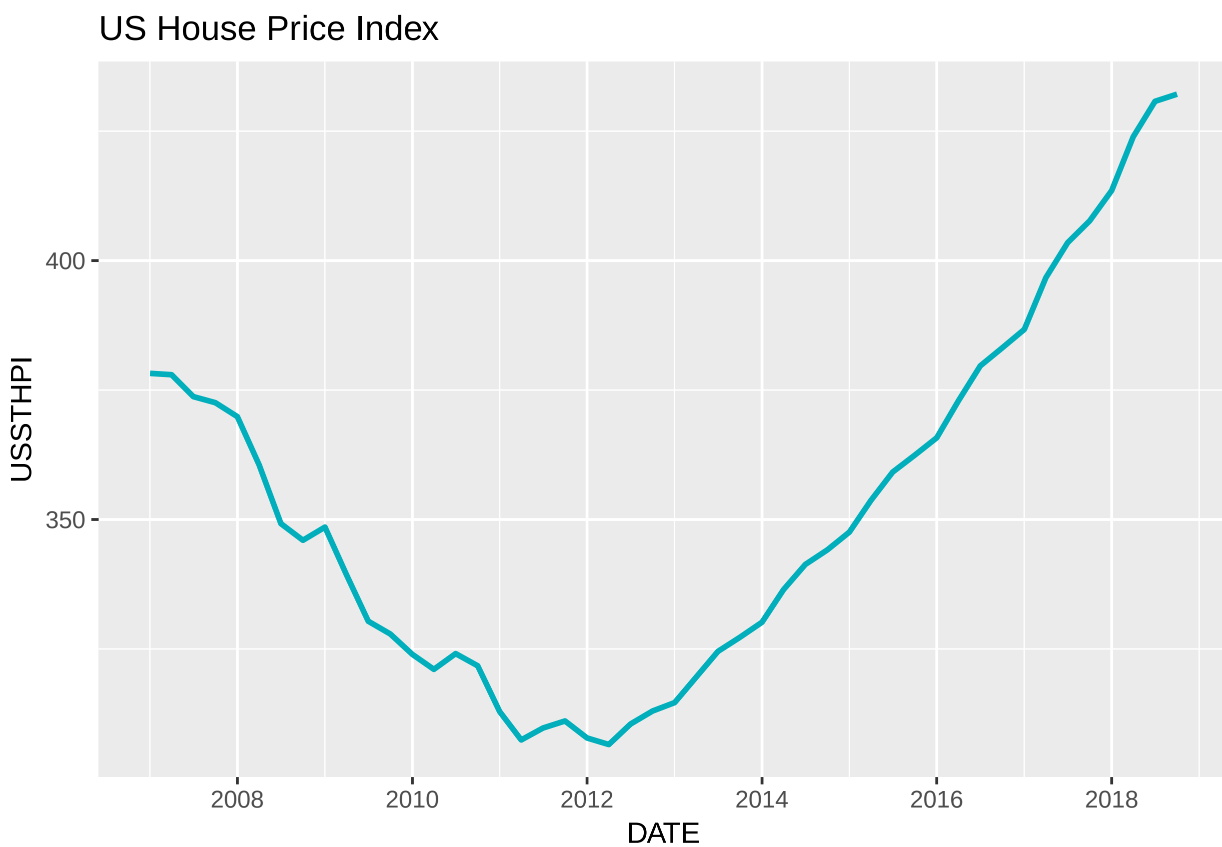


US House Price Index

Us house price index crashed from its peak after 2008 recession and the prices were very low during 2012 and has been increasing from then.

This raph also shows Trend and random component wih seasonal component missing from data.

```
ushpi$DATE <- as.Date(ushpi$DATE)
ushpi$USSTHPI<-as.numeric(ushpi$USSTHPI)
ggplot(data = ushpi, aes(x = DATE, y = USSTHPI)) + ggtitle("US House Price Index") +
     geom_line(color = "#00AFBB", size = 1)
```



Correlations

Unemployment rate and sp500 have high negative correlation meaning as sp500 increases unemployment rate decreases

sp 500 and yield curve also have high negative correlation

sp500 and US House price index have positive correlation

Unemployment rate and yield curve have positive correlation

Unemployment rate and US house price index have negative correlation

Yield curve and Us house price have negative corelation which signifies US house price index doesn't responds fast to the recession centiments.

Making Time series Object and seeing first 20 values of each

```r
sp_500 <- ts(stockprice$SP500, start=c(2007,1), freq = 12)

unemp <- ts(unemployment$UNRATE, start = c(2007,1), freq = 12)

yield <- ts(yieldcurve$T10Y2Y, start = c(2007,1), freq = 12)


hpi <- ts(ushpi$USSTHPI, start = c(2007,1), freq = 4)

head(sp_500, 20)
```

```
##          Jan     Feb     Mar     Apr     May     Jun     Jul     Aug
## 2007 1438.24 1406.82 1420.86 1482.37 1530.62 1503.35 1455.27 1473.99
## 2008 1378.55 1330.63 1322.70 1385.59 1400.38 1280.00 1267.38 1282.83
##          Sep     Oct     Nov     Dec
## 2007 1526.75 1549.38 1481.14 1468.36
## 2008
```

```r
head(unemp,20)
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 2007 4.6 4.5 4.4 4.5 4.4 4.6 4.7 4.6 4.7 4.7 4.7 5.0
## 2008 5.0 4.9 5.1 5.0 5.4 5.6 5.8 6.1
```

```r
head(yield,20)
```

```
##        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov
## 2007 -0.11 -0.13 -0.05  0.01 -0.11  0.00  0.20  0.26  0.43  0.57  0.68
## 2008  1.10  1.65  2.09  1.62  1.55  1.54  1.48  1.50
##        Dec
## 2007  0.99
## 2008
```

```r
head(hpi,20)
```

```
##        Qtr1   Qtr2   Qtr3   Qtr4
## 2007 378.22 377.96 373.73 372.56
## 2008 369.86 360.54 349.20 345.99
## 2009 348.53 339.32 330.35 327.89
## 2010 323.96 321.06 324.10 321.75
## 2011 312.90 307.43 309.73 311.09
```

Doing Mean and Naive Forecast

Mean and Naive Forecast of SP500

Naive forecast Predicts the next 10 values as 2900.45 as it naively takes last value.

Mean forecast Predicts Next 10 Values as 1729.289 as it takes Mean of the values.

Bias Adjusted and Simple Back transformation points towards the same direction of forecasting that is lesss increase of sp500 Index

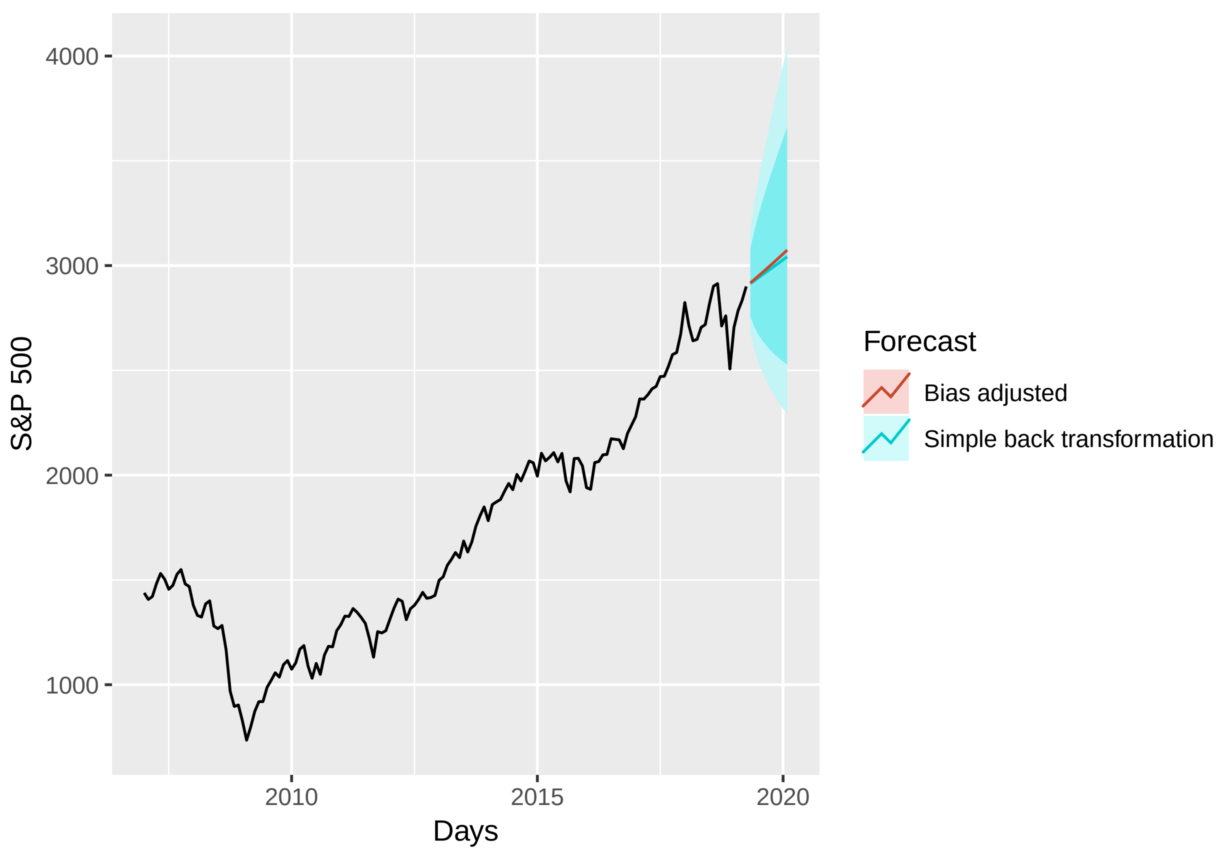
```r
fit_nsp500 <- naive(sp_500, h=10)
print(fit_nsp500)
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## May 2019        2900.45 2814.787 2986.113 2769.440 3031.460
## Jun 2019        2900.45 2779.304 3021.596 2715.174 3085.726
## Jul 2019        2900.45 2752.077 3048.822 2673.534 3127.366
## Aug 2019        2900.45 2729.124 3071.776 2638.430 3162.470
## Sep 2019        2900.45 2708.902 3091.998 2607.502 3193.398
## Oct 2019        2900.45 2690.620 3110.280 2579.542 3221.358
## Nov 2019        2900.45 2673.807 3127.093 2553.830 3247.070
## Dec 2019        2900.45 2658.159 3142.741 2529.897 3271.003
## Jan 2020        2900.45 2643.461 3157.439 2507.420 3293.480
## Feb 2020        2900.45 2629.560 3171.340 2486.160 3314.740
```

```r
fit_msp500 <- meanf(sp_500, h=10)
print(fit_msp500)
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## May 2019       1729.289 994.6247 2463.953 601.4819 2857.096
## Jun 2019       1729.289 994.6247 2463.953 601.4819 2857.096
## Jul 2019       1729.289 994.6247 2463.953 601.4819 2857.096
## Aug 2019       1729.289 994.6247 2463.953 601.4819 2857.096
## Sep 2019       1729.289 994.6247 2463.953 601.4819 2857.096
## Oct 2019       1729.289 994.6247 2463.953 601.4819 2857.096
## Nov 2019       1729.289 994.6247 2463.953 601.4819 2857.096
## Dec 2019       1729.289 994.6247 2463.953 601.4819 2857.096
## Jan 2020       1729.289 994.6247 2463.953 601.4819 2857.096
## Feb 2020       1729.289 994.6247 2463.953 601.4819 2857.096
```

```r
fc <- rwf(sp_500, drift=TRUE, lambda=0, h=10)
fc2 <- rwf(sp_500, drift=TRUE, lambda=0, h=10,
           biasadj=TRUE)


autoplot(sp_500) +
  autolayer(fc, series="Simple back transformation") +
  autolayer(fc2, series="Bias adjusted",PI=FALSE) +
  guides(color=guide_legend(title="Forecast"))  +labs(y= "S&P 500", x = "Days")
```



Forecast of Unemployment

Naive forecast Predicts the next 10 values as 3.8 as it naively takes last value.

Mean forecast Predicts Next 10 Values as 6.46 as it takes Mean of the values.

Bias Adjusted and Simple Back transformation points towards the same direction of forecasting that is slight decrease in Unemployment rate.

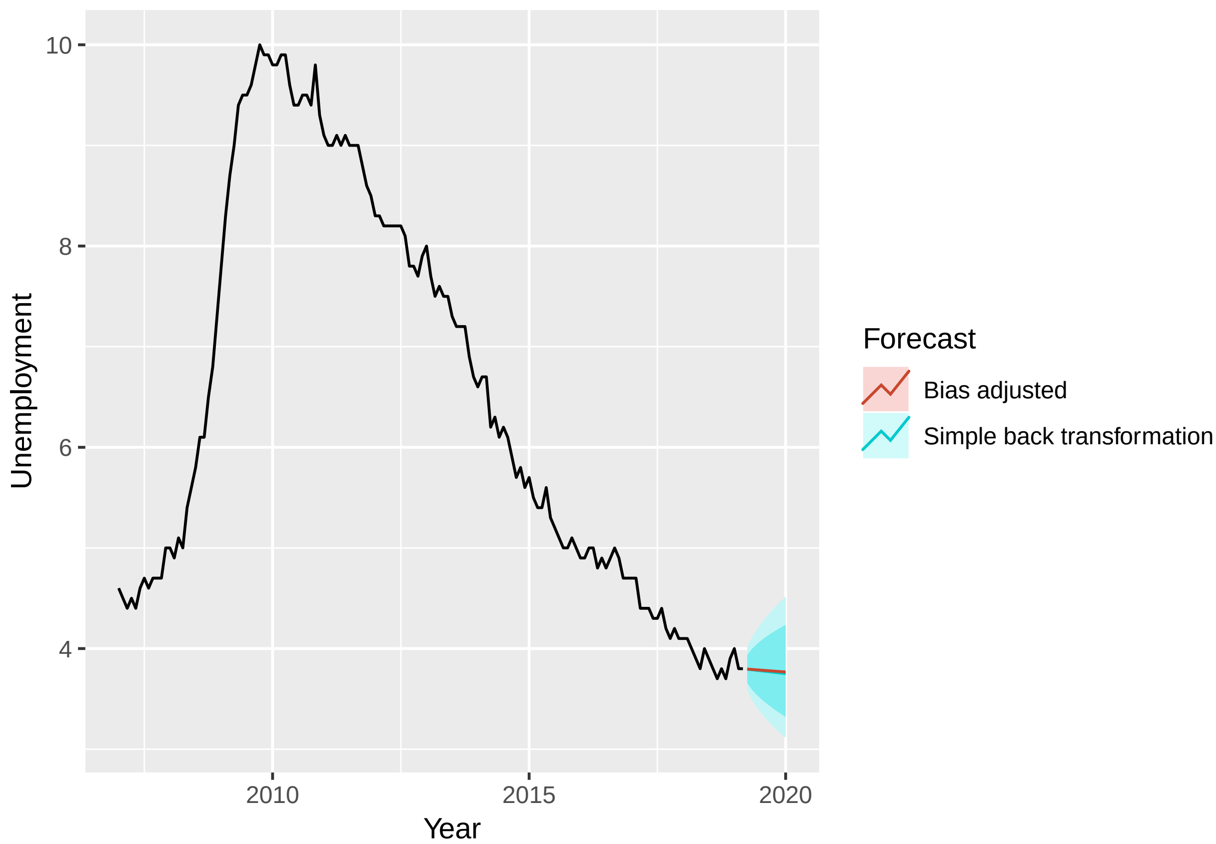
```
fit_nunemp <- naive(unemp, h=10)
print(fit_nunemp)
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Apr 2019            3.8 3.571044 4.028956 3.449842 4.150158
## May 2019            3.8 3.476207 4.123793 3.304801 4.295199
## Jun 2019            3.8 3.403436 4.196564 3.193508 4.406492
## Jul 2019            3.8 3.342087 4.257913 3.099683 4.500317
## Aug 2019            3.8 3.288038 4.311962 3.017022 4.582978
## Sep 2019            3.8 3.239174 4.360826 2.942290 4.657710
## Oct 2019            3.8 3.194239 4.405761 2.873568 4.726432
## Nov 2019            3.8 3.152414 4.447586 2.809602 4.790398
## Dec 2019            3.8 3.113131 4.486869 2.749525 4.850475
## Jan 2020            3.8 3.075977 4.524023 2.692702 4.907298
```

```
fit_munemp<- meanf(unemp, h=10)
# mean forecast
print(fit_munemp)
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Apr 2019       6.464626 3.859157 9.070095 2.464781 10.46447
## May 2019       6.464626 3.859157 9.070095 2.464781 10.46447
## Jun 2019       6.464626 3.859157 9.070095 2.464781 10.46447
## Jul 2019       6.464626 3.859157 9.070095 2.464781 10.46447
## Aug 2019       6.464626 3.859157 9.070095 2.464781 10.46447
## Sep 2019       6.464626 3.859157 9.070095 2.464781 10.46447
## Oct 2019       6.464626 3.859157 9.070095 2.464781 10.46447
## Nov 2019       6.464626 3.859157 9.070095 2.464781 10.46447
## Dec 2019       6.464626 3.859157 9.070095 2.464781 10.46447
## Jan 2020       6.464626 3.859157 9.070095 2.464781 10.46447
```

```
fc <- rwf(unemp, drift=TRUE, lambda=0, h=10)
fc2 <- rwf(unemp, drift=TRUE, lambda=0, h=10,
           biasadj=TRUE)


autoplot(unemp) +
  autolayer(fc, series="Simple back transformation") +
  autolayer(fc2, series="Bias adjusted",PI=FALSE) +
  guides(color=guide_legend(title="Forecast"))  +labs(y= "Unemployment", x = "Year")
```



Forecast of Yield curve

Naive forecast Predicts the next 10 values as 0.17 as it naively takes last value.

Mean forecast Predicts Next 10 Values as 1.51 as it takes Mean of the values.

Bias Adjusted and Simple Back transformation points towards the same direction of forecasting that is constant over a period of time.

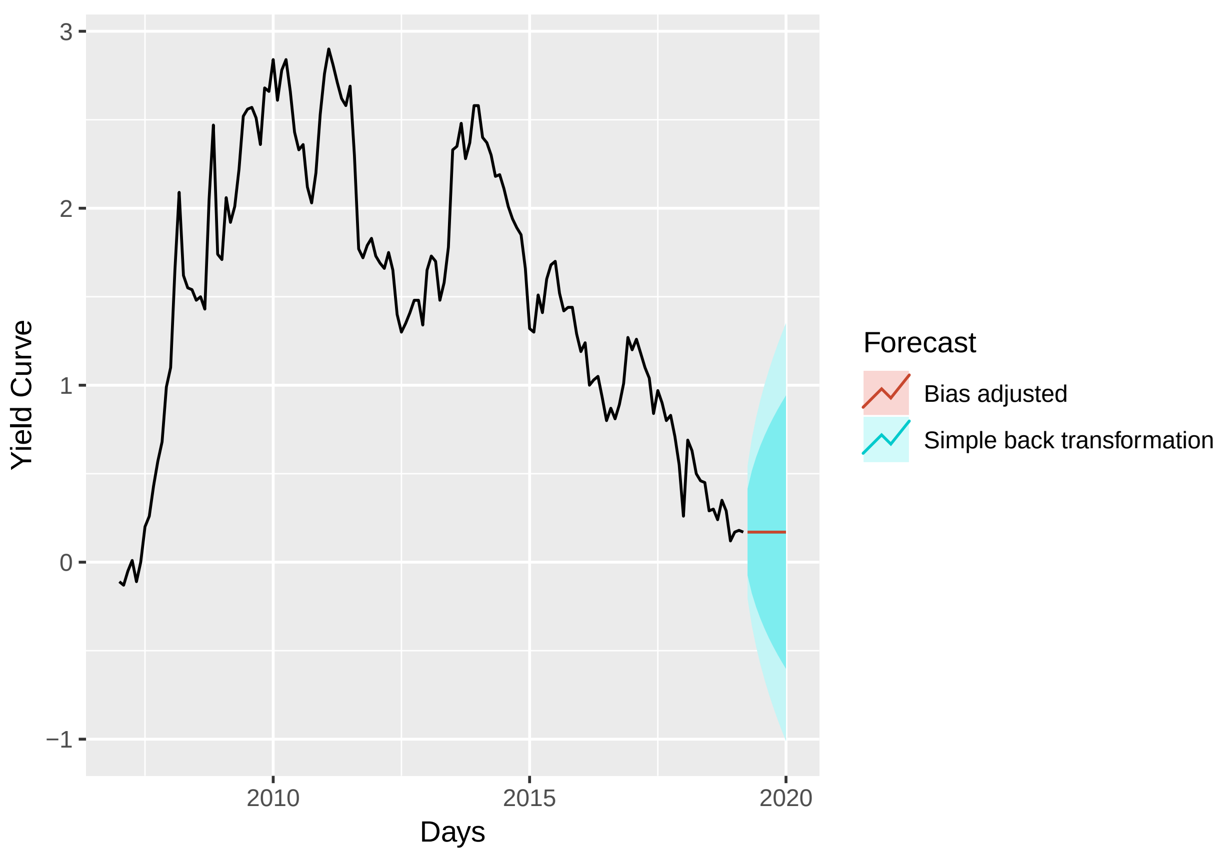
```
fit_nyield <- naive(yield, h=10)
print(fit_nyield)
```

```
##          Point Forecast       Lo 80     Hi 80      Lo 95     Hi 95
## Apr 2019           0.17 -0.07432496 0.4143250 -0.2036628 0.5436628
## May 2019           0.17 -0.17552767 0.5155277 -0.3584390 0.6984390
## Jun 2019           0.17 -0.25318324 0.5931832 -0.4772029 0.8172029
## Jul 2019           0.17 -0.31864992 0.6586499 -0.5773256 0.9173256
## Aug 2019           0.17 -0.37632722 0.7163272 -0.6655354 1.0055354
## Sep 2019           0.17 -0.42847148 0.7684715 -0.7452831 1.0852831
## Oct 2019           0.17 -0.47642308 0.8164231 -0.8186188 1.1586188
## Nov 2019           0.17 -0.52105534 0.8610553 -0.8868779 1.2268779
## Dec 2019           0.17 -0.56297488 0.9029749 -0.9509883 1.2909883
## Jan 2020           0.17 -0.60262336 0.9426234 -1.0116255 1.3516255
```

```
fit_myield <- meanf(yield, h=10)
# mean forecast
print(fit_myield)
```

```
##          Point Forecast     Lo 80    Hi 80       Lo 95    Hi 95
## Apr 2019       1.518231 0.4780745 2.558388 -0.07858884 3.115051
## May 2019       1.518231 0.4780745 2.558388 -0.07858884 3.115051
## Jun 2019       1.518231 0.4780745 2.558388 -0.07858884 3.115051
## Jul 2019       1.518231 0.4780745 2.558388 -0.07858884 3.115051
## Aug 2019       1.518231 0.4780745 2.558388 -0.07858884 3.115051
## Sep 2019       1.518231 0.4780745 2.558388 -0.07858884 3.115051
## Oct 2019       1.518231 0.4780745 2.558388 -0.07858884 3.115051
## Nov 2019       1.518231 0.4780745 2.558388 -0.07858884 3.115051
## Dec 2019       1.518231 0.4780745 2.558388 -0.07858884 3.115051
## Jan 2020       1.518231 0.4780745 2.558388 -0.07858884 3.115051
```

```r
fc <- rwf(yield, h=10)
fc2 <- rwf(yield, h=10,
           biasadj=TRUE)


autoplot(yield) +
  autolayer(fc, series="Simple back transformation") +
  autolayer(fc2, series="Bias adjusted",PI=FALSE) +
  guides(color=guide_legend(title="Forecast"))  +labs(y= "Yield Curve", x = "Days")
```



Forecast of House Price index

Naive forecast Predicts the next 10 values as 432.14 as it naively takes last value.

Mean forecast Predicts Next 10 Values as 352.11 as it takes Mean of the values.

Bias Adjusted and Simple Back transformation points towards the same direction of forecasting that is slight increase over a period of time.

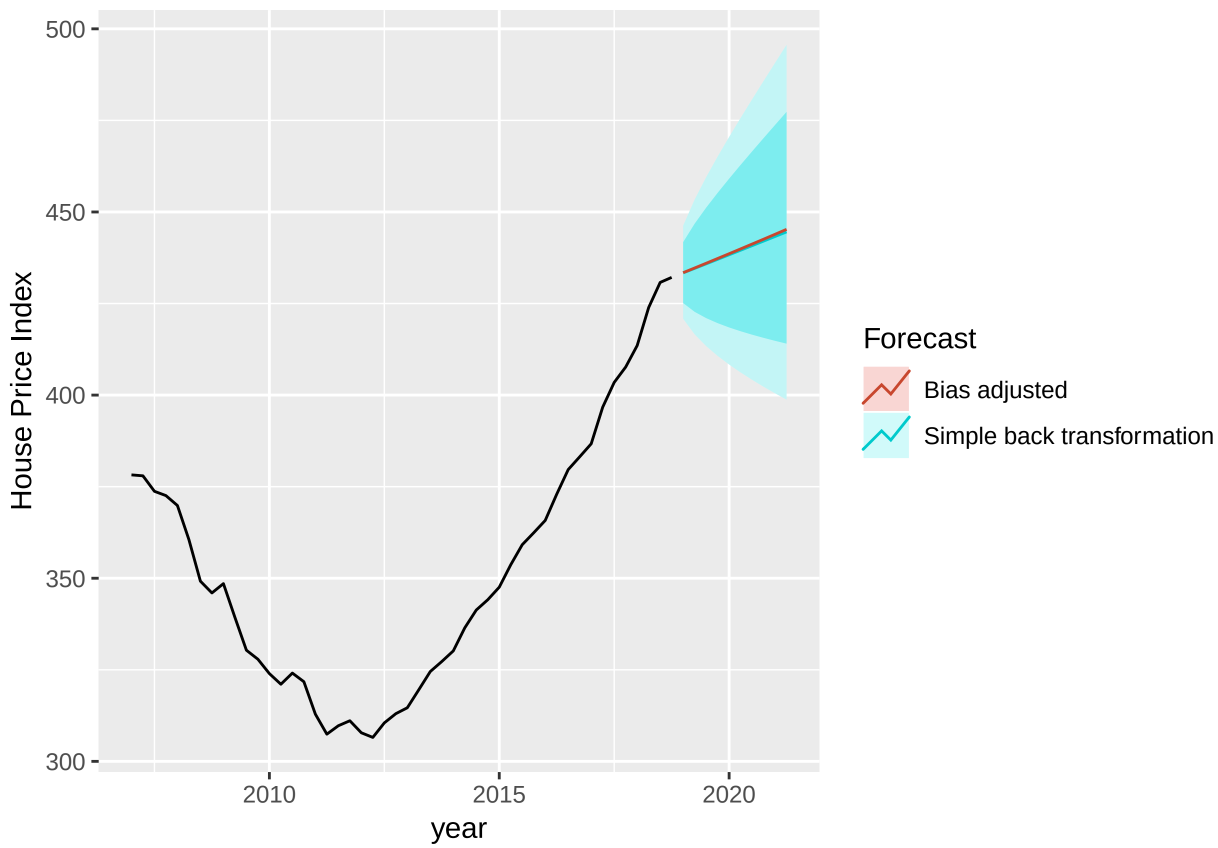
```
fit_nhpi <- naive(hpi, h=10)
print(fit_nhpi)
```

```
##         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2019 Q1         432.14 425.3128 438.9672 421.6987 442.5813
## 2019 Q2         432.14 422.4849 441.7951 417.3738 446.9062
## 2019 Q3         432.14 420.3150 443.9650 414.0552 450.2248
## 2019 Q4         432.14 418.4856 445.7944 411.2575 453.0225
## 2020 Q1         432.14 416.8740 447.4060 408.7926 455.4874
## 2020 Q2         432.14 415.4169 448.8631 406.5642 457.7158
## 2020 Q3         432.14 414.0770 450.2030 404.5150 459.7650
## 2020 Q4         432.14 412.8298 451.4502 402.6076 461.6724
## 2021 Q1         432.14 411.6585 452.6215 400.8162 463.4638
## 2021 Q2         432.14 410.5506 453.7294 399.1218 465.1582
```

```
fit_mhpi <- meanf(hpi, h=10)
# mean forecast
print(fit_mhpi)
```

```
##         Point Forecast    Lo 80   Hi 80    Lo 95    Hi 95
## 2019 Q1       352.1119 305.4328 398.791 279.8666 424.3572
## 2019 Q2       352.1119 305.4328 398.791 279.8666 424.3572
## 2019 Q3       352.1119 305.4328 398.791 279.8666 424.3572
## 2019 Q4       352.1119 305.4328 398.791 279.8666 424.3572
## 2020 Q1       352.1119 305.4328 398.791 279.8666 424.3572
## 2020 Q2       352.1119 305.4328 398.791 279.8666 424.3572
## 2020 Q3       352.1119 305.4328 398.791 279.8666 424.3572
## 2020 Q4       352.1119 305.4328 398.791 279.8666 424.3572
## 2021 Q1       352.1119 305.4328 398.791 279.8666 424.3572
## 2021 Q2       352.1119 305.4328 398.791 279.8666 424.3572
```

```
fc <- rwf(hpi, drift=TRUE, lambda=0, h=10)
fc2 <- rwf(hpi, drift=TRUE, lambda=0, h=10,
           biasadj=TRUE)


autoplot(hpi) +
  autolayer(fc, series="Simple back transformation") +
  autolayer(fc2, series="Bias adjusted",PI=FALSE) +
  guides(color=guide_legend(title="Forecast"))  +labs(y= "House Price Index", x = "year")
```



Decomposition of the data

we decompose the data to get better understanding of seasonal, trend and Remainder components

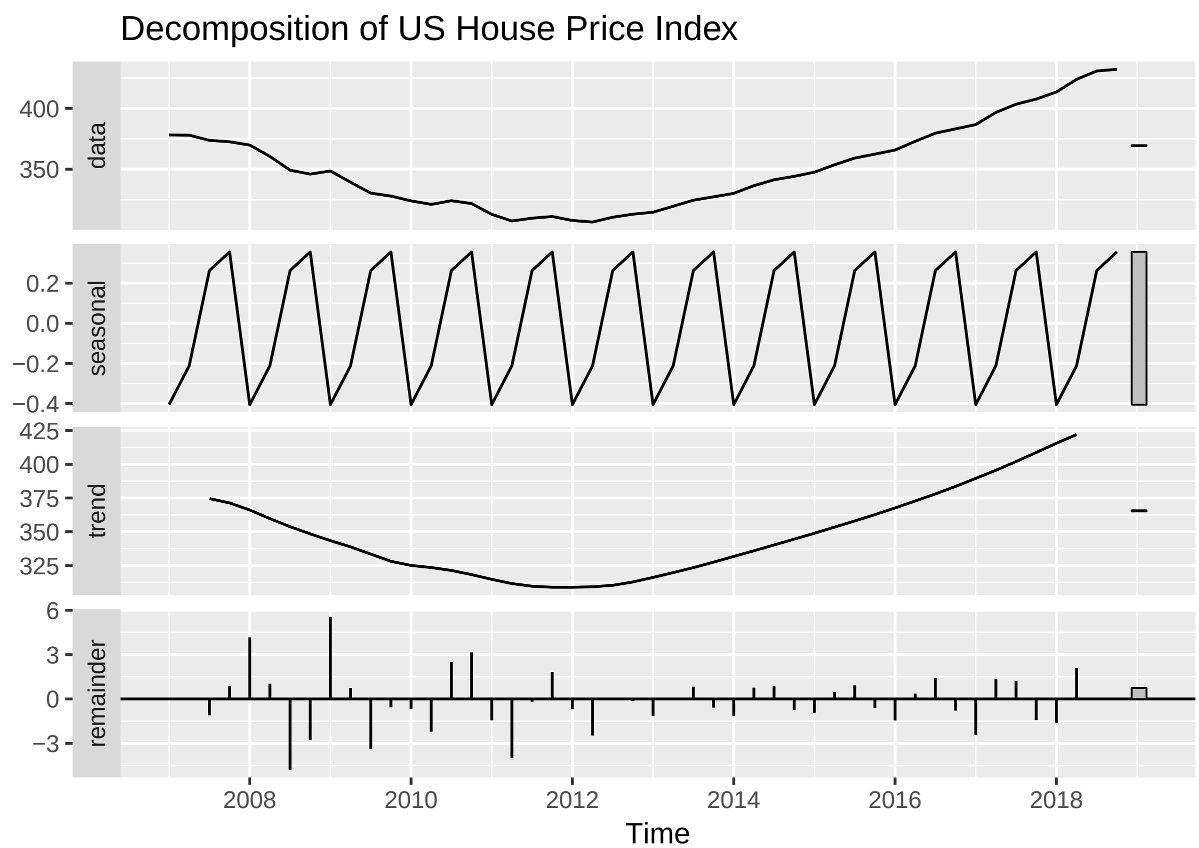
Decomposition of S&P 500, Unemployment rate, House price Index and Yield Curve Data:

Trend: Data has a trend component and that validate our point, when we looked data graph Naively

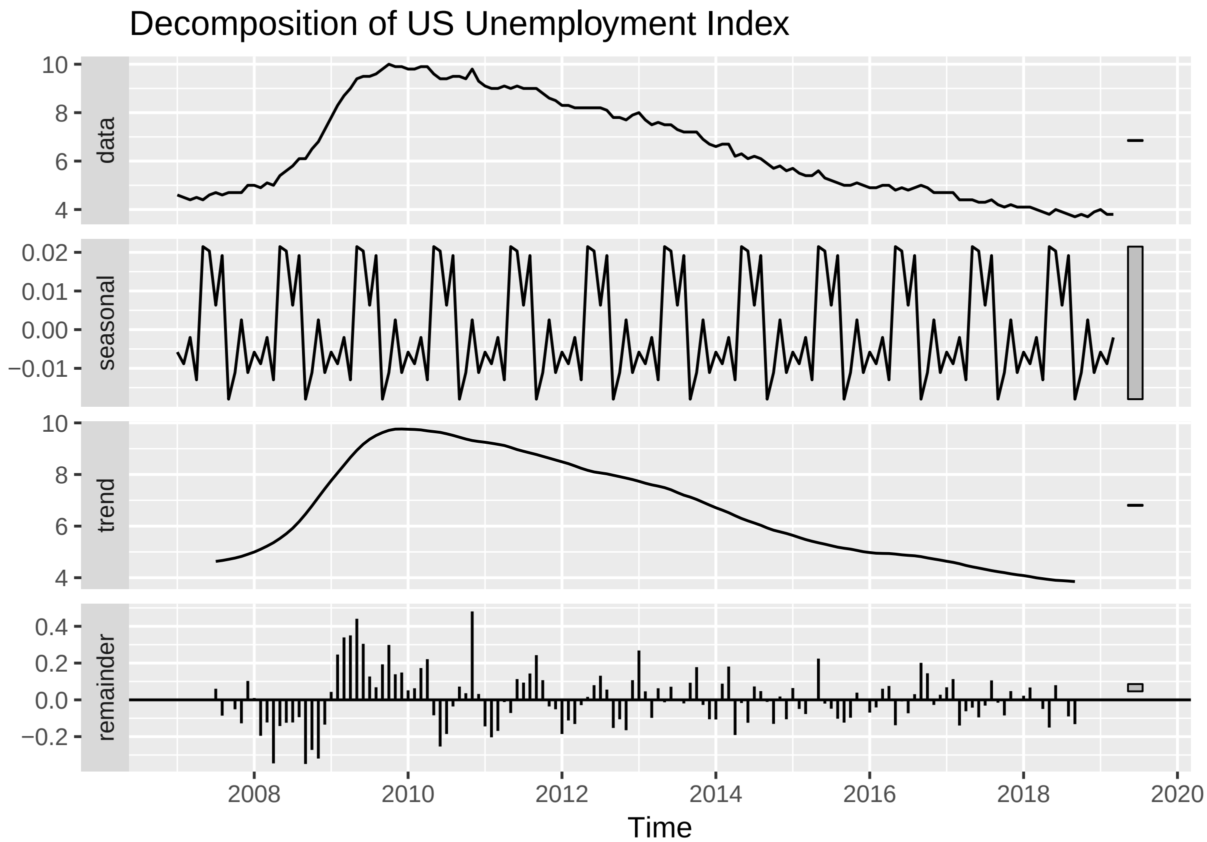
Seasonal: Seasonal component is not varying much over period of time so might not have much of seasonal component.

Reminder: Remainder component that is full data minus seasonal and trend is present

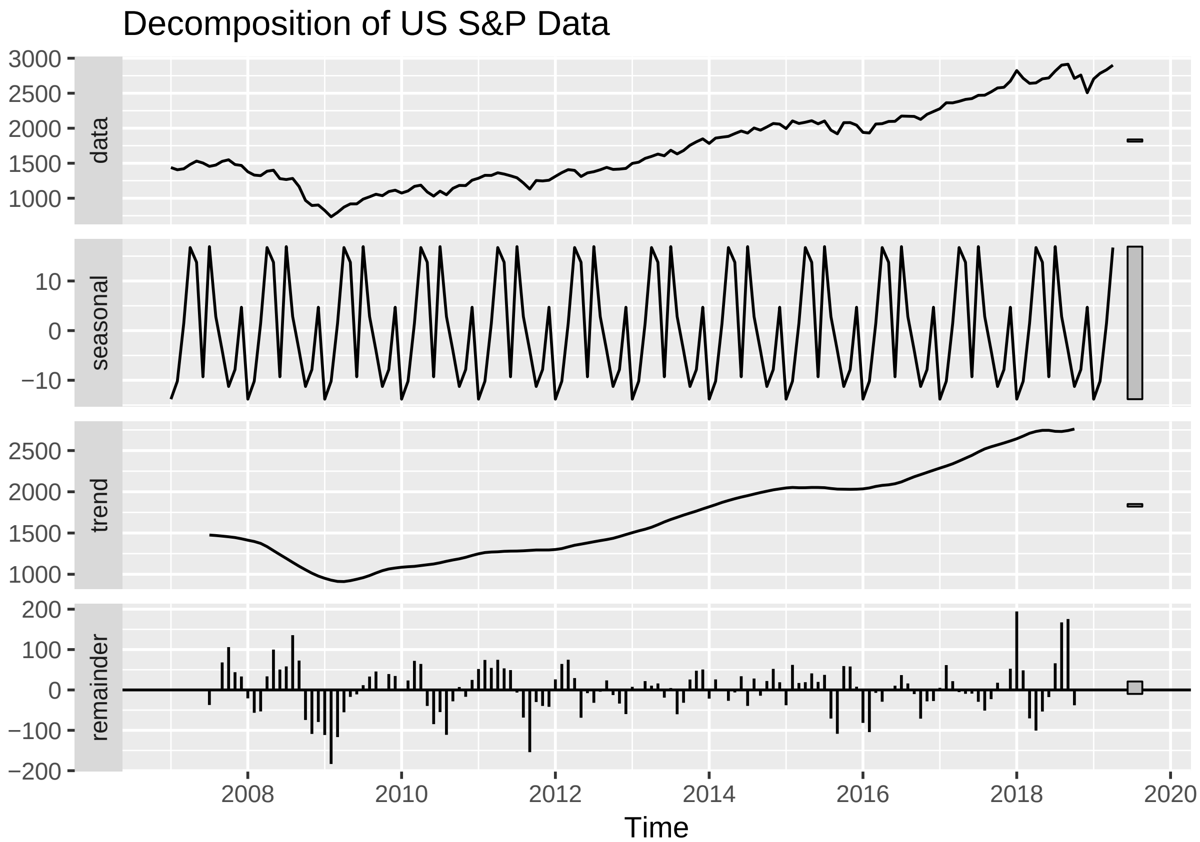
```
decom1 <- decompose(hpi)
autoplot(decom1) + ggtitle("Decomposition of US House Price Index")
```



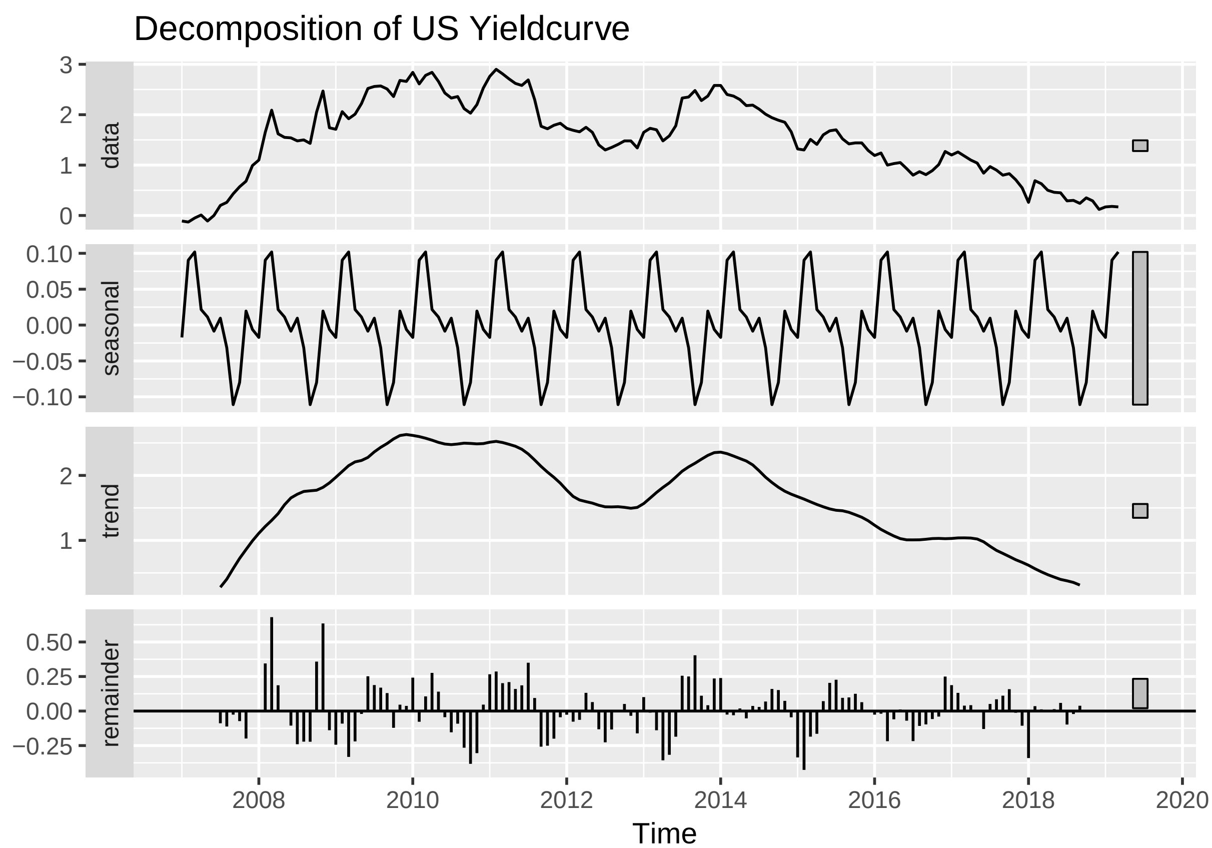

```
decom2 <- decompose(unemp)
autoplot(decom2) + ggtitle("Decomposition of US Unemployment Index")
```




```
decom3 <- decompose(sp_500)
autoplot(decom3) + ggtitle("Decomposition of US S&P Data")
```




```
decom4 <- decompose(yield)
autoplot(decom4) + ggtitle("Decomposition of US Yieldcurve")
```



Exponential Smoothening

Based on the description of Trend and Seasonality of the Data

The more recent the observation the higher the associated weights. This framework generates reliable forecasts quickly and for a wide range of time series, which is a great advantage and of major importance to applications in industry.

Simple Exponential smoothening

This method is suitable for forecasting data with no clear trend and or seasonal pattern.

We can see that simple exponential smoothening has predicted all the values of four data set to be constant.

```
sp <- window(sp_500, start=2007)
  ### Estimate parameters
fc <- ses(sp, h=5)
summary(fc[["model"]])
```

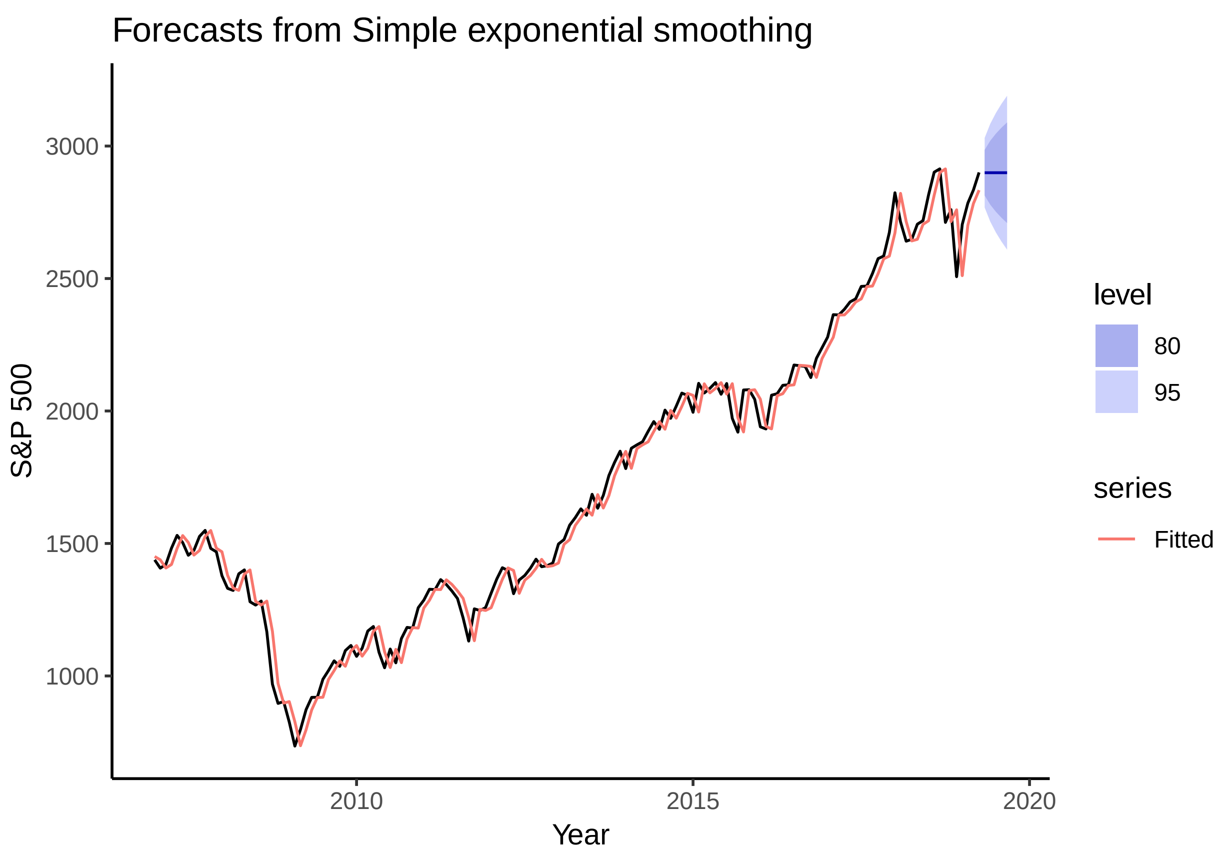
```
## Simple exponential smoothing
##
## Call:
##  ses(y = sp, h = 5)
##
##   Smoothing parameters:
##     alpha = 0.9846
##
```

```
##   Initial states:
##     l = 1450.7767
##
##   sigma:  67.1058
##
##      AIC     AICc      BIC
## 1988.630 1988.796 1997.621
##
## Training set error measures:
##                    ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 9.941382 66.65082 51.14969 0.3796088 3.258575 0.2274918
##                    ACF1
## Training set 0.01156233
```

```
  ### Accuracy of one-step-ahead
round(accuracy(fc, 2))
```

```
##                 ME RMSE  MAE     MPE   MAPE MASE ACF1
## Training set    10   67   51       0      3    1    0
## Test set     -2897 2897 2897 -144871 144871   57   NA
```

```
autoplot(fc) +
  autolayer(fitted(fc), series="Fitted") + theme_classic() +
  ylab("S&P 500") + xlab("Year")
```




```
hpi <- window(hpi, start=2007)
  ### Estimate parameters
fc <- ses(hpi, h=5)
```

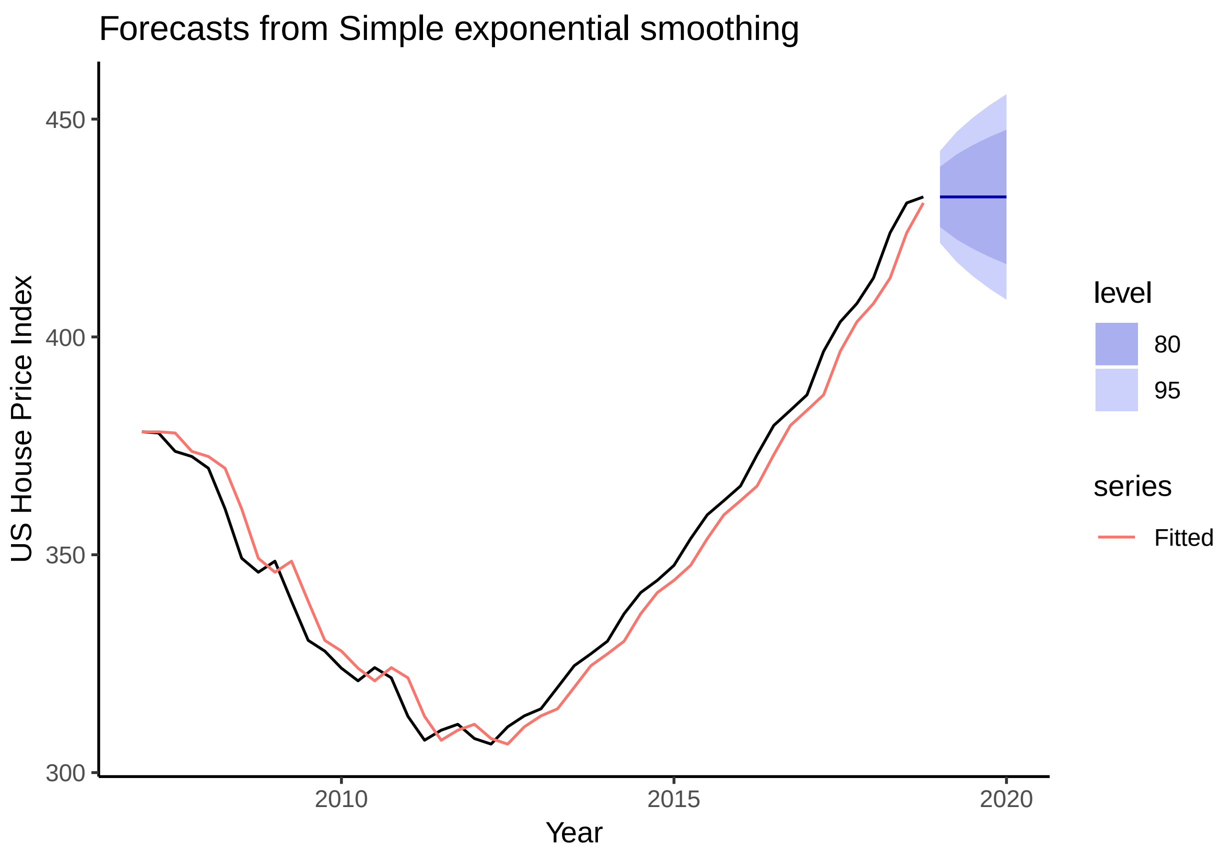
```r
summary(fc[["model"]])
```

```
## Simple exponential smoothing
##
## Call:
##  ses(y = hpi, h = 5)
##
##   Smoothing parameters:
##     alpha = 0.9999
##
##   Initial states:
##     l = 378.2003
##
##   sigma:  5.3852
##
##      AIC     AICc      BIC
## 351.4065 351.9519 357.0201
##
## Training set error measures:
##                    ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 1.123853 5.271863 4.495774 0.2668724 1.271906 0.2700644
##                   ACF1
## Training set 0.6895493
```

```r
  ### Accuracy of one-step-ahead
round(accuracy(fc, 2))
```

```
##                ME RMSE MAE    MPE  MAPE MASE ACF1
## Training set    1    5   4      0     1    1    1
## Test set     -430  430 430 -21507 21507   94   NA
```

```r
autoplot(fc) +
  autolayer(fitted(fc), series="Fitted") + theme_classic() +
  ylab("US House Price Index") + xlab("Year")
```



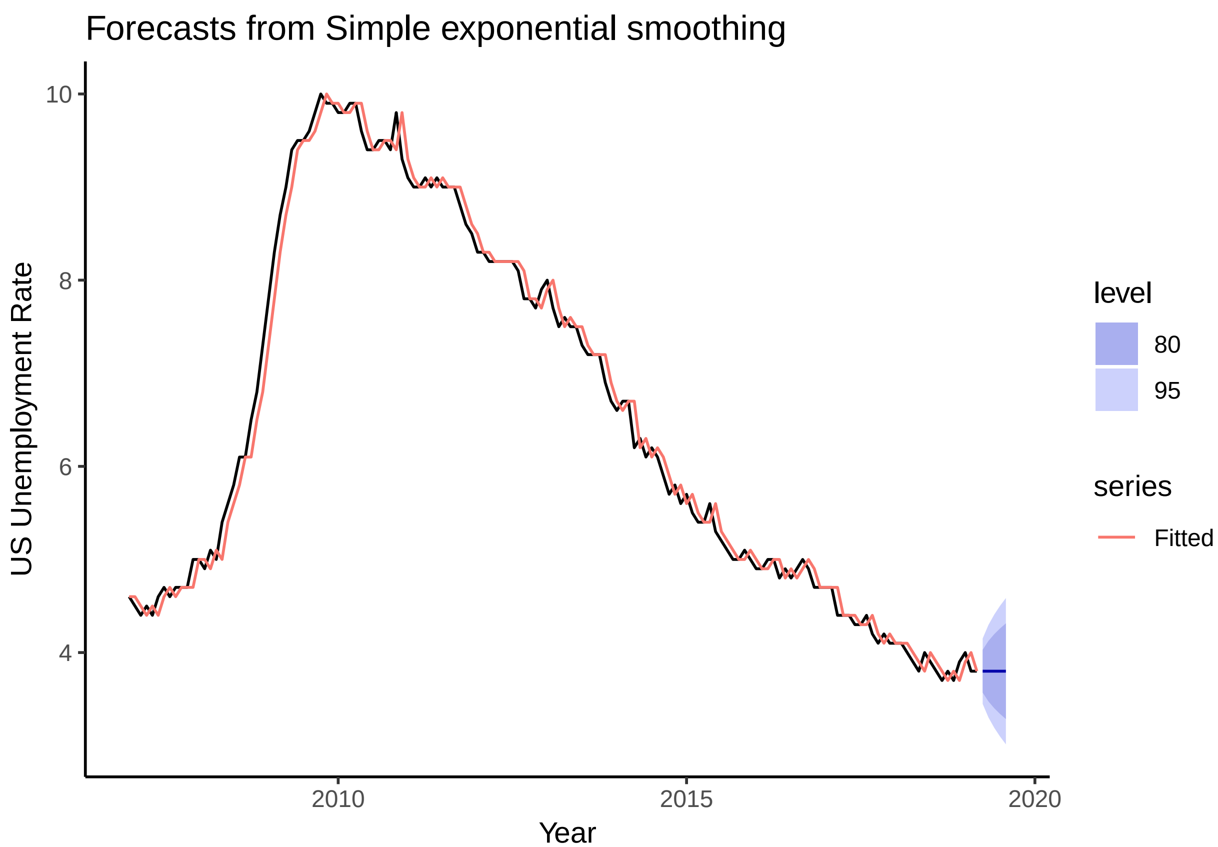

```r
unemp <- window(unemp, start=2007)
  ### Estimate parameters
fc <- ses(unemp, h=5)
summary(fc[["model"]])
```

```
## Simple exponential smoothing
##
## Call:
##  ses(y = unemp, h = 5)
##
##   Smoothing parameters:
##     alpha = 0.9999
##
##   Initial states:
##     l = 4.6007
##
##   sigma:  0.1793
##
##      AIC     AICc      BIC
## 232.2434 232.4112 241.2147
##
## Training set error measures:
##                        ME      RMSE       MAE        MPE     MAPE
## Training set -0.005447817 0.1780518 0.1306225 -0.1695686 2.120655
##                   MASE      ACF1
## Training set 0.1326865 0.2741259
```

```r
### Accuracy of one-step-ahead
round(accuracy(fc, 2))
```

```
##               ME RMSE MAE MPE MAPE MASE ACF1
## Training set  0    0   0   0    2    1    0
## Test set     -2    2   2 -90   90   14   NA
```

```r
autoplot(fc) +
  autolayer(fitted(fc), series="Fitted") + theme_classic() +
  ylab("US Unemployment Rate") + xlab("Year")
```




```r
yieldcurve1 <- window(yield, start=2007)
  ### Estimate parameters
fc <- ses(yieldcurve1, h=5)
summary(fc[["model"]])
```

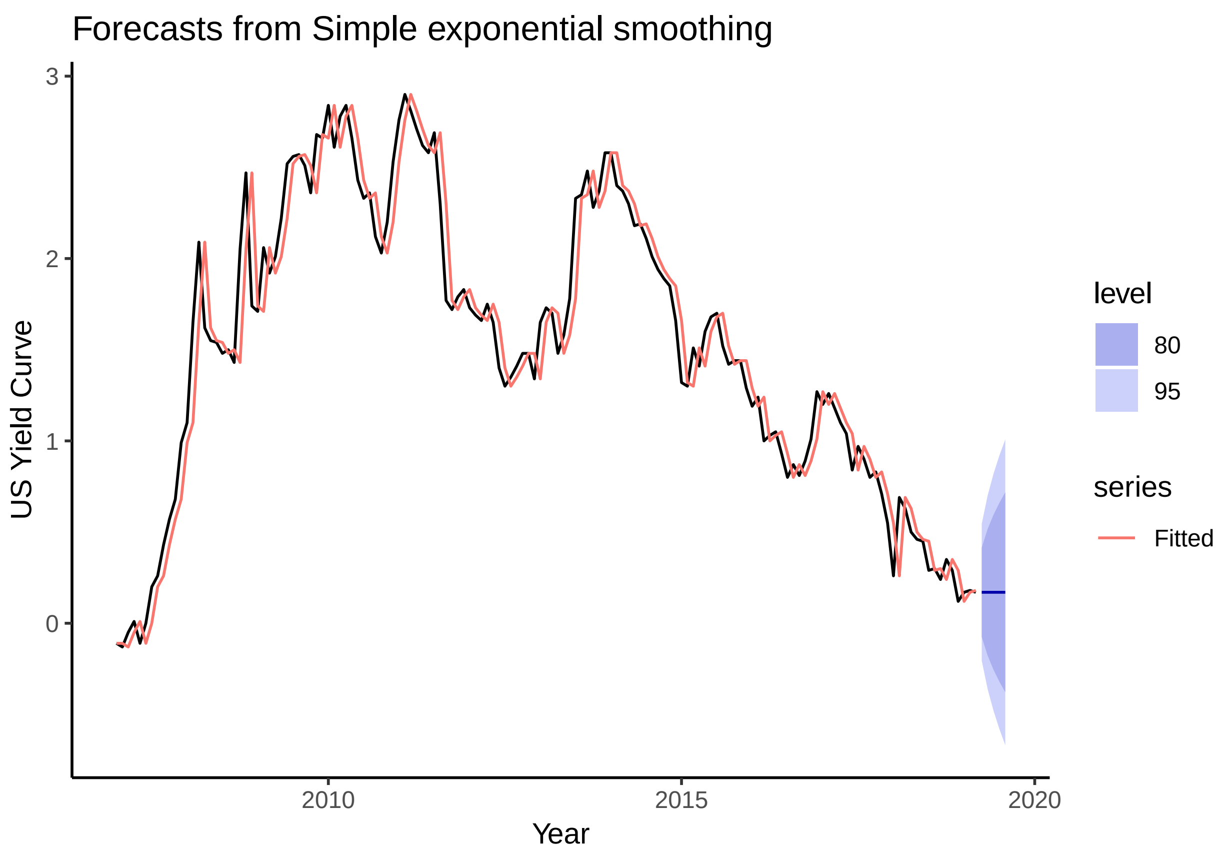
```
## Simple exponential smoothing
##
## Call:
##  ses(y = yieldcurve1, h = 5)
##
##   Smoothing parameters:
##     alpha = 0.9999
##
##   Initial states:
##     l = -0.11
##
```

```
##   sigma:  0.1913
##
##      AIC     AICc      BIC
## 251.3394 251.5072 260.3107
##
## Training set error measures:
##                       ME      RMSE       MAE MPE MAPE      MASE     ACF1
## Training set 0.001904817 0.1900005 0.1368742 Inf  Inf 0.2279548 0.119087
```

```
  ### Accuracy of one-step-ahead
round(accuracy(fc, 2))
```

```
##              ME RMSE MAE MPE MAPE MASE ACF1
## Training set  0    0   0 Inf  Inf    1    0
## Test set      2    2   2  91   91   13   NA
```

```
autoplot(fc) +
  autolayer(fitted(fc), series="Fitted") + theme_classic() +
  ylab("US Yield Curve") + xlab("Year")
```



Trends Method

This method is the extension of simple exponential smoothening to allow forecasting data with trend.

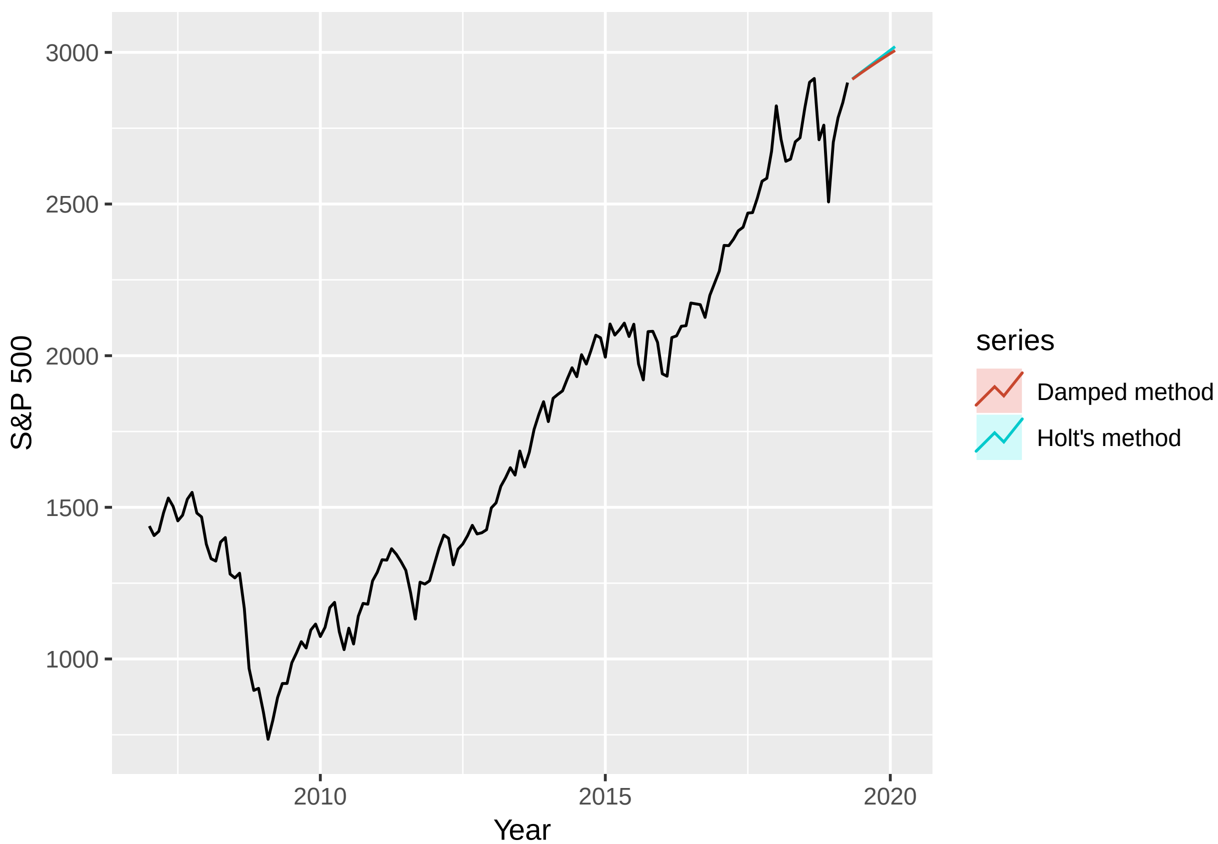
S&P 500: Both Damped and Holt's Trend Method predict stock market to grow further

Unemployment Rate:Both Damped and Holt's trend Method predict the Unemployment rate to go down.

House Price Index: Although Holt's method is bulish on predictions with upward trend, Damped seems a little cautious.

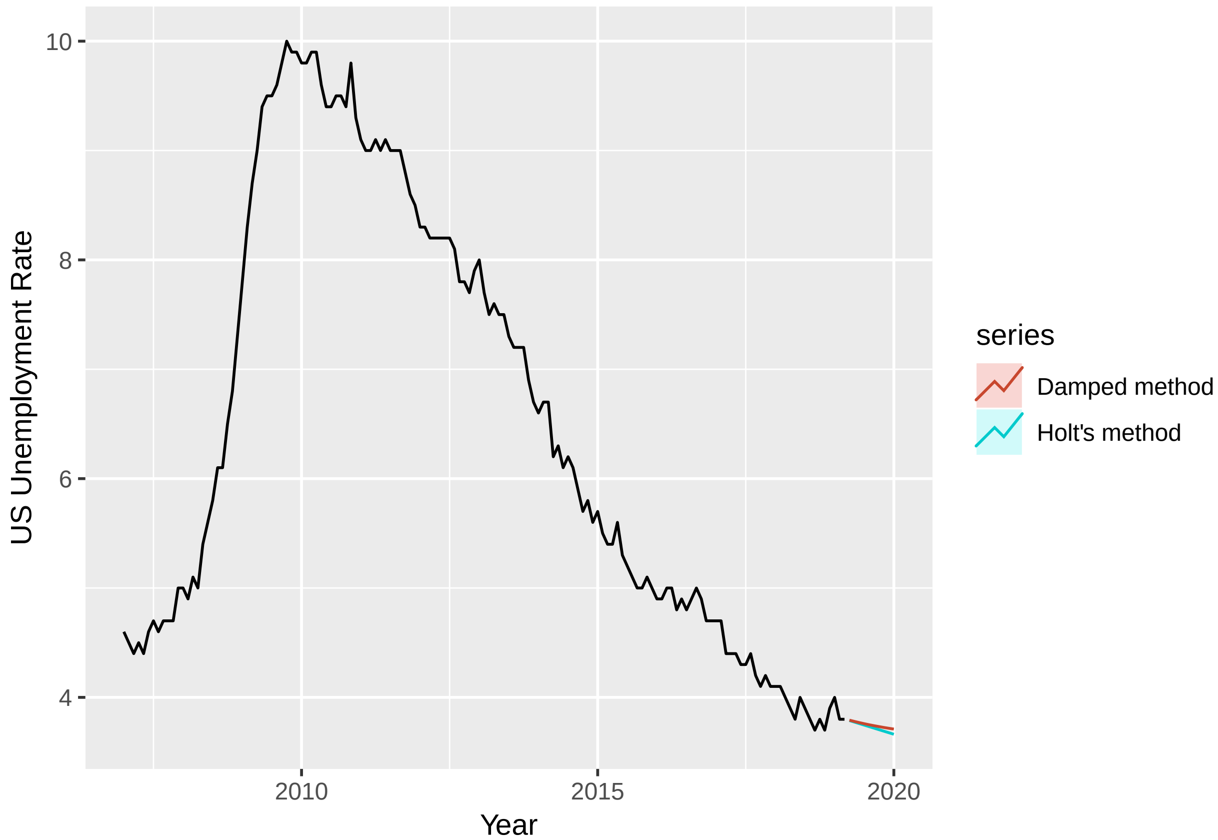
Yield Curve: Both method Shows yield curve to remain constant going further with very litle upward trend in Holt's method.

```
sp <- window(sp_500, start= 2007)
holtfc1 <- holt(sp, h = 10)
holtfc2 <- holt(sp, damped = TRUE, h = 10)
autoplot(sp) +
  autolayer(holtfc1, series = "Holt's method", PI = FALSE) +
  autolayer(holtfc2, series = "Damped method", PI = FALSE) +
  xlab("Year") + ylab("S&P 500")
```

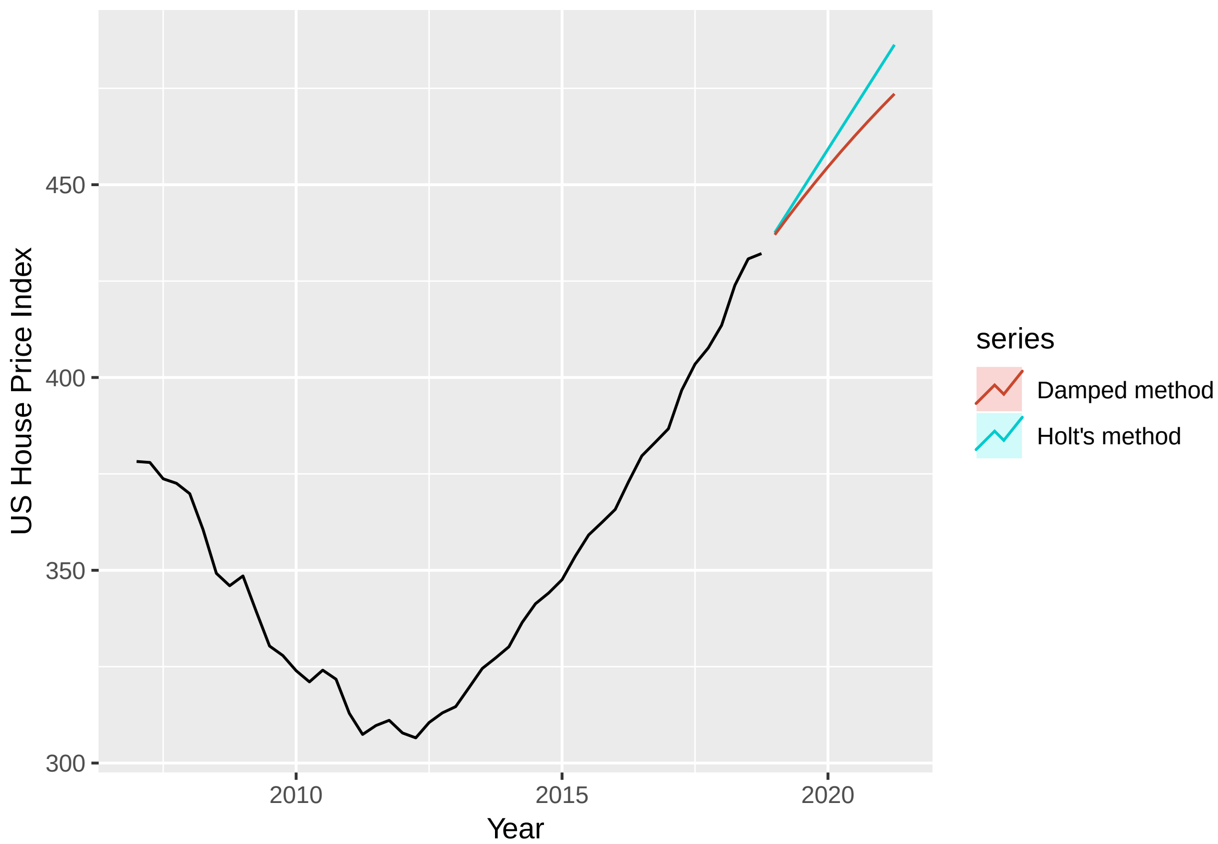



```
un <- window(unemp, start= 2007)
holtfc1 <- holt(un, h = 10)
holtfc2 <- holt(un, damped = TRUE, h = 10)
autoplot(un) +
  autolayer(holtfc1, series = "Holt's method", PI = FALSE) +
```

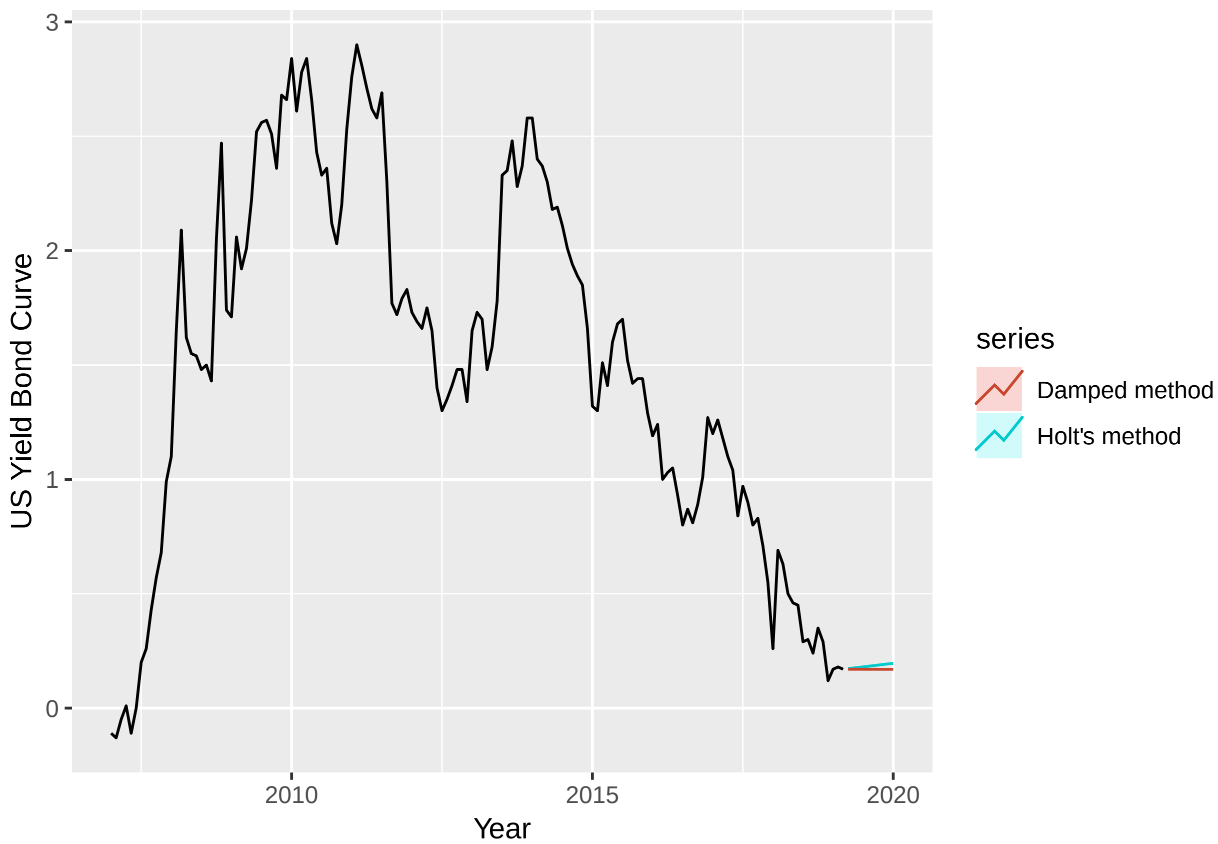
```
  autolayer(holtfc2, series = "Damped method", PI = FALSE) +
  xlab("Year") + ylab("US Unemployment Rate")
```




```
hp <- window(hpi, start= 2007)
holtfc1 <- holt(hp, h = 10)
holtfc2 <- holt(hp, damped = TRUE, h = 10)
autoplot(hp) +
  autolayer(holtfc1, series = "Holt's method", PI = FALSE) +
  autolayer(holtfc2, series = "Damped method", PI = FALSE) +
  xlab("Year") + ylab("US House Price Index")
```




```r
yi <- window(yield, start= 2007)
holtfc1 <- holt(yi, h = 10)
holtfc2 <- holt(yi, damped = TRUE, h = 10)
autoplot(yi) +
  autolayer(holtfc1, series = "Holt's method", PI = FALSE) +
  autolayer(holtfc2, series = "Damped method", PI = FALSE) +
  xlab("Year") + ylab("US Yield Bond Curve")
```



Seasonal Methods

Holt and winters extended Holt's method to capture seasonability.The Holt-Winters seasonal method comprises the forecast equation and three smoothing equations of level, trend and seasonal component.

In these we have showed holt-winter additive and multiplicative methods with their decomposition to get the full picture.

S&P 500: Multiplicative forecast is more bulish than additive forecast.

Unemployment Rate: Though additive methods shows a little constant forecast, Holt-winters multiplicative forecast shows an upward trend in unemloyment rate.

House Price Index: Both Multiplicative and additive components show same upward trend line predictions with decomposition showing some varying seasonal component.

Yield Curve: Both Multiplicative and additive methods shows downward trend line and hence indicates inversion of yield curve.

```
  # HOlt-Winter's Additive & Multiplicative methods
sp <- window(sp_500, start=2007)
hwfc1 <- hw(sp, seasonal="additive")
hwfc2 <- hw(sp, seasonal="multiplicative")
hwfc1[["model"]]
```

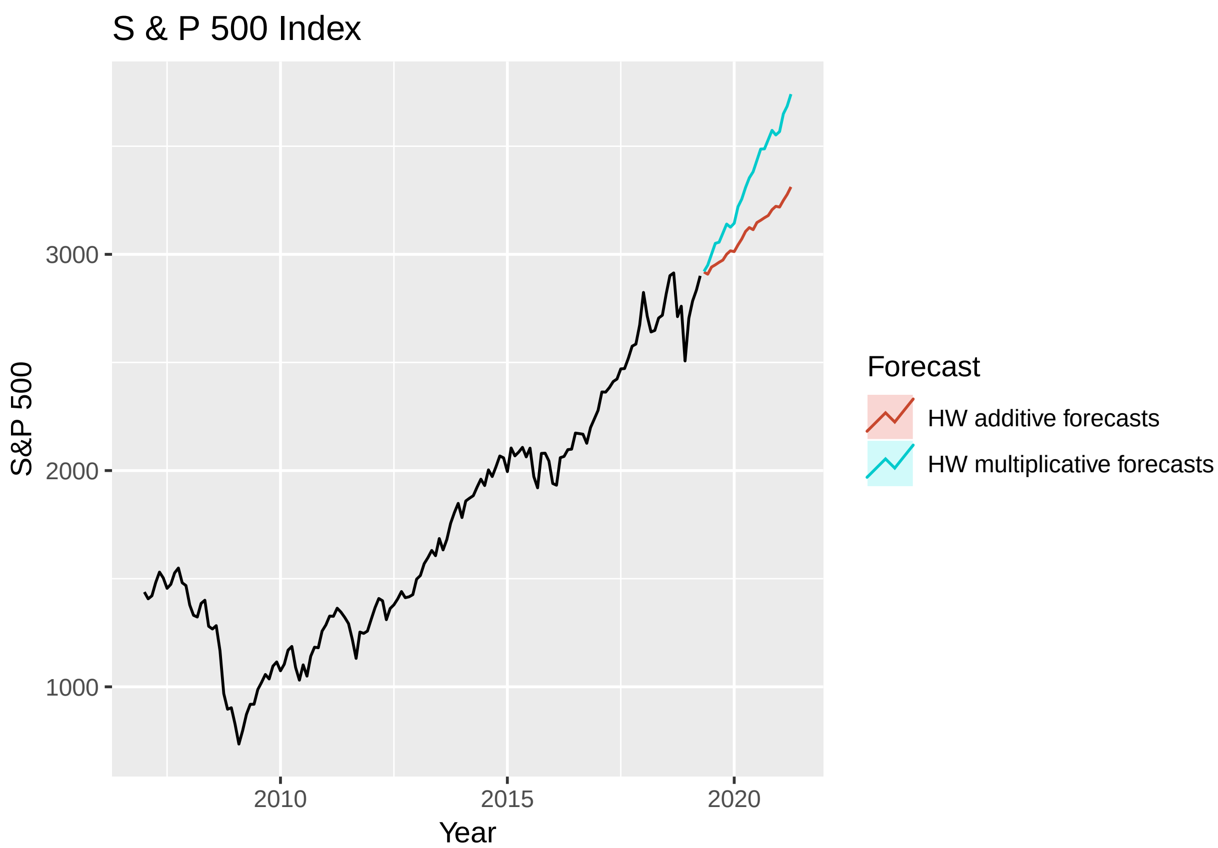
```
## Holt-Winters' additive method
##
```

```
## Call:
##  hw(y = sp, seasonal = "additive")
##
##   Smoothing parameters:
##     alpha = 0.985
##     beta  = 0.0281
##     gamma = 1e-04
##
##   Initial states:
##     l = 1618.0046
##     b = -9.7148
##     s = -1.5002 -0.3012 -10.0389 -3.0987 2.3033 9.0316
##            -6.8322 20.3497 19.3664 1.6813 -8.5065 -22.4547
##
##   sigma:  70.2798
##
##      AIC     AICc      BIC
## 2015.390 2020.098 2066.343
```

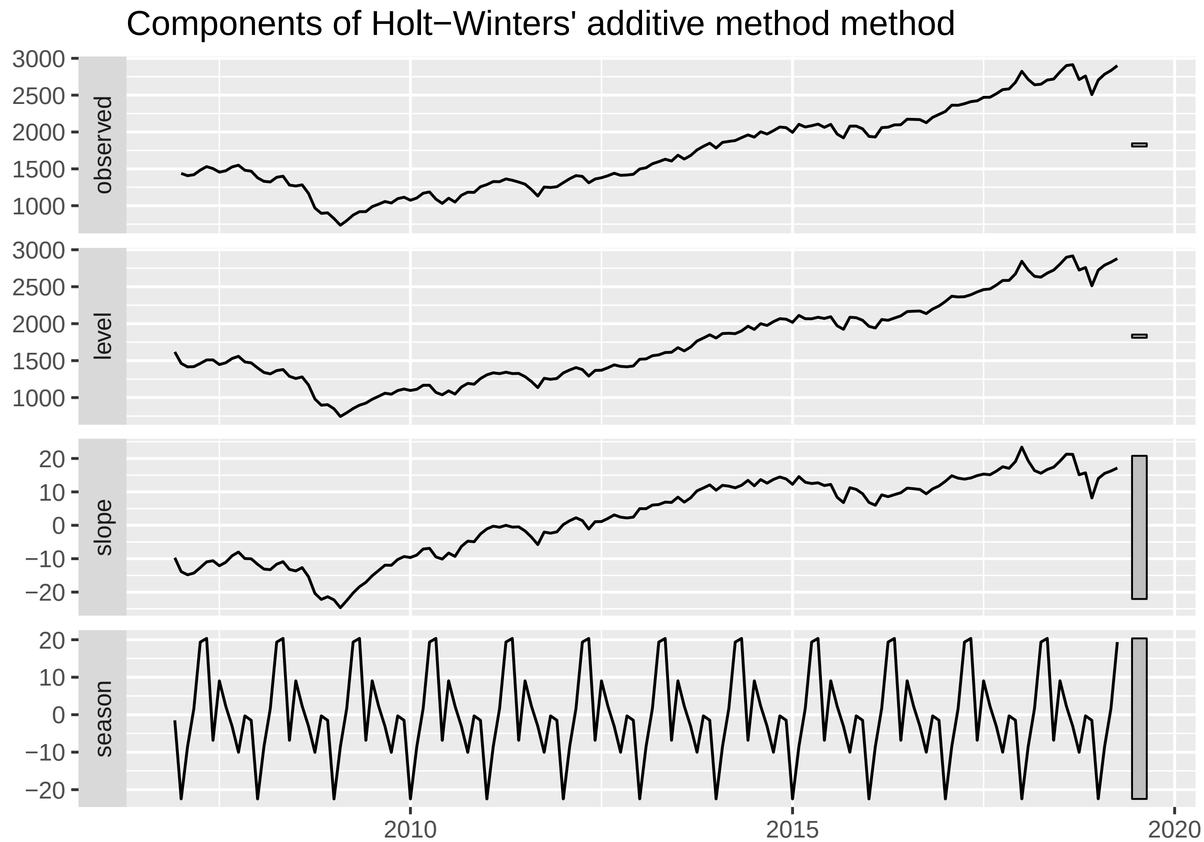
```r
hwfc1[["model"]]
```

```
## Holt-Winters' additive method
##
## Call:
##  hw(y = sp, seasonal = "additive")
##
##   Smoothing parameters:
##     alpha = 0.985
##     beta  = 0.0281
##     gamma = 1e-04
##
##   Initial states:
##     l = 1618.0046
##     b = -9.7148
##     s = -1.5002 -0.3012 -10.0389 -3.0987 2.3033 9.0316
##            -6.8322 20.3497 19.3664 1.6813 -8.5065 -22.4547
##
##   sigma:  70.2798
##
##      AIC     AICc      BIC
## 2015.390 2020.098 2066.343
```

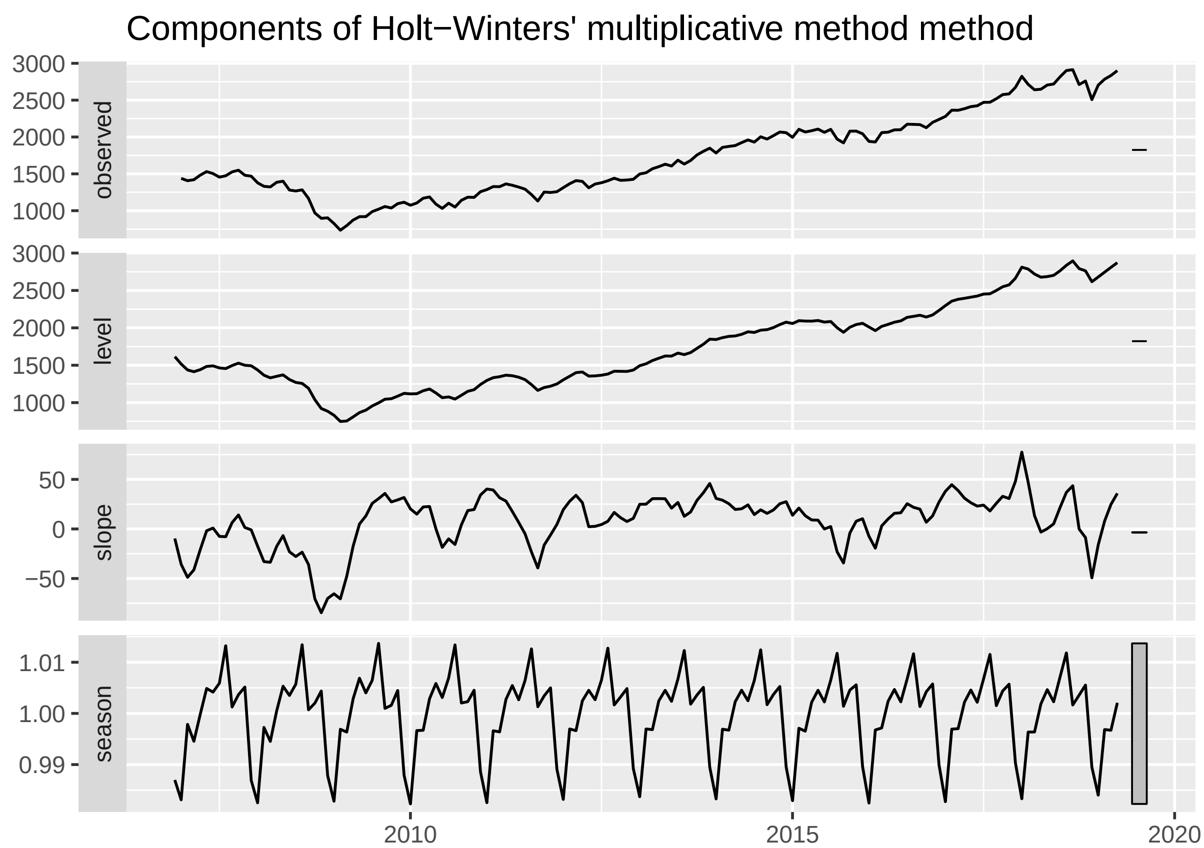
```r
autoplot(sp) +
  autolayer(hwfc1, series="HW additive forecasts", PI=FALSE) +
  autolayer(hwfc2, series="HW multiplicative forecasts",
            PI=FALSE) +
  xlab("Year") + ylab("S&P 500") +
  ggtitle("S & P 500 Index") +
  guides(colour=guide_legend(title="Forecast"))
```




```r
autoplot(hwfc1[['model']])
```




```r
autoplot(hwfc2[['model']])
```



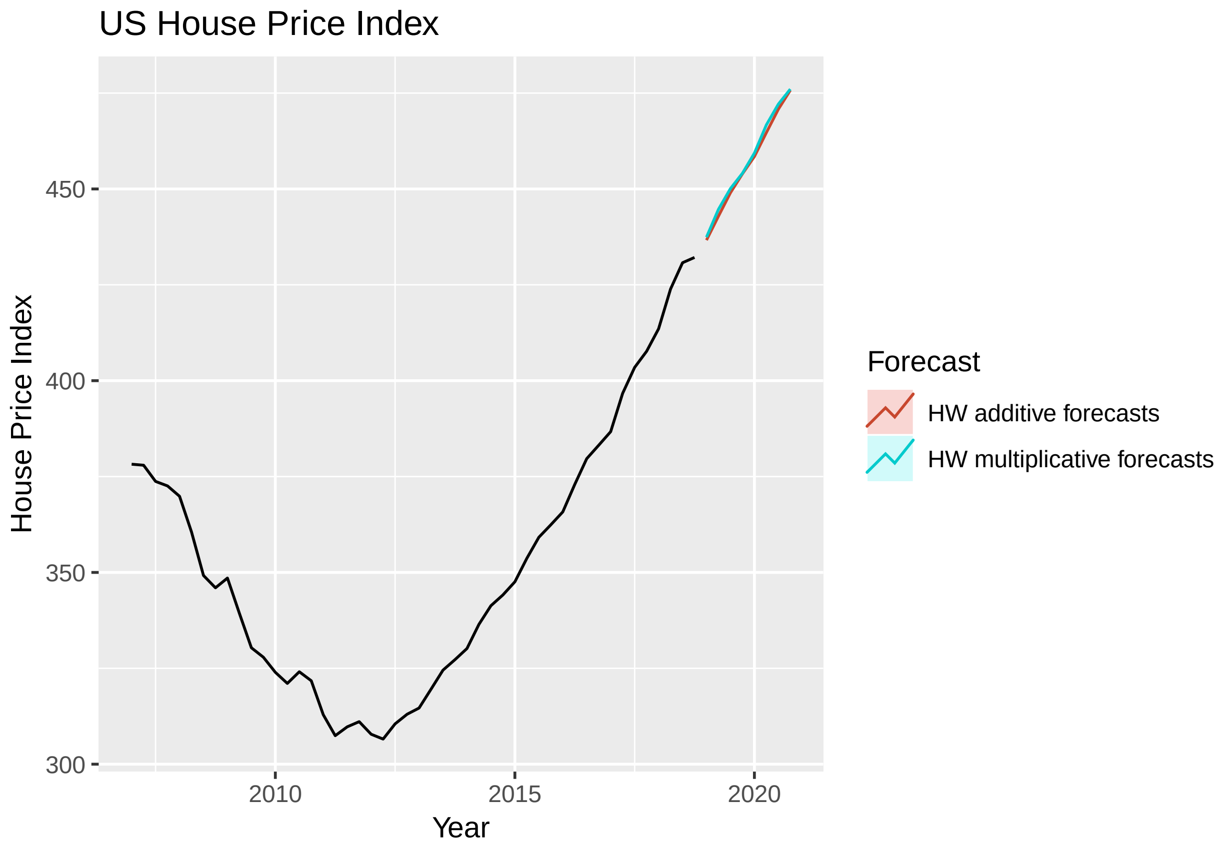

```r
hp <- window(hpi, start=2007)
hwfc1 <- hw(hp, seasonal="additive")
hwfc2 <- hw(hp, seasonal="multiplicative")
hwfc1[["model"]]
```

```
## Holt-Winters' additive method
##
## Call:
##  hw(y = hp, seasonal = "additive")
##
##   Smoothing parameters:
##     alpha = 0.9936
##     beta  = 0.2603
##     gamma = 0.0064
##
##   Initial states:
##     l = 387.6636
##     b = -2.299
##     s = 0.1186 0.6442 0.0213 -0.7841
##
##   sigma:  4.0088
##
##      AIC     AICc      BIC
## 328.3618 333.0986 345.2026
```

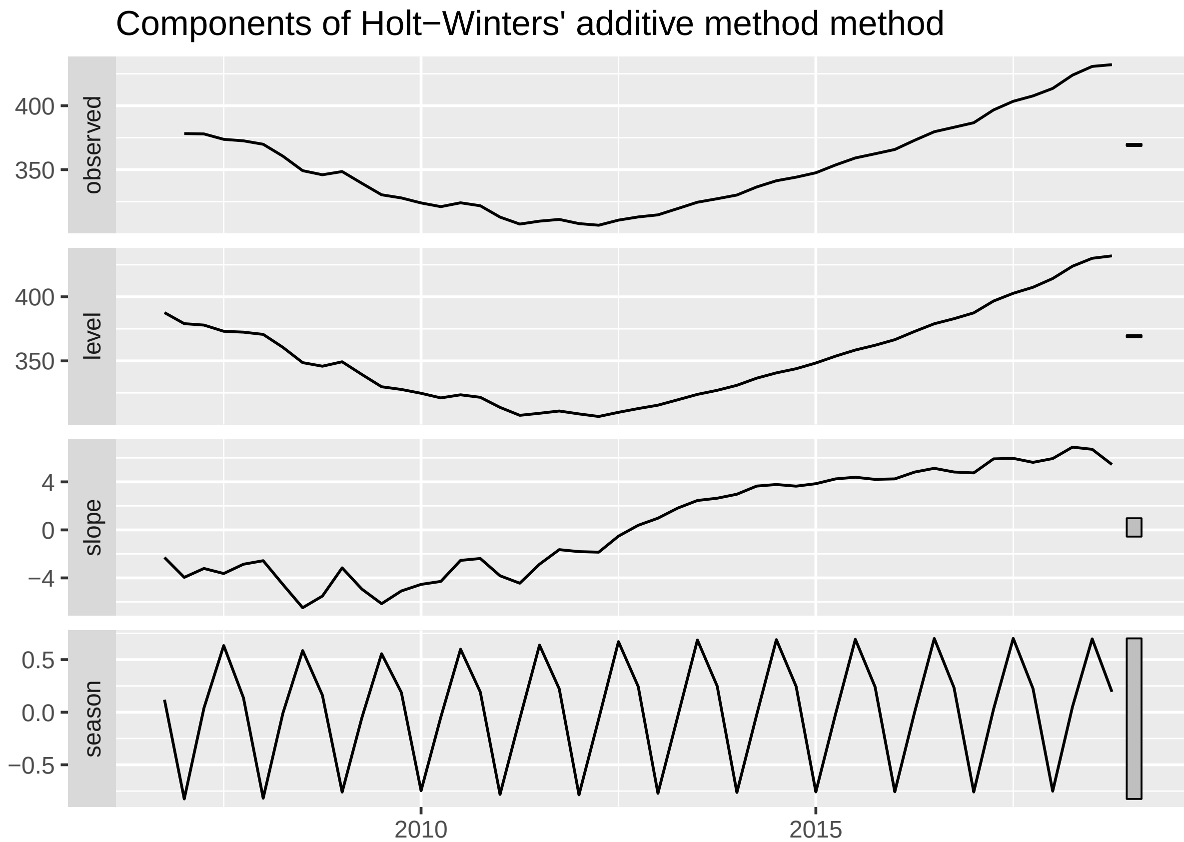
```r
hwfc1[["model"]]
```

```
## Holt-Winters' additive method
##
## Call:
##  hw(y = hp, seasonal = "additive")
##
##   Smoothing parameters:
##     alpha = 0.9936
##     beta  = 0.2603
##     gamma = 0.0064
##
##   Initial states:
##     l = 387.6636
##     b = -2.299
##     s = 0.1186 0.6442 0.0213 -0.7841
##
##   sigma:  4.0088
##
##      AIC     AICc      BIC
## 328.3618 333.0986 345.2026
```

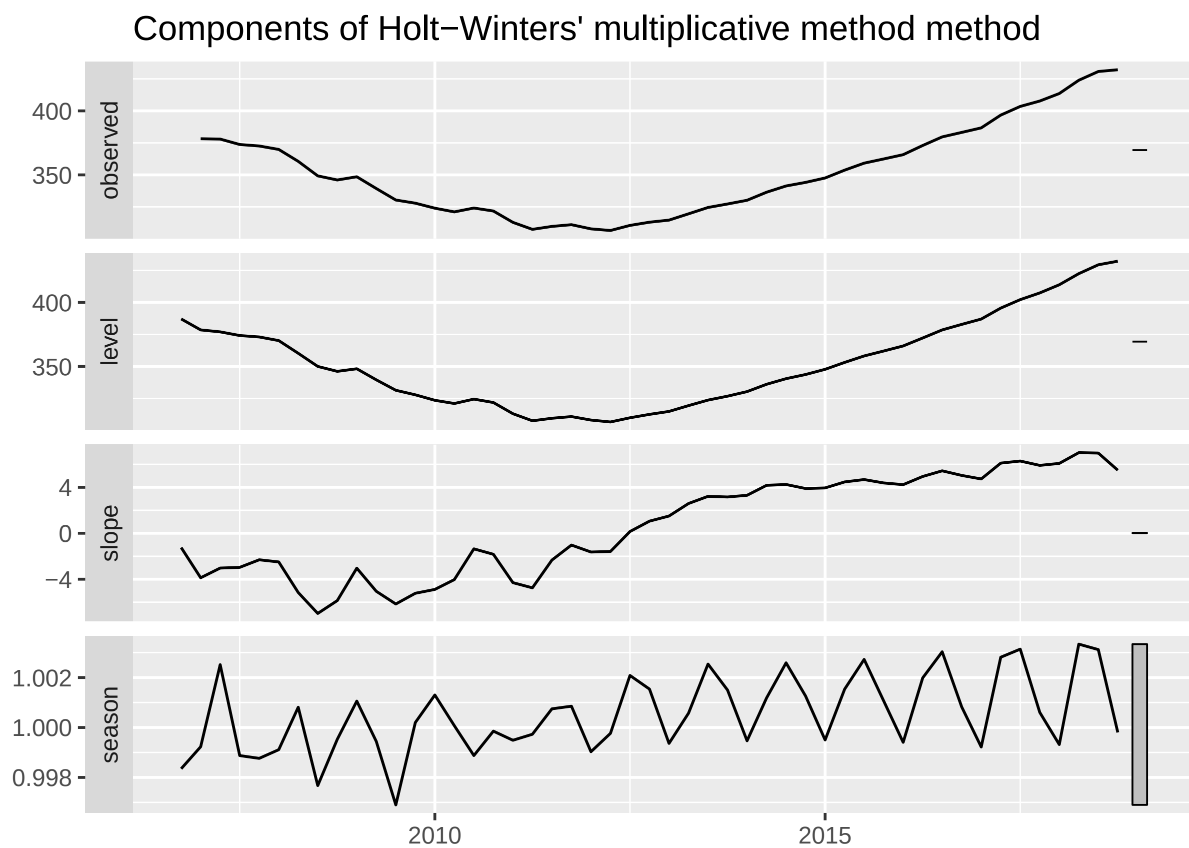
```r
autoplot(hp) +
  autolayer(hwfc1, series="HW additive forecasts", PI=FALSE) +
  autolayer(hwfc2, series="HW multiplicative forecasts",
            PI=FALSE) +
  xlab("Year") + ylab("House Price Index") +
  ggtitle("US House Price Index") +
  guides(colour=guide_legend(title="Forecast"))
```




```
autoplot(hwfc1[['model']])
```




```r
autoplot(hwfc2[['model']])
```



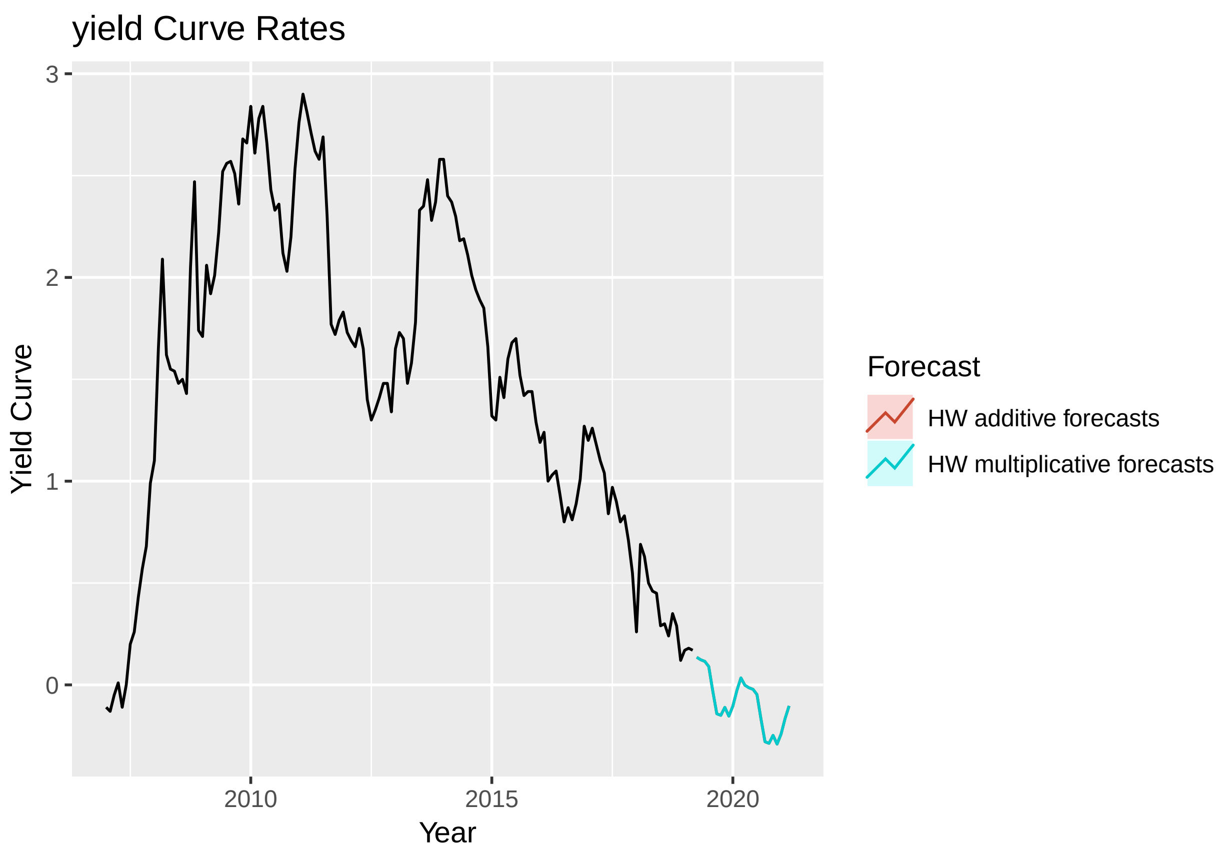

```
yi <- window(yield, start=2007)
hwfc1 <- hw(yi, seasonal="additive")
hwfc2 <- hw(yi )
hwfc1[["model"]]
```

```
## Holt-Winters' additive method
##
## Call:
##  hw(y = yi, seasonal = "additive")
##
##   Smoothing parameters:
##     alpha = 0.9777
##     beta  = 0.0104
##     gamma = 1e-04
##
##   Initial states:
##     l = -0.4575
##     b = -0.0072
##     s = -0.1065 -0.075 -0.1258 -0.1293 -0.0296 0.0797
##            0.0939 0.09 0.091 0.1156 0.042 -0.0459
##
##   sigma:  0.2025
##
##      AIC     AICc      BIC
## 281.0695 285.8137 331.9068
```

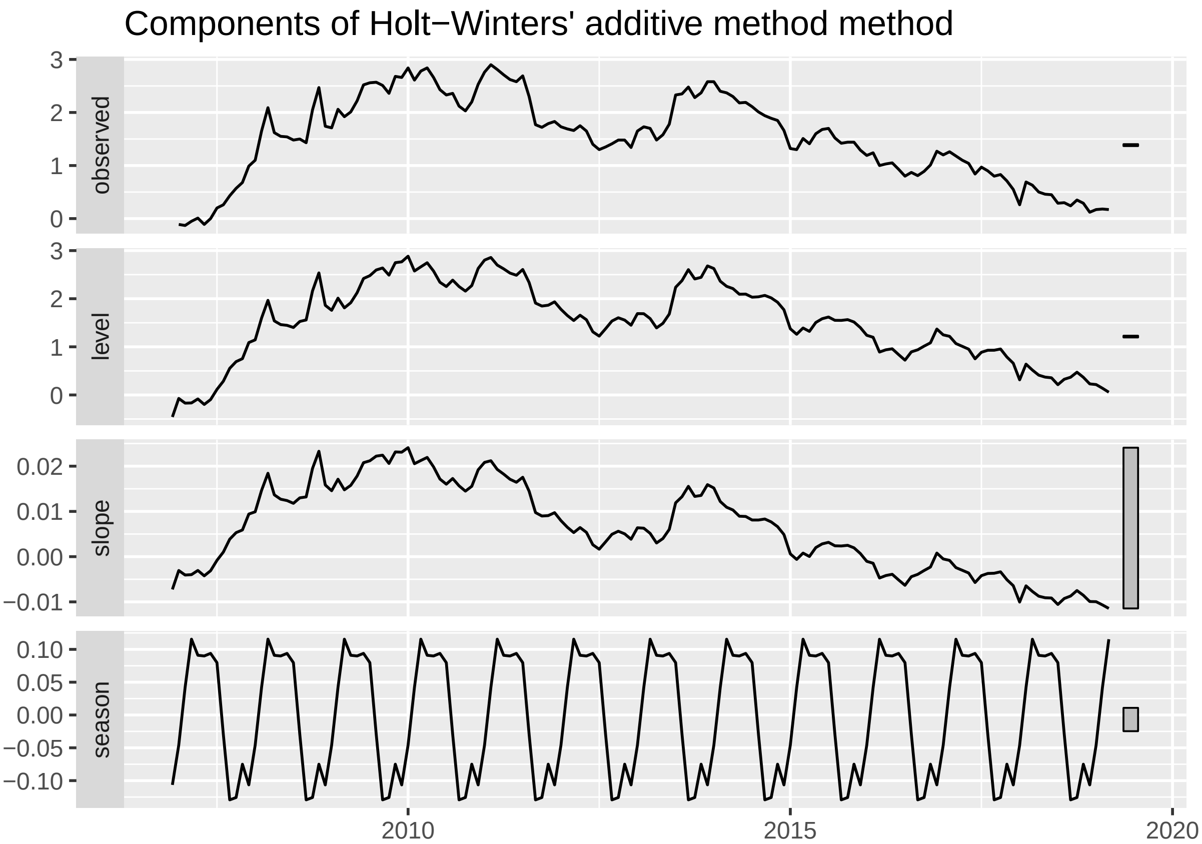
```r
hwfc1[["model"]]
```

```
## Holt-Winters' additive method
##
## Call:
##  hw(y = yi, seasonal = "additive")
##
##   Smoothing parameters:
##     alpha = 0.9777
##     beta  = 0.0104
##     gamma = 1e-04
##
##   Initial states:
##     l = -0.4575
##     b = -0.0072
##     s = -0.1065 -0.075 -0.1258 -0.1293 -0.0296 0.0797
##            0.0939 0.09 0.091 0.1156 0.042 -0.0459
##
##   sigma:  0.2025
##
##      AIC     AICc      BIC
## 281.0695 285.8137 331.9068
```

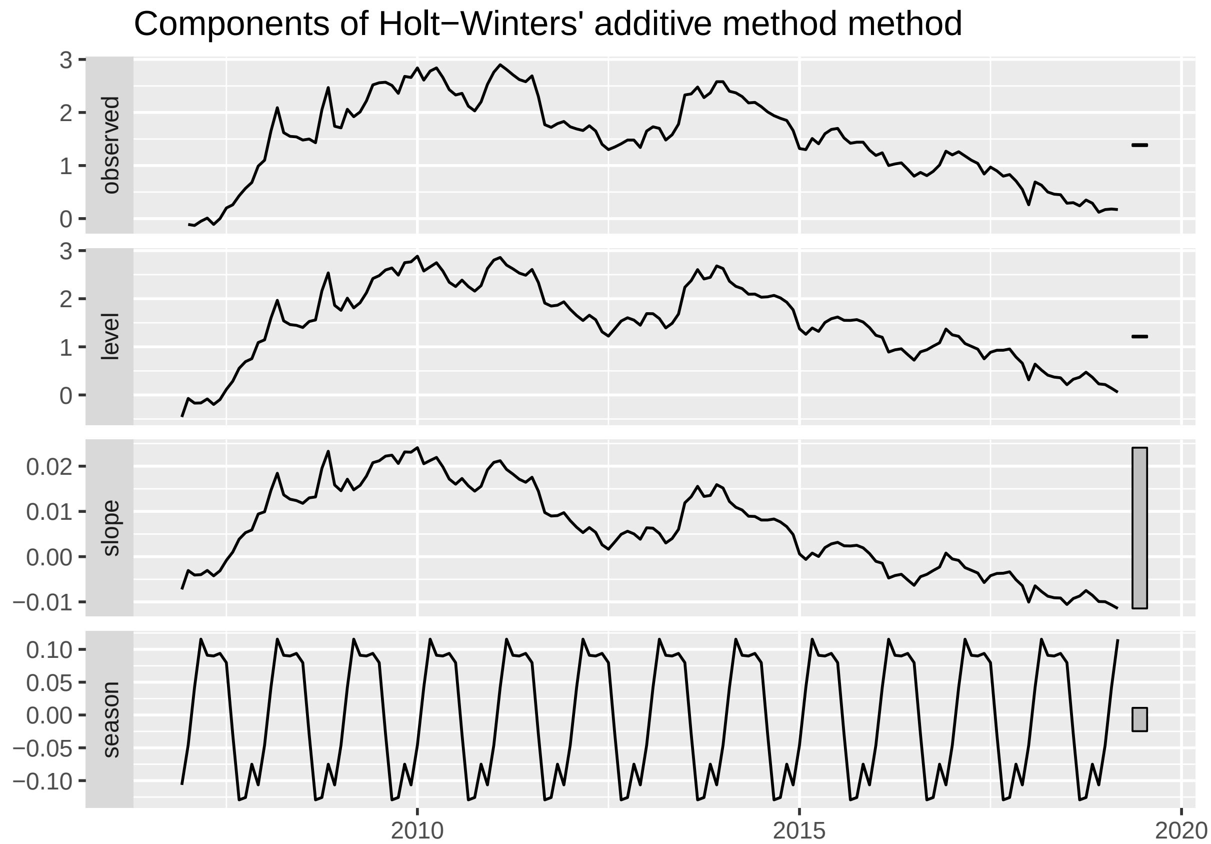
```r
autoplot(yi) +
  autolayer(hwfc1, series="HW additive forecasts", PI=FALSE) +
  autolayer(hwfc2, series="HW multiplicative forecasts",
            PI=FALSE) +
  xlab("Year") + ylab("Yield Curve") +
  ggtitle("yield Curve Rates") +
  guides(colour=guide_legend(title="Forecast"))
```




```r
autoplot(hwfc1[['model']])
```




```r
autoplot(hwfc2[['model']])
```



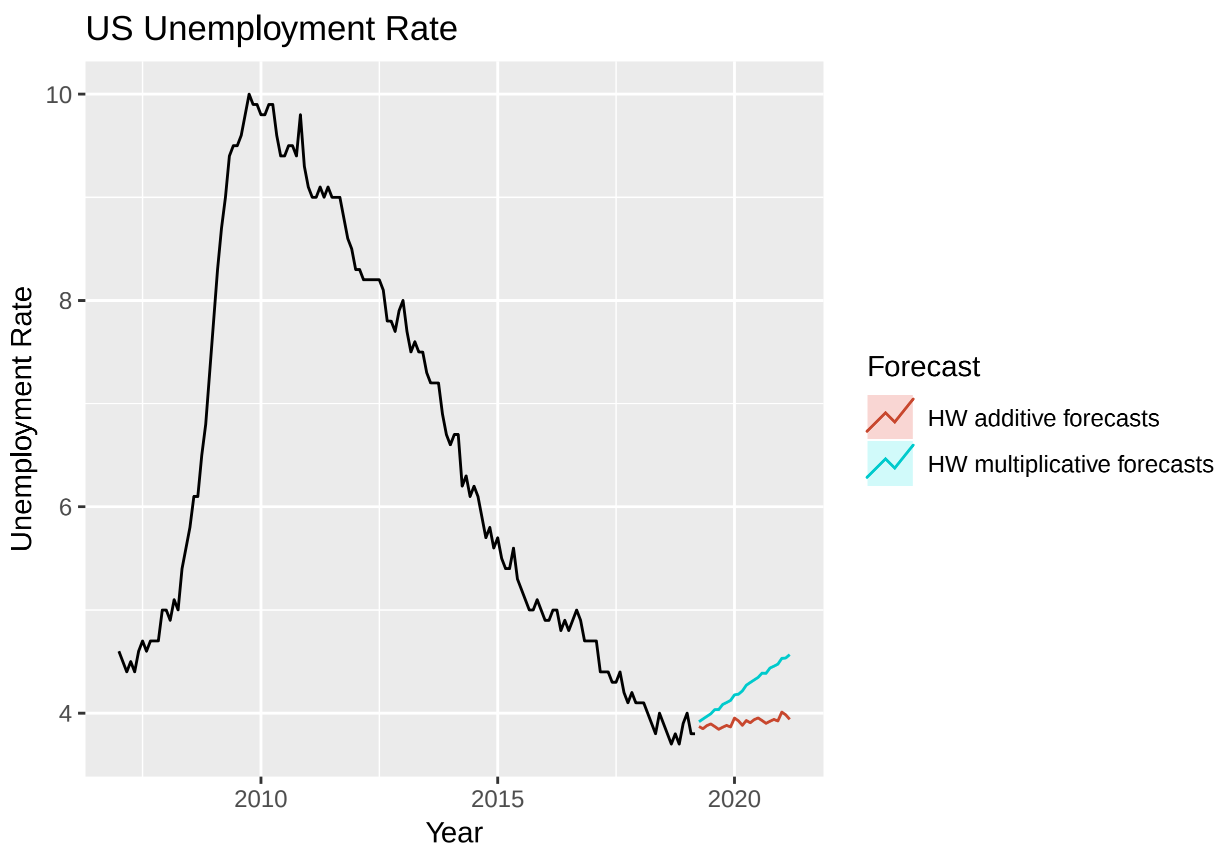

```r
un <- window(unemp, start=2007)
hwfc1 <- hw(un, seasonal="additive")
hwfc2 <- hw(un, seasonal="multiplicative")
hwfc1[["model"]]
```

```
## Holt-Winters' additive method
##
## Call:
##  hw(y = un, seasonal = "additive")
##
##   Smoothing parameters:
##     alpha = 0.4392
##     beta  = 0.2017
##     gamma = 0.0288
##
##   Initial states:
##     l = 4.1352
##     b = -0.1449
##     s = -0.033 0.0029 -0.01 -0.032 -0.0124 0.0425
##            0.0165 -0.0279 0.0227 -0.0522 0.0185 0.0643
##
##   sigma:  0.195
##
##      AIC     AICc      BIC
## 270.0374 274.7816 320.8748
```

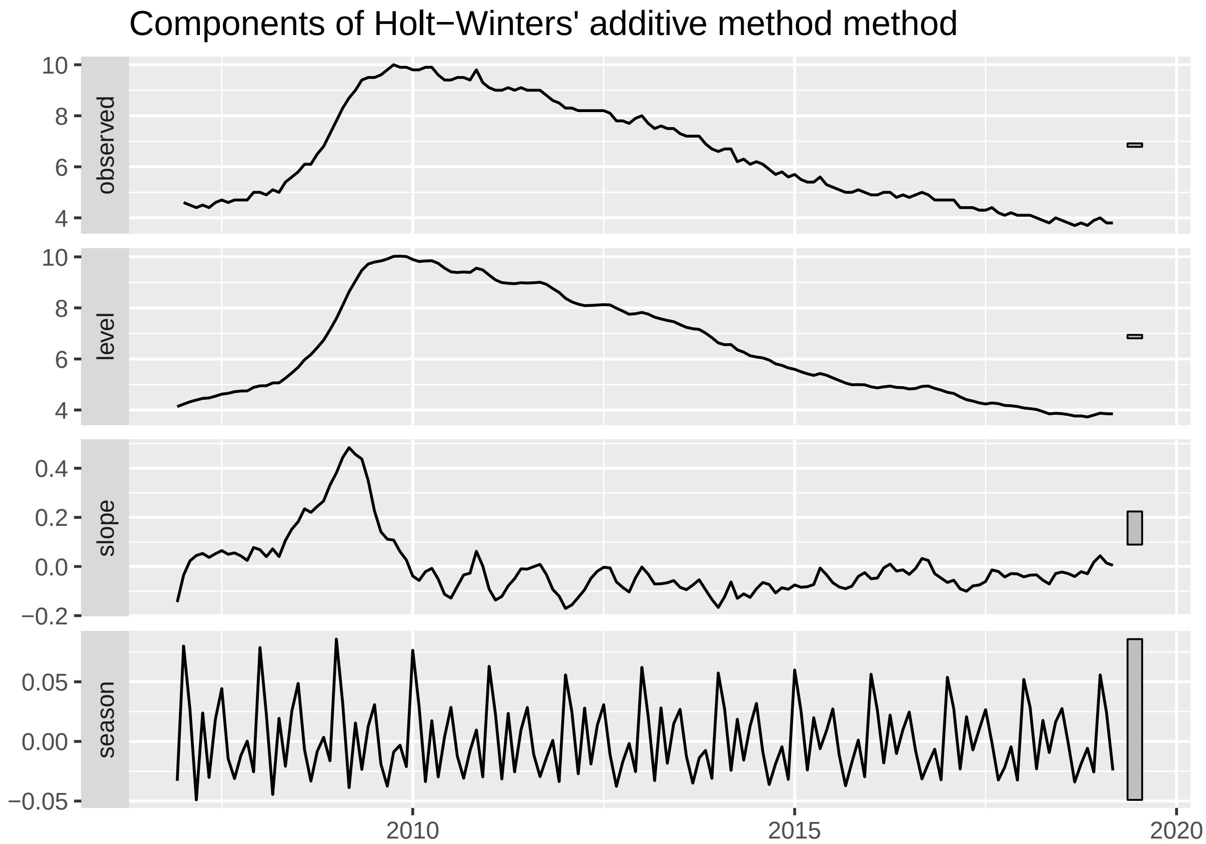
```r
hwfc1[["model"]]
```

```
## Holt-Winters' additive method
##
## Call:
##  hw(y = un, seasonal = "additive")
##
##   Smoothing parameters:
##     alpha = 0.4392
##     beta  = 0.2017
##     gamma = 0.0288
##
##   Initial states:
##     l = 4.1352
##     b = -0.1449
##     s = -0.033 0.0029 -0.01 -0.032 -0.0124 0.0425
##            0.0165 -0.0279 0.0227 -0.0522 0.0185 0.0643
##
##   sigma:  0.195
##
##      AIC     AICc      BIC
## 270.0374 274.7816 320.8748
```

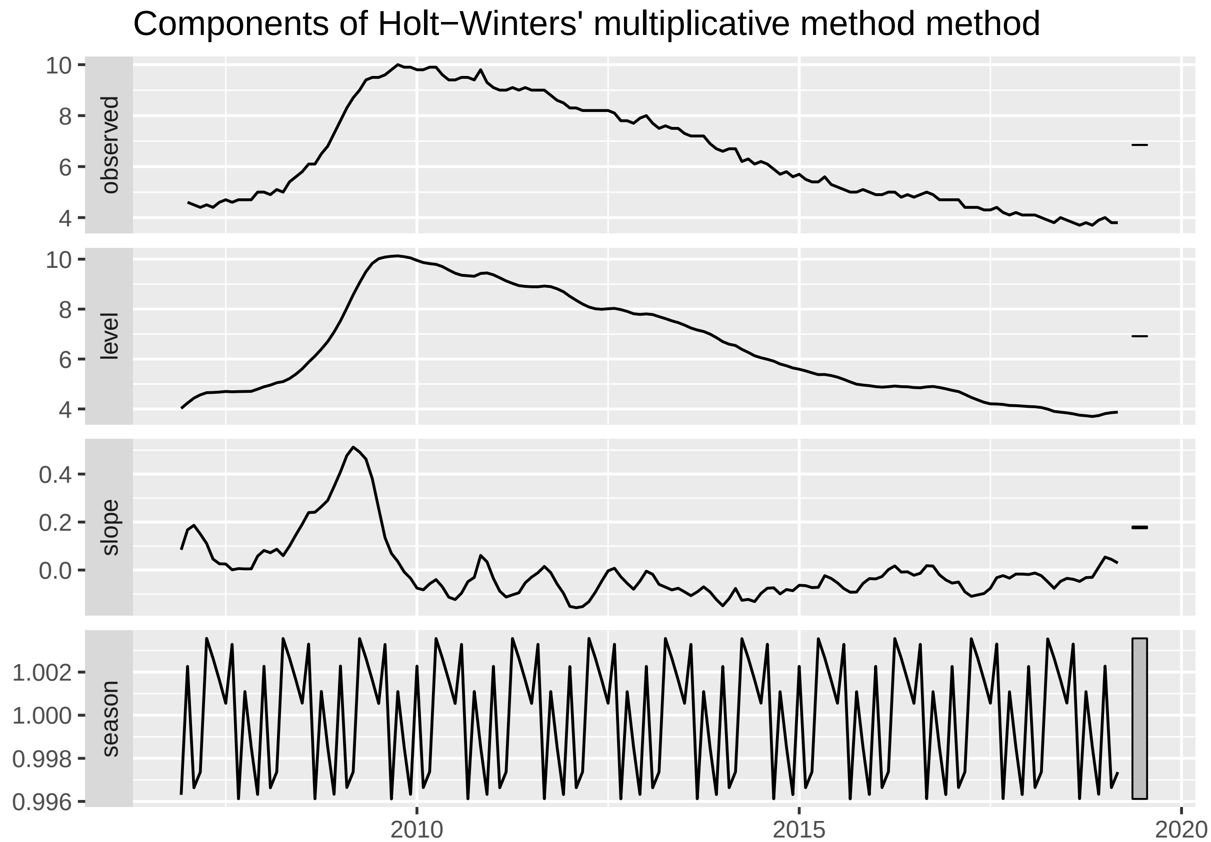
```r
autoplot(un) +
  autolayer(hwfc1, series="HW additive forecasts", PI=FALSE) +
  autolayer(hwfc2, series="HW multiplicative forecasts",
            PI=FALSE) +
  xlab("Year") + ylab("Unemployment Rate") +
  ggtitle("US Unemployment Rate") +
  guides(colour=guide_legend(title="Forecast"))
```




```
autoplot(hwfc1[['model']])
```




```r
autoplot(hwfc2[['model']])
```



ETS Models

These are reffered to as State space model.Each model consists of a measurement equation that describes the observed data, and some state equations that describe how the unobserved components or states (Error, trend, seasonal) change over time.It selects model looing into AIC values.

S&P 500:It's an additive error model (A,N,N).This predicts no change in sttock price with time.

Unemployment Rate: It's an multiplicative error with additive trend (M,A,N).Unemployment prediction shows a downward trend.

House Price Index: It's an Additive trend and Additive erroer model (A,A,N). It predicts US house price Index to grow as we move forward.

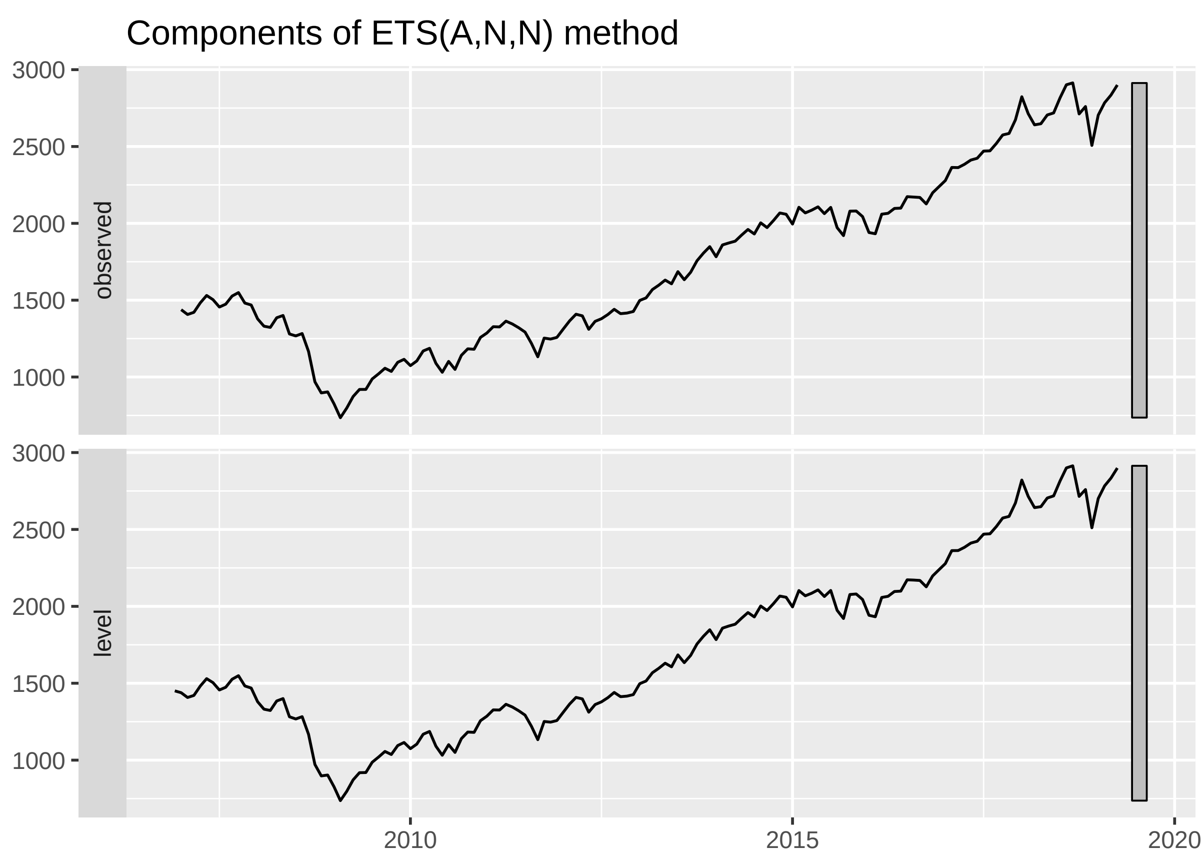
Yield Curve:It shows an Additive erroe with no trend and seasonality(A,N,N).It shows a constant yield curve prediction.

```r
sp <- window(sp_500, start=2007)
fit <- ets(sp)
summary(fit)
```

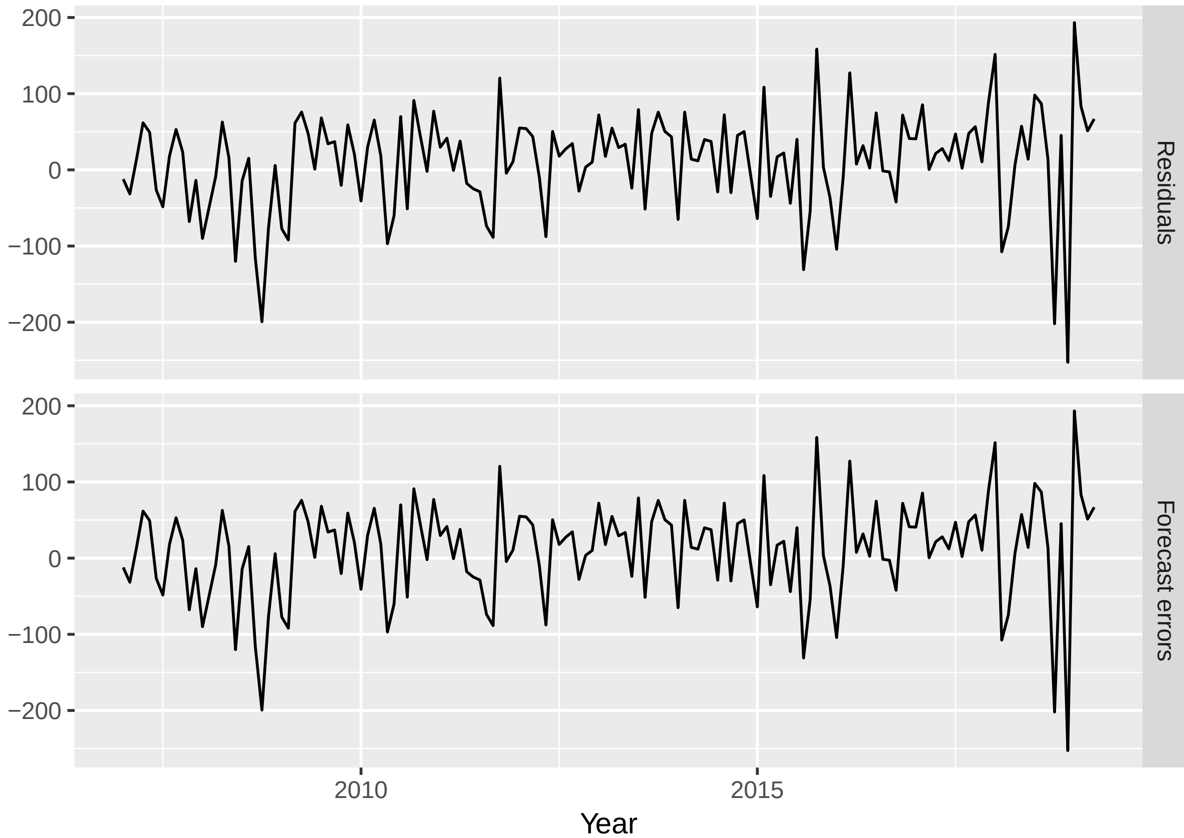
```
## ETS(A,N,N)
##
## Call:
##  ets(y = sp)
##
```

```
##   Smoothing parameters:
##     alpha = 0.9844
##
##   Initial states:
##     l = 1450.427
##
##   sigma:  67.1058
##
##      AIC     AICc      BIC
## 1988.630 1988.796 1997.621
##
## Training set error measures:
##                    ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 9.945782 66.65083 51.14945 0.3798462 3.258563 0.2274907
##                   ACF1
## Training set 0.0117453
```

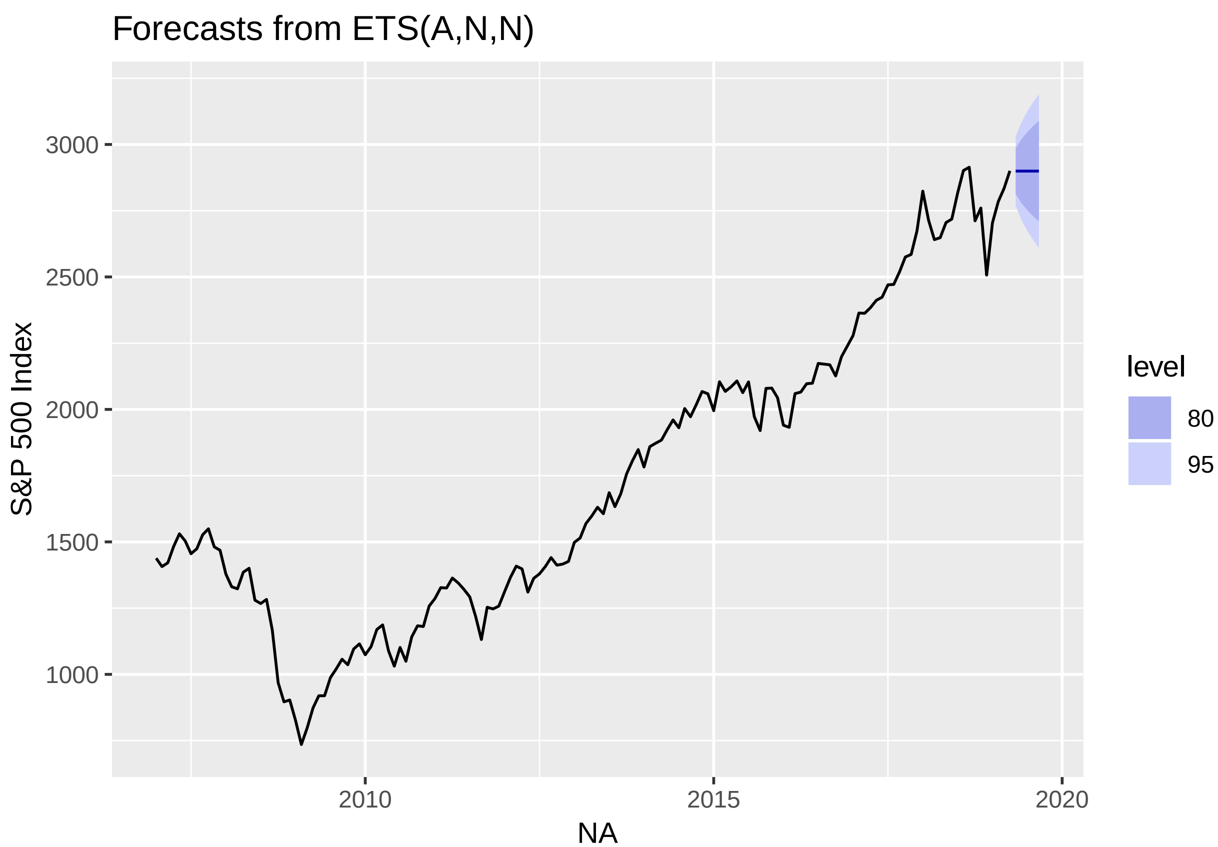
```
autoplot(fit)
```




```
cbind('Residuals' = residuals(fit), 'Forecast errors' = residuals(fit,type='response')) %>%
  autoplot(facet=TRUE) +
  xlab("Year") + ylab("")
```




```
### Forecasts with ETS Models
fit %>% forecast(h=5) %>%
  autoplot() + ylab("S&P 500 Index")
```

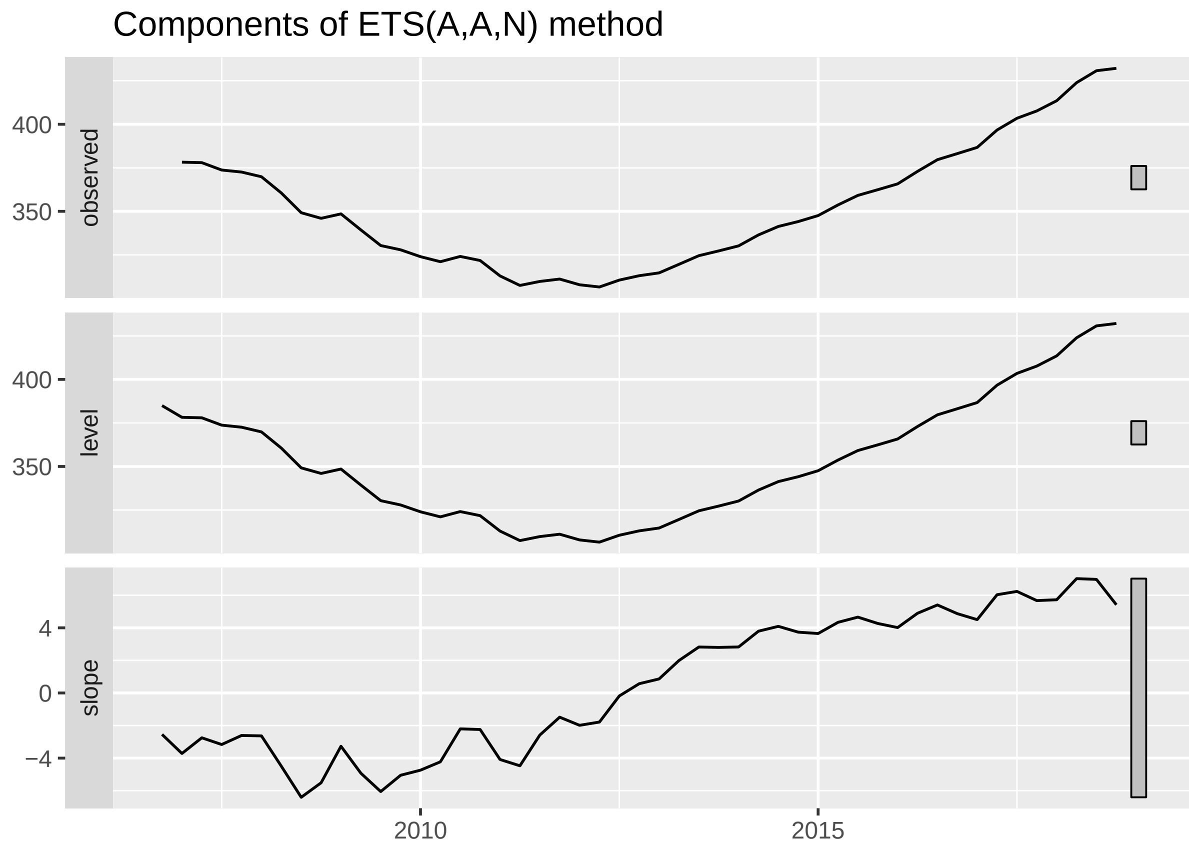



```r
hp <- window(hpi, start=2007)
fit <- ets(hp)
summary(fit)
```

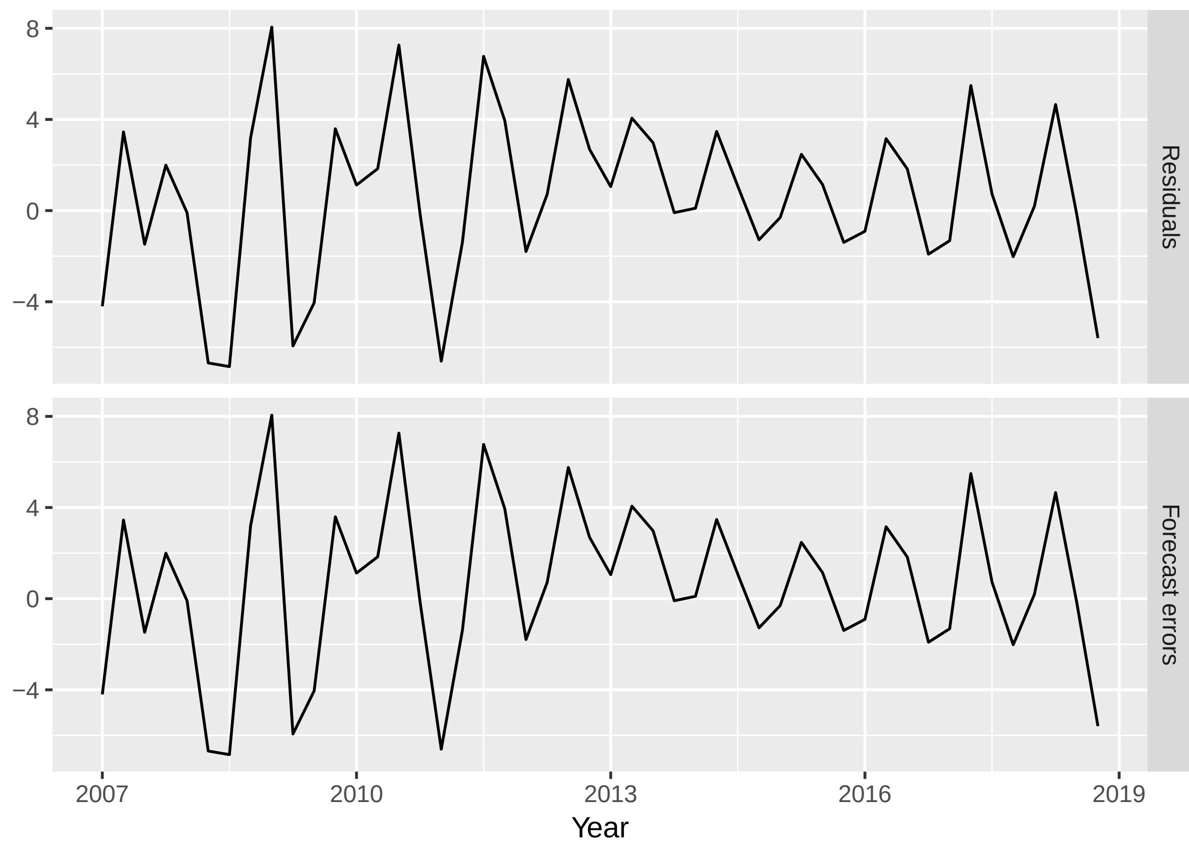
```
## ETS(A,A,N)
##
## Call:
##  ets(y = hp)
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 0.2786
##
##   Initial states:
##     l = 384.9641
##     b = -2.5437
##
##   sigma:  3.7945
##
##      AIC     AICc      BIC
## 319.6615 321.0900 329.0175
##
## Training set error measures:
##                     ME     RMSE      MAE       MPE      MAPE      MASE
## Training set 0.5951719 3.632931 2.854276 0.1872739 0.8248899 0.1714584
##                     ACF1
```

```
## Training set 0.07249745
```

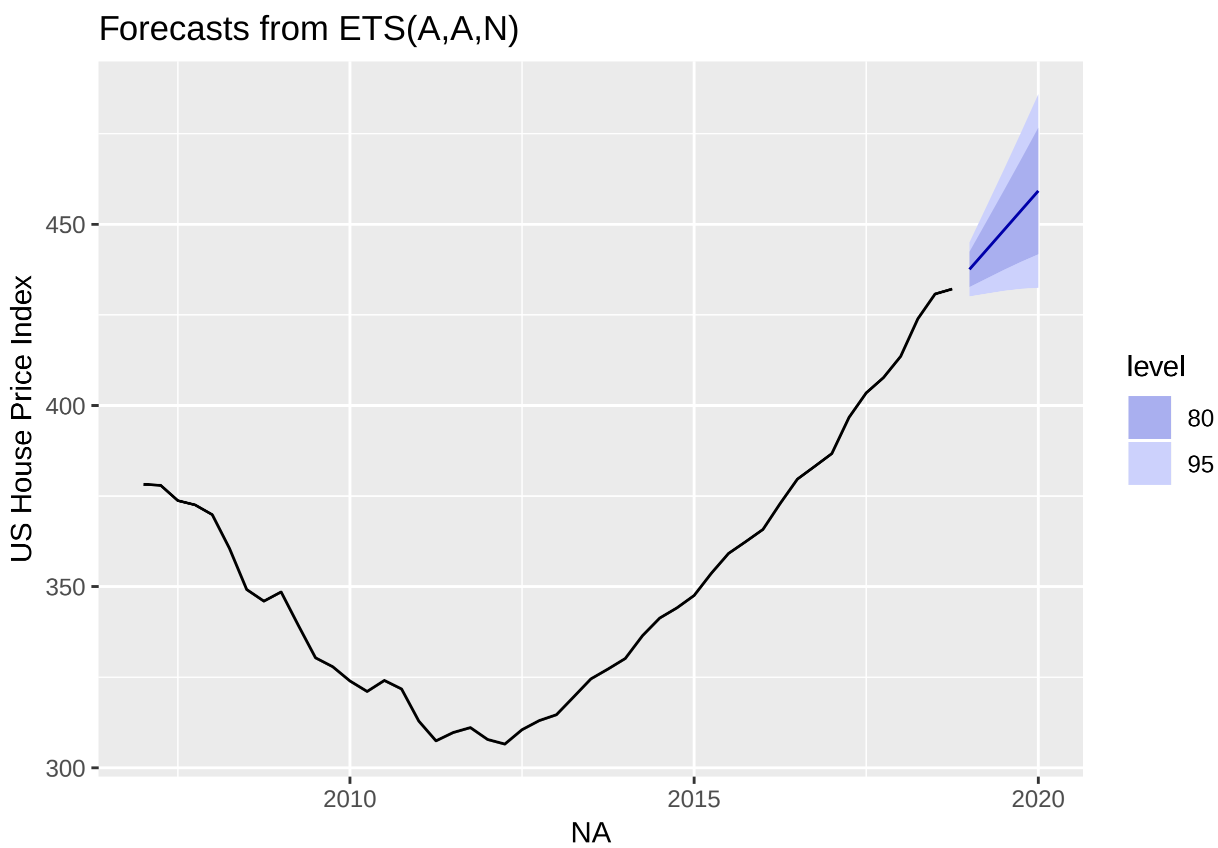
```r
autoplot(fit)
```




```r
cbind('Residuals' = residuals(fit), 'Forecast errors' = residuals(fit,type='response')) %>%
  autoplot(facet=TRUE) +
  xlab("Year") + ylab("")
```




```
### Forecasts with ETS Models
fit %>% forecast(h=5) %>%
  autoplot() + ylab("US House Price Index")
```

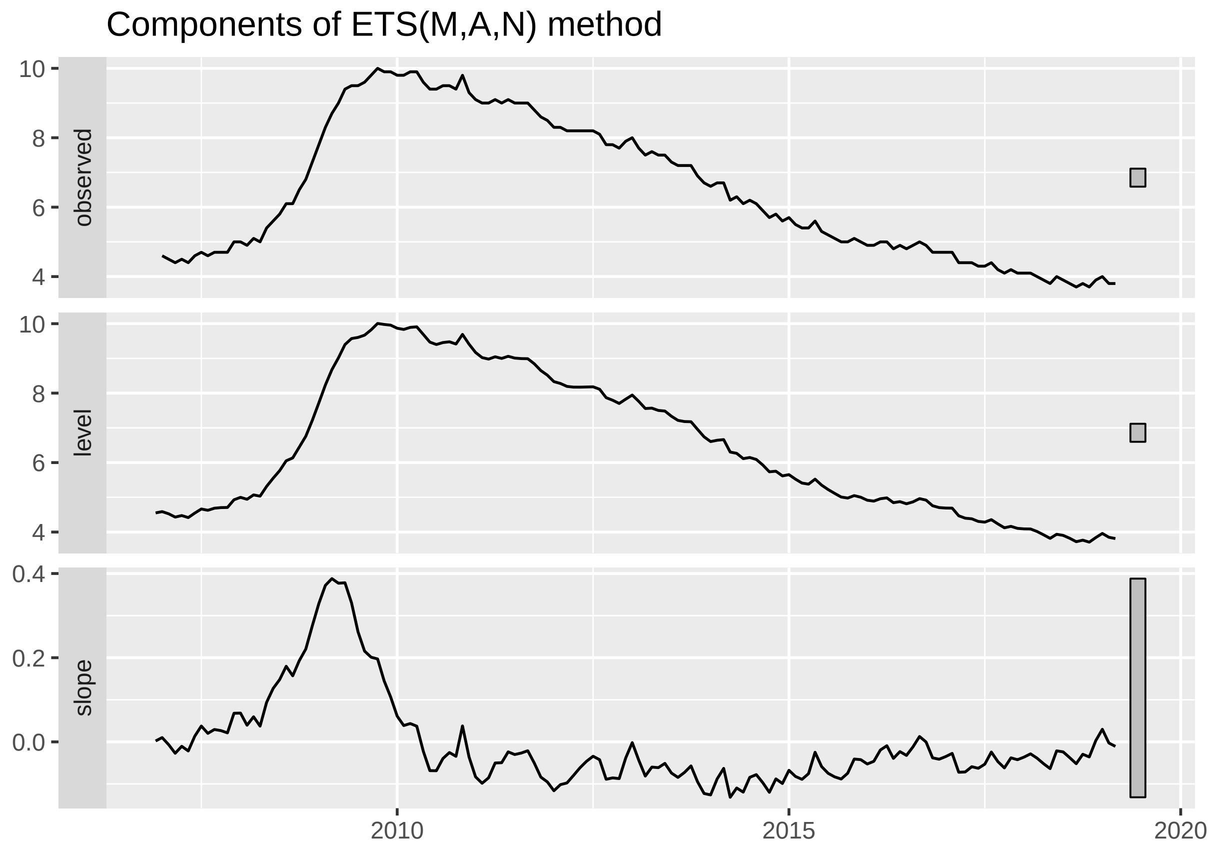



```r
un <- window(unemp, start=2007)
fit <- ets(un)
summary(fit)
```

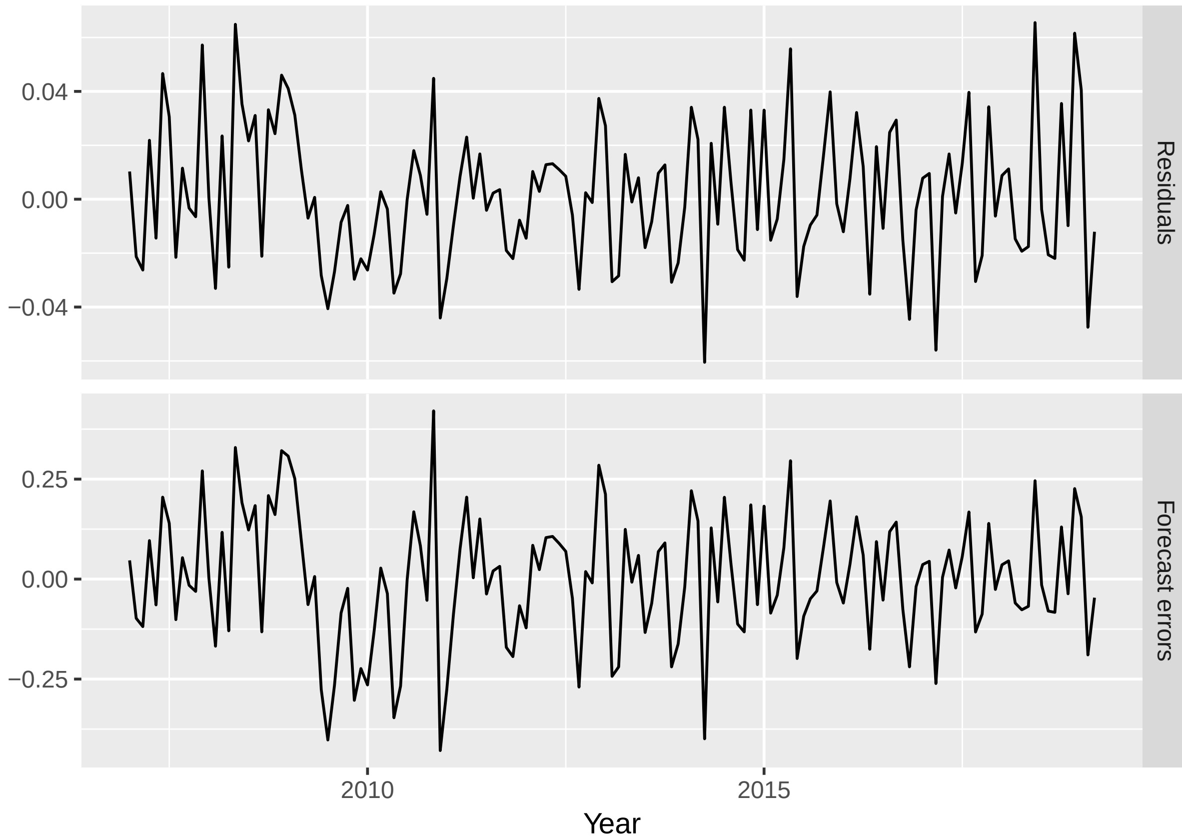
```
## ETS(M,A,N)
##
## Call:
##  ets(y = un)
##
##   Smoothing parameters:
##     alpha = 0.7405
##     beta  = 0.172
##
##   Initial states:
##     l = 4.5511
##     b = 0.0021
##
##   sigma:  0.0259
##
##      AIC     AICc      BIC
## 199.6142 200.0398 214.5664
##
## Training set error measures:
##                         ME      RMSE       MAE        MPE     MAPE      MASE
## Training set -0.00050698 0.1606949 0.1277779 0.08298968 2.055443 0.1297969
##                     ACF1
```

```
## Training set 0.1677853
```

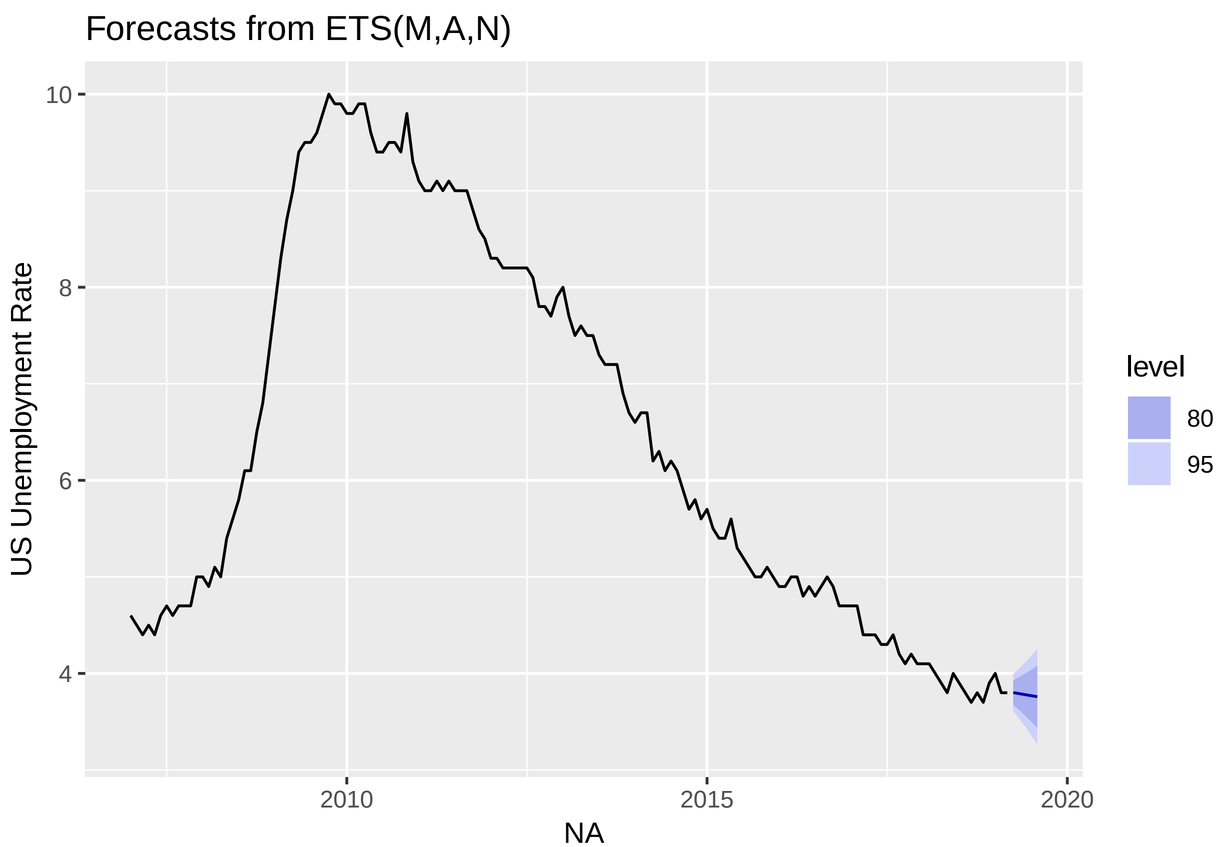
```r
autoplot(fit)
```




```r
cbind('Residuals' = residuals(fit), 'Forecast errors' = residuals(fit,type='response')) %>%
  autoplot(facet=TRUE) +
  xlab("Year") + ylab("")
```



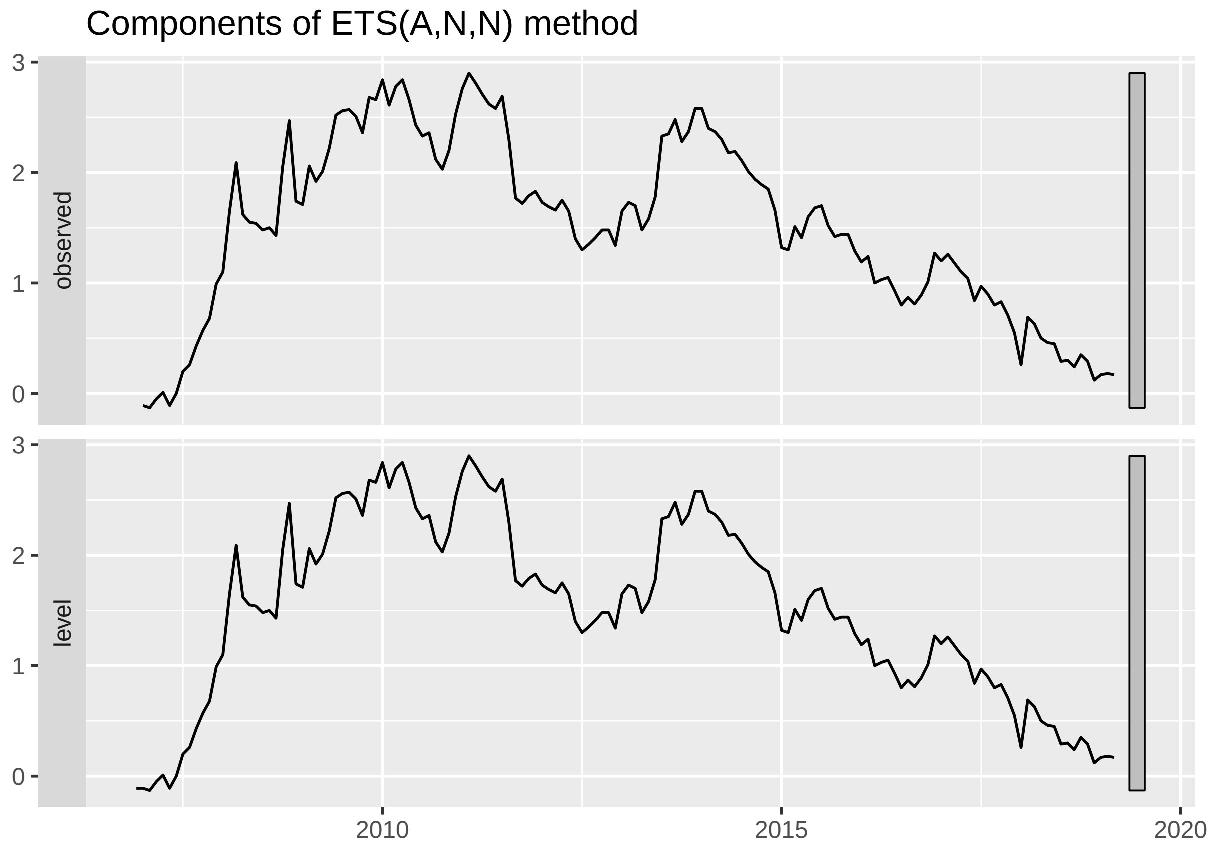

```
### Forecasts with ETS Models
fit %>% forecast(h=5) %>%
  autoplot() + ylab("US Unemployment Rate")
```



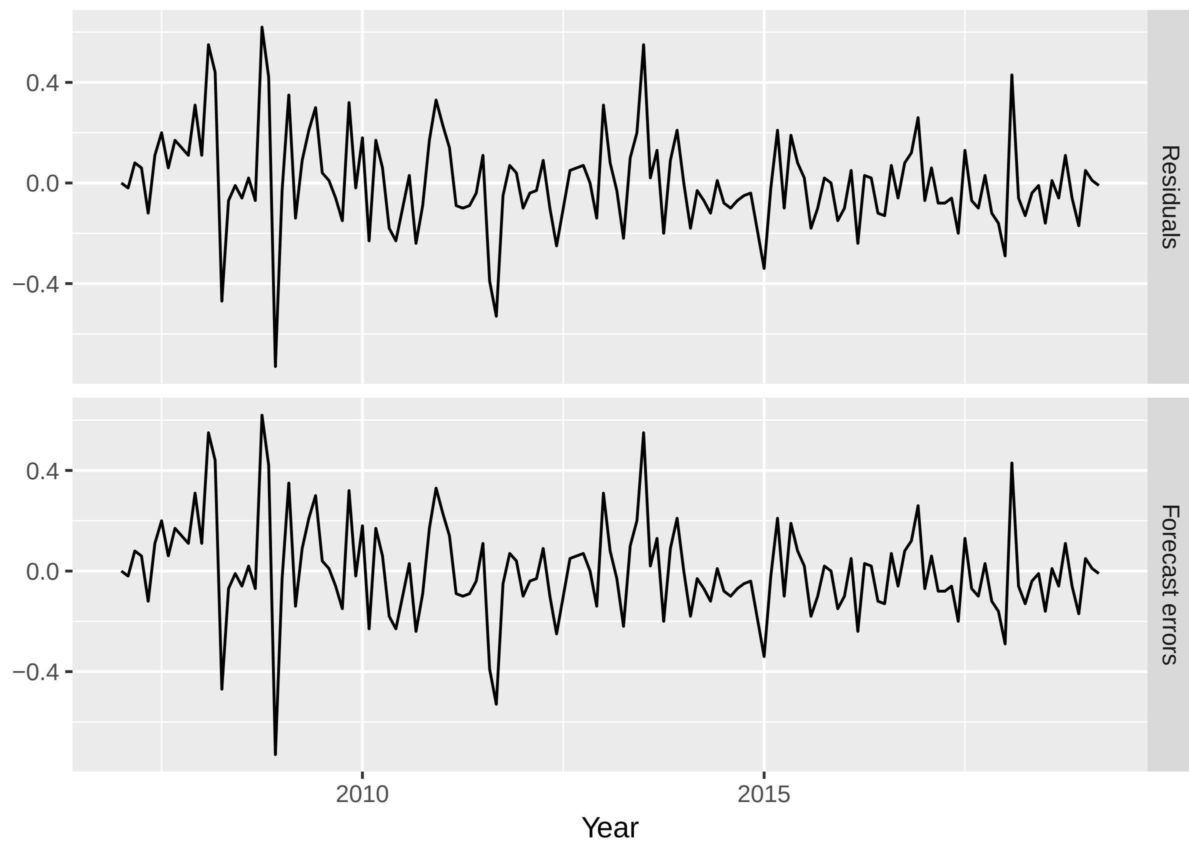

```r
yi <- window(yield, start=2007)
fit <- ets(yi)
summary(fit)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = yi)
##
##   Smoothing parameters:
##     alpha = 0.9999
##
##   Initial states:
##     l = -0.11
##
##   sigma:  0.1913
##
##      AIC     AICc      BIC
## 251.3394 251.5072 260.3107
##
## Training set error measures:
##                       ME      RMSE       MAE MPE MAPE      MASE     ACF1
## Training set 0.001904817 0.1900005 0.1368742 Inf  Inf 0.2279548 0.119087
```

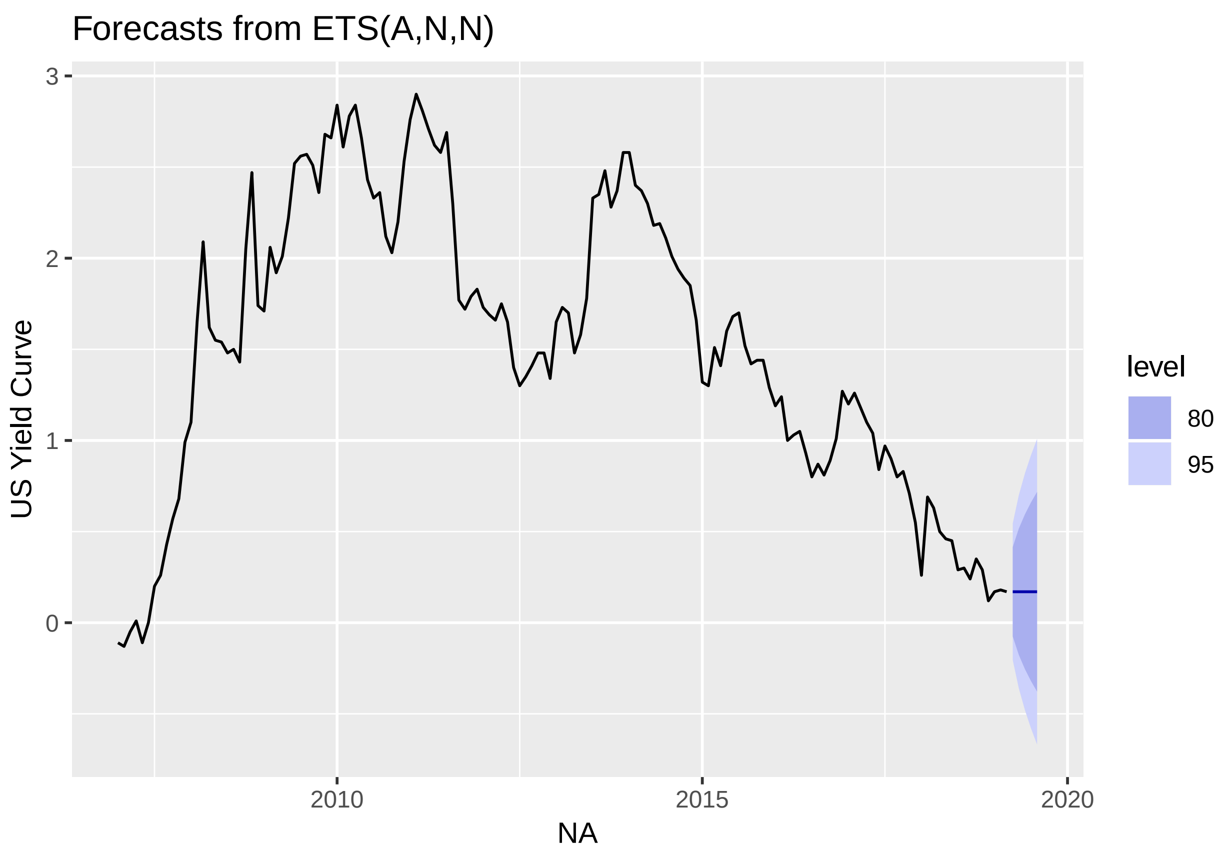
```r
autoplot(fit)
```




```
cbind('Residuals' = residuals(fit), 'Forecast errors' = residuals(fit,type='response')) %>%
  autoplot(facet=TRUE) +
  xlab("Year") + ylab("")
```




```
### Forecasts with ETS Models
fit %>% forecast(h=5) %>%
  autoplot() + ylab("US Yield Curve")
```



Forecasting Accuracy of ETS

We will see MAPE and MAE forecasting errors to predict our accuracy of the model.

We can see that ets model have high forecasting errors.

```
# Forecasting accuracy for ETS

sp500 <- window(sp_500, start = 2007, end = c(2017,12))
test1 <- window(sp_500, start = c(2018))

hpiu <- window(hpi, start = 2007, end = c(2016,4))
test2 <- window(hpi, start = 2017)

unempu <- window(unemp, start = 2007, end = c(2017,12))
test3 <- window(unemp, start = c(2018))

yieldu <- window(yield, start = 2007, end = c(2017,12))
test4 <- window(yield, start = c(2018))


fit1 <- ets(sp500)
train1 <- forecast(fit1, h=16)
accuracy(train1, test1)
```

```
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set  9.364321 57.51689 45.70232 0.3766103 3.215566 0.2087850
```

```
## Test set     81.660703 133.76733 109.78987 2.8175821 3.924598 0.5015605
##                    ACF1 Theil's U
## Training set 0.08332435        NA
## Test set     0.37823297  1.123981
```

```r
fit2 <- ets(sp500)
train2 <- forecast(fit2, h=08)
accuracy(train2, test2)
```

```
##                        ME       RMSE        MAE          MPE       MAPE
## Training set     9.364321   57.51689   45.70232    0.3766103   3.215566
## Test set     -2250.867866 2250.87901 2250.86787 -532.6340976 532.634098
##                   MASE        ACF1 Theil's U
## Training set  0.208785  0.08332435        NA
## Test set     10.282791 -0.00919882  254.2074
```

```r
fit3 <- ets(sp500)
train3 <- forecast(fit3, h=15)
accuracy(train3, test3)
```

```
##                        ME       RMSE        MAE           MPE         MAPE
## Training set     9.364321   57.51689   45.70232  3.766103e-01     3.215566
## Test set     -2669.714532 2669.71454 2669.71453 -6.876066e+04 68760.660435
##                   MASE       ACF1 Theil's U
## Training set  0.208785 0.08332435        NA
## Test set     12.196237 0.44588015  21625.94
```

```r
fit4 <- ets(sp500)
train4 <- forecast(fit4, h=15)
accuracy(train4, test4)
```

```
##                        ME       RMSE        MAE           MPE         MAPE
## Training set     9.364321   57.51689   45.70232  3.766103e-01 3.215566e+00
## Test set     -2673.261199 2673.26120 2673.26120 -1.001362e+06 1.001362e+06
##                   MASE       ACF1 Theil's U
## Training set  0.208785 0.08332435        NA
## Test set     12.212439 0.61436170  20939.03
```

Arima Model Forecasting

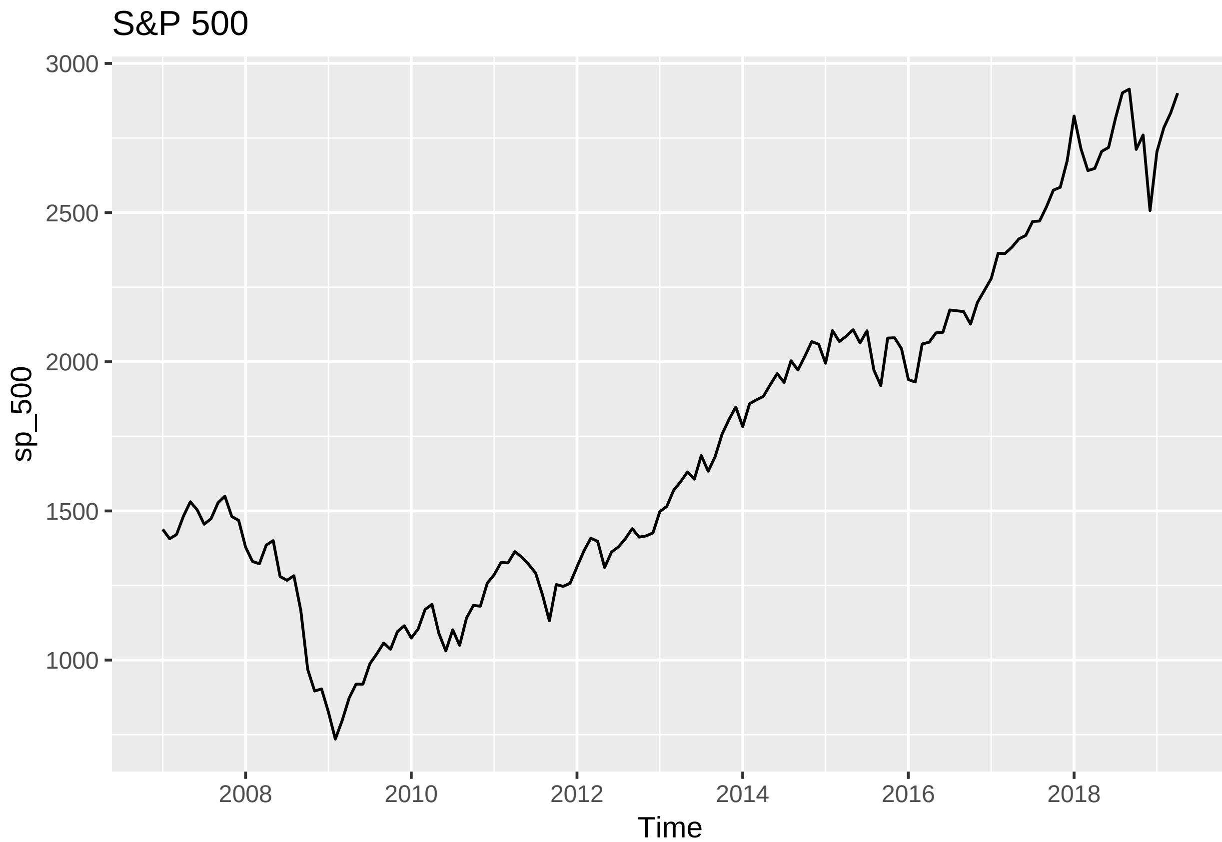
Arima Models aims to describe auto correlations in the data. Exponential smothening and ARIMA models are two widely used approaches.

Stationarity

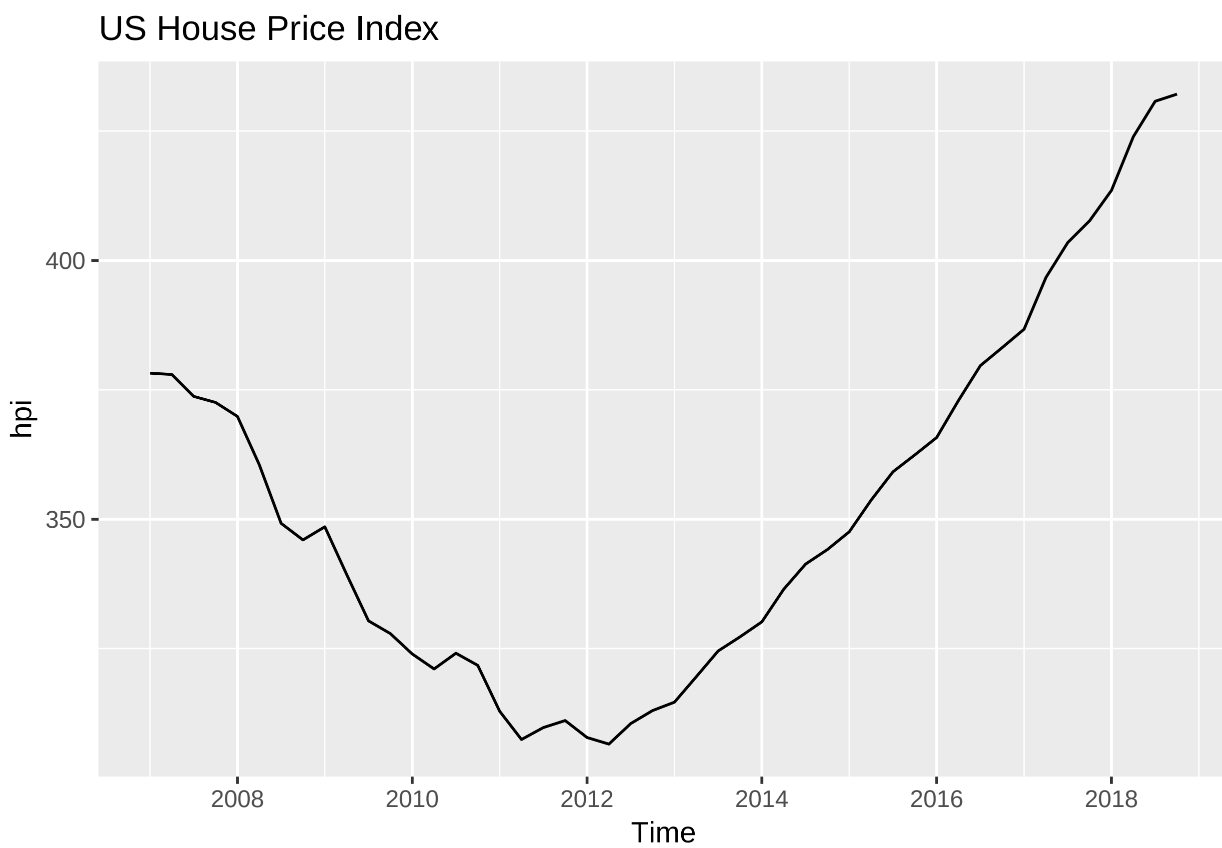
Time series with Trend or seasonality is not stationary but can be of cyclic. For ARIMA model predictions we need to make sure that time series is stationary.

We can see clearly that all these series are not stationary by simply plotting them again as all these have trend component in it.

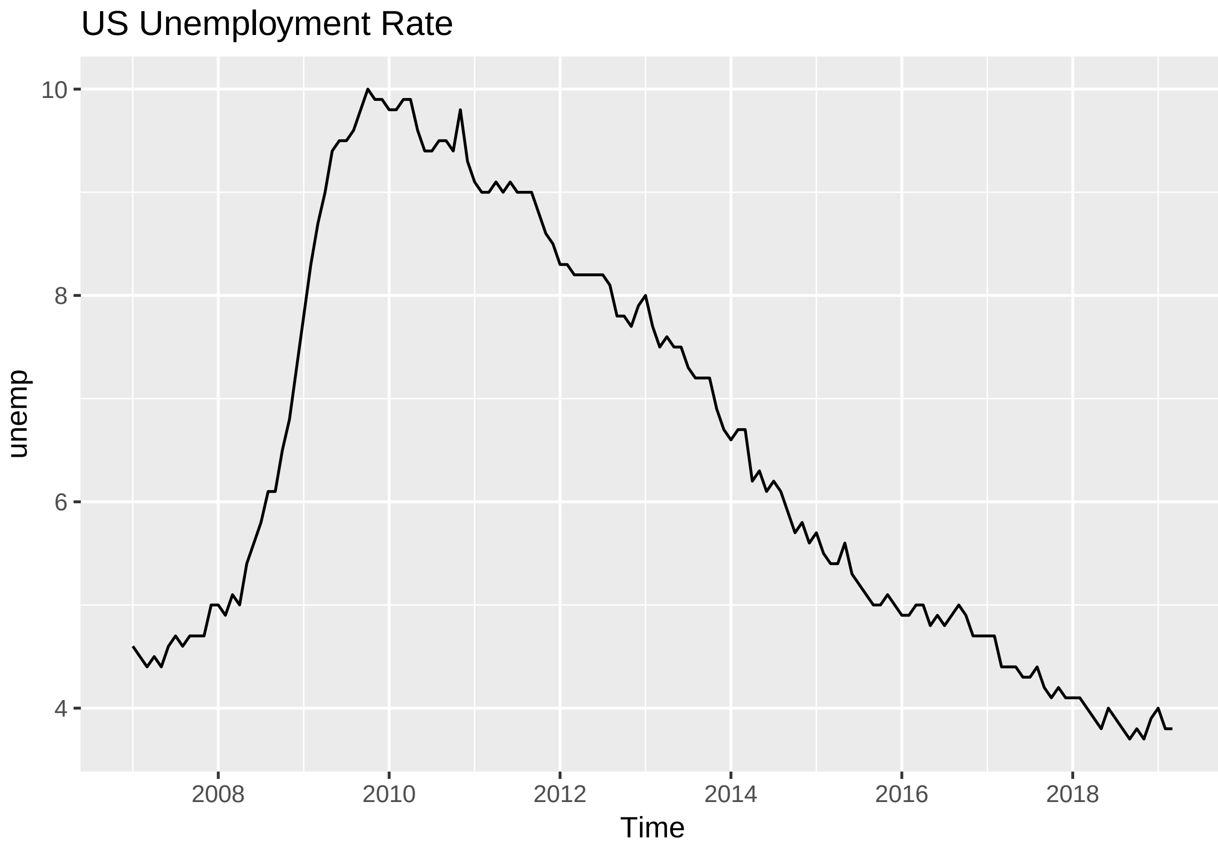
```r
autoplot(sp_500) + ggtitle("S&P 500")
```



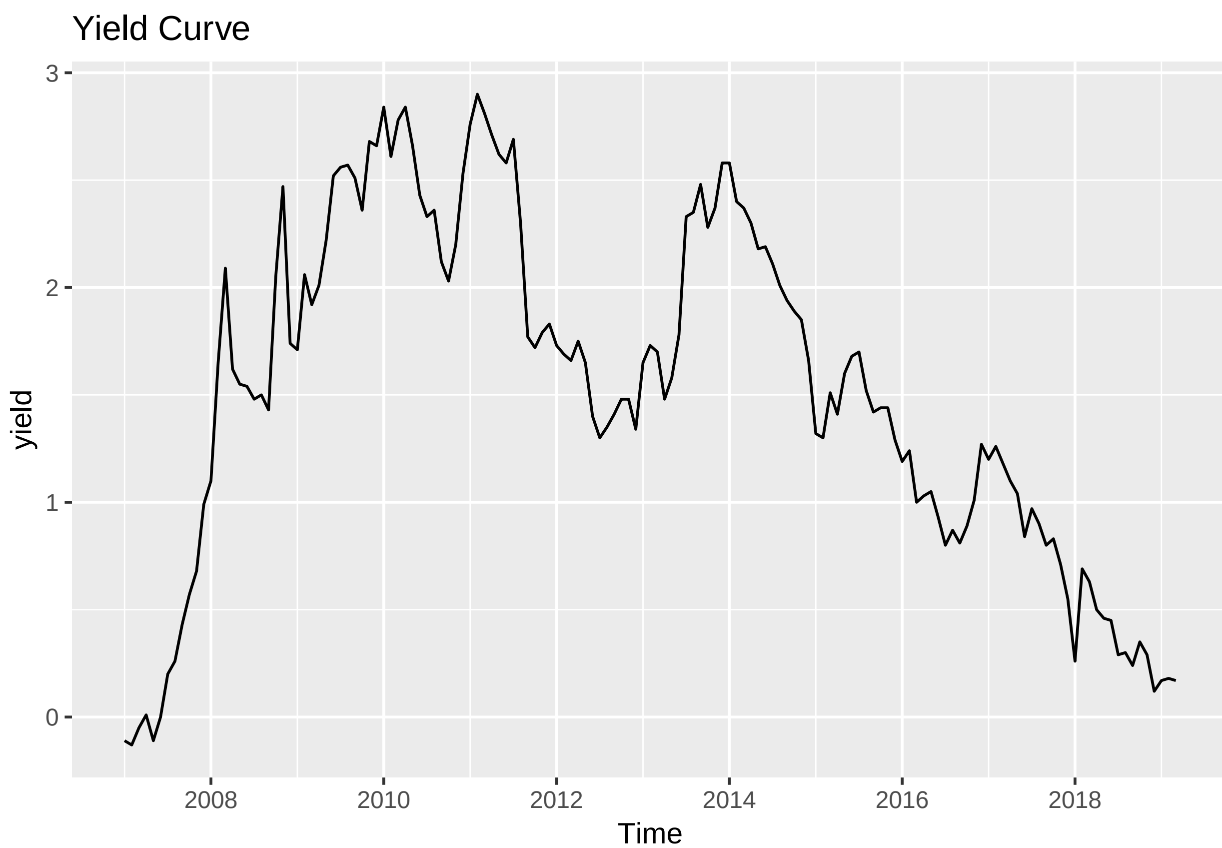

```r
autoplot(hpi) + ggtitle("US House Price Index")
```




```
autoplot(unemp) + ggtitle("US Unemployment Rate")
```




```r
autoplot(yield) + ggtitle("Yield Curve")
```



Test for stationarity and Differencing required

We test for staionarity and check how many differencing required to make it stationary.

```
# Null Hypothesis is that the series is stationary
library(urca)
```

```
## Warning: package 'urca' was built under R version 3.5.2
```

```
summary(ur.kpss(sp_500))
```

```
##
## #######################
## # KPSS Unit Root Test #
## #######################
##
## Test is of type: mu with 4 lags.
##
## Value of test-statistic is: 2.6619
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

```
summary(ur.kpss(hpi))
```

```
##
## #######################
```

```
## # KPSS Unit Root Test #
## #######################
##
## Test is of type: mu with 3 lags.
##
## Value of test-statistic is: 0.5886
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

```
summary(ur.kpss(unemp))
```

```
##
## #######################
## # KPSS Unit Root Test #
## #######################
##
## Test is of type: mu with 4 lags.
##
## Value of test-statistic is: 1.3667
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

```
summary(ur.kpss(yield))
```

```
##
## #######################
## # KPSS Unit Root Test #
## #######################
##
## Test is of type: mu with 4 lags.
##
## Value of test-statistic is: 0.8576
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

```
sp1<- ndiffs(sp_500)
hp1 <-ndiffs(hpi)
un1<- ndiffs(unemp)
yi1 <-ndiffs(yield)


print(paste0("Differencing required to make S&P 500 Index stationary: " , sp1))
```

```
## [1] "Differencing required to make S&P 500 Index stationary: 1"
```

```
print(paste0("Differencing required to make House Price Index stationary: " , hp1))
```

```
## [1] "Differencing required to make House Price Index stationary: 2"
```

```
print(paste0("Differencing required to make Unemployment Rate stationary: " , un1))
```

```
## [1] "Differencing required to make Unemployment Rate stationary: 2"
```

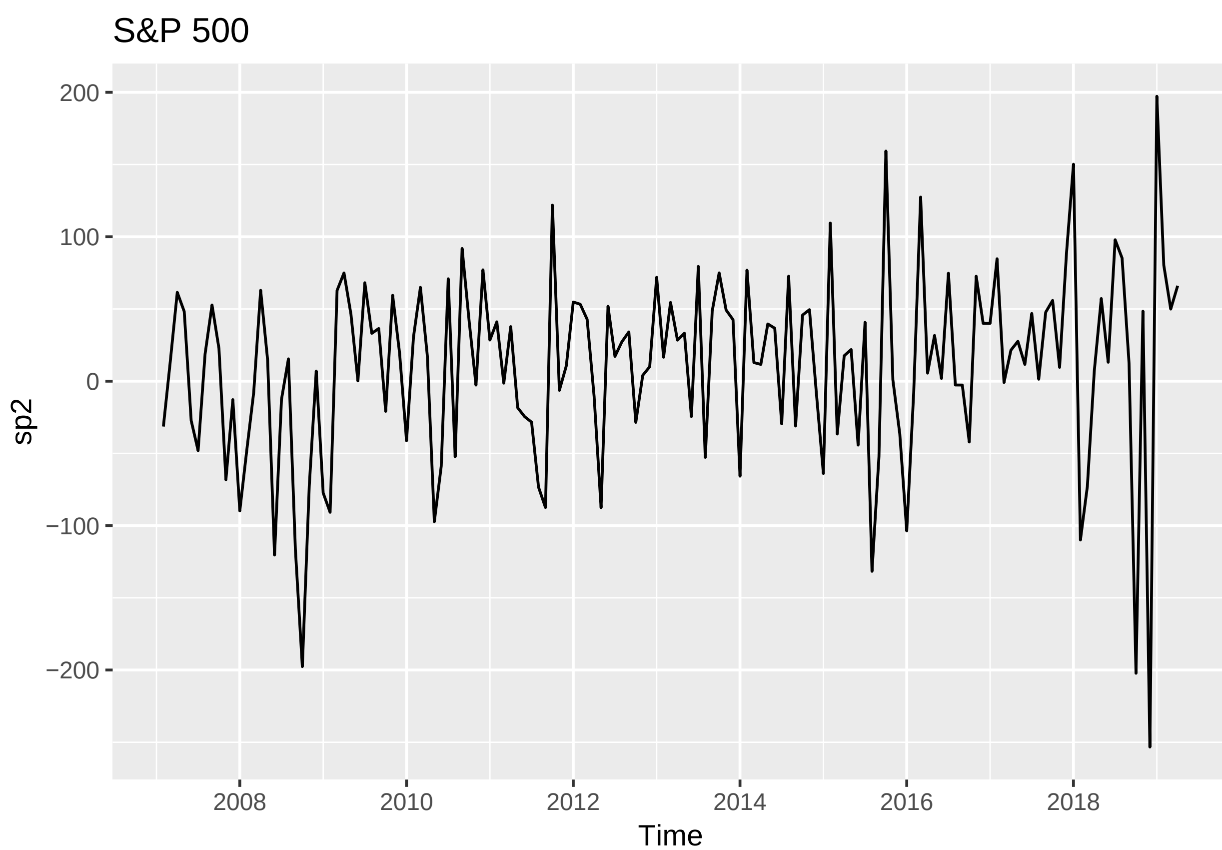
```
print(paste0("Differencing required to make yieldcurve stationary: " , yi1))
```

```
## [1] "Differencing required to make yieldcurve stationary: 2"
```

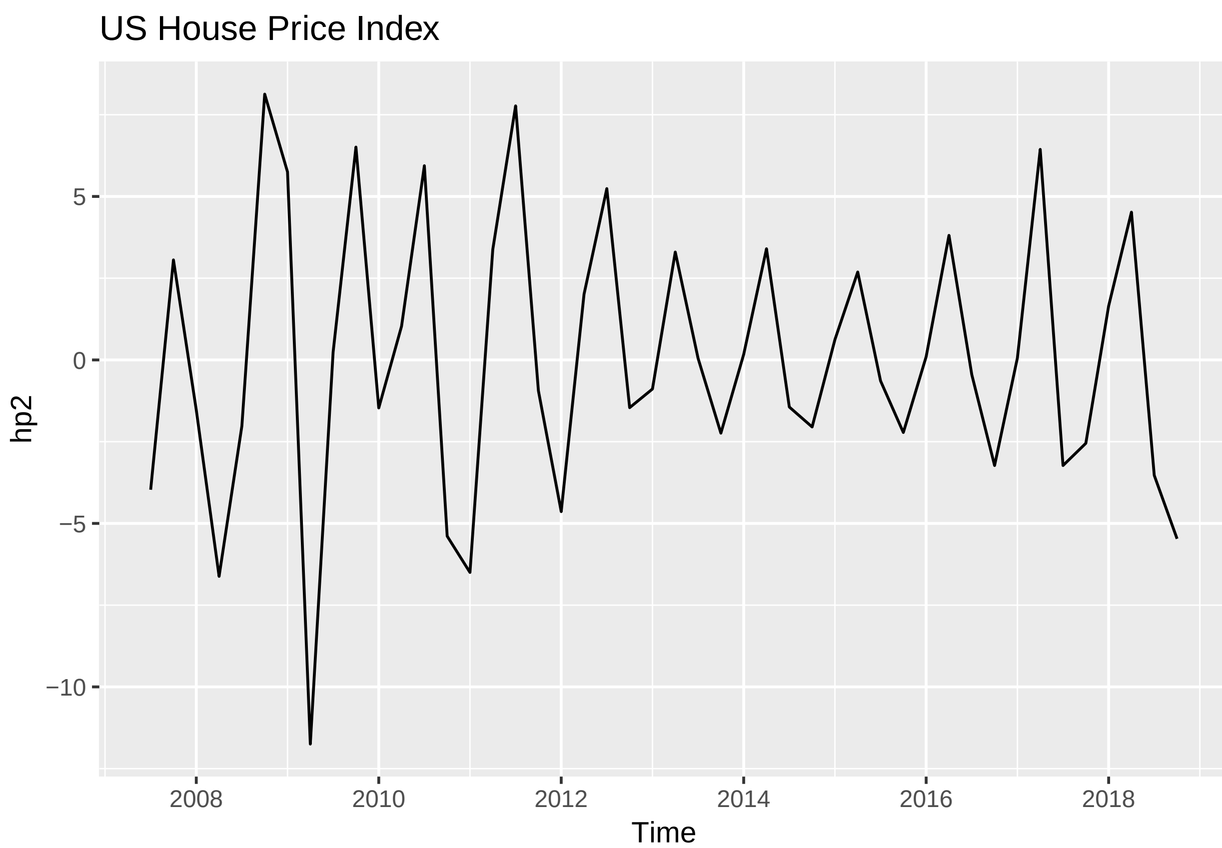
Making Series Stationary

We makee the series Stationary and see by plotting them.All observations should have same mean.

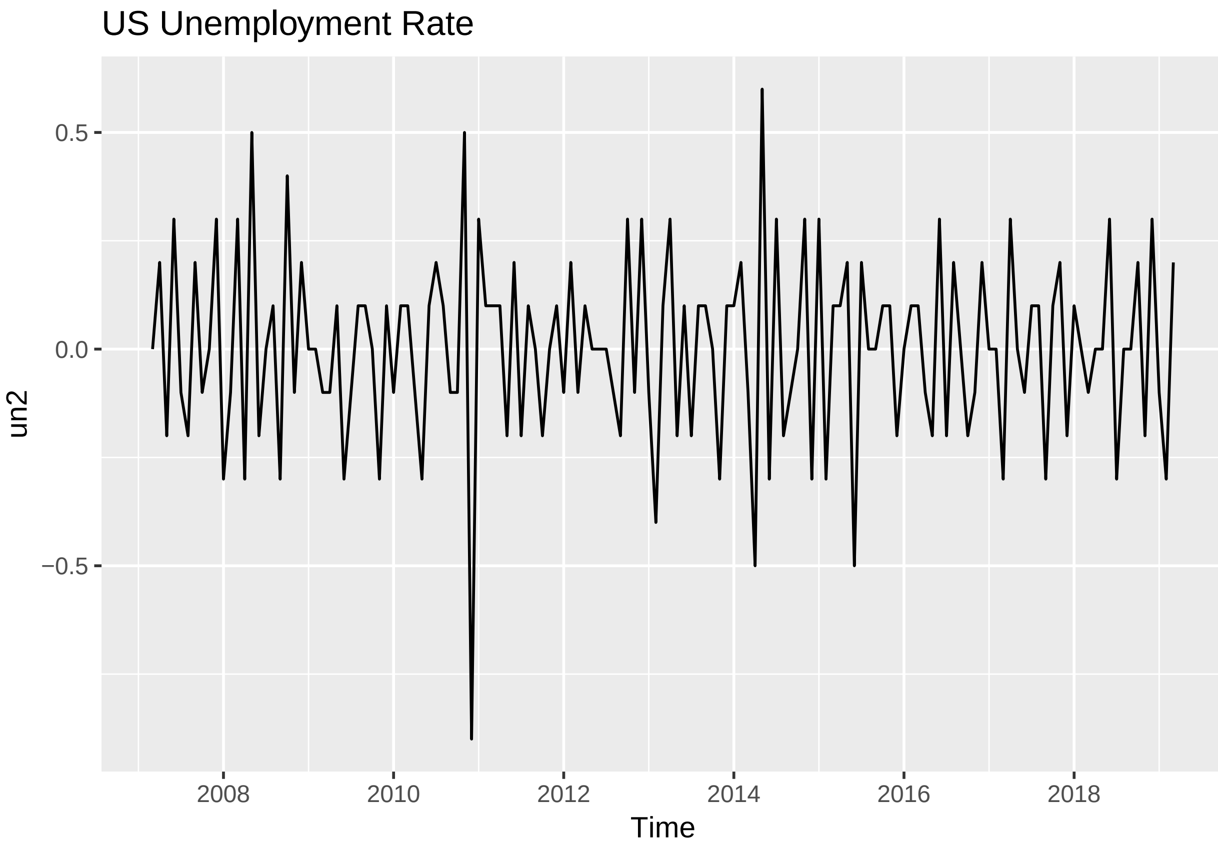
```
sp2 <- diff(sp_500)
hp_fd <- diff(hpi)
hp2 <- diff(hp_fd)
un_fd <- diff(unemp)
un2 <- diff(un_fd)
yi_fd <- diff(yieldcurve1)
yi2 <- diff(yi_fd)

autoplot(sp2) + ggtitle("S&P 500")
```



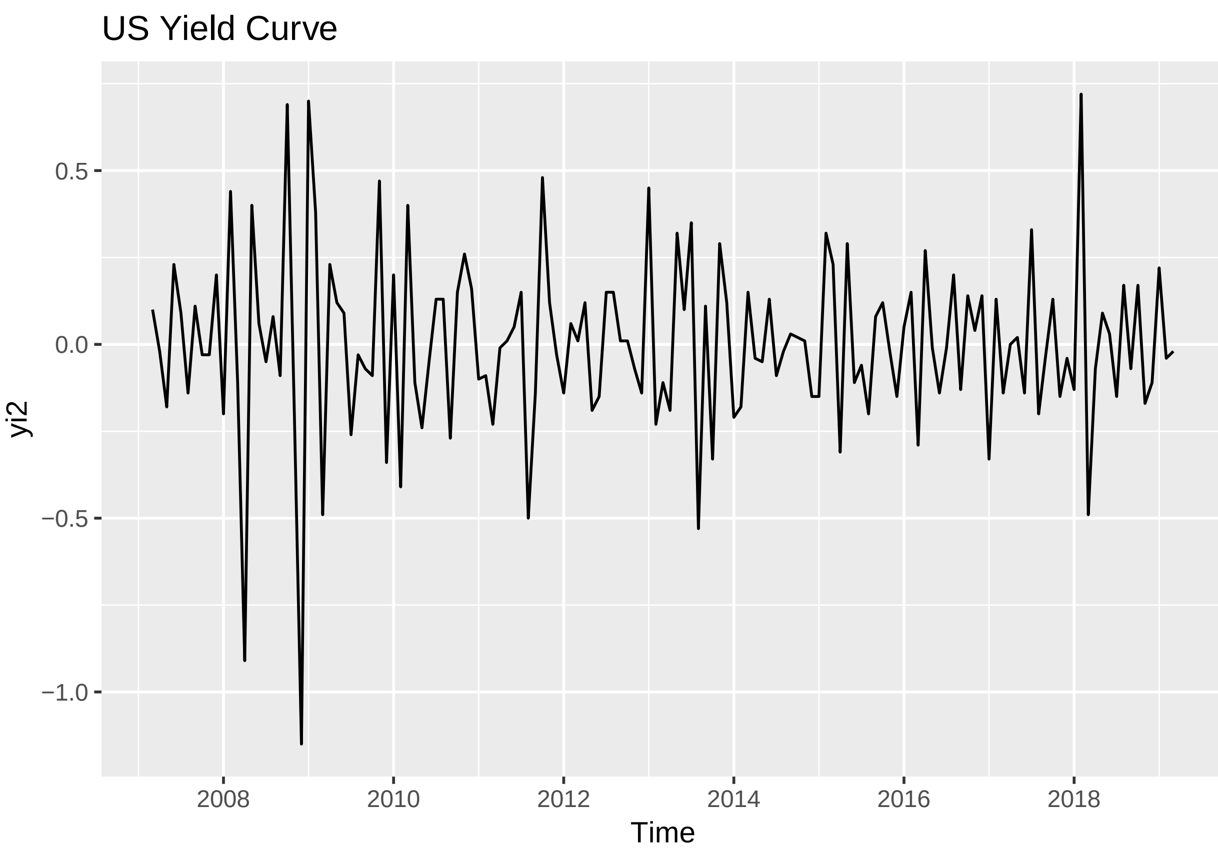

```
autoplot(hp2) + ggtitle("US House Price Index")
```




```r
autoplot(un2) + ggtitle("US Unemployment Rate")
```




```r
autoplot(yi2) + ggtitle("US Yield Curve")
```



Finding (p,d,q) values for ARIMA Modelling

P specifies the lagged value predictor of order p

d specifies degree of first differencing involved

q specifies forcast errors in regression like model of order q

It is sometimes possible to use the ACF and PACF plot, to determine appropriate values of p and q

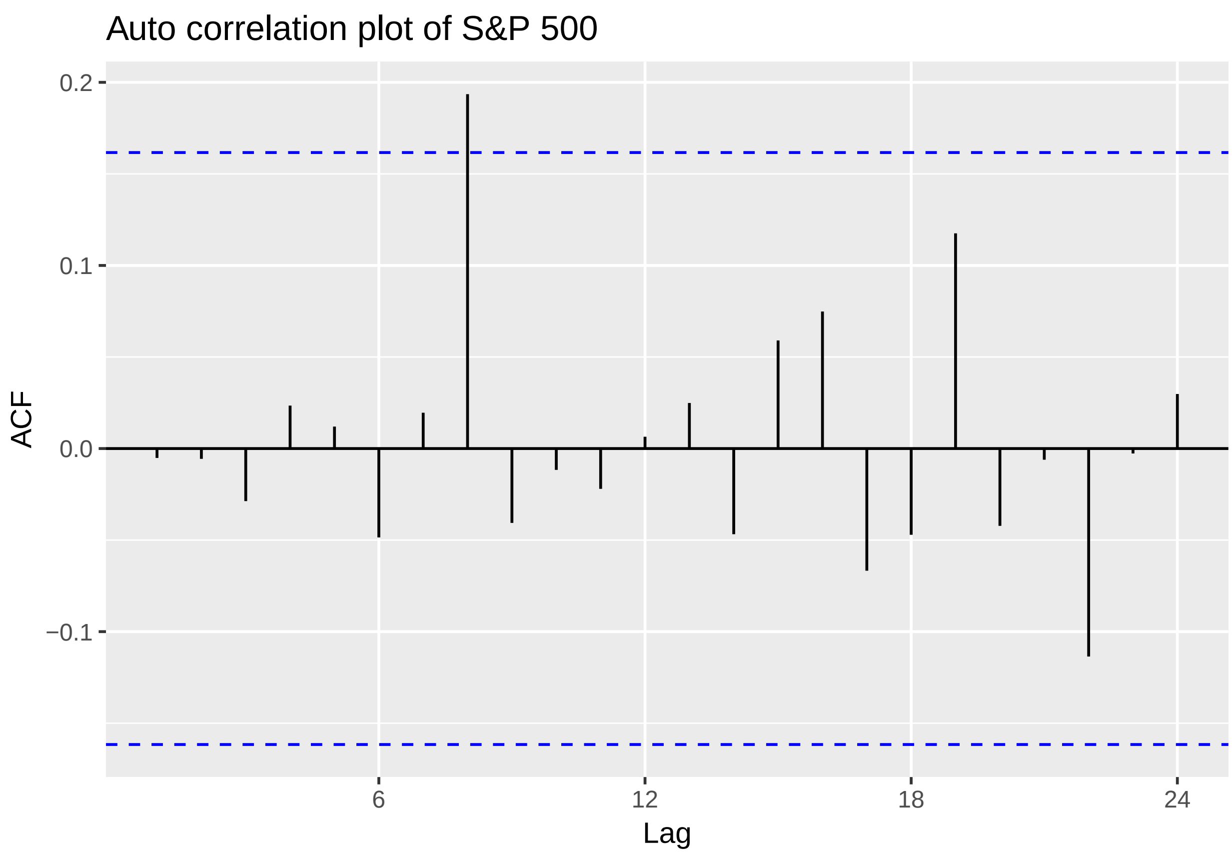
(p,d,q) values for US s&P 500: (0,1,0)

(p,d,q) values for US House Price Index: (2,2,0)

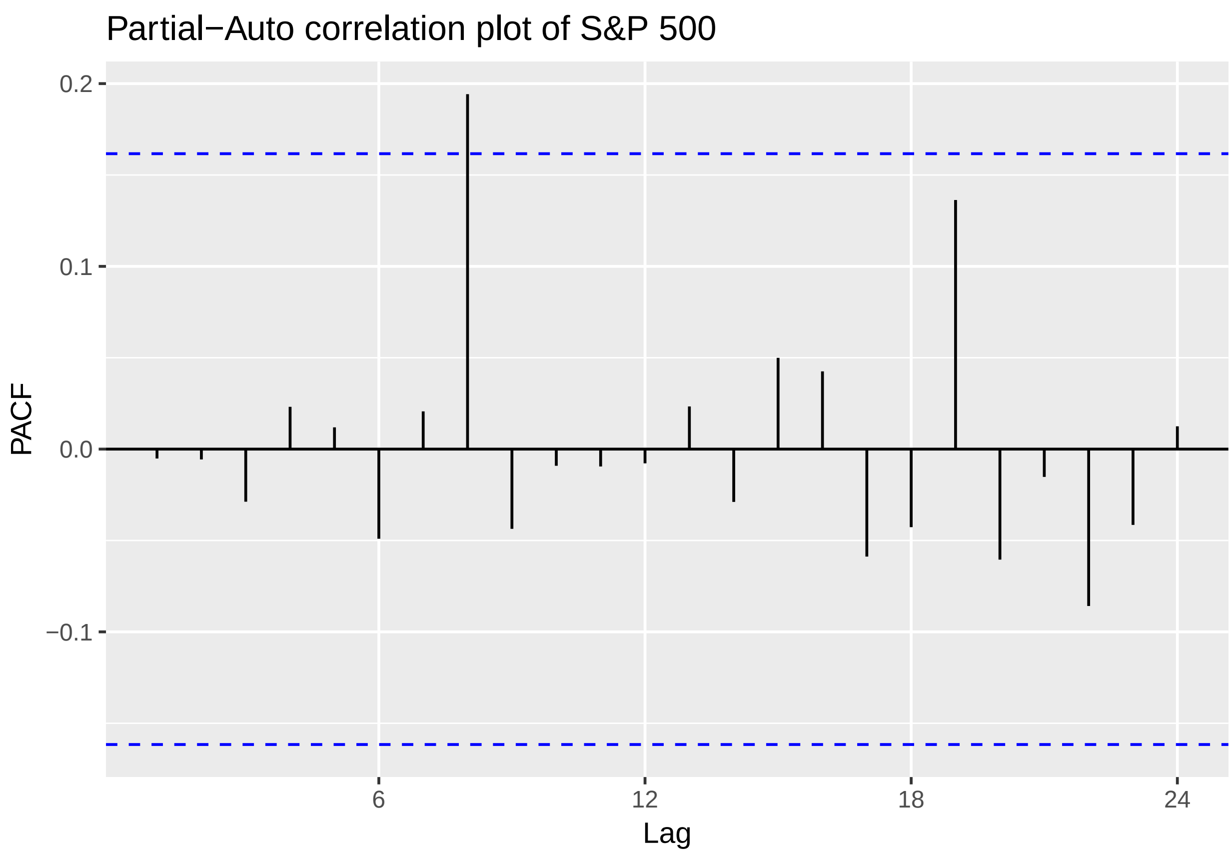
(p,d,q) values for US Unemployment Rate: (5,2,0)

(p,d,q) values for US yield Curve: (4,1,0)

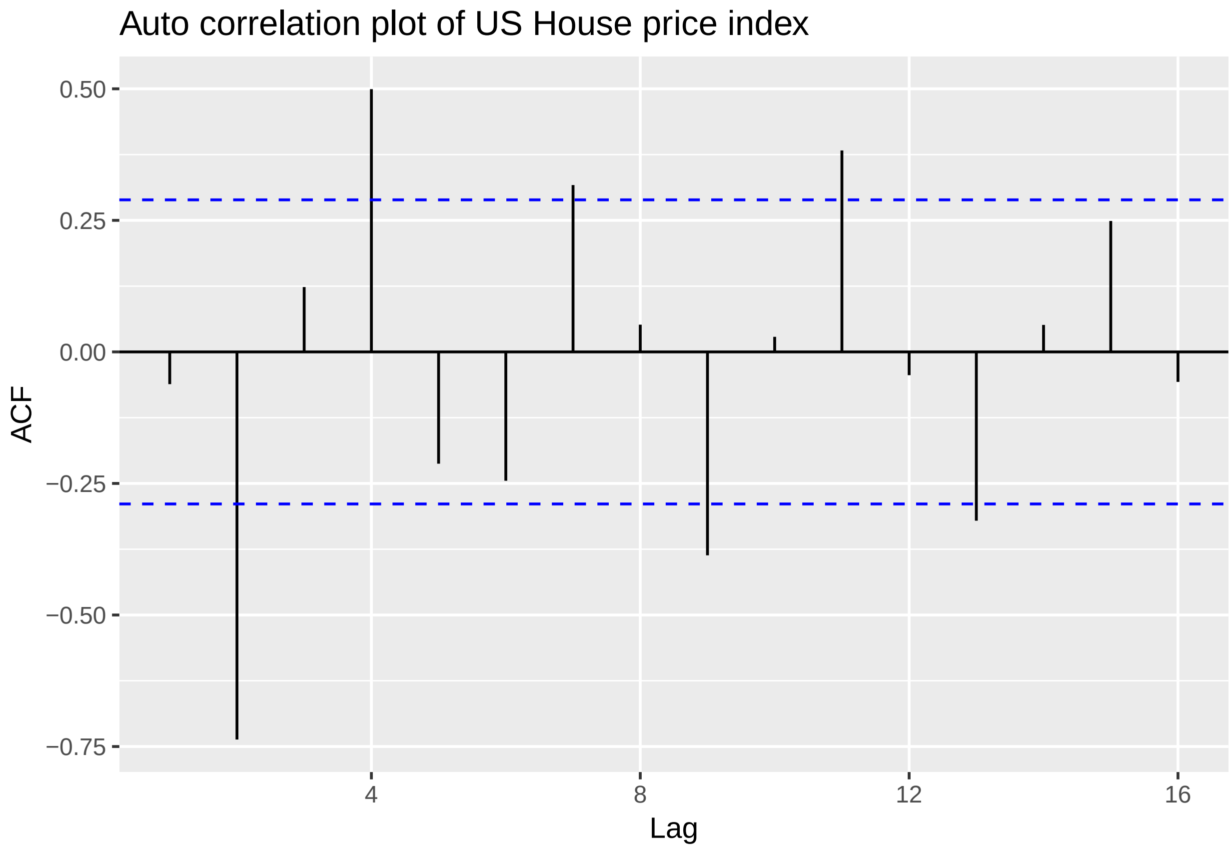
```
ggAcf(sp2) + ggtitle("Auto correlation plot of S&P 500")
```



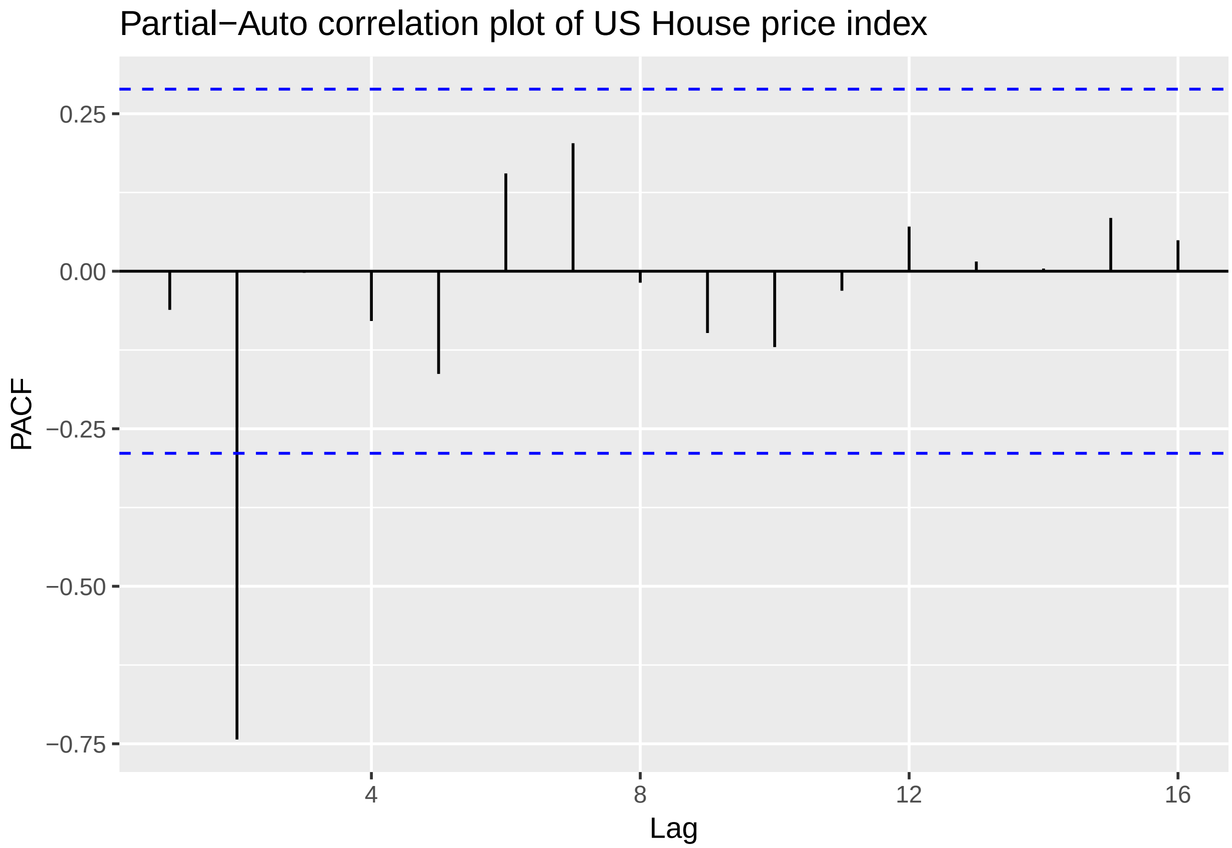

```r
ggPacf(sp2)+ ggtitle("Partial-Auto correlation plot of S&P 500")
```



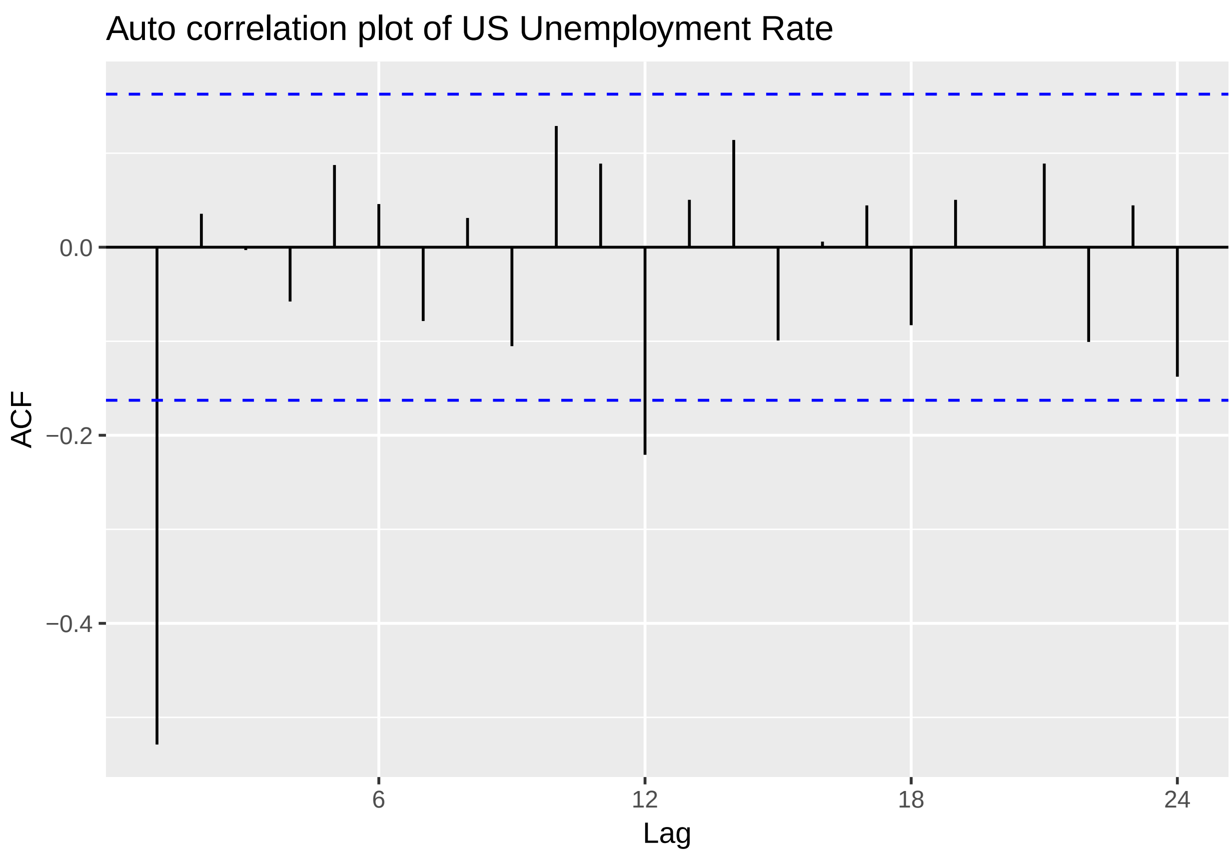

```
ggAcf(hp2) + ggtitle("Auto correlation plot of US House price index ")
```



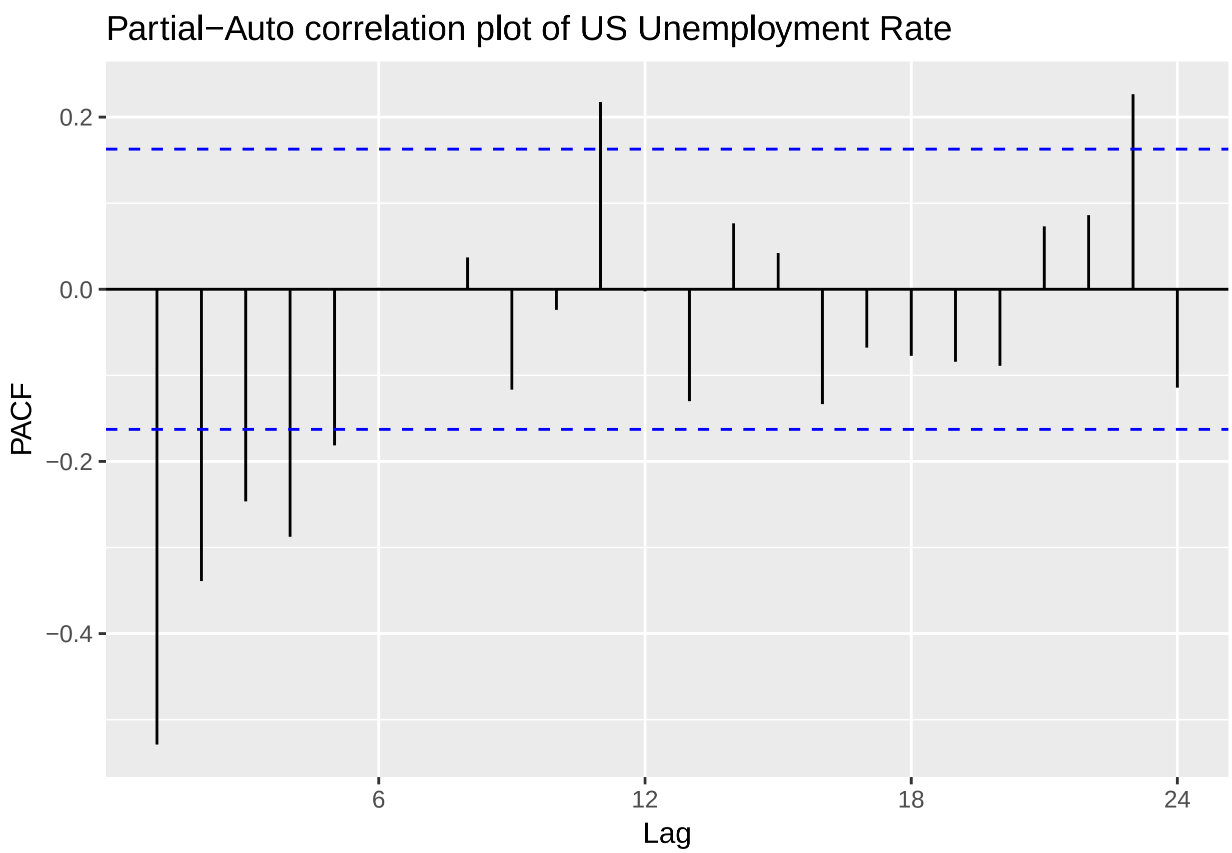

```r
ggPacf(hp2) +ggtitle("Partial-Auto correlation plot of US House price index ")
```



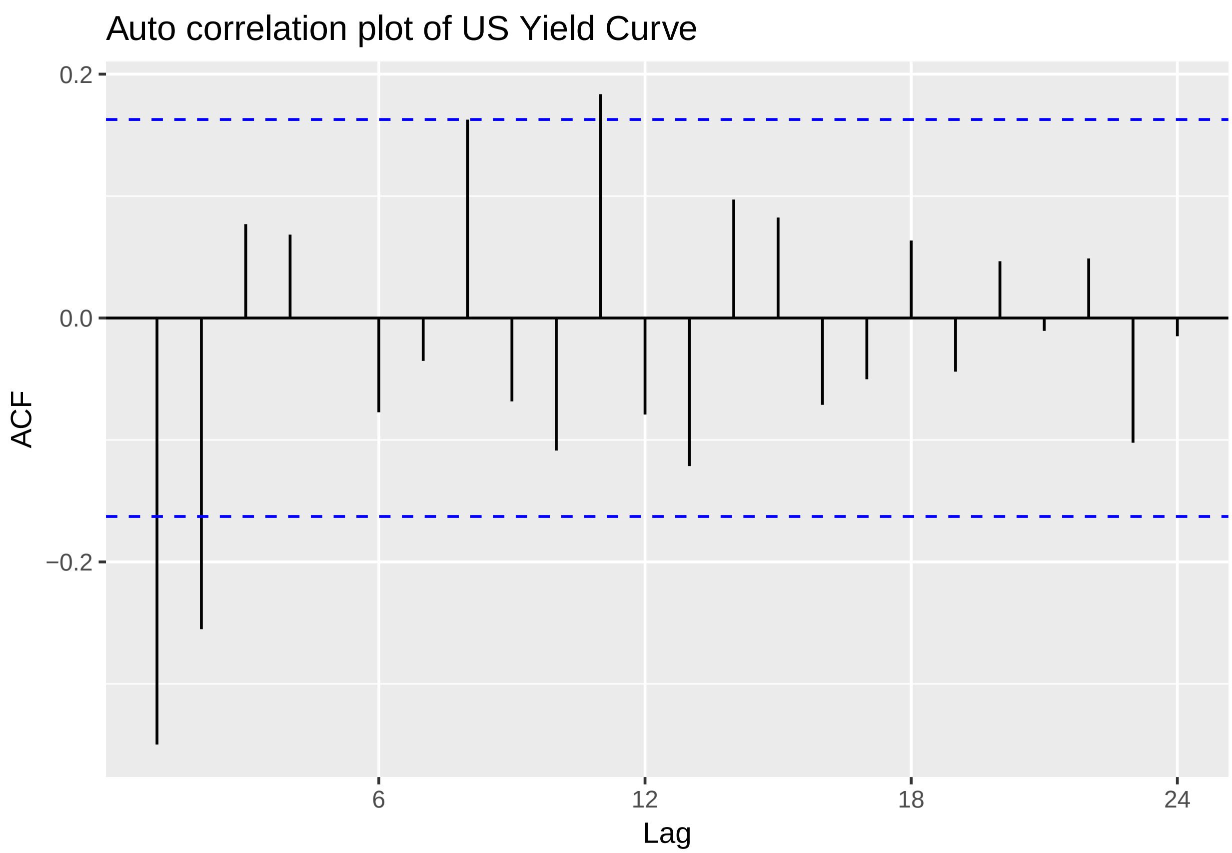

```r
ggAcf(un2) + ggtitle("Auto correlation plot of US Unemployment Rate ")
```



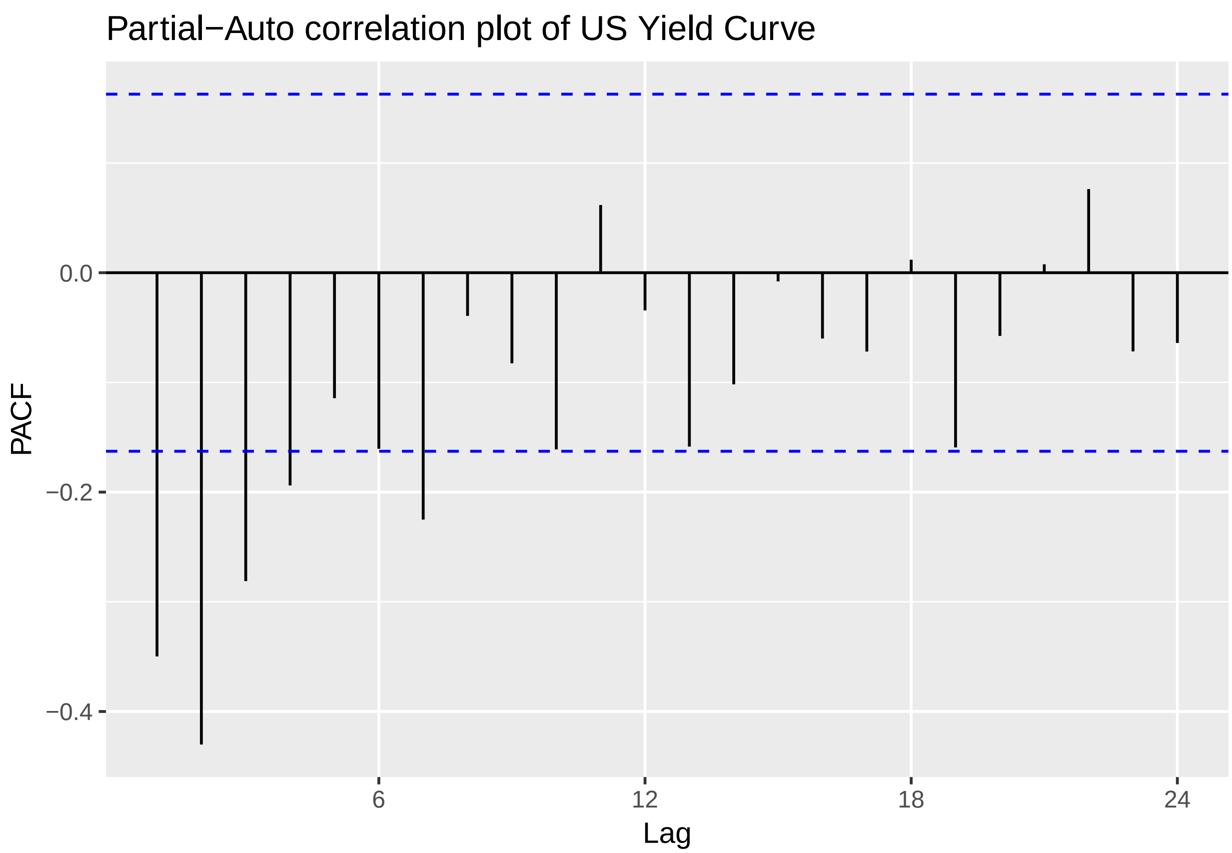

```
ggPacf(un2) +ggtitle("Partial-Auto correlation plot of US Unemployment Rate ")
```




```r
ggAcf(yi2) + ggtitle("Auto correlation plot of US Yield Curve ")
```




```r
ggPacf(yi2) +ggtitle("Partial-Auto correlation plot of US Yield Curve ")
```



We can also use Auto arima function of forecast package to get (p,d,q) values

```
auto.arima(sp_500)
```

```
## Series: sp_500
## ARIMA(0,1,0) with drift
##
## Coefficients:
##        drift
##       9.9470
## s.e.  5.4517
##
## sigma^2 estimated as 4399:  log likelihood=-824.68
## AIC=1653.37   AICc=1653.45   BIC=1659.35
```

```
auto.arima(hpi)
```

```
## Series: hpi
## ARIMA(2,2,0)
##
## Coefficients:
##           ar1      ar2
##       -0.1218  -0.7775
## s.e.   0.0955   0.0892
##
## sigma^2 estimated as 7.076:  log likelihood=-110.16
```

```
## AIC=226.32   AICc=226.89   BIC=231.81
```

```r
auto.arima(unemp)
```

```
## Series: unemp
## ARIMA(1,2,2)(0,0,2)[12]
##
## Coefficients:
##          ar1      ma1     ma2     sma1     sma2
##       0.5679  -1.5094  0.6285  -0.4531  -0.2312
## s.e.  0.2177   0.1905  0.1553   0.0853   0.0833
##
## sigma^2 estimated as 0.0206:  log likelihood=74.5
## AIC=-137   AICc=-136.4   BIC=-119.14
```

```r
auto.arima(yield)
```

```
## Series: yield
## ARIMA(0,2,3)
##
## Coefficients:
##           ma1      ma2     ma3
##       -0.8459  -0.2802  0.1650
## s.e.   0.0817   0.1011  0.0784
##
## sigma^2 estimated as 0.03578:  log likelihood=35.88
## AIC=-63.76   AICc=-63.47   BIC=-51.85
```

Comapairing AIC/ BIC values, fitting and forecasting the final models

S&P 500: It predict 10 month upward trend but with pessimism in the stock market at 95% confidence interval

Unemployment Rate: It predicts constant downward trend in US unemployment rate at 95% confidence interval

House Price Index: It predicts upward trend in US house price Index at 95 % confidence interval

Yield Curve: It predicts yield curve inverts towards the end of 2019 and towards the starting of 2010.

```r
Arima(unemp, order = c(5,2,0))
```

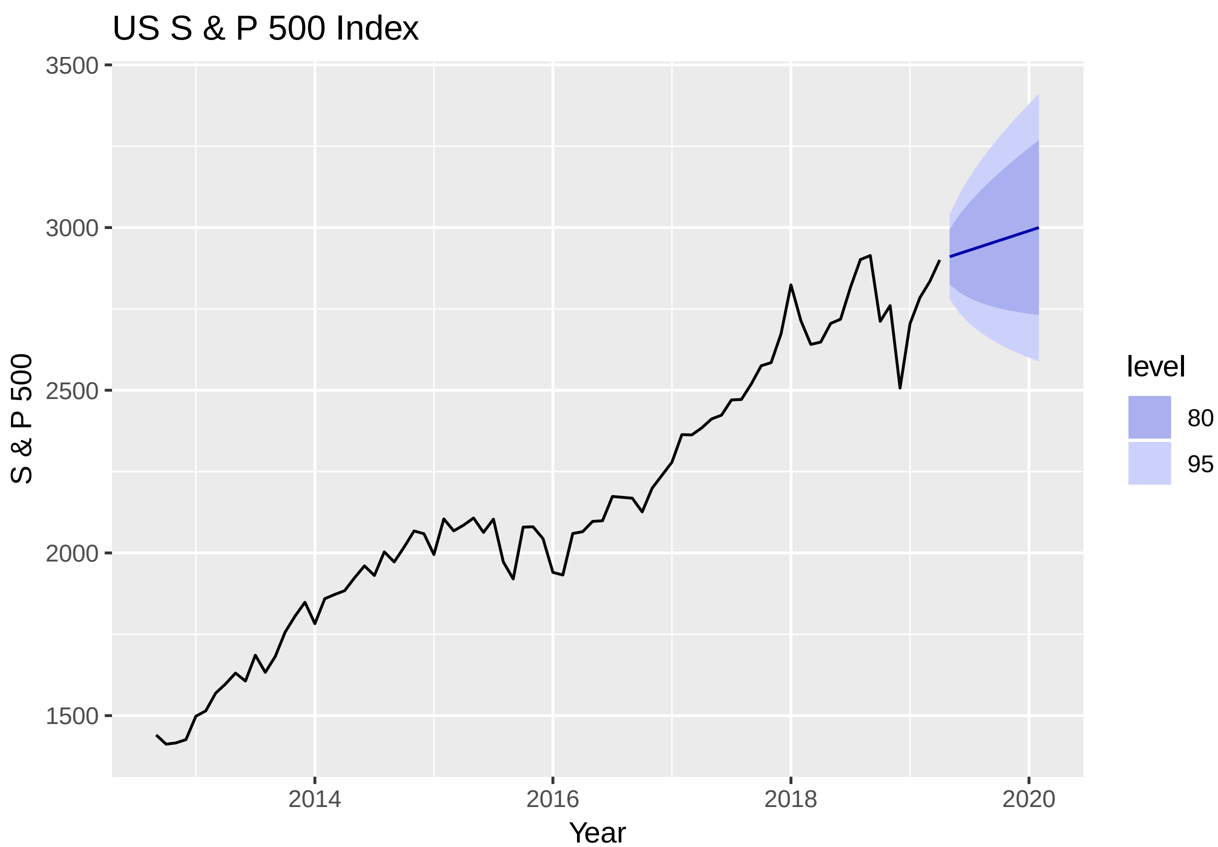
```
## Series: unemp
## ARIMA(5,2,0)
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5
##       -0.9196  -0.7579  -0.6045  -0.4592  -0.1900
## s.e.   0.0817   0.1065   0.1125   0.1062   0.0826
##
## sigma^2 estimated as 0.02534:  log likelihood=62.6
## AIC=-113.2   AICc=-112.59   BIC=-95.34
```

```r
Arima(yield, order = c(4,1,0))
```

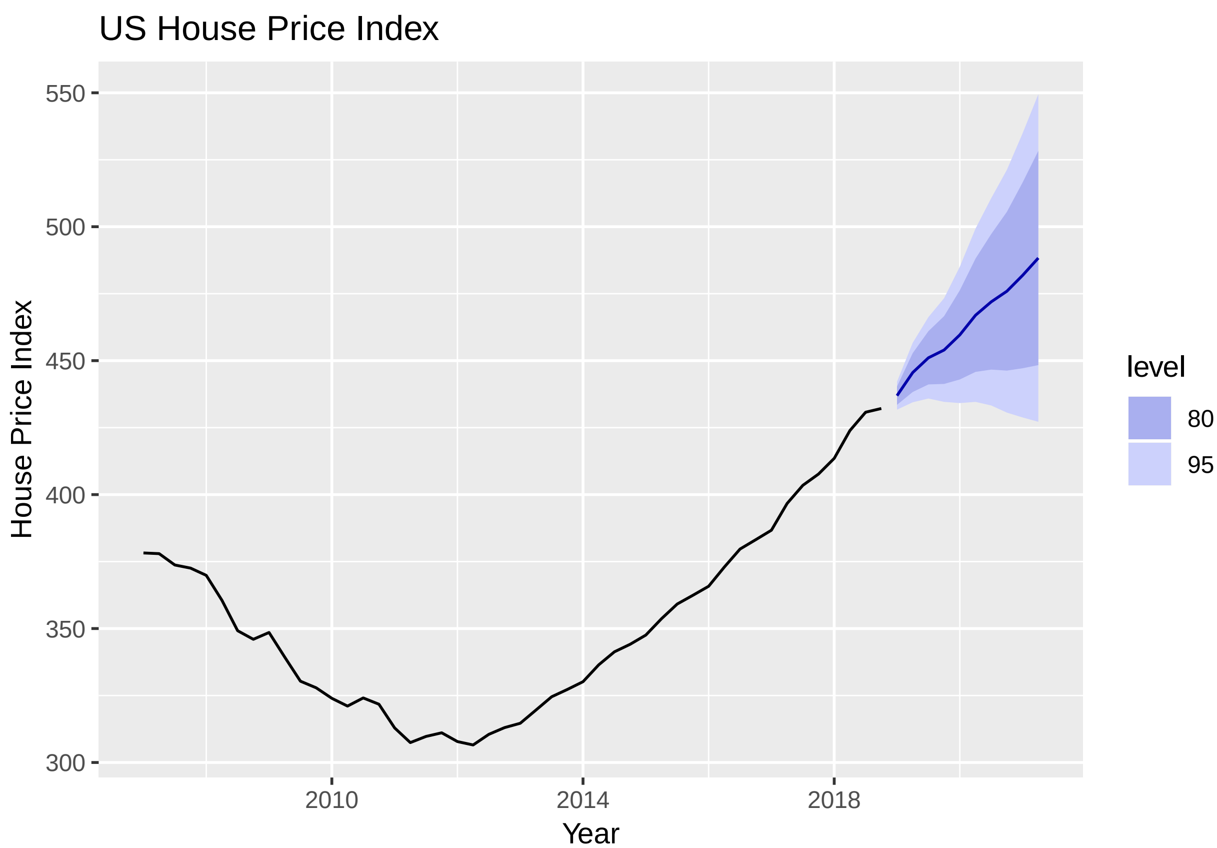
```
## Series: yield
```

```
## ARIMA(4,1,0)
##
## Coefficients:
##          ar1      ar2     ar3     ar4
##       0.1466  -0.1625  0.0734  0.0493
## s.e.  0.0824   0.0827  0.0826  0.0820
##
## sigma^2 estimated as 0.03554:  log likelihood=38.42
## AIC=-66.84   AICc=-66.42   BIC=-51.93
```

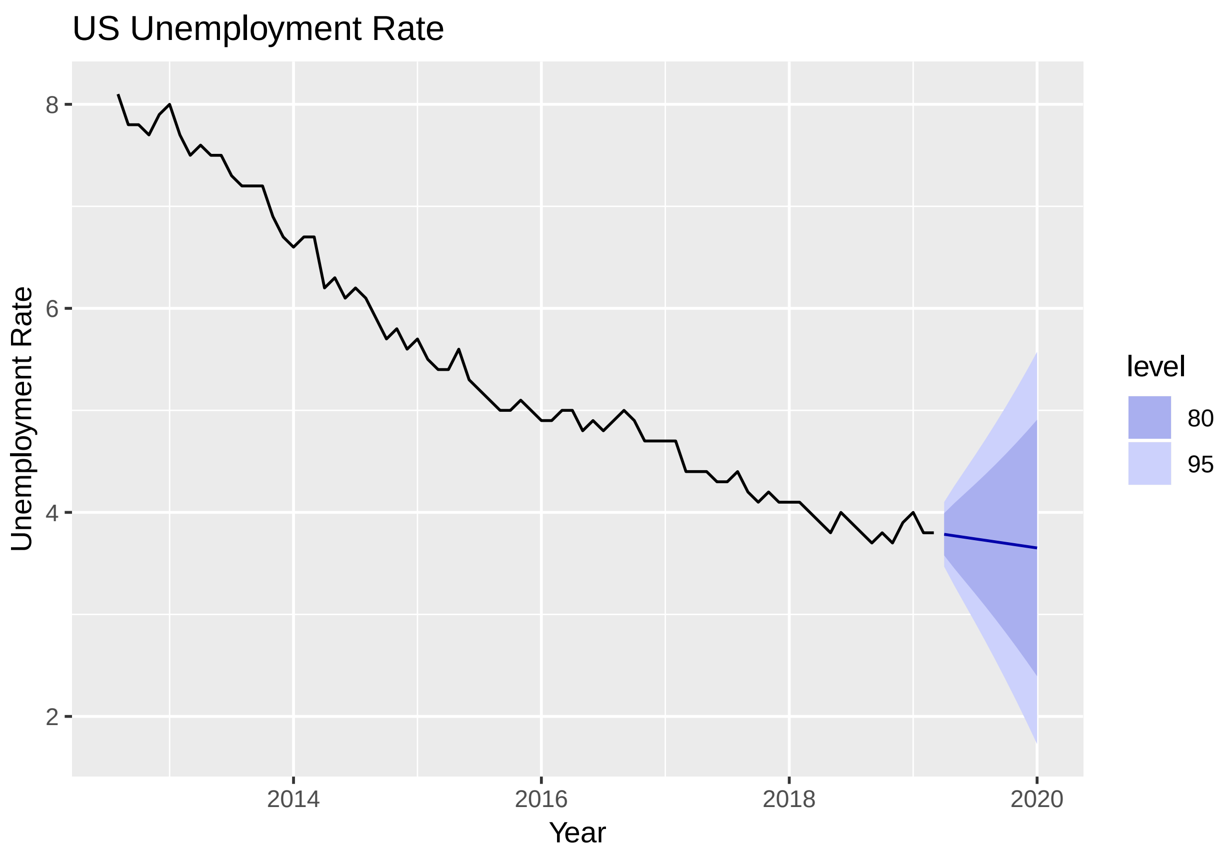
```
fit1 <- auto.arima(sp_500, seasonal = FALSE)
fit1 %>% forecast(h=10) %>% autoplot(include=80) + xlab("Year") +  ylab("S & P 500") + ggtitle("US S & I
```



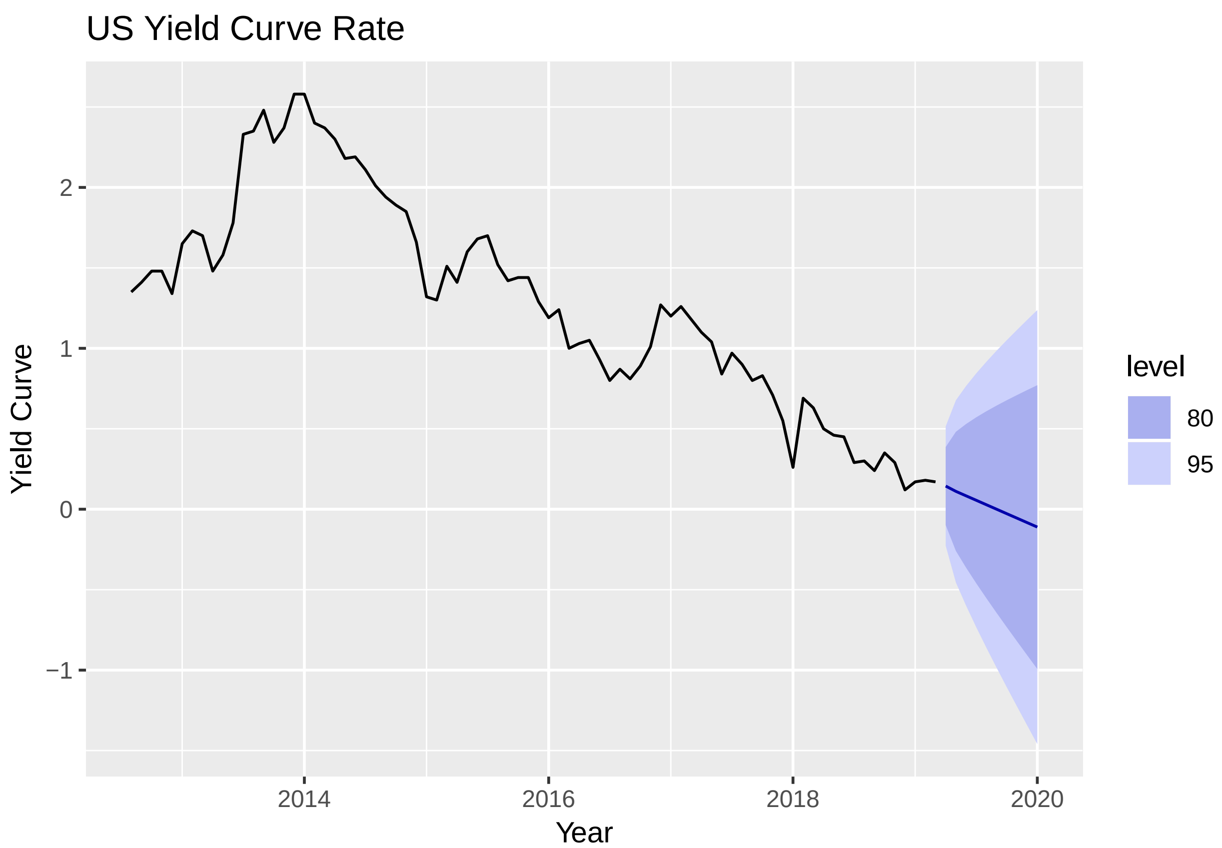

```
fit2 <- auto.arima(hpi, seasonal = FALSE)
fit2 %>% forecast(h=10) %>% autoplot(include=80) + xlab("Year") +  ylab("House Price Index") + ggtitle(
```




```
fit3 <- auto.arima(unemp, seasonal = FALSE)
fit3 %>% forecast(h=10) %>% autoplot(include=80) + xlab("Year") +  ylab("Unemployment Rate") + ggtitle(
```




```
fit4 <- auto.arima(yield, seasonal = FALSE)
fit4 %>% forecast(h=10) %>% autoplot(include=80) + xlab("Year") +  ylab("Yield Curve") + ggtitle("US Yi
```



Forecasting Accuracy of Arima Model

Arima Model have High Forecasting Accuracy as MAPE, MAE and even RMSE have very less values.So, we will Use ARIMA model for our final conclusions.

```
# Forecasting accuracy for ARIMA

sp500 <- window(sp_500, start = 2007, end = c(2017,12))
test1 <- window(sp_500, start = c(2018))

hpiu <- window(hpi, start = 2007, end = c(2016,4))
test2 <- window(hpi, start = 2017)

unempu <- window(unemp, start = 2007, end = c(2017,12))
test3 <- window(unemp, start = c(2018))

yieldu <- window(yield, start = 2007, end = c(2017,12))
test4 <- window(yield, start = c(2018))


fit1 <- auto.arima(sp500, seasonal = FALSE)
train1 <- forecast(fit1, h=16)
accuracy(train1, test1)
```

```
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set  4.780965 56.67981 44.72753 0.3005813 3.195987 0.2043318
```

```
## Test set     -118.394364 179.24536 142.75094 -4.4238311 5.277764 0.6521387
##                    ACF1 Theil's U
## Training set 0.03925546        NA
## Test set     0.54569862  1.596559
```

```r
fit2 <- auto.arima(hpiu, seasonal = FALSE)
train2 <- forecast(fit2, h=08)
accuracy(train2, test2)
```

```
##                     ME     RMSE      MAE        MPE      MAPE      MASE
## Training set 0.2575488 2.590969 1.995721 0.08920473 0.5949802 0.1346211
## Test set     5.3867562 6.431794 5.539618 1.27940150 1.3189303 0.3736743
##                   ACF1 Theil's U
## Training set 0.0318574        NA
## Test set     0.5612678 0.9334991
```

```r
fit3 <- auto.arima(unempu, seasonal = FALSE)
train3 <- forecast(fit3, h=15)
accuracy(train3, test3)
```

```
##                       ME      RMSE       MAE        MPE     MAPE      MASE
## Training set 0.000283463 0.1629713 0.1279162 0.09254736 1.990062 0.1210563
## Test set     0.146901333 0.2154826 0.1595328 3.77307131 4.103690 0.1509774
##                     ACF1 Theil's U
## Training set -0.08435746        NA
## Test set      0.63253261  1.821023
```

```r
fit4 <- auto.arima(yieldu, seasonal = FALSE)
train4 <- forecast(fit4, h=15)
accuracy(train4, test4)
```

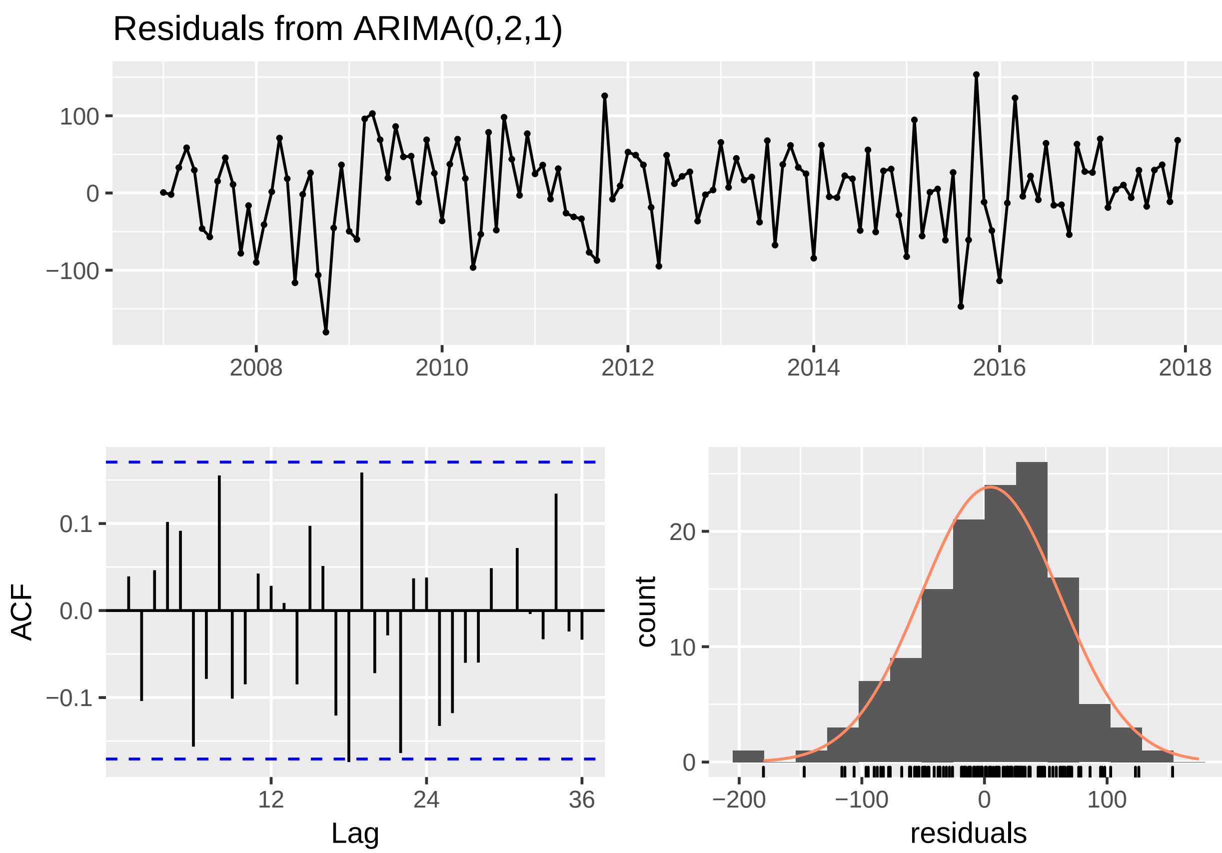
```
##                        ME      RMSE       MAE       MPE     MAPE      MASE
## Training set -0.017720234 0.1881677 0.1381642       Inf      Inf 0.2264987
## Test set      0.007641502 0.1032426 0.0801708 -6.473884 27.22896 0.1314275
##                     ACF1 Theil's U
## Training set -0.01322789        NA
## Test set     -0.09302886  0.505827
```

Residual Diagnosics

Our next step is to run a residual diagnostics to ensure our residuals are white noise under our initial assumptions.

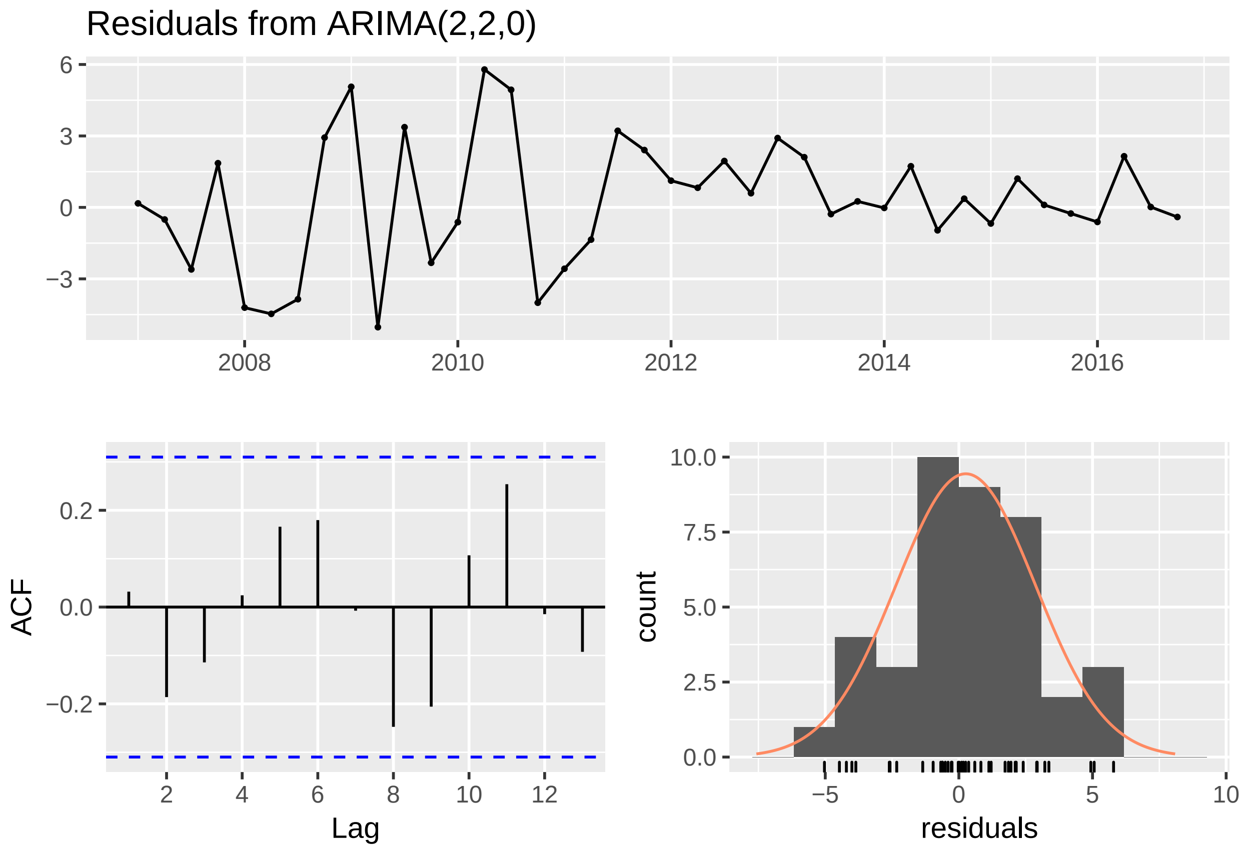
Although not perfect we can see that the residuals do display a normal distribution.

```r
checkresiduals(fit1)
```



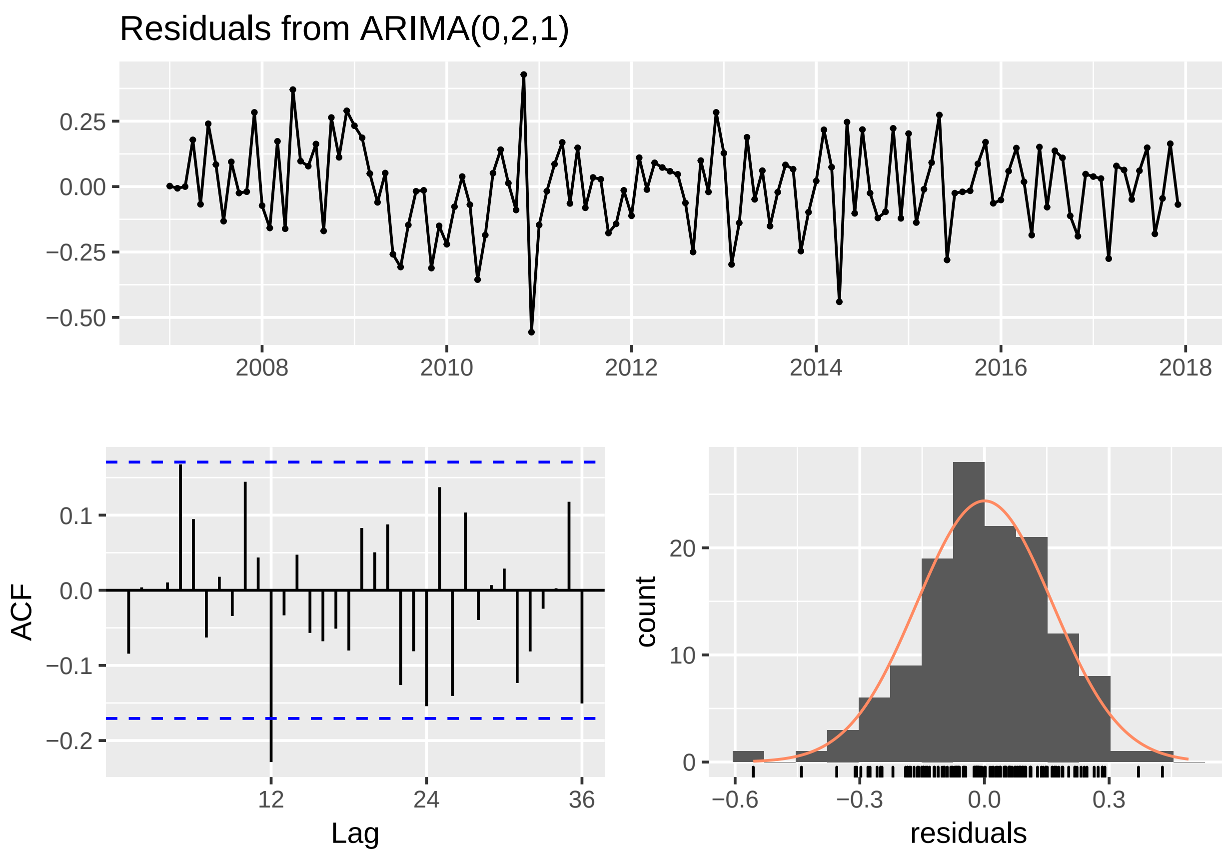

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,2,1)
## Q* = 34.743, df = 23, p-value = 0.05515
##
## Model df: 1.   Total lags used: 24
```

```r
checkresiduals(fit2)
```



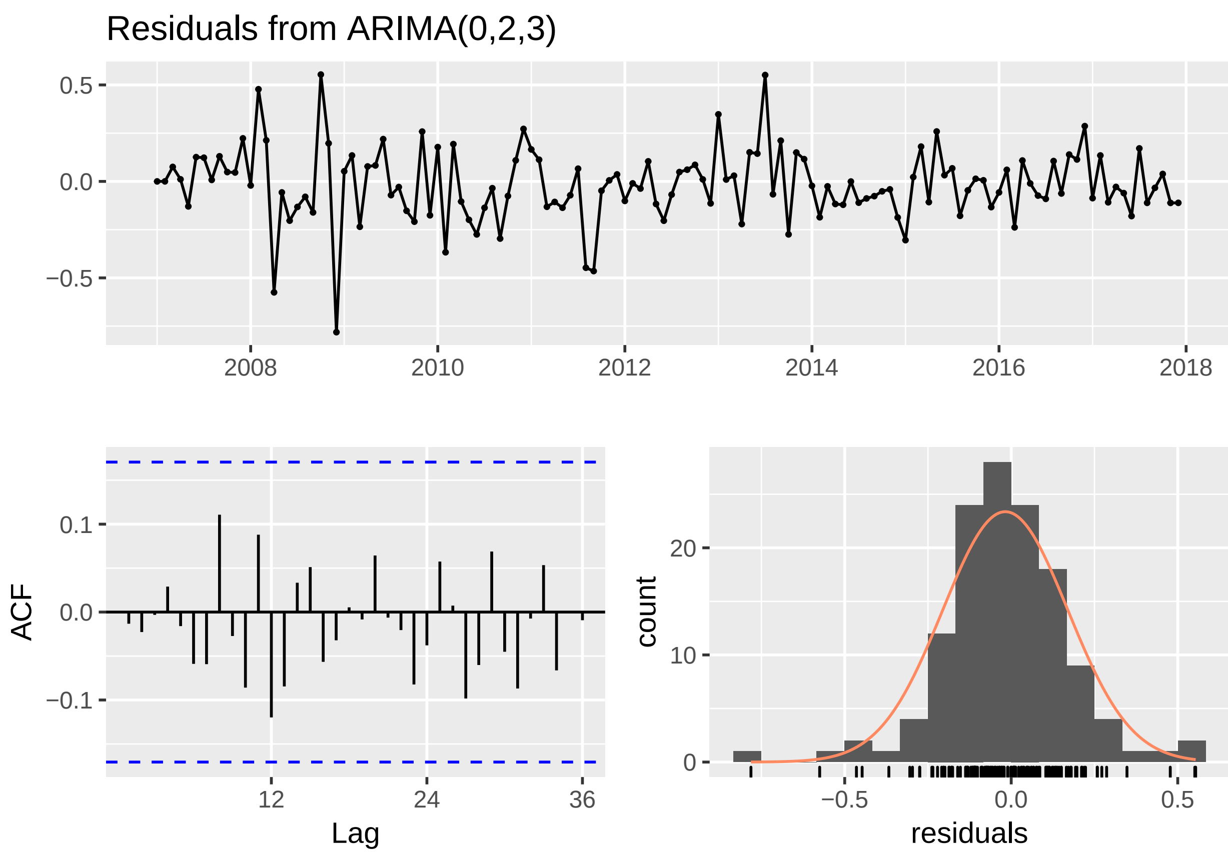

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,2,0)
## Q* = 8.3304, df = 6, p-value = 0.2149
##
## Model df: 2.   Total lags used: 8
```

```r
checkresiduals(fit3)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,2,1)
## Q* = 31.244, df = 23, p-value = 0.1169
##
## Model df: 1.   Total lags used: 24
```

```
checkresiduals(fit4)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,2,3)
## Q* = 11.755, df = 21, p-value = 0.946
##
## Model df: 3.   Total lags used: 24
```

Neural Networks Forecasting

Artificial neural networks are forecasting methods that are based on simple mathematical models of the brain. They allow complex nonlinear relationships between the response variable and its predictors.

S&P 500: Neural Network Predicts downward trend in the S&P 500 Index, so there appears to be caution in the market going forward.

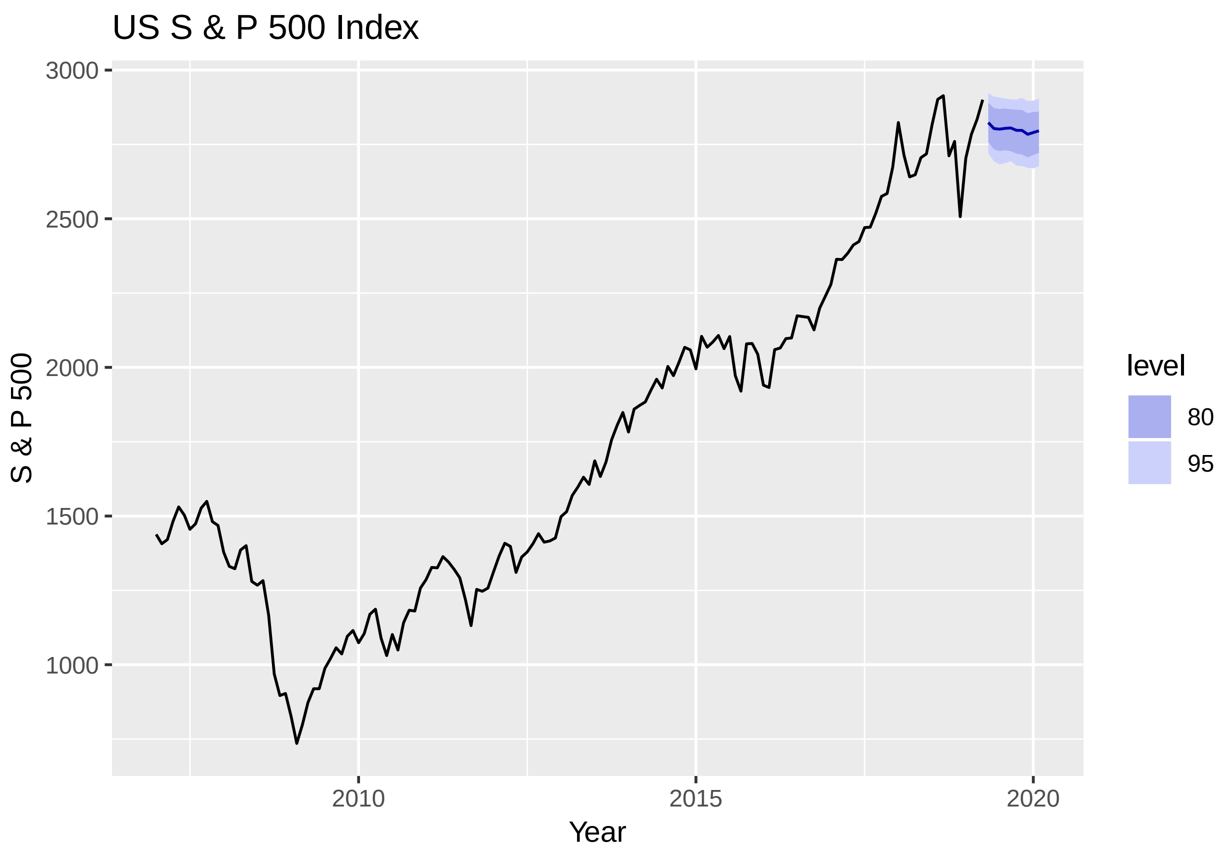
Unemployment Rate: Neural Network predicts a upward constant unemployment rate, again shows a caution in the market towards recession.

House Price Index: Neural Network predicts house prices start to fall towards the end of 2019 and start of 2020

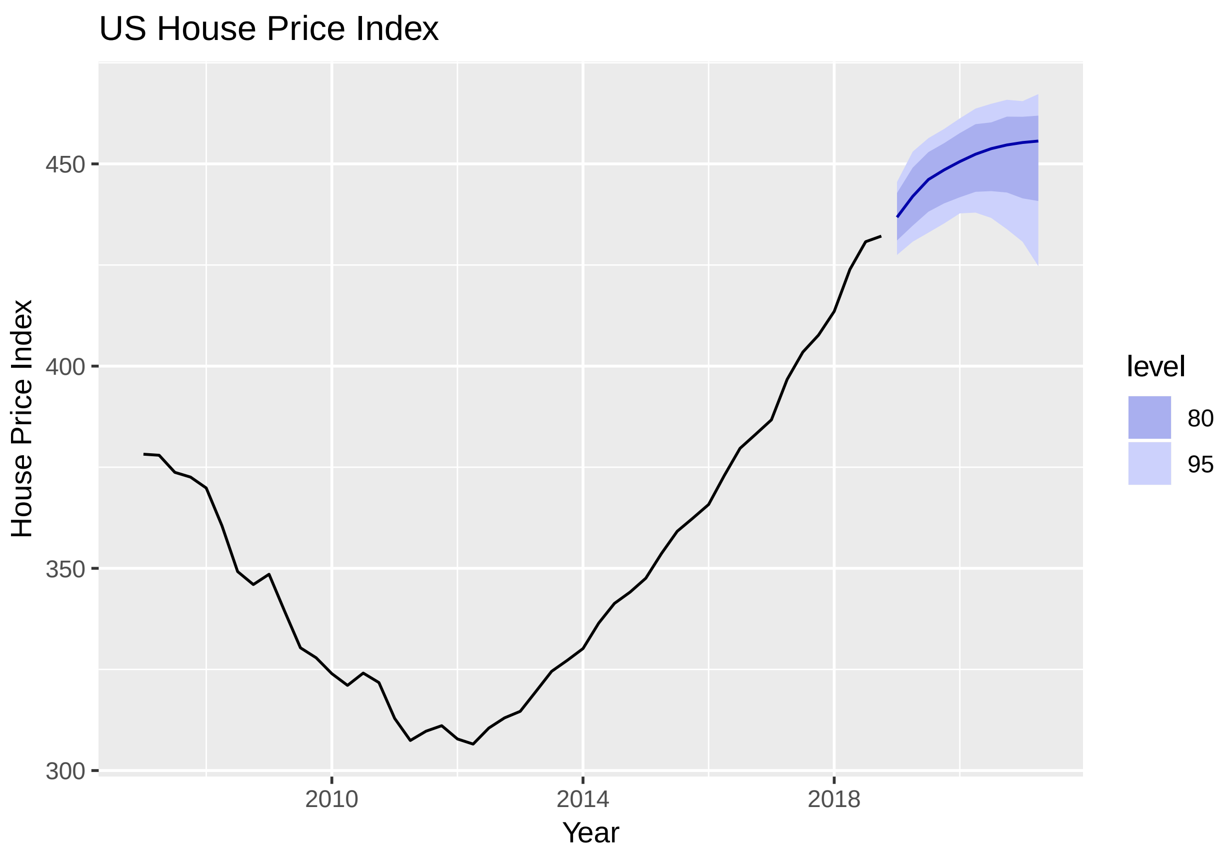
Yield Curve: Yield Curve increases but has a huge error in MAPE training set and that explains the exception .

```
lambda1 <- BoxCox.lambda(sp_500)
lambda2 <- BoxCox.lambda(hpi)
```

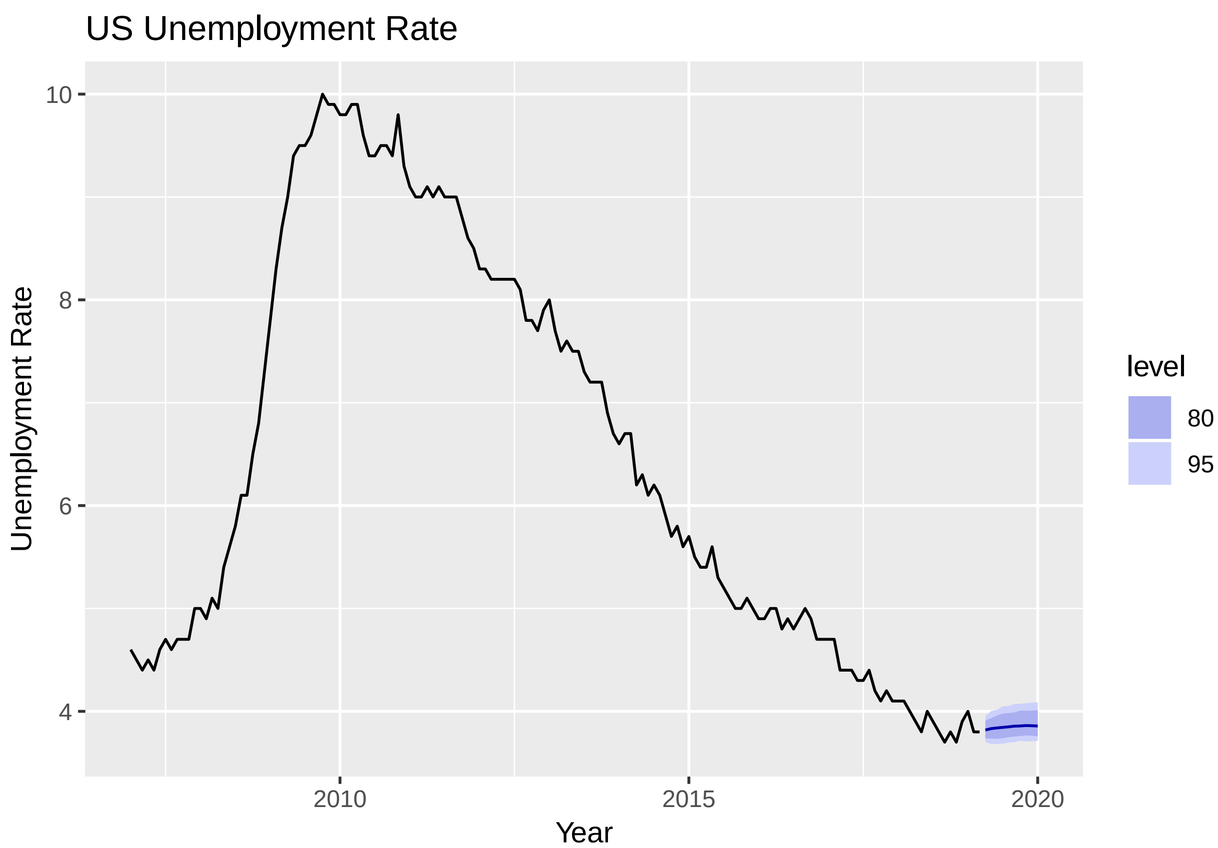
```r
lambda3 <- BoxCox.lambda(unemp)
lambda4 <- BoxCox.lambda(yield)
fit1 <- nnetar(sp_500, lambda = lambda1)
fit_net <- forecast(fit1, h = 10, PI = TRUE)
autoplot(fit_net,
    holdout = sp_500) + xlab("Year") +  ylab("S & P 500") + ggtitle("US S & P 500 Index")
```



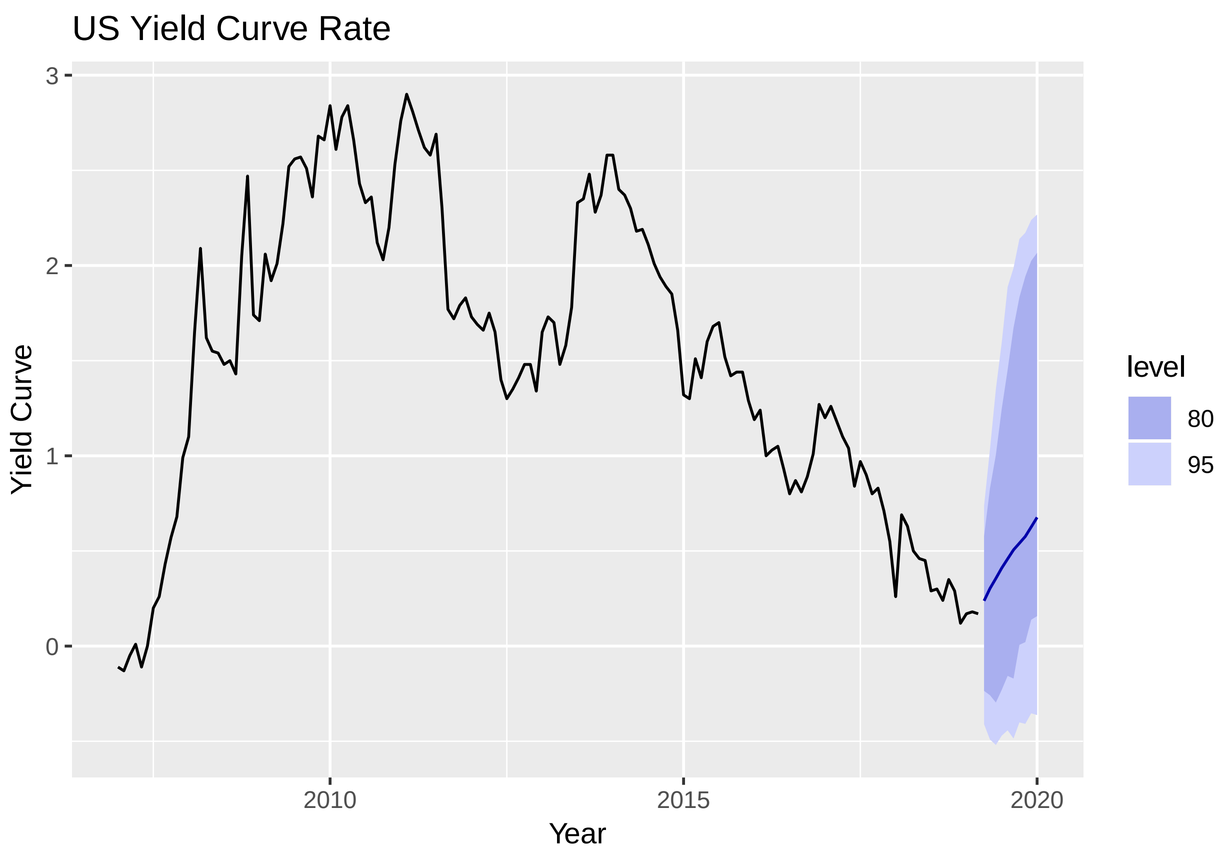

```r
fit2 <- nnetar(hpi, lambda = lambda2)
fit_net <- forecast(fit2, h = 10, PI = TRUE)
autoplot(fit_net,
    holdout = hpi) + xlab("Year") +  ylab("House Price Index") + ggtitle("US House Price Index")
```




```
fit3 <- nnetar(unemp, lambda = lambda3)
fit_net <- forecast(fit3, h = 10, PI = TRUE)
autoplot(fit_net,
    holdout = unemp) + xlab("Year") +  ylab("Unemployment Rate") + ggtitle("US Unemployment Rate")
```




```
fit4 <- nnetar(yield, lambda = lambda4)
fit_net <- forecast(fit4, h = 10, PI = TRUE)
autoplot(fit_net,
    holdout = yield)+ xlab("Year") +  ylab("Yield Curve") + ggtitle("US Yield Curve Rate")
```



Forecasting Accuracy for Neural Network Time series Analysis

Neural Network did pretty good on accuracy as MAE and MAPE errors had very less values except for the test set of yield curve.

```
# Forecasting accuracy for Neural Network

SP500 <- window(sp_500, start = 2007, end = c(2017,12))
test1 <- window(sp_500, start = c(2018))

hpiu <- window(hpi, start = 2007, end = c(2016,4))
test2 <- window(hpi, start = 2017)

unempu <- window(unemp, start = 2007, end = c(2017,12))
test3 <- window(unemp, start = c(2018))

yieldu <- window(yield, start = 2007, end = c(2017,12))
test4 <- window(yield, start = c(2018))


fit1 <- nnetar(SP500, lambda = lambda1)
train1 <- forecast(fit1, h=16, PI = TRUE)
accuracy(train1, test1)
```

```
##                        ME     RMSE      MAE        MPE     MAPE
## Training set   -0.5104534 55.48884 43.48498 -0.2167857 3.085928
```

```
## Test set     -354.0298352 400.09826 365.18418 -12.9753943 13.370405
##                   MASE        ACF1 Theil's U
## Training set 0.1986554 0.007113437        NA
## Test set     1.6682955 0.612357505  3.601867
```

```r
fit2 <- nnetar(hpiu, lambda = lambda2)
train2 <- forecast(fit2, h=08, PI = TRUE)
accuracy(train2, test2)
```

```
##                       ME      RMSE       MAE         MPE      MAPE
## Training set  0.02909082  2.627823  2.141891 0.000710134 0.6434871
## Test set     22.47755805 26.692915 22.477558 5.335632426 5.3356324
##                  MASE      ACF1 Theil's U
## Training set 0.144481 0.1171647        NA
## Test set     1.516221 0.6331938  3.854632
```

```r
fit3 <- nnetar(unempu, lambda = lambda3)
train3 <- forecast(fit3, h=15, PI = TRUE)
accuracy(train3, test3)
```

```
##                        ME      RMSE       MAE           MPE      MAPE
## Training set  0.003286837 0.1444131 0.1153201   0.001431267  1.758434
## Test set     -0.406530264 0.4734579 0.4065303 -10.632461536 10.632462
##                   MASE       ACF1 Theil's U
## Training set 0.1091357 -0.1739002        NA
## Test set     0.3847290  0.6738986  4.015894
```

```r
fit4 <- nnetar(yieldu, lambda = lambda4)
train4 <- forecast(fit4, h=15, PI = TRUE)
accuracy(train4, test4)
```

```
##                        ME      RMSE       MAE         MPE       MAPE
## Training set -0.002848568 0.1855713 0.1406402   -1.500037   8.544489
## Test set     -0.302908427 0.3584553 0.3166683 -145.592070 147.613164
##                   MASE      ACF1 Theil's U
## Training set 0.2305577 0.1639044        NA
## Test set     0.5191284 0.6984039  3.570568
```

CONCLUSIONS

Model Selection

We Did Forecasting with Naive Models, ETS models, ARIMA and Neural Network models for S&P 500 Data, US Unemployment Rate, US House Price Index and US Yield Curve. We saw the Errors associated with all the models and found that perhaps ARIMA model and Neural Networks have the lowest forecasting Errors.So, We can see combined ARIMA and Neural Network results with 95% confidence Interval to get the sense of the data.

We see S&P 500 data have clear deteorating optimism in both ARIMA and Neural Networks. It might mean inverstors will have their reservations going forward at the end of 2019.

According to ARIMA Ever Rising US house price Index is also a worrying sign as it might show the inflated market and psuedo-optimism. On the other hand neural network shows a caution in the market regarding the house price Index.

According to ARIMA Unemployment Rate shows a downward constant trend as this rate is the lowest of the decade and again a worrying sign of pseudo- optimism. On the other hand Neural network shows a slight decrease in employment rate and hence can be considered a caution by the market.

Yield curve is and has been the most accurate predictor of Recession. Yield Curve Inversion means that investors are less optimist about about the short term return on Bonds than long term (10 years) return. ARIMA model prediction of yield curve Inversion at the end of 2019 is a sign of Investor sentiments and the most worrying picture we have. Neural Network showsheathy yield curve but has huge error in MAPE.

All said, These are only predictors and are basically dependent on previous or lagged values. We can see the positive and negative sentiments but can't forecast anything with utmost certainity.

```
##################### THE END ##########################
```