Oil_Time_Series.R

rstudio-user

2019-12-23

```r
# Author:
#   RIHAD VATIAWA
#
# Time Series is a sequence of well-defined data points measured at consistent time
# intervals over a period of time. ... Time series analysis is the use of statistical
# methods to analyze time series data and extract meaningful statistics and characteristics about the d
#
# The goal of this project was to use various (ts) time series methods in order to
# make predictions on a dataset of our choice. The modeling techniques used were:
#
# ARIMA models
# Exponential Smoothing models
# Facebooks Prophet model
#
# The data sourced for our ts project came from the link below
# This data consists of monthly WTI (West Texas Intermediate) oil prices from Cushing,
# Oklahoma. The data extends from Jan 1986 upto 21 Dec 2019.
#   https://fred.stlouisfed.org/series/MCOILWTICO
#
# --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- #

# import libraries
library(fpp2)
```

```
## Loading required package: ggplot2

## 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
```

```r
library(astsa)
```

```
##
## Attaching package: 'astsa'

## The following object is masked from 'package:fpp2':
```

```
##
##     oil
## The following objects are masked from 'package:fma':
##
##     chicken, sales
## The following object is masked from 'package:forecast':
##
##     gas
```

```r
library(tseries)
library(forecast)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

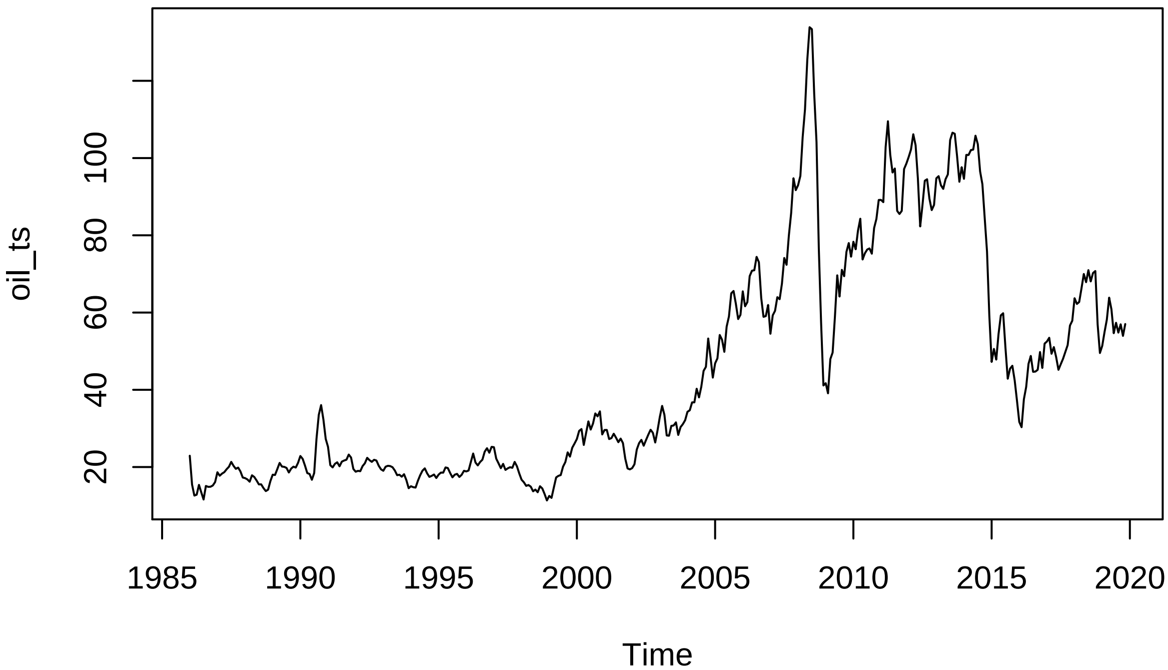
```r
library(prophet)
```

```
## Loading required package: Rcpp
## Loading required package: rlang
```

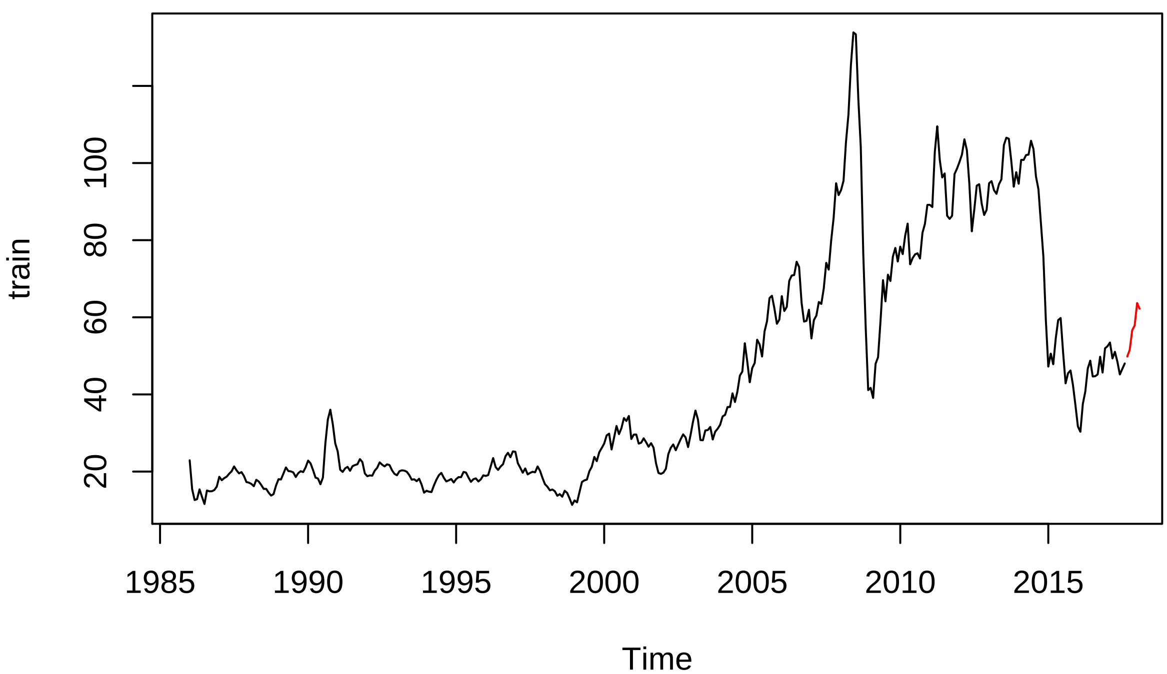
```r
# load in data
oil <- read.csv('MCOILWTICO.csv')

# preview the data
head(oil)
```

```
##         DATE MCOILWTICO
## 1 1986-01-01      22.93
## 2 1986-02-01      15.46
## 3 1986-03-01      12.61
## 4 1986-04-01      12.84
## 5 1986-05-01      15.38
## 6 1986-06-01      13.43
```

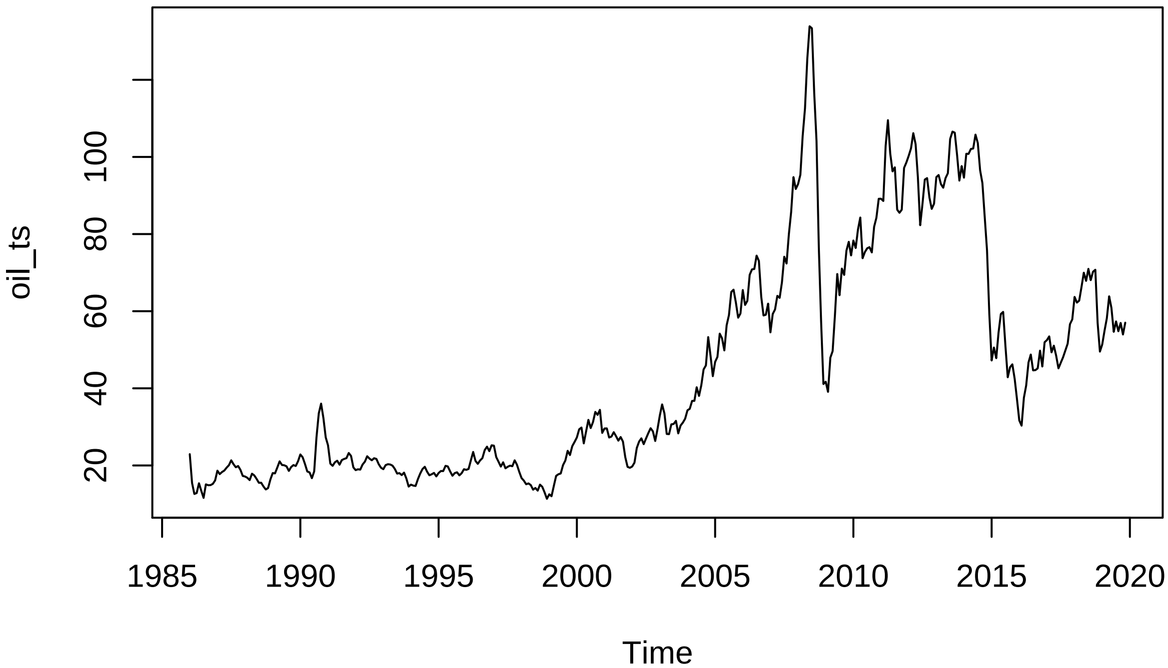
```r
# glimpse(oil)

# transforming data into ts
oil_ts <- ts(oil$MCOILWTICO, start = c(1986, 1), frequency = 12)

# visualizing our ts data
plot(oil_ts)
```



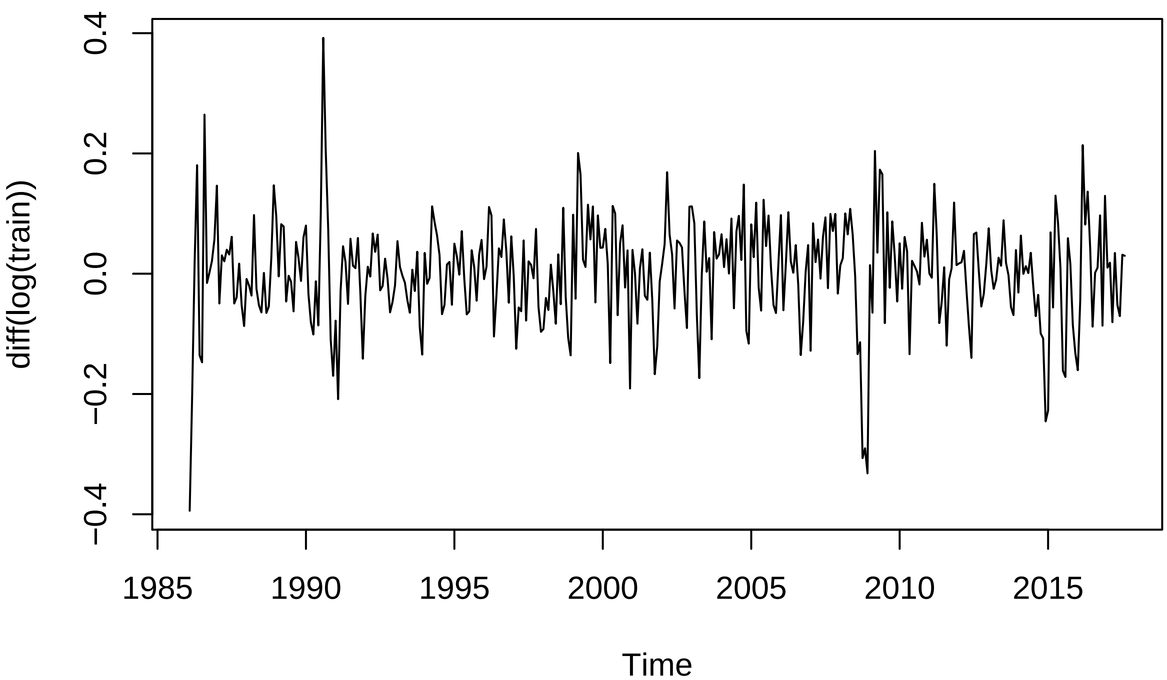

```r
# training data - all data except last six months: YR, MONTH
train <- window(oil_ts, c(1986, 1), c(2017, 8))
# plot(train)

# testing data - just the last six months: YR, MONTH
test <- window(oil_ts, c(2017, 9), c(2018, 2))
# plot(test)

# visualizing train and test data
plot(train)
lines(test, col = 'red')
```



```r
################
# ARIMA Model
################

## Step 1: Visualize the ts
# for obvious trend and increasing variance
plot(oil_ts)
```




```r
## Step 2: Check for stationarity
# using a diff and a log to stationarize the data
plot(diff(log(train)))
```



```r
adf.test(train)
```

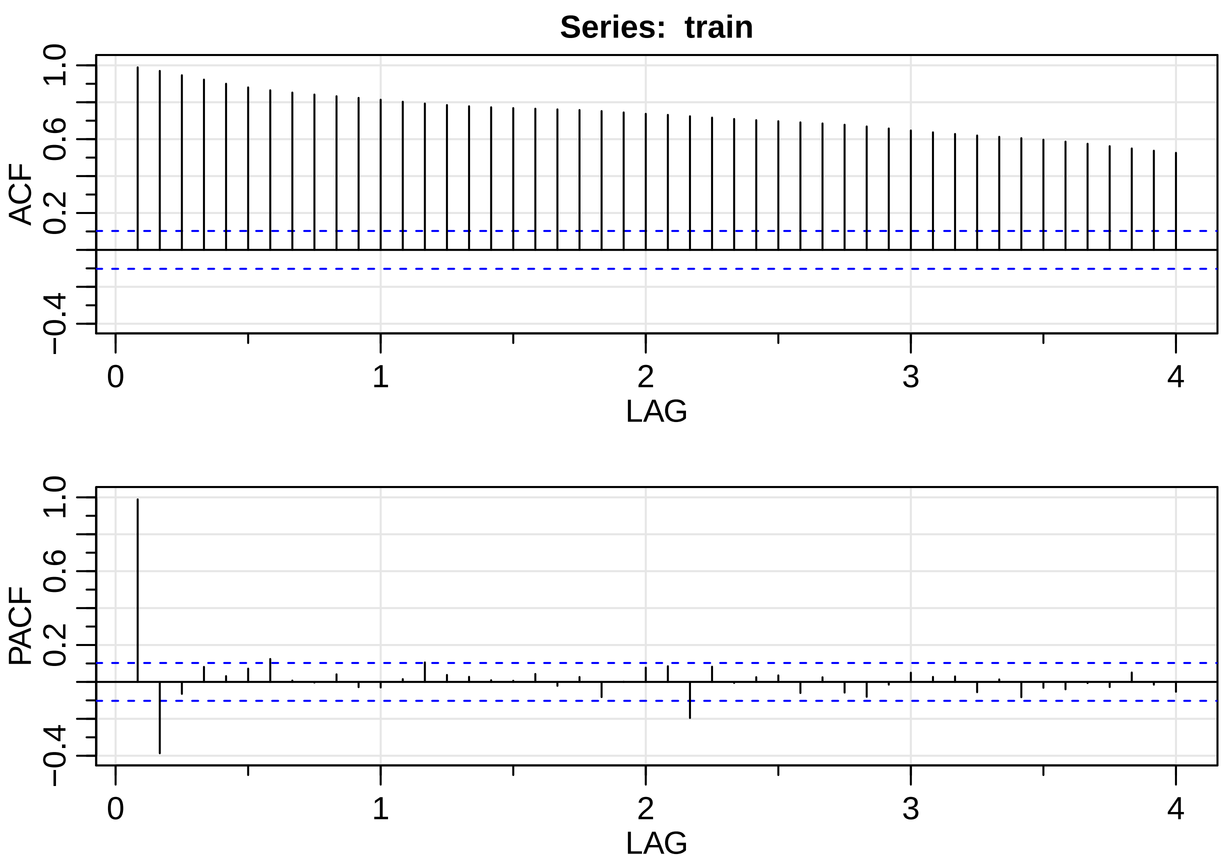
```
##
##  Augmented Dickey-Fuller Test
##
## data:  train
## Dickey-Fuller = -2.2622, Lag order = 7, p-value = 0.4664
## alternative hypothesis: stationary
```

```r
# passes the Dickey Fuller test
adf.test(diff(log(train)))
```

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(log(train))
## Dickey-Fuller = -7.7017, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```r
# acf tails off - pacf kinda tails sorta
acf2(train)
```




```
##        ACF  PACF
##  [1,] 0.99  0.99
##  [2,] 0.97 -0.39
##  [3,] 0.95 -0.06
##  [4,] 0.92  0.08
```

```
##  [5,] 0.90  0.03
##  [6,] 0.88  0.07
##  [7,] 0.86  0.12
##  [8,] 0.85  0.01
##  [9,] 0.84  0.00
## [10,] 0.83  0.04
## [11,] 0.82 -0.03
## [12,] 0.81 -0.03
## [13,] 0.80  0.02
## [14,] 0.79  0.11
## [15,] 0.78  0.04
## [16,] 0.78  0.03
## [17,] 0.77  0.01
## [18,] 0.77  0.01
## [19,] 0.77  0.04
## [20,] 0.76 -0.02
## [21,] 0.76  0.03
## [22,] 0.75 -0.08
## [23,] 0.74  0.00
## [24,] 0.74  0.08
## [25,] 0.73  0.09
## [26,] 0.72 -0.20
## [27,] 0.72  0.08
## [28,] 0.71 -0.01
## [29,] 0.70  0.03
## [30,] 0.70  0.04
## [31,] 0.69 -0.06
## [32,] 0.69  0.03
## [33,] 0.68 -0.06
## [34,] 0.67 -0.08
## [35,] 0.66 -0.01
## [36,] 0.65  0.05
## [37,] 0.64  0.03
## [38,] 0.63  0.03
## [39,] 0.62 -0.06
## [40,] 0.61  0.01
## [41,] 0.61 -0.08
## [42,] 0.60 -0.03
## [43,] 0.59 -0.04
## [44,] 0.58 -0.01
## [45,] 0.56 -0.03
## [46,] 0.55  0.05
## [47,] 0.54 -0.02
## [48,] 0.53 -0.05
```

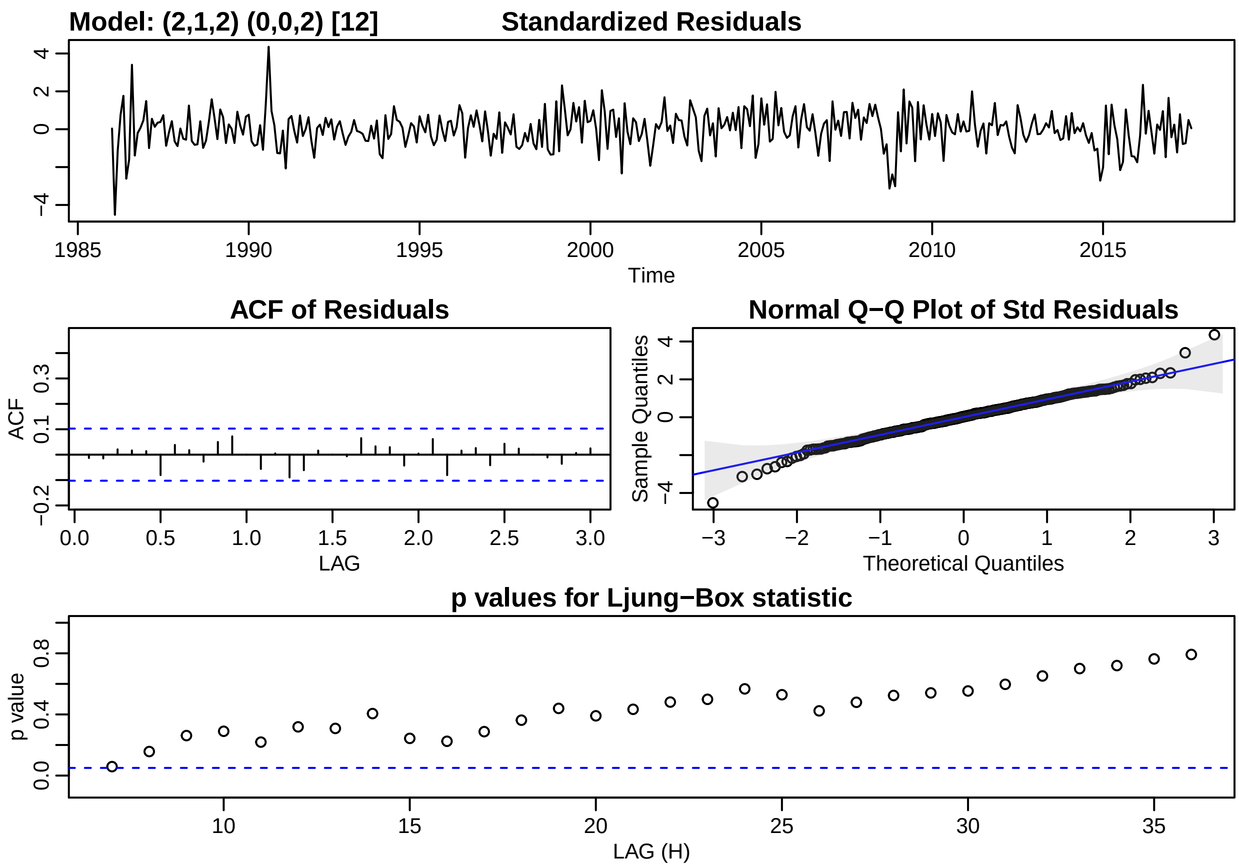
```r
## ACF and PACF
# Autocorrelation refers to how correlated a ts (future value) is with its past values
# whereas the ACF is the plot used to see the correlation (relationship) between
# the points, upto and including the lag unit.
# In ACF, the correlation coefficient is in the y-axis whereas the number
# of lags is shown in the x-axis.

# suggests: p=2, d=1, q=2, P=0, D=0, Q=2, S=12
auto.arima(train)
```

```
## Series: train
## ARIMA(2,1,2)(0,0,2)[12]
##
## Coefficients:
##          ar1      ar2      ma1     ma2    sma1     sma2
##       1.5596  -0.6568  -1.1949  0.2760  0.1019  -0.1567
## s.e.  0.0829   0.0787   0.1039  0.0976  0.0561   0.0537
##
## sigma^2 estimated as 14.86:  log likelihood=-1046.74
## AIC=2107.49   AICc=2107.79   BIC=2135.05
```

```
# first model --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
fit <- sarima(log(train), p=2, d=1, q=2, P=0, D=0, Q=2, S=12)
```

```
## initial  value -2.469201
## iter   2 value -2.473196
## iter   3 value -2.507314
## iter   4 value -2.507671
## iter   5 value -2.508150
## iter   6 value -2.508465
## iter   7 value -2.509452
## iter   8 value -2.510230
## iter   9 value -2.510993
## iter  10 value -2.511360
## iter  11 value -2.511628
## iter  12 value -2.511816
## iter  13 value -2.511859
## iter  14 value -2.511885
## iter  15 value -2.511928
## iter  16 value -2.511982
## iter  17 value -2.511991
## iter  18 value -2.511996
## iter  19 value -2.511999
## iter  20 value -2.512001
## iter  21 value -2.512003
## iter  22 value -2.512004
## iter  23 value -2.512006
## iter  24 value -2.512007
## iter  25 value -2.512007
## iter  25 value -2.512007
## iter  25 value -2.512007
## final  value -2.512007
## converged
## initial  value -2.473238
## iter   2 value -2.473383
## iter   3 value -2.475430
## iter   4 value -2.475865
## iter   5 value -2.476714
## iter   6 value -2.478409
## iter   7 value -2.479678
## iter   8 value -2.479849
## iter   9 value -2.479890
## iter  10 value -2.479954
## iter  11 value -2.479958
## iter  12 value -2.479960
```

```
## iter  13 value -2.479960
## iter  14 value -2.479962
## iter  15 value -2.479966
## iter  16 value -2.479977
## iter  17 value -2.479981
## iter  18 value -2.479989
## iter  19 value -2.479990
## iter  20 value -2.479996
## iter  21 value -2.480006
## iter  22 value -2.480017
## iter  23 value -2.480025
## iter  24 value -2.480034
## iter  25 value -2.480054
## iter  26 value -2.480218
## iter  27 value -2.480314
## iter  28 value -2.480373
## iter  29 value -2.480417
## iter  30 value -2.480507
## iter  31 value -2.480630
## iter  32 value -2.480673
## iter  33 value -2.480689
## iter  34 value -2.480716
## iter  35 value -2.480801
## iter  36 value -2.481024
## iter  37 value -2.481400
## iter  38 value -2.482000
## iter  39 value -2.482356
## iter  40 value -2.483354
## iter  41 value -2.484759
## iter  42 value -2.485638
## iter  43 value -2.487146
## iter  44 value -2.487382
## iter  45 value -2.487570
## iter  46 value -2.487763
## iter  47 value -2.488081
## iter  48 value -2.488452
## iter  49 value -2.488586
## iter  50 value -2.488598
## iter  51 value -2.488609
## iter  52 value -2.488610
## iter  53 value -2.488610
## iter  54 value -2.488610
## iter  55 value -2.488611
## iter  56 value -2.488611
## iter  56 value -2.488611
## final  value -2.488611
## converged
```



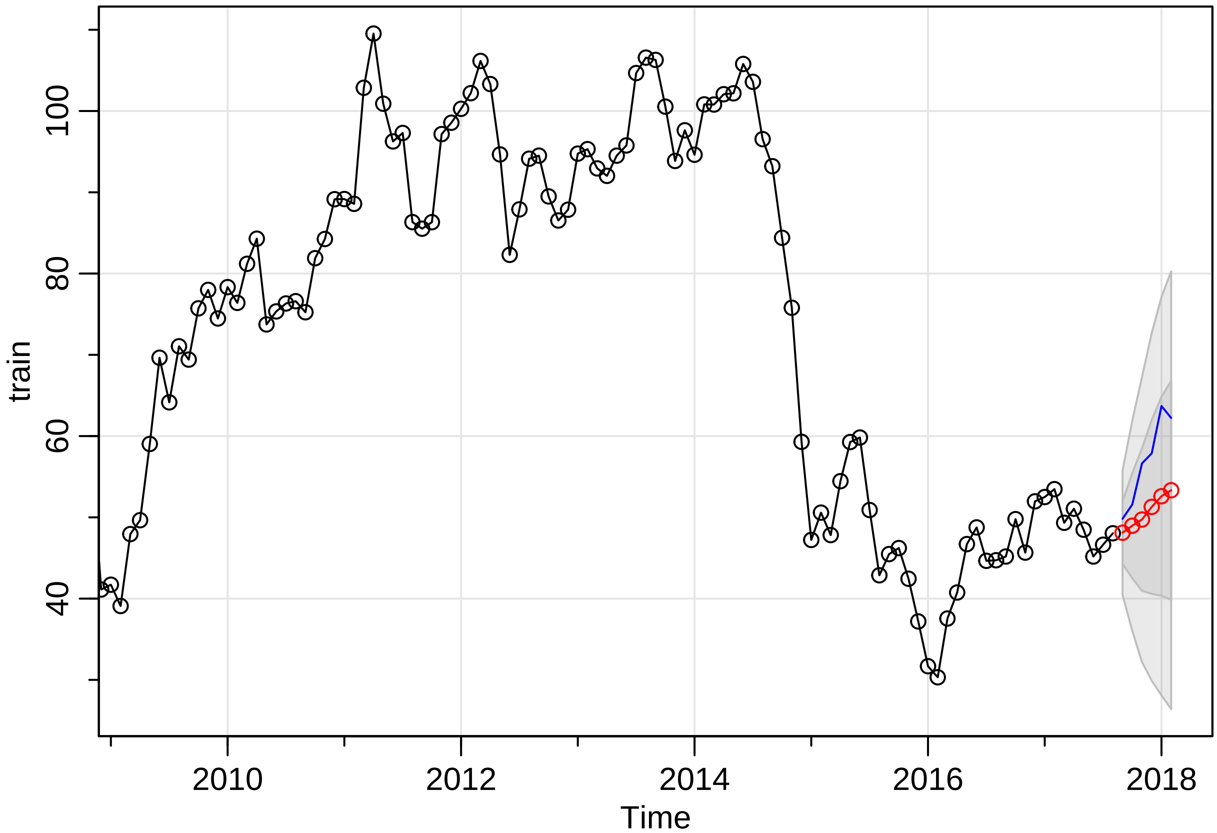

```r
# preview AIC and BIC of model
print(cbind(fit$AIC, fit$BIC))
```

```
##              [,1]      [,2]
## [1,] -2.097129 -2.014015
```

```r
fit$ttable
```

```
##          Estimate     SE t.value p.value
## ar1        1.2917 0.1657  7.7968  0.0000
## ar2       -0.3973 0.1621 -2.4502  0.0147
## ma1       -1.0174 0.1792 -5.6768  0.0000
## ma2        0.0921 0.1757  0.5244  0.6003
## sma1       0.0448 0.0577  0.7762  0.4381
## sma2      -0.0676 0.0545 -1.2398  0.2158
## constant   0.0024 0.0030  0.7891  0.4306
```

```r
sarima_oil <- as.ts(sarima.for(train, n.ahead = 6, p=2, d=1, q=2, P=0, D=0, Q=2, S=12))

# compare predicted and actual
lines(test, col = 'blue')
```



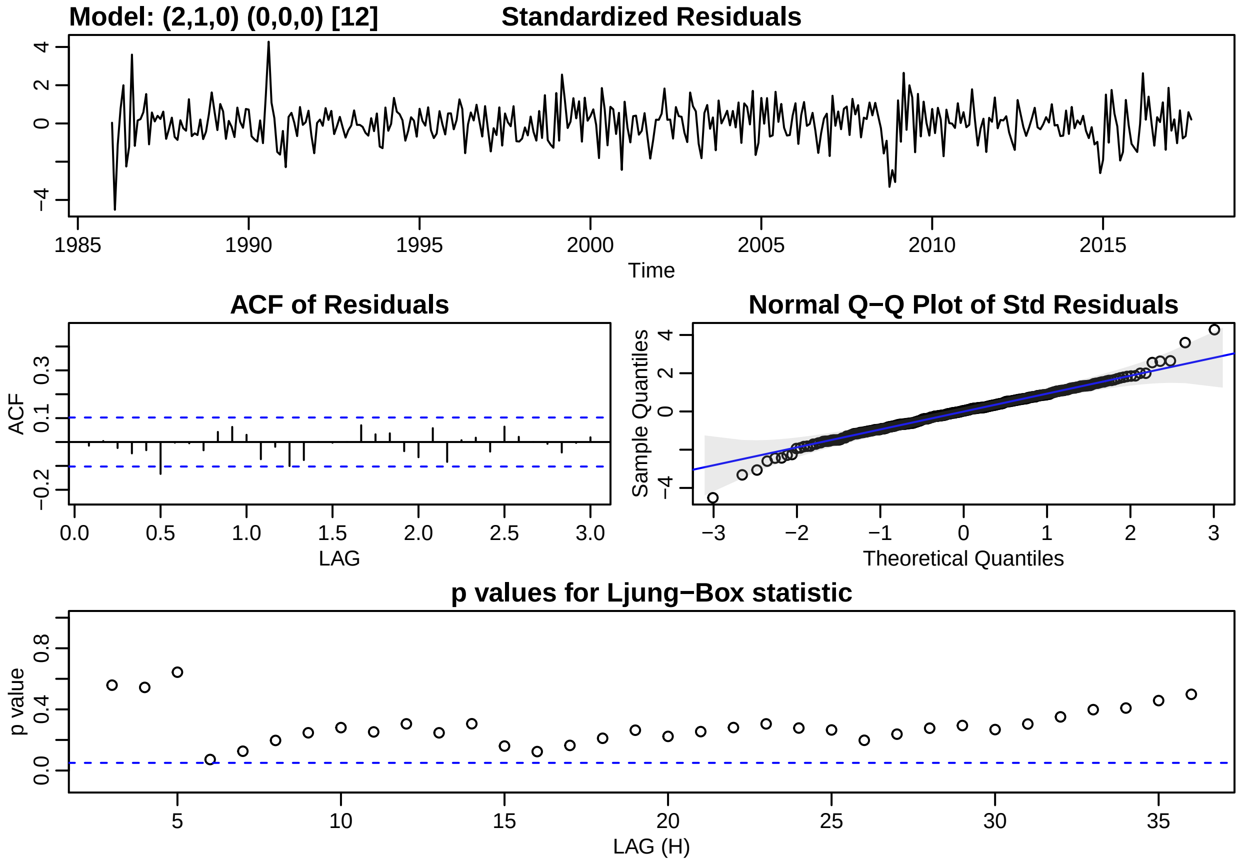

```r
# preview accuracy of model
accuracy(sarima_oil$pred, test)
```

```
##                 ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set 6.304671 7.111124 6.304671 10.63671 10.63671 0.4423583  2.029021
```

```r
# second model --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
fit2 <- sarima(log(train), p=2, d=1, q=0, P=0, D=0, Q=0, S=12)
```

```
## initial  value -2.469201
## iter   2 value -2.501140
## iter   3 value -2.504276
## iter   4 value -2.504299
## iter   5 value -2.504319
## iter   6 value -2.504330
## iter   7 value -2.504331
## iter   8 value -2.504332
## iter   8 value -2.504332
## final  value -2.504332
## converged
## initial  value -2.476313
## iter   2 value -2.476779
## iter   3 value -2.476819
## iter   4 value -2.476884
## iter   5 value -2.476896
## iter   6 value -2.476897
## iter   6 value -2.476897
## iter   6 value -2.476897
```

```
## final  value -2.476897
## converged
```



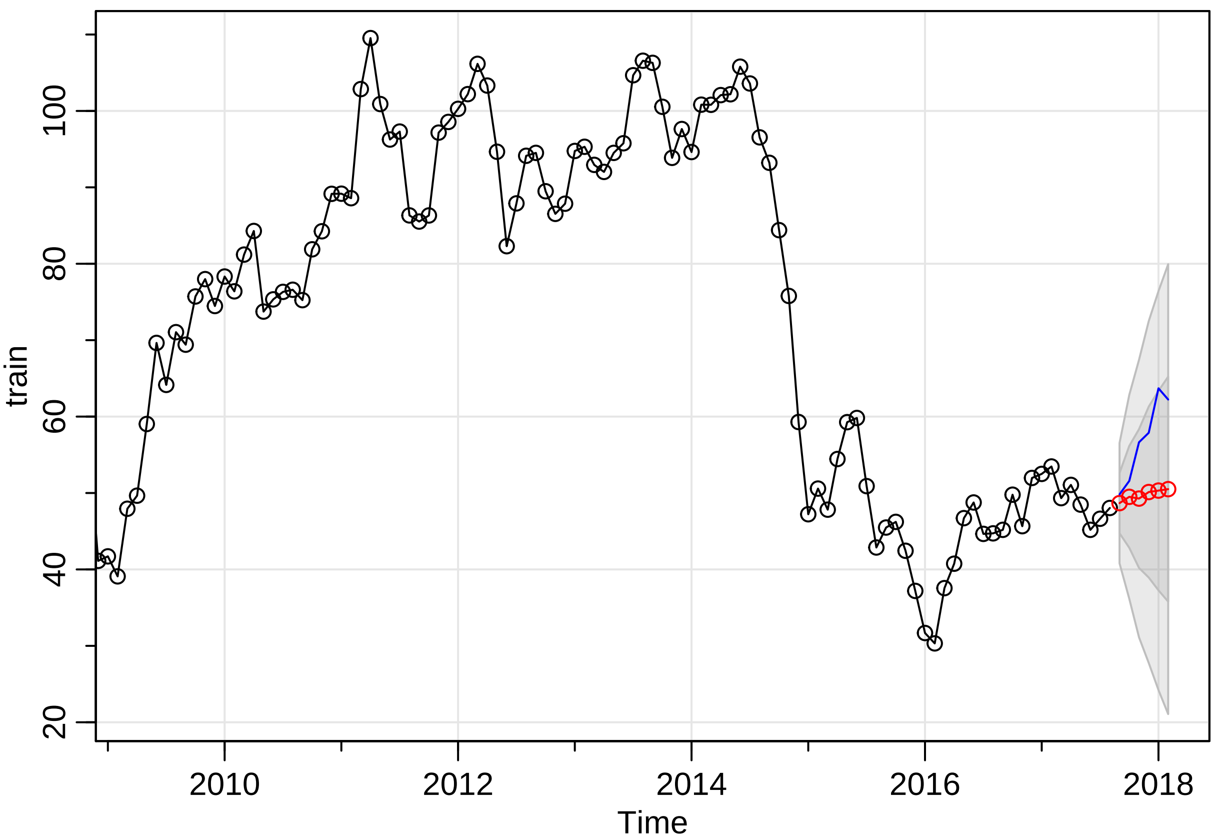

```r
# preview AIC and BIC of model
print(cbind(fit2$AIC, fit2$BIC))
```

```
##           [,1]      [,2]
## [1,] -2.094808 -2.053251
```

```r
fit2$ttable
```

```
##          Estimate     SE t.value p.value
## ar1        0.2932 0.0524  5.5945  0.0000
## ar2       -0.0303 0.0528 -0.5744  0.5661
## constant   0.0016 0.0059  0.2773  0.7817
```

```r
sarima_oil2 <- as.ts(sarima.for(train, n.ahead = 6, p=2, d=1, q=0, P=0, D=0, Q=1, S=12))

# compare predicted and actual
lines(test, col = 'blue')
```




```r
# preview accuracy of model
accuracy(sarima_oil2$pred, test)
```

```
##                 ME     RMSE      MAE     MPE    MAPE      ACF1 Theil's U
## Test set 7.242352 8.530619 7.242352 12.0975 12.0975 0.5049333  2.414631
```

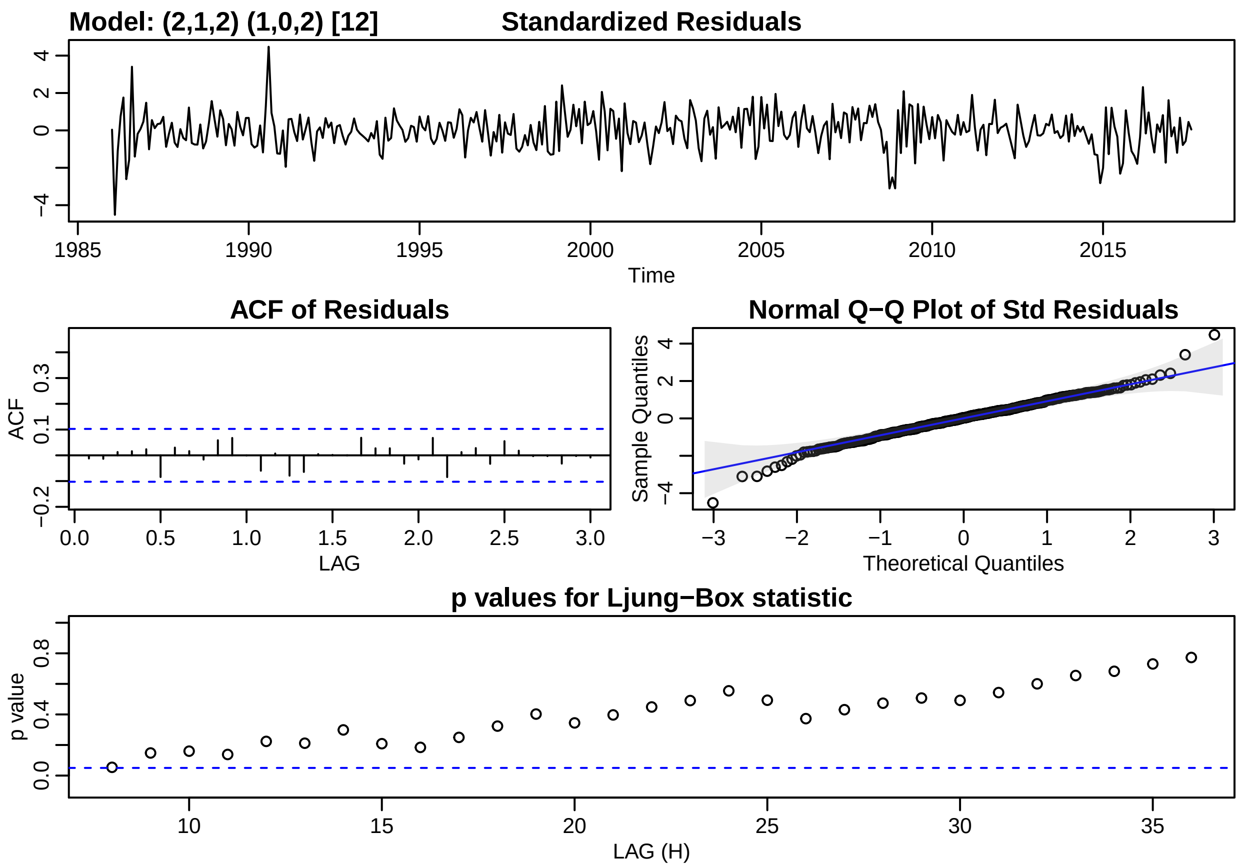
```r
# third model --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
fit3 <- sarima(log(train), p=2, d=1, q=2, P=1, D=0, Q=2, S=12)
```

```
## initial  value -2.485585
## iter   2 value -2.493045
## iter   3 value -2.537189
## iter   4 value -2.537724
## iter   5 value -2.538939
## iter   6 value -2.539211
## iter   7 value -2.541125
## iter   8 value -2.542098
## iter   9 value -2.542872
## iter  10 value -2.543224
## iter  11 value -2.543589
## iter  12 value -2.544106
## iter  13 value -2.544279
## iter  14 value -2.544345
## iter  15 value -2.544400
## iter  16 value -2.544445
## iter  17 value -2.544546
## iter  18 value -2.544610
## iter  19 value -2.544650
```

```
## iter  20 value -2.544763
## iter  21 value -2.544814
## iter  22 value -2.544848
## iter  23 value -2.544934
## iter  24 value -2.545387
## iter  25 value -2.545919
## iter  26 value -2.546460
## iter  27 value -2.547245
## iter  28 value -2.547501
## iter  29 value -2.547810
## iter  30 value -2.549027
## iter  31 value -2.549815
## iter  32 value -2.550803
## iter  33 value -2.551083
## iter  34 value -2.551307
## iter  35 value -2.551353
## iter  36 value -2.551371
## iter  37 value -2.551409
## iter  38 value -2.551456
## iter  39 value -2.551512
## iter  40 value -2.551537
## iter  41 value -2.551537
## iter  42 value -2.551538
## iter  43 value -2.551538
## iter  44 value -2.551538
## iter  45 value -2.551539
## iter  46 value -2.551539
## iter  47 value -2.551541
## iter  48 value -2.551548
## iter  49 value -2.551557
## iter  50 value -2.551639
## iter  51 value -2.551694
## iter  52 value -2.551711
## iter  53 value -2.551718
## iter  54 value -2.551725
## iter  55 value -2.551728
## iter  56 value -2.551730
## iter  57 value -2.551731
## iter  58 value -2.551732
## iter  59 value -2.551732
## iter  60 value -2.551733
## iter  61 value -2.551733
## iter  62 value -2.551733
## iter  63 value -2.551733
## iter  64 value -2.551733
## iter  65 value -2.551733
## iter  65 value -2.551733
## iter  65 value -2.551733
## final  value -2.551733
## converged
## initial  value -2.483153
## iter   2 value -2.484372
## iter   3 value -2.484456
## iter   4 value -2.485151
```

```
## iter   5 value -2.485515
## iter   6 value -2.485607
## iter   7 value -2.486053
## iter   8 value -2.486894
## iter   9 value -2.487590
## iter  10 value -2.488060
## iter  11 value -2.488326
## iter  12 value -2.488758
## iter  13 value -2.488780
## iter  14 value -2.488805
## iter  15 value -2.488814
## iter  16 value -2.488817
## iter  17 value -2.488820
## iter  18 value -2.488825
## iter  19 value -2.488825
## iter  20 value -2.488827
## iter  21 value -2.488828
## iter  22 value -2.488831
## iter  23 value -2.488835
## iter  24 value -2.488841
## iter  25 value -2.488844
## iter  26 value -2.488844
## iter  27 value -2.488845
## iter  28 value -2.488845
## iter  29 value -2.488846
## iter  30 value -2.488847
## iter  31 value -2.488848
## iter  32 value -2.488849
## iter  33 value -2.488849
## iter  34 value -2.488850
## iter  35 value -2.488852
## iter  36 value -2.488853
## iter  37 value -2.488854
## iter  38 value -2.488855
## iter  39 value -2.488855
## iter  40 value -2.488857
## iter  41 value -2.488860
## iter  42 value -2.488863
## iter  43 value -2.488864
## iter  44 value -2.488865
## iter  45 value -2.488866
## iter  46 value -2.488869
## iter  47 value -2.488877
## iter  48 value -2.488896
## iter  49 value -2.488905
## iter  50 value -2.488935
## iter  51 value -2.488953
## iter  52 value -2.488997
## iter  53 value -2.489051
## iter  54 value -2.489094
## iter  55 value -2.489208
## iter  56 value -2.489228
## iter  57 value -2.489424
## iter  58 value -2.489537
```

```
## iter  59 value -2.489696
## iter  60 value -2.489766
## iter  61 value -2.489854
## iter  62 value -2.490028
## iter  63 value -2.490299
## iter  64 value -2.490806
## iter  65 value -2.490868
## iter  66 value -2.491072
## iter  67 value -2.491154
## iter  68 value -2.491206
## iter  69 value -2.491244
## iter  70 value -2.491246
## iter  71 value -2.491250
## iter  72 value -2.491301
## iter  73 value -2.491392
## iter  74 value -2.491439
## iter  75 value -2.491458
## iter  76 value -2.491463
## iter  77 value -2.491466
## iter  78 value -2.491471
## iter  79 value -2.491477
## iter  80 value -2.491483
## iter  81 value -2.491485
## iter  82 value -2.491485
## iter  83 value -2.491486
## iter  84 value -2.491486
## iter  85 value -2.491487
## iter  86 value -2.491489
## iter  87 value -2.491489
## iter  88 value -2.491490
## iter  89 value -2.491490
## iter  90 value -2.491490
## iter  91 value -2.491490
## iter  92 value -2.491491
## iter  93 value -2.491492
## iter  94 value -2.491492
## iter  94 value -2.491492
## iter  94 value -2.491492
## final  value -2.491492
## converged
```



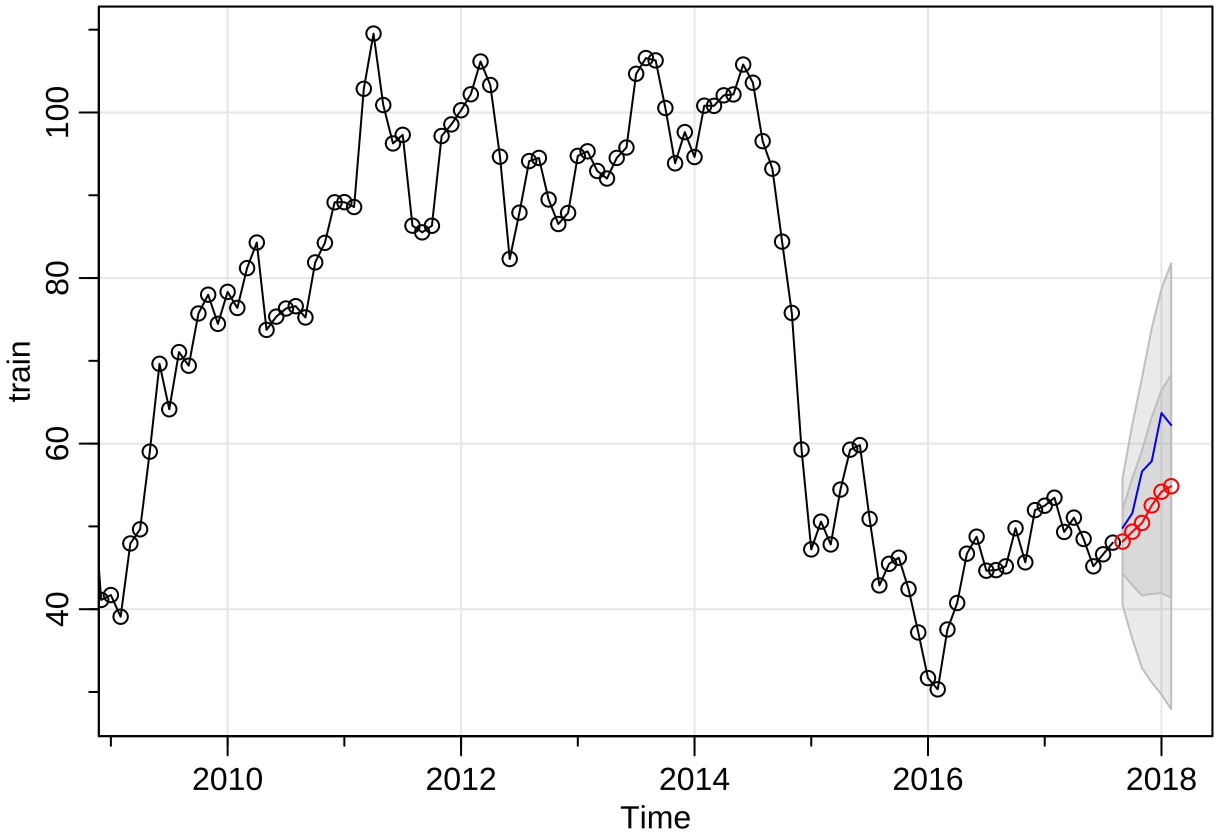

```r
# preview AIC and BIC of model
print(cbind(fit3$AIC, fit3$BIC))
```

```
##           [,1]     [,2]
## [1,] -2.097613 -2.00411
```

```r
fit3$ttable
```

```
##          Estimate     SE t.value p.value
## ar1        1.2800 0.1658  7.7199  0.0000
## ar2       -0.3836 0.1628 -2.3558  0.0190
## ma1       -1.0034 0.1785 -5.6220  0.0000
## ma2        0.0755 0.1757  0.4297  0.6676
## sar1      -0.8884 0.1121 -7.9290  0.0000
## sma1       0.9457 0.1370  6.9043  0.0000
## sma2      -0.0067 0.0696 -0.0963  0.9234
## constant   0.0023 0.0031  0.7465  0.4559
```

```r
sarima_oil3 <- as.ts(sarima.for(train, n.ahead = 6, p=2, d=1, q=2, P=1, D=0, Q=2, S=12))

# compare predicted and actual
lines(test, col = 'blue')
```




```
# preview accuracy of model
accuracy(sarima_oil3$pred, test)
```

```
##                ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set 5.389568 6.054024 5.389568 9.107386 9.107386 0.3763519  1.730619
```

```
# forth model --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
fit4 <- sarima(log(train), p=2, d=1, q=1, P=1, D=0, Q=1, S=12)
```

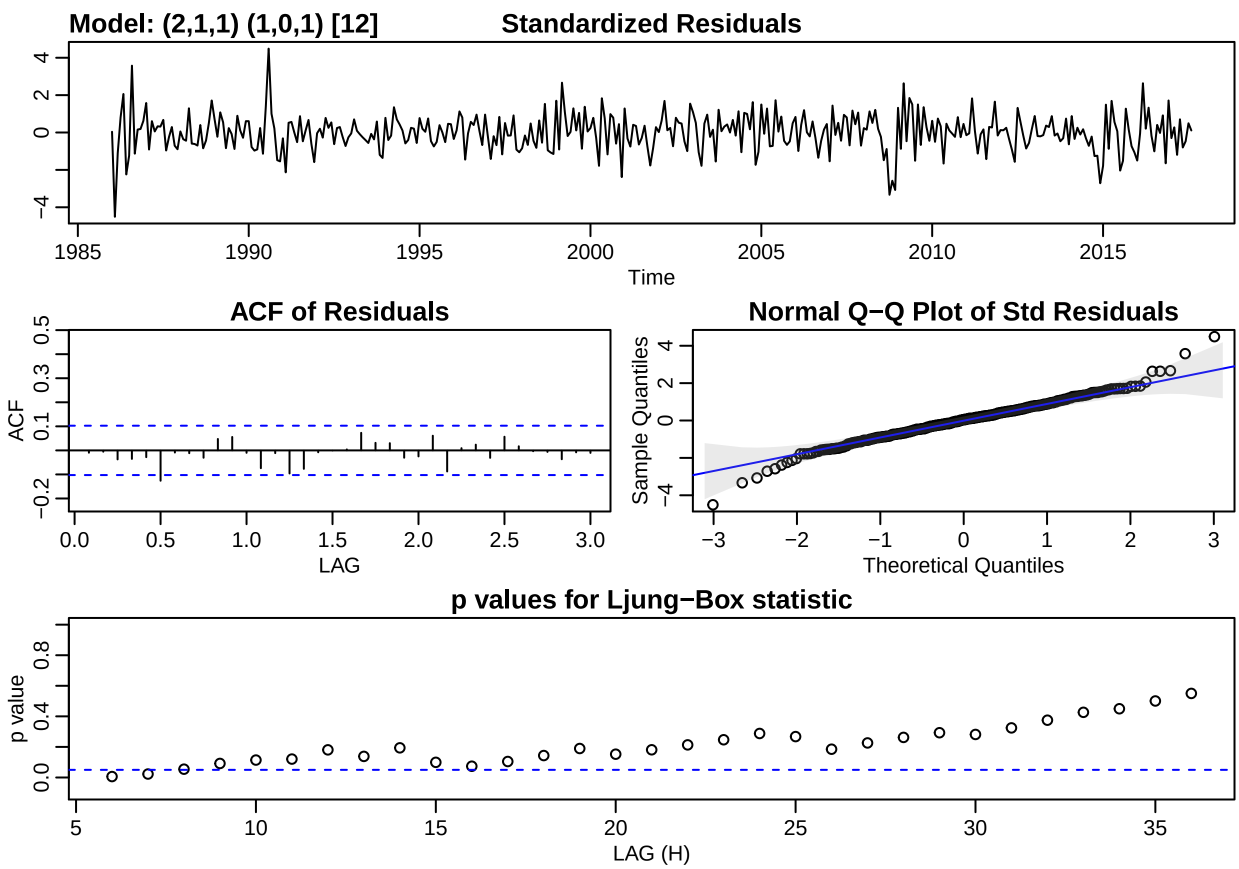
```
## initial  value -2.485585
## iter   2 value -2.487772
## iter   3 value -2.531278
## iter   4 value -2.532389
## iter   5 value -2.532546
## iter   6 value -2.532822
## iter   7 value -2.533527
## iter   8 value -2.534885
## iter   9 value -2.535602
## iter  10 value -2.536139
## iter  11 value -2.536745
## iter  12 value -2.537083
## iter  13 value -2.537138
## iter  14 value -2.537181
## iter  15 value -2.537191
## iter  16 value -2.537218
## iter  17 value -2.537221
## iter  18 value -2.537222
## iter  19 value -2.537222
```

```
## iter  20 value -2.537223
## iter  21 value -2.537226
## iter  22 value -2.537227
## iter  23 value -2.537228
## iter  24 value -2.537230
## iter  25 value -2.537235
## iter  26 value -2.537245
## iter  27 value -2.537262
## iter  28 value -2.537264
## iter  29 value -2.537280
## iter  30 value -2.537281
## iter  31 value -2.537289
## iter  32 value -2.537290
## iter  33 value -2.537290
## iter  34 value -2.537290
## iter  35 value -2.537290
## iter  36 value -2.537291
## iter  37 value -2.537292
## iter  38 value -2.537294
## iter  39 value -2.537297
## iter  40 value -2.537300
## iter  41 value -2.537301
## iter  42 value -2.537302
## iter  42 value -2.537302
## iter  42 value -2.537302
## final  value -2.537302
## converged
## initial  value -2.476130
## iter   2 value -2.477311
## iter   3 value -2.478376
## iter   4 value -2.478395
## iter   5 value -2.478416
## iter   6 value -2.478435
## iter   7 value -2.478467
## iter   8 value -2.478489
## iter   9 value -2.478512
## iter  10 value -2.478548
## iter  11 value -2.478634
## iter  12 value -2.478755
## iter  13 value -2.479018
## iter  14 value -2.479514
## iter  15 value -2.479611
## iter  16 value -2.479850
## iter  17 value -2.479885
## iter  18 value -2.480058
## iter  19 value -2.480196
## iter  20 value -2.480402
## iter  21 value -2.480672
## iter  22 value -2.481116
## iter  23 value -2.481888
## iter  24 value -2.482037
## iter  25 value -2.482118
## iter  26 value -2.482164
## iter  27 value -2.482172
```

```
## iter  28 value -2.482173
## iter  29 value -2.482177
## iter  30 value -2.482177
## iter  31 value -2.482178
## iter  32 value -2.482180
## iter  33 value -2.482181
## iter  34 value -2.482182
## iter  35 value -2.482184
## iter  36 value -2.482188
## iter  37 value -2.482191
## iter  38 value -2.482192
## iter  39 value -2.482192
## iter  40 value -2.482192
## iter  40 value -2.482193
## final  value -2.482193
## converged
```

```
## Warning in sqrt(diag(fitit$var.coef)): NaNs produced
```

```
## Warning in sqrt(diag(fitit$var.coef)): NaNs produced
```



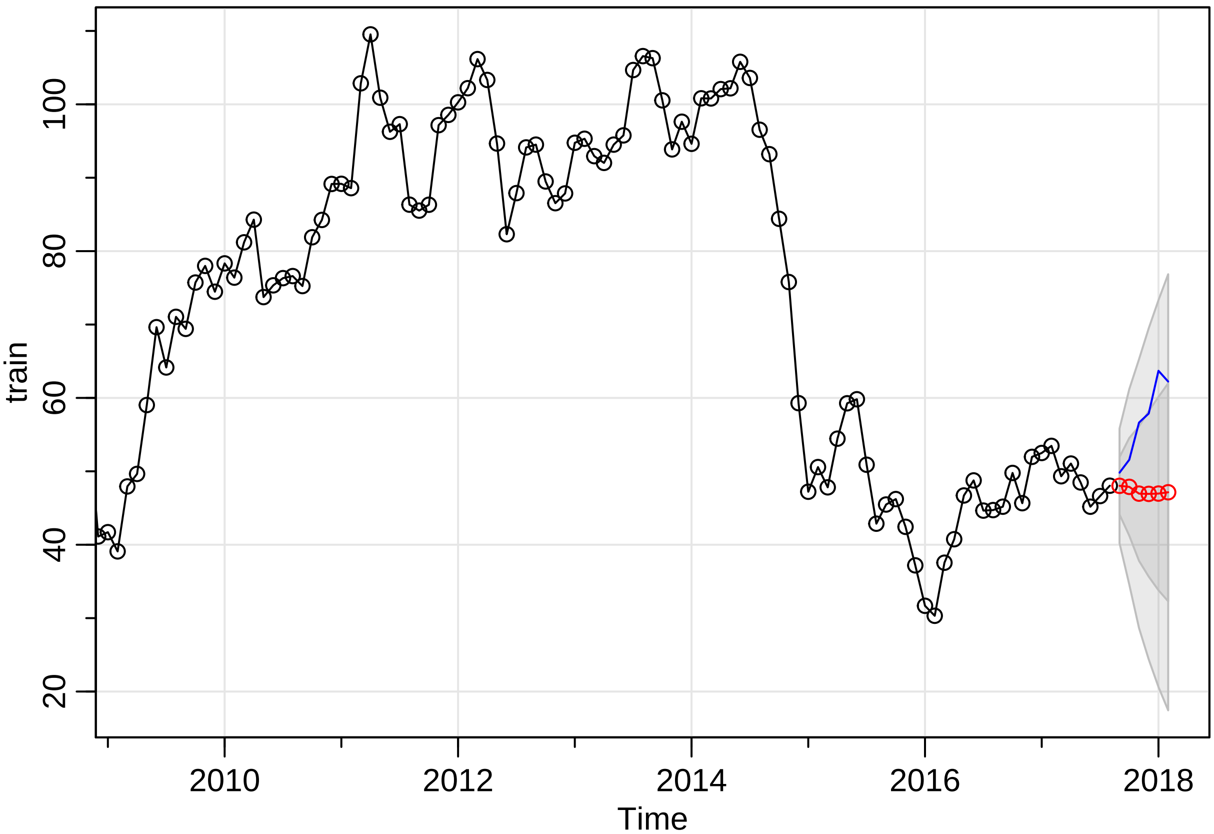

```
# preview AIC and BIC of model
print(cbind(fit4$AIC, fit4$BIC))
```

```
##           [,1]      [,2]
## [1,] -2.089569 -2.016844
```

```r
fit4$ttable
```

```
##          Estimate     SE t.value p.value
## ar1       -0.1896    NaN     NaN     NaN
## ar2        0.1138    NaN     NaN     NaN
## ma1        0.4823    NaN     NaN     NaN
## sar1      -0.9049 0.0920 -9.8344  0.0000
## sma1       0.9661 0.0909 10.6311  0.0000
## constant   0.0015 0.0061  0.2444  0.8071
```

```r
sarima_oil4 <- as.ts(sarima.for(train, n.ahead = 6, p=2, d=1, q=1, P=1, D=0, Q=1, S=12))

# compare predicted and actual
lines(test, col = 'blue')
```




```r
# preview accuracy of model
accuracy(sarima_oil4$pred, test)
```

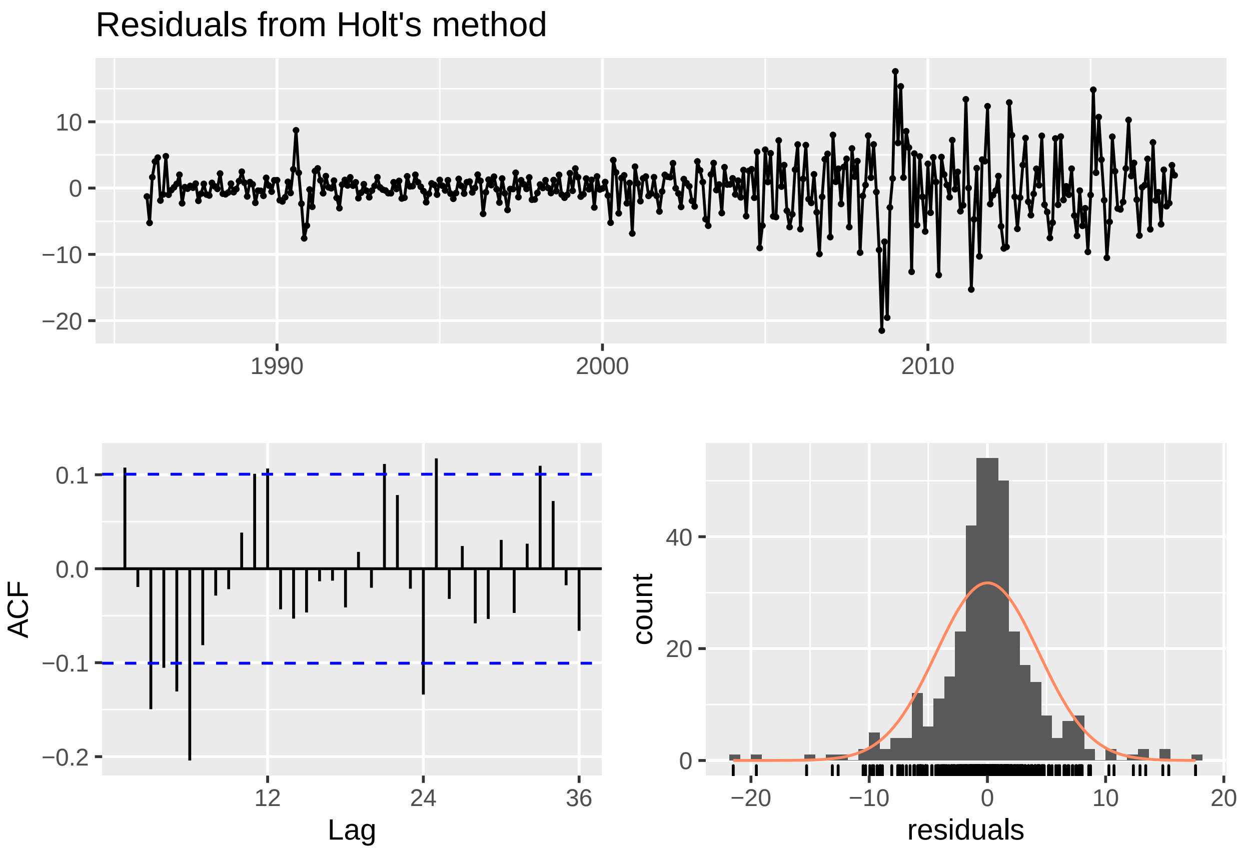
```
##                ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set 9.655375 11.08977 9.655375 16.21246 16.21246 0.5282617  3.146215
```

```r
# ARIMA Summary:
# first model was best - slightly less accurate than third model but less complex
# AR only model was not as good as first model even though suggested by ACF and PACF plots


#############################
# Exponetial Smoothing Model
#############################
```

```r
# now trying simple exponetial smoothing due to trend in data

# holt model
fit_holt <- holt(train, h = 12)
summary(fit_holt)
```

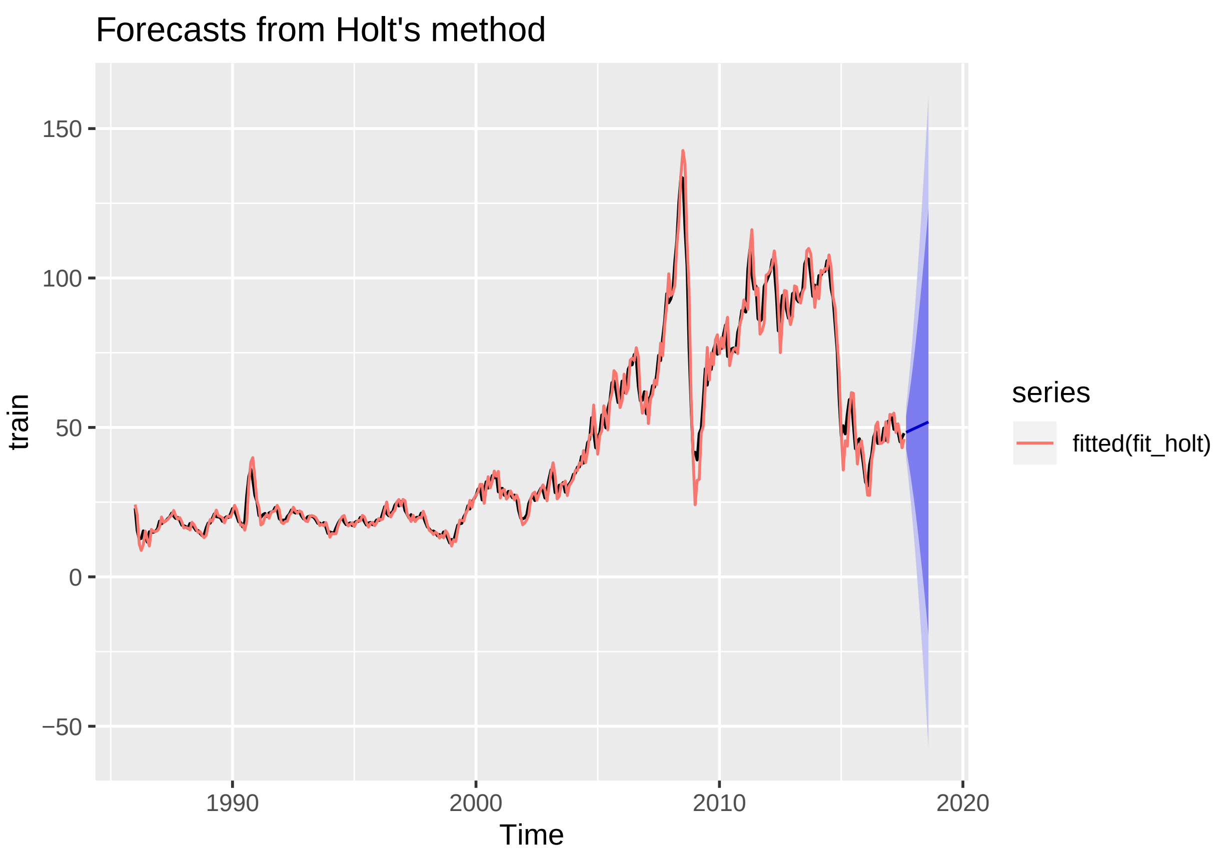
```
##
## Forecast method: Holt's method
##
## Model Information:
## Holt's method
##
## Call:
##  holt(y = train, h = 12)
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 0.4311
##
##   Initial states:
##     l = 25.8715
##     b = -1.6742
##
##   sigma:  4.3603
##
##      AIC     AICc      BIC
## 3382.374 3382.534 3402.075
##
## Error measures:
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 0.01214228 4.337284 2.891557 0.3268413 7.081585 0.2619254
##                   ACF1
## Training set 0.1076263
##
## Forecasts:
##          Point Forecast      Lo 80     Hi 80      Lo 95     Hi 95
## Sep 2017       48.35465  42.766707  53.94259  39.808629  56.90067
## Oct 2017       48.66949  38.914188  58.42479  33.750045  63.58893
## Nov 2017       48.98433  34.721287  63.24737  27.170888  70.79777
## Dec 2017       49.29917  30.125325  68.47301  19.975305  78.62303
## Jan 2018       49.61401  25.131823  74.09620  12.171736  87.05628
## Feb 2018       49.92885  19.758275  80.09943   3.786938  96.07076
## Mar 2018       50.24369  14.023525  86.46386  -5.150273 105.63765
## Apr 2018       50.55853   7.945226  93.17184 -14.612895 115.72996
## May 2018       50.87337   1.539313 100.20743 -24.576559 126.32330
## Jun 2018       51.18821  -5.179978 107.55640 -35.019495 137.39592
## Jul 2018       51.50305 -12.199931 115.20604 -45.922253 148.92836
## Aug 2018       51.81789 -19.509148 123.14493 -57.267403 160.90319
```

```r
# checking for autocorrelation in residuals
checkresiduals(fit_holt)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Holt's method
## Q* = 71.252, df = 20, p-value = 1.136e-07
##
## Model df: 4.   Total lags used: 24
```

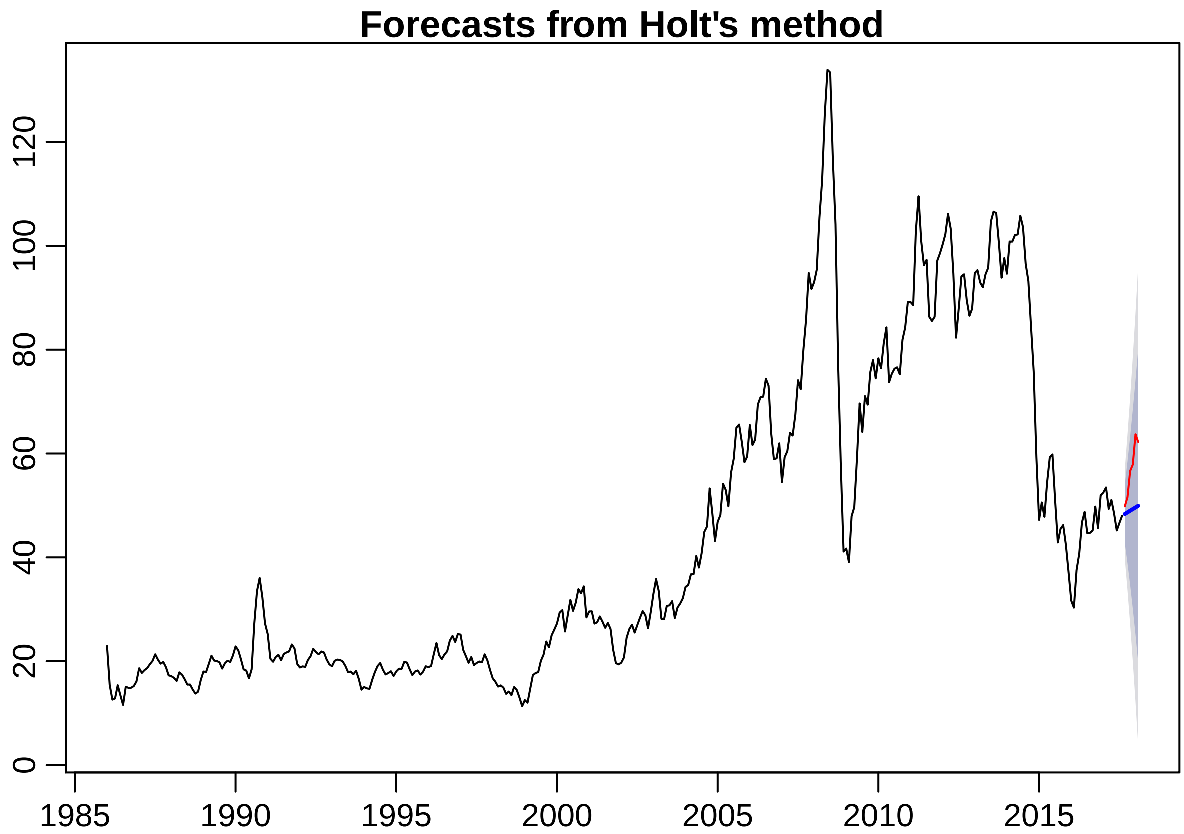
```
# plotting model fit
autoplot(fit_holt) + autolayer(fitted(fit_holt))
```




```
# checking model accuracy
accuracy(fit_holt, test)
```

```
##                       ME     RMSE      MAE        MPE      MAPE      MASE
## Training set 0.01214228 4.337284 2.891557  0.3268413  7.081585 0.2619254
## Test set     7.83325097 9.060829 7.833251 13.1343054 13.134305 0.7095581
##                   ACF1 Theil's U
## Training set 0.1076263        NA
## Test set     0.5199591  2.566832
```

```
# forecast test values
for_fit <- forecast(fit_holt, h = 6)

# plot the predicted data
plot(for_fit)
lines(test, col = 'red')
```

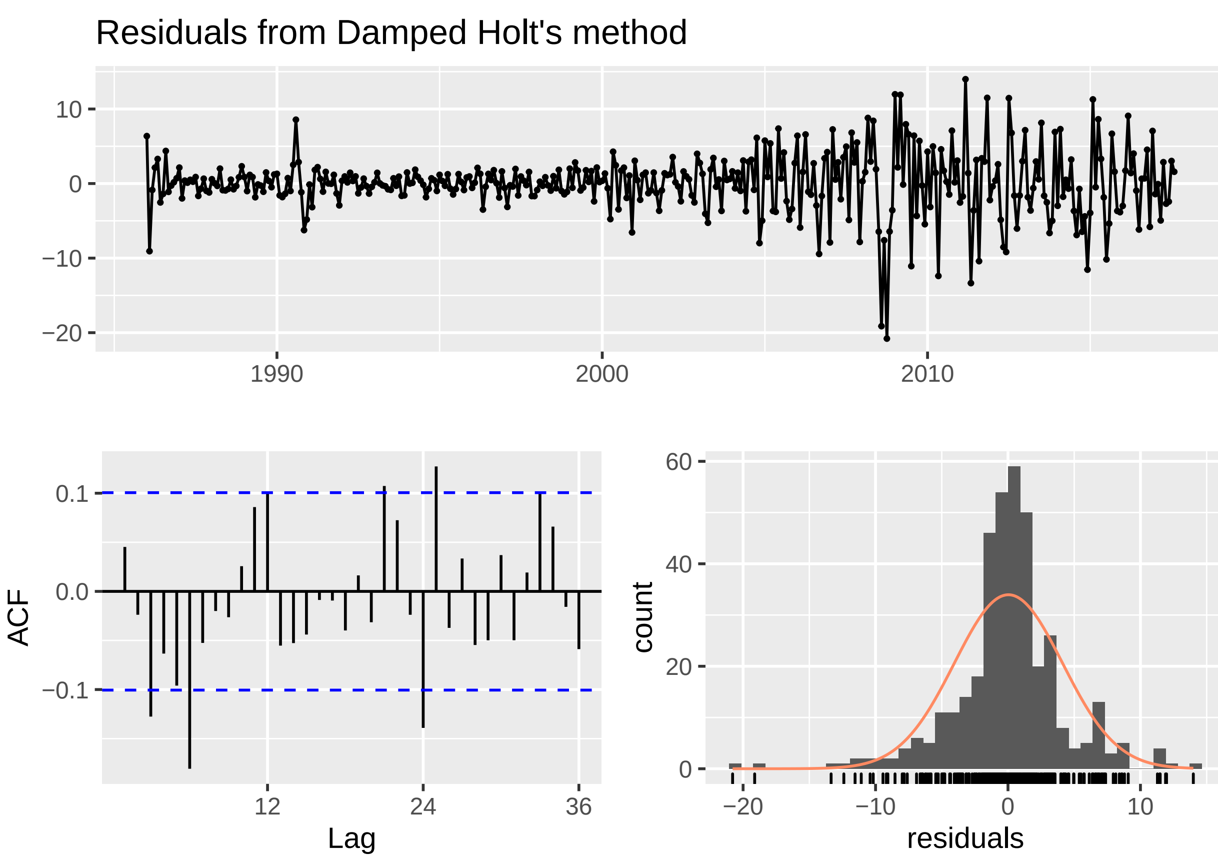



```r
# holt model damped
fit_holt_d <- holt(train, h = 12, damped = TRUE)
summary(fit_holt_d)
```

```
##
## Forecast method: Damped Holt's method
##
## Model Information:
## Damped Holt's method
##
## Call:
##  holt(y = train, h = 12, damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 0.4508
##     phi   = 0.8
##
##   Initial states:
##     l = 17.4339
##     b = -1.0832
##
##   sigma:  4.1087
##
##      AIC     AICc      BIC
## 3338.185 3338.411 3361.826
##
## Error measures:
```

```
##                       ME     RMSE     MAE       MPE     MAPE      MASE
## Training set 0.03484041 4.081539 2.76284 0.1178085 6.779181 0.2502659
##                    ACF1
## Training set 0.04524763
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80      Lo 95     Hi 95
## Sep 2017       48.47409 43.20863 53.73955  40.421268  56.52692
## Oct 2017       48.82149 39.93079 57.71220  35.224330  62.41865
## Nov 2017       49.09941 36.67217 61.52666  30.093582  68.10524
## Dec 2017       49.32175 33.43437 65.20913  25.024101  73.61940
## Jan 2018       49.49962 30.24936 68.74987  20.058894  78.94034
## Feb 2018       49.64191 27.14192 72.14191  15.231139  84.05269
## Mar 2018       49.75575 24.12745 75.38404  10.560652  88.95084
## Apr 2018       49.84682 21.21410 78.47954   6.056851  93.63678
## May 2018       49.91967 18.40486 81.43448   1.721931  98.11741
## Jun 2018       49.97796 15.69933 84.25658  -2.446674 102.40259
## Jul 2018       50.02458 13.09488 86.95429  -6.454529 106.50370
## Aug 2018       50.06189 10.58750 89.53627 -10.308970 110.43274
```

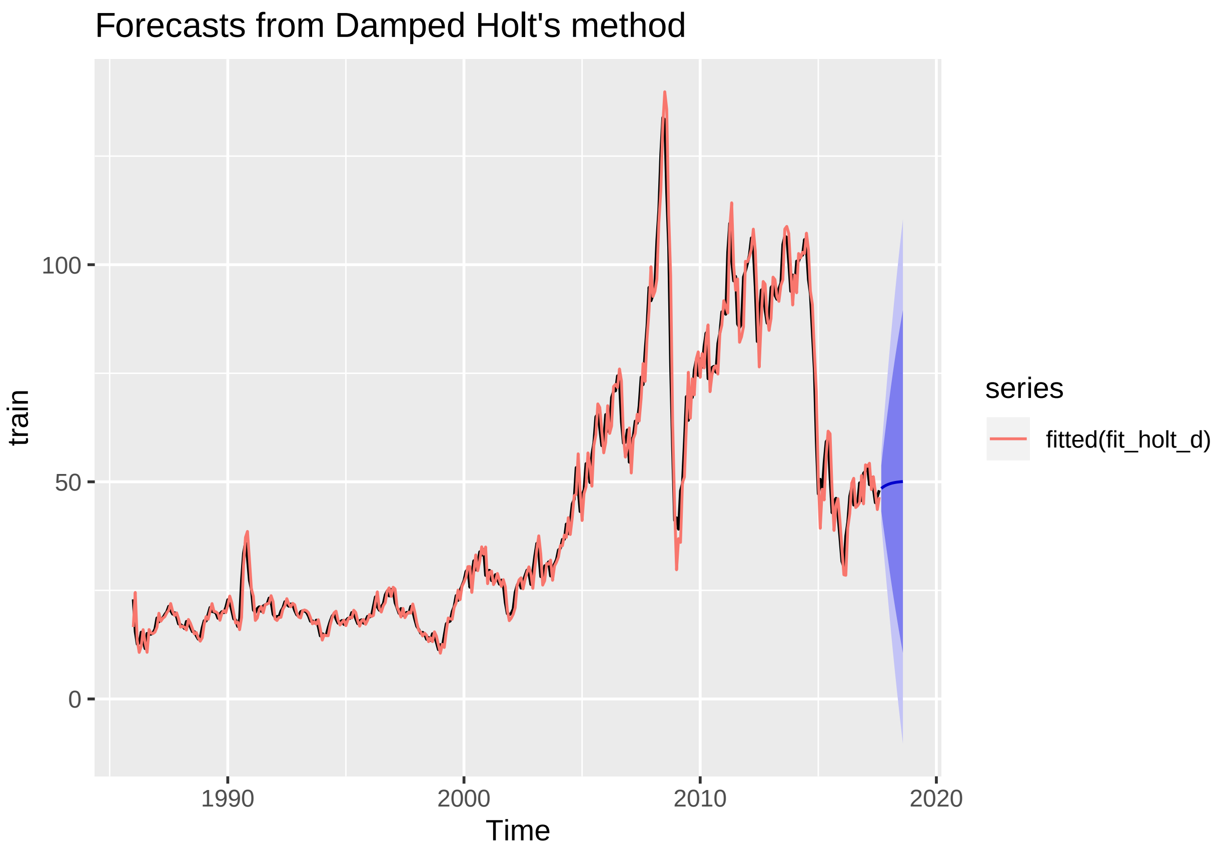
```r
checkresiduals(fit_holt_d)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Damped Holt's method
## Q* = 52.815, df = 19, p-value = 4.991e-05
##
```

```
## Model df: 5.   Total lags used: 24
```

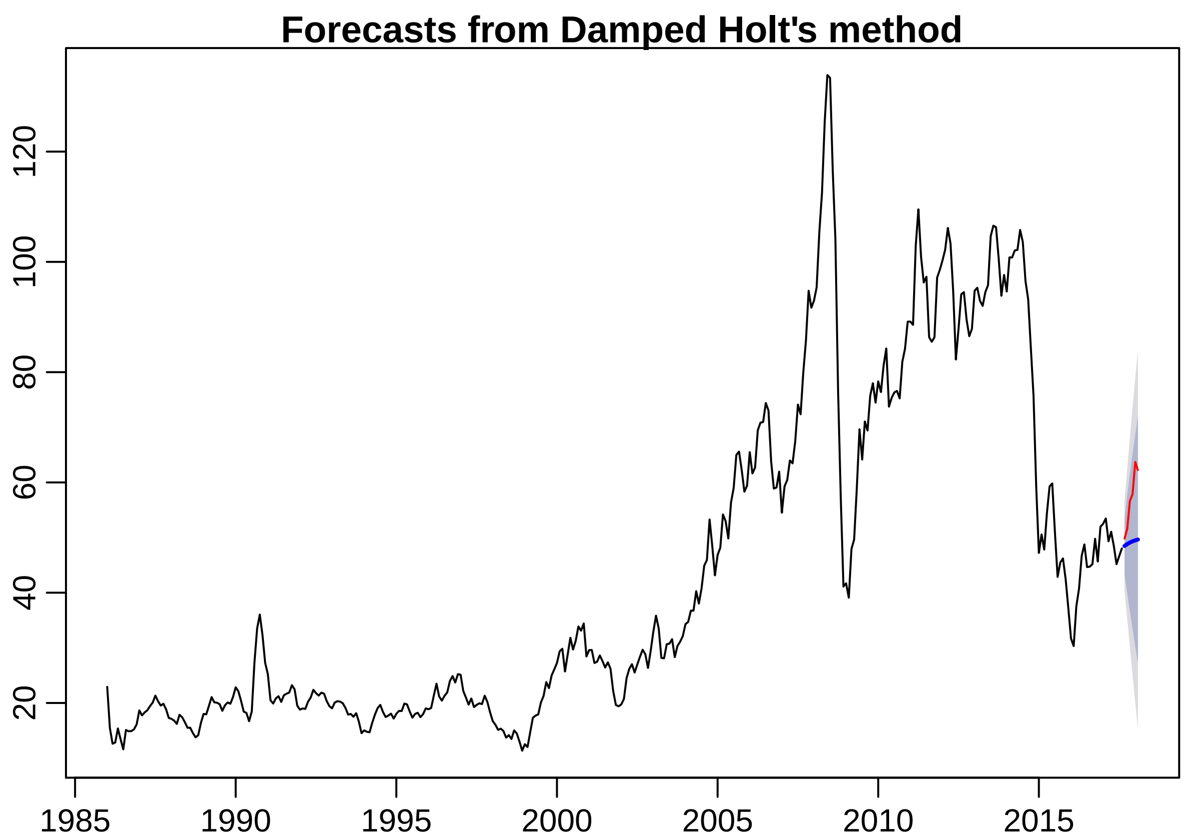
```
autoplot(fit_holt_d) + autolayer(fitted(fit_holt_d))
```




```
accuracy(fit_holt_d, test)
```

```
##                       ME     RMSE      MAE        MPE      MAPE      MASE
## Training set 0.03484041 4.081539 2.762840  0.1178085  6.779181 0.2502659
## Test set     7.83195437 9.125384 7.831954 13.1116442 13.111644 0.7094406
##                    ACF1 Theil's U
## Training set 0.04524763        NA
## Test set     0.52498746   2.58097
```

```
for_fit2 <- forecast(fit_holt_d, h = 6)
plot(for_fit2)
lines(test, col = 'red')
```

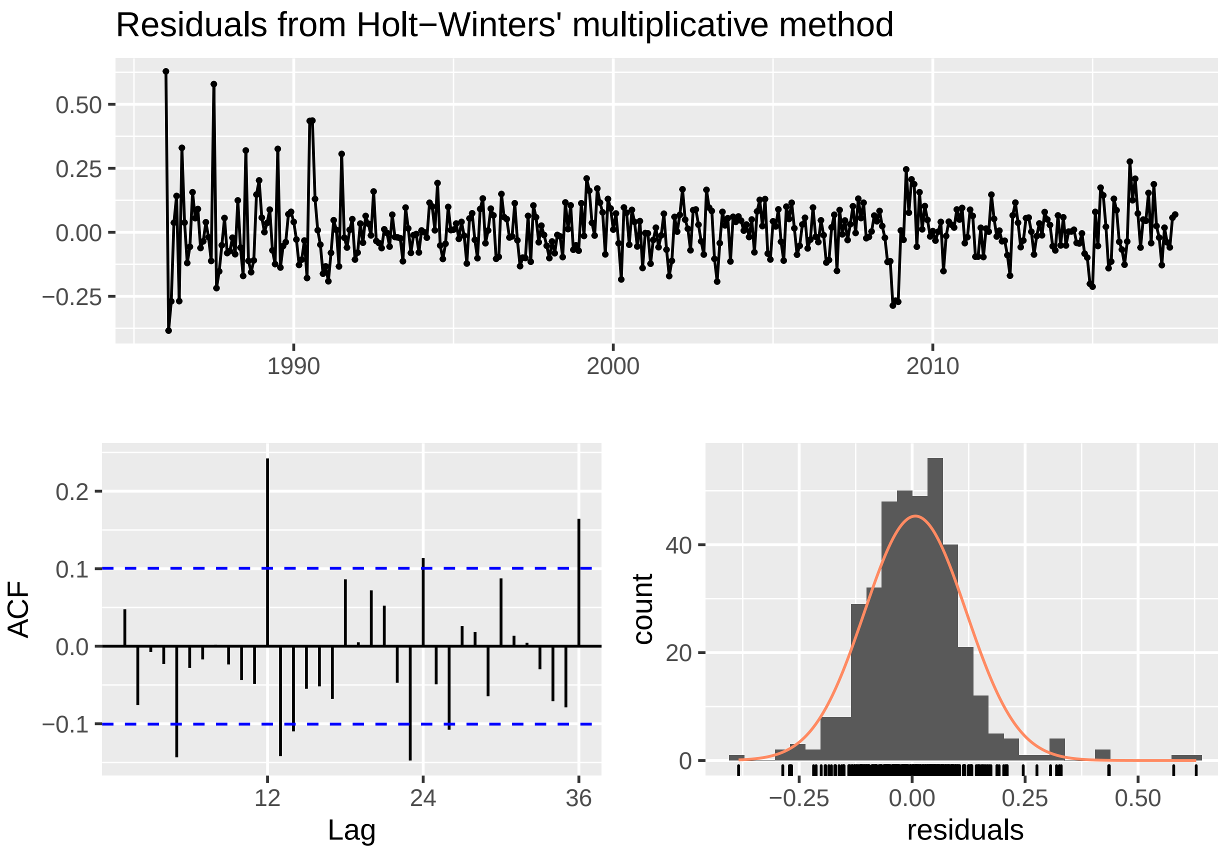



```r
# holt winters model - dont expect this model to work super well - no clear seasonality in the data
fit_hw <- hw(train, seasonal = "multiplicative")
summary(fit_hw)
```

```
##
## Forecast method: Holt-Winters' multiplicative method
##
## Model Information:
## Holt-Winters' multiplicative method
##
## Call:
##  hw(y = train, seasonal = "multiplicative")
##
##   Smoothing parameters:
##     alpha = 0.9078
##     beta  = 0.0442
##     gamma = 0.0922
##
##   Initial states:
##     l = 14.7024
##     b = 0.3698
##     s = 0.9437 0.9447 1.0892 1.0537 0.956 0.7575
##            1.2178 1.027 0.9909 1.0592 1.0262 0.9341
##
##   sigma:  0.1155
##
##      AIC     AICc      BIC
## 3321.590 3323.281 3388.573
```

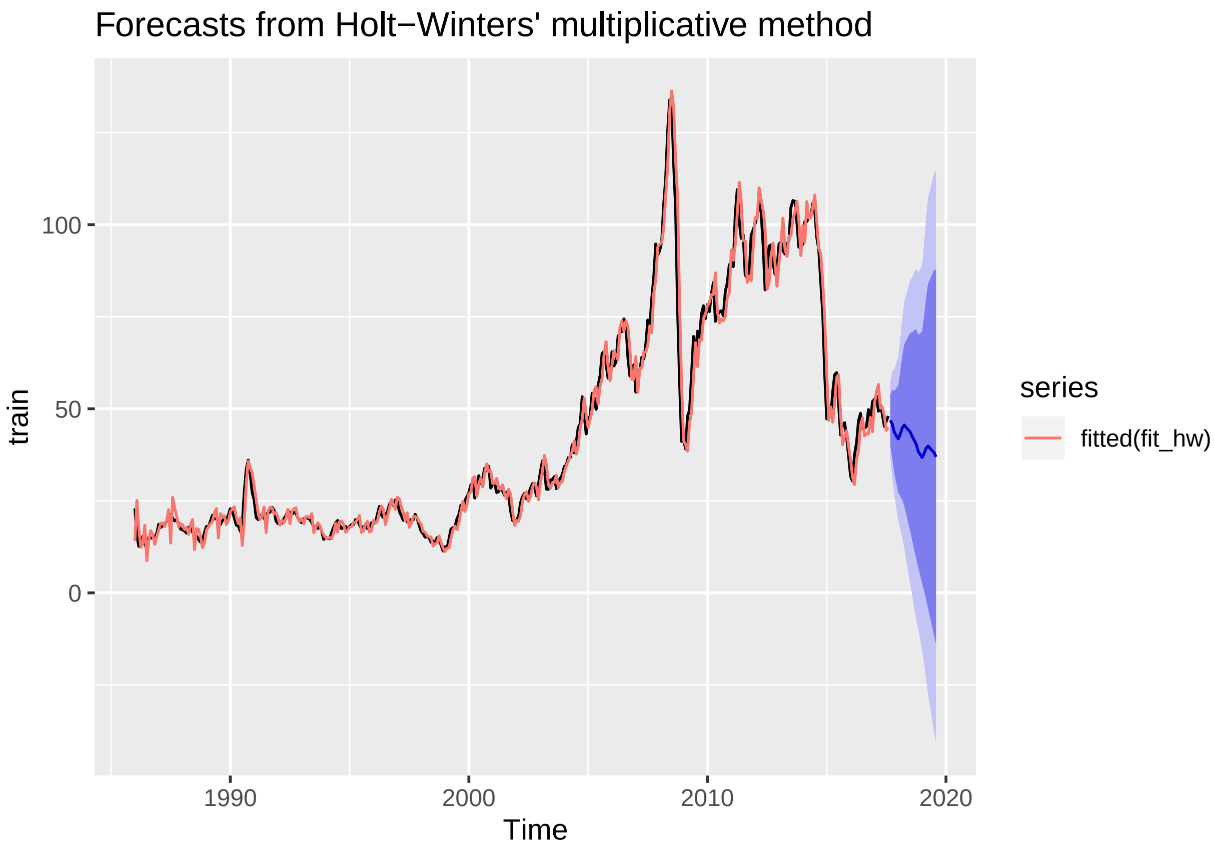
```
##
## Error measures:
##                       ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.09302371 4.623187 3.118393 -0.4995332 8.134427 0.2824729
##                   ACF1
## Training set 0.3277393
##
## Forecasts:
##          Point Forecast       Lo 80    Hi 80       Lo 95     Hi 95
## Sep 2017       46.87253  39.9357465 53.80931  36.2636366  57.48142
## Oct 2017       45.79068  36.3602478 55.22111  31.3680780  60.21328
## Nov 2017       43.64089  32.4648272 54.81696  26.5485744  60.73322
## Dec 2017       42.67346  29.7537946 55.59313  22.9145357  62.43239
## Jan 2018       41.82564  27.2811861 56.37010  19.5818156  64.06947
## Feb 2018       43.10529  26.2080083 60.00258  17.2631264  68.94746
## Mar 2018       44.80010  25.2616627 64.33853  14.9186398  74.68156
## Apr 2018       45.58369  23.6801382 67.48725  12.0850960  79.08229
## May 2018       44.88387  21.2993539 68.46838   8.8144667  80.95327
## Jun 2018       44.28739  18.9912415 69.58354   5.6002680  82.97452
## Jul 2018       43.69738  16.6978727 70.69688   2.4051990  84.98955
## Aug 2018       42.45633  14.1934195 70.71925  -0.7680637  85.68073
## Sep 2018       41.43828  11.6052633 71.27130  -4.1873830  87.06395
## Oct 2018       40.42847   9.2939083 71.56303  -7.1877336  88.04468
## Nov 2018       38.47853   6.8552611 70.10179  -9.8850849  86.84214
## Dec 2018       37.57376   4.6880179 70.45950 -12.7206408  87.86815
## Jan 2019       36.77547   2.5597967 70.99115 -15.5528873  89.10383
## Feb 2019       37.84614   0.4753869 75.21689 -19.3074952  94.99978
## Mar 2019       39.27638  -1.8254895 80.37824 -23.5835025 102.13625
## Apr 2019       39.90331  -4.2945848 84.10121 -27.6915396 107.49817
## May 2019       39.23032  -6.7082190 85.16885 -31.0266112 109.48725
## Jun 2019       38.64812  -9.1484632 86.44470 -34.4504441 111.74668
## Jul 2019       38.07188 -11.6080362 87.75179 -37.9069928 114.05074
## Aug 2019       36.92968 -13.8739264 87.73329 -40.7677315 114.62709
```

```
checkresiduals(fit_hw)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Holt-Winters' multiplicative method
## Q* = 74.791, df = 8, p-value = 5.432e-13
##
## Model df: 16.   Total lags used: 24
```

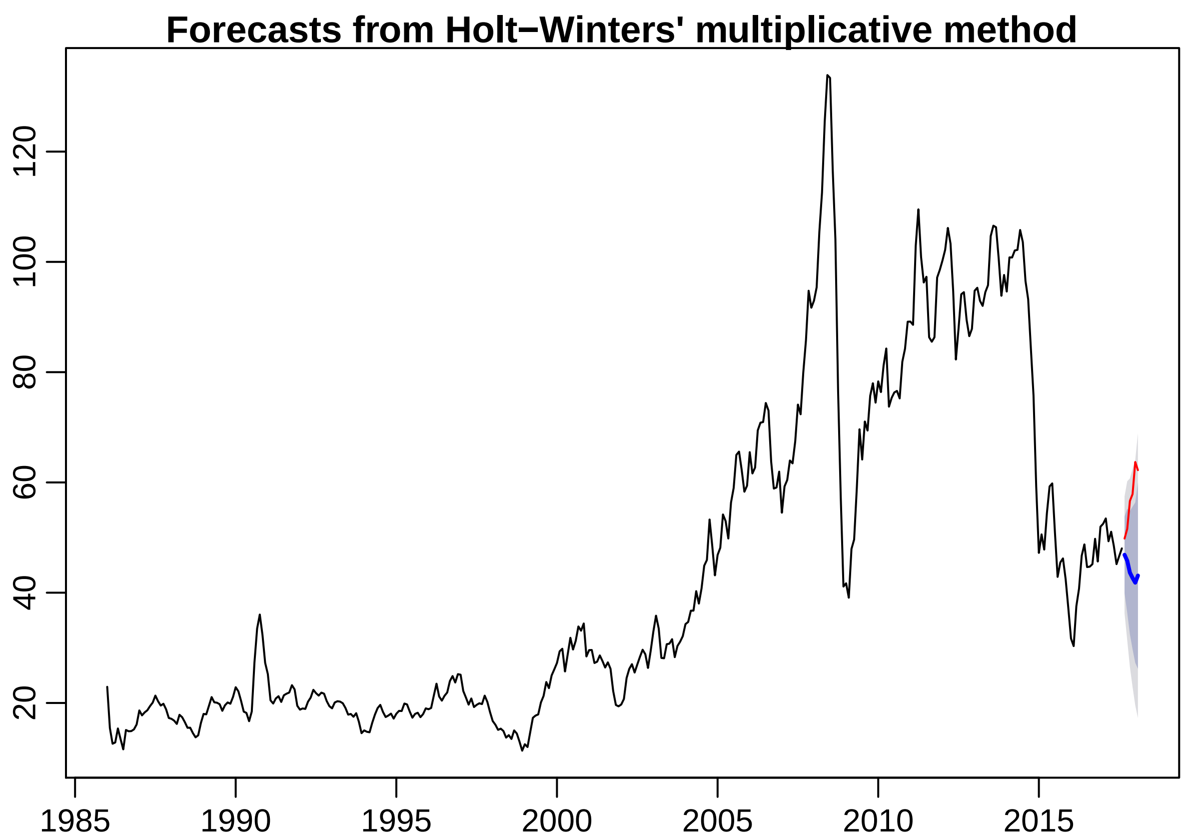
```
autoplot(fit_hw) + autolayer(fitted(fit_hw))
```




```r
accuracy(fit_hw, test)
```

```
##                        ME      RMSE       MAE        MPE      MAPE
## Training set -0.09302371  4.623187  3.118393 -0.4995332  8.134427
## Test set     12.99024979 14.643891 12.990250 21.9058454 21.905845
##                   MASE      ACF1 Theil's U
## Training set 0.2824729 0.3277393        NA
## Test set     1.1766936 0.5341790  4.167851
```

```r
for_fit3 <- forecast(fit_hw, h = 6)
plot(for_fit3)
lines(test, col = 'red')
```




```
# Exponential Smoothing Summary:
# holt model seems to predict the best
# makes sense as there is a trend but no seasonality

################
# Phophet Model
################

head(oil)
```

```
##          DATE MCOILWTICO
## 1 1986-01-01      22.93
## 2 1986-02-01      15.46
## 3 1986-03-01      12.61
## 4 1986-04-01      12.84
## 5 1986-05-01      15.38
## 6 1986-06-01      13.43
```

```
# renaming columns for prophet modeling
colnames(oil)[1] <- 'ds'
colnames(oil)[2] <- 'y'

# re-subsetting data - prophet requires dataframe instead of ts
oil_train = oil[1:380,]
oil_test = oil[381:386,]

prophet_fit <- prophet(oil_train)
```

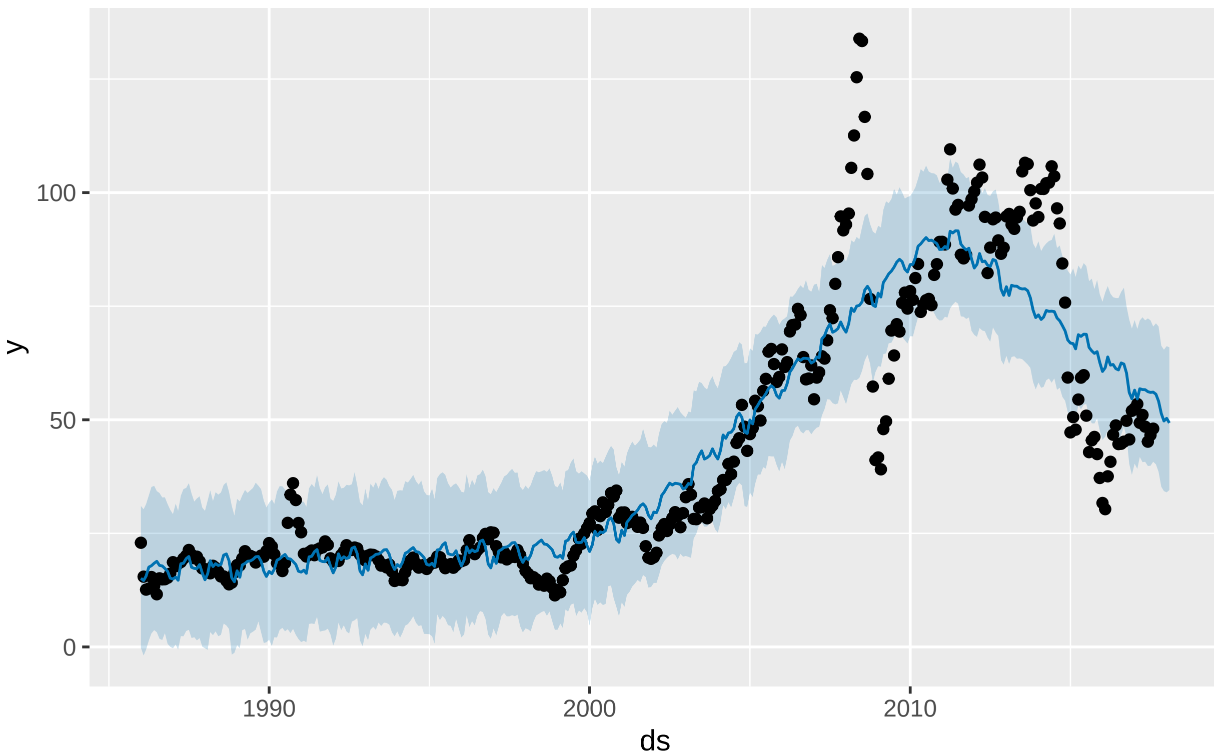
```
## Disabling weekly seasonality. Run prophet with weekly.seasonality=TRUE to override this.
```

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

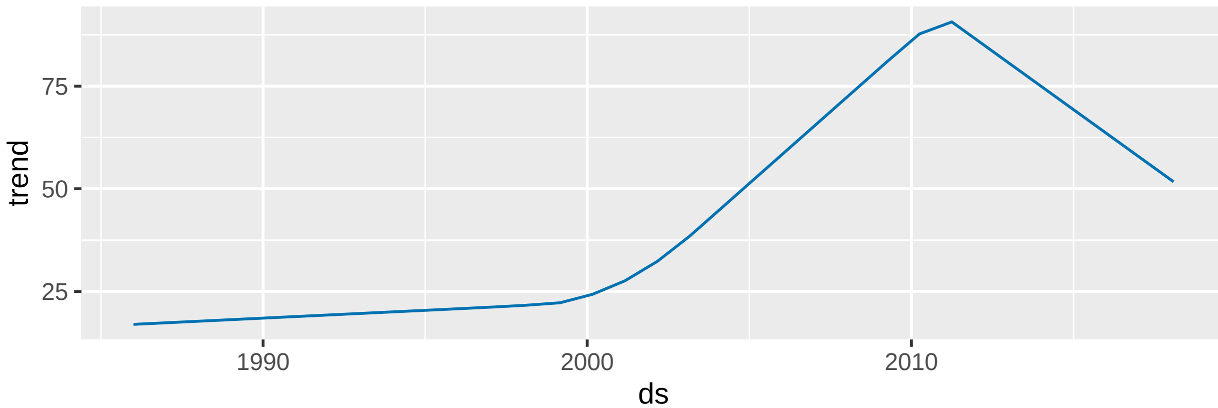
```r
# make df for forecasted predictions
future <- make_future_dataframe(prophet_fit, periods = 6, freq = 'month')
forecast <- predict(prophet_fit, future)
tail(forecast[c('ds', 'yhat', 'yhat_lower', 'yhat_upper')])
```

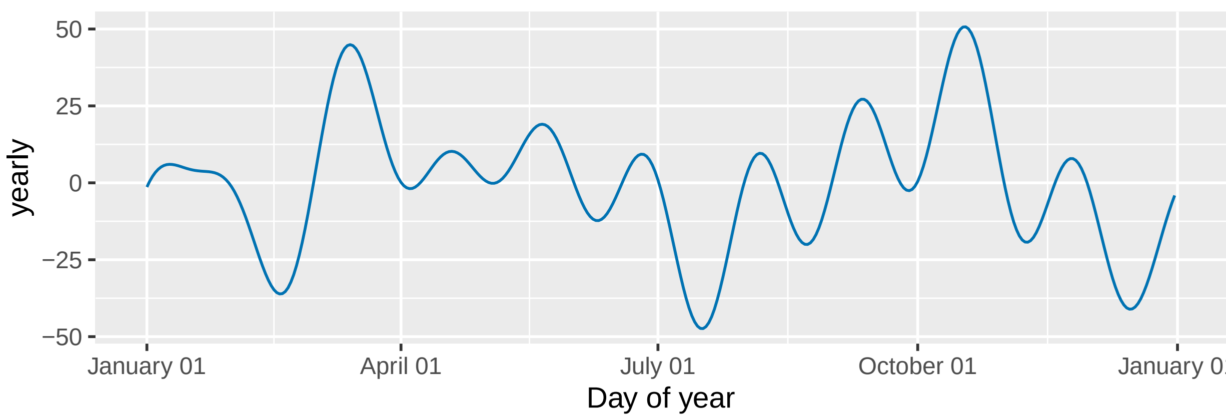
```
##             ds     yhat yhat_lower yhat_upper
## 381 2017-09-01 55.63151   40.18352   71.22727
## 382 2017-10-01 54.09709   38.54682   70.55393
## 383 2017-11-01 51.38124   35.24158   66.29676
## 384 2017-12-01 49.72739   34.28923   65.42814
## 385 2018-01-01 50.31957   34.02240   66.20291
## 386 2018-02-01 49.30186   34.52141   65.95236
```

```r
plot(prophet_fit, forecast)
```




```r
prophet_plot_components(prophet_fit, forecast)
```






```r
# checking model accuracy
accuracy(forecast[c('yhat')][380:386,], test)
```

```
##                 ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set 4.101113 8.577843 7.540457 6.059809 12.87085 0.5703779  2.291049
```

```r
# Prophet model summary:
# fits better than exponential smoothing models but not as well as final ARIMA model
```