Sales Forecast, Example

Practice exercise: Seasonal Naive Method, Exponential Smoothing Model and ARIMA Model

JDM

Sales Forecast Considerations:

- This file shows the result of "Sales_per_day_Forecast.R" file. It is the same code.
- Although RMarkdown idea is to hide the script, some important scripts will be shown.

This EXAMPLE shows in a simple way how to use some popular methods for Forecast: Benchmark Method (Seasonal Naive Method), Exponential Smoothing Model and ARIMA Model.

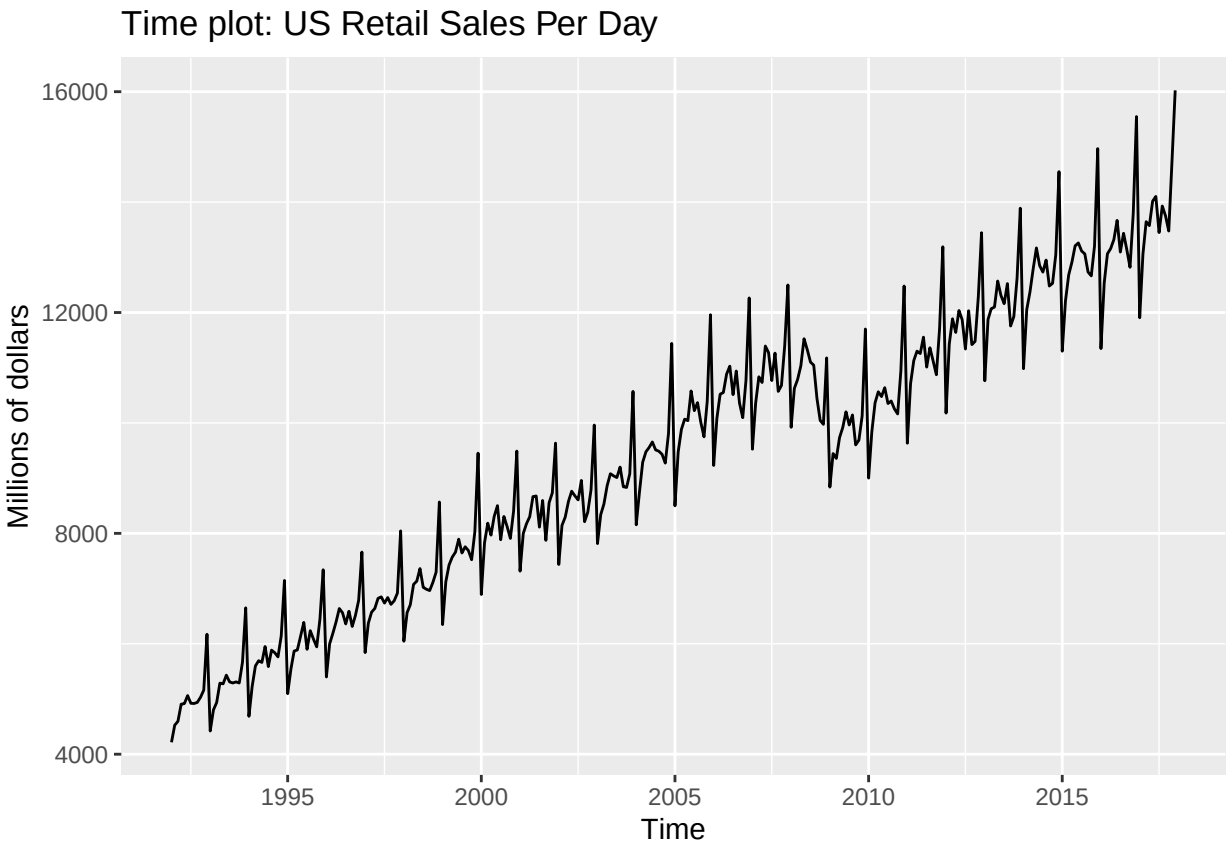
This data is not adjusted for inflation. So these are nomimal values. (practice excersice) Monthly retail sales per day

```
# Load data
data <- read.csv("https://raw.githubusercontent.com/Jdmuci/R_practices/main/sales_per_day.csv")

# declare as time series data
Y <- ts(data[,2], start = c(1992,1), end = c(2017,12), frequency = 12)
```

Preliminary analysis

Time plot



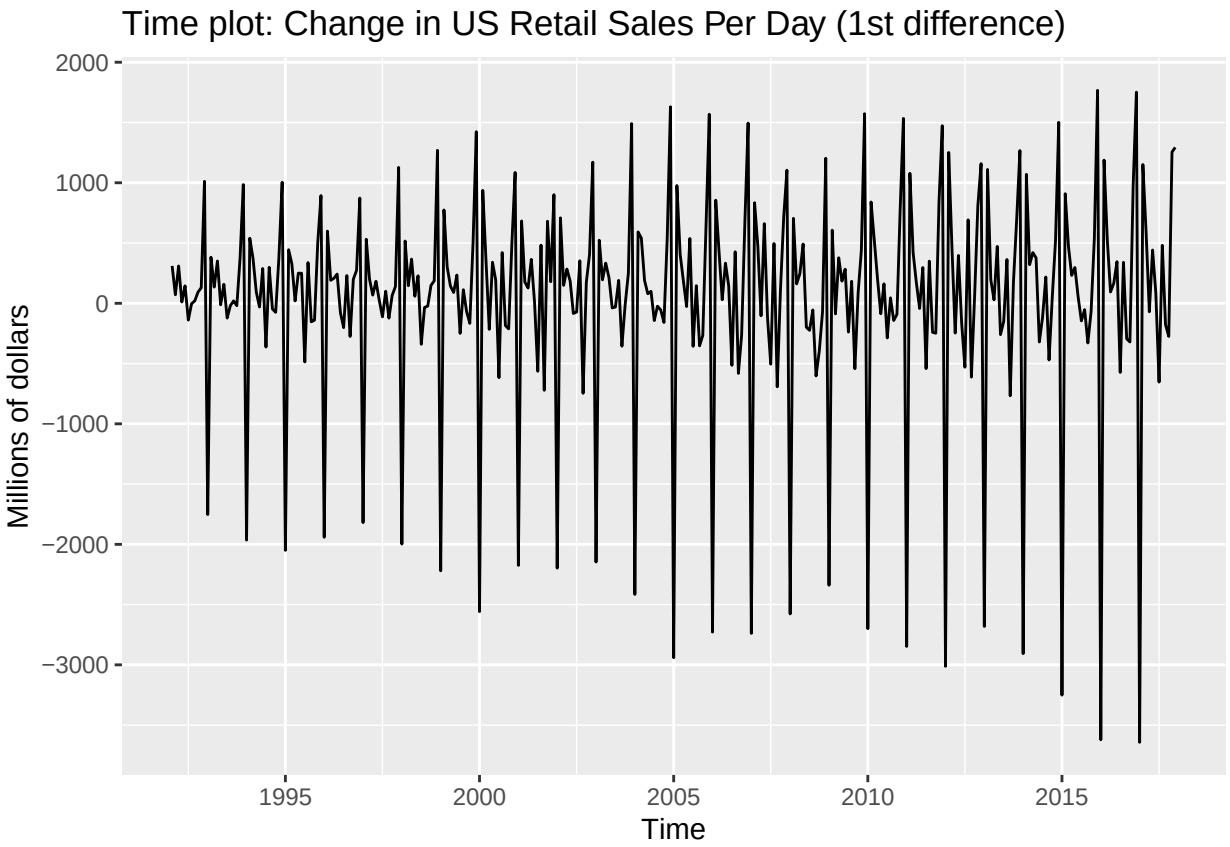
1- Seasonality and Trend

Data has a strong trend and seasonality. Need to investigate transformations.

- Take the first difference to remove trend.

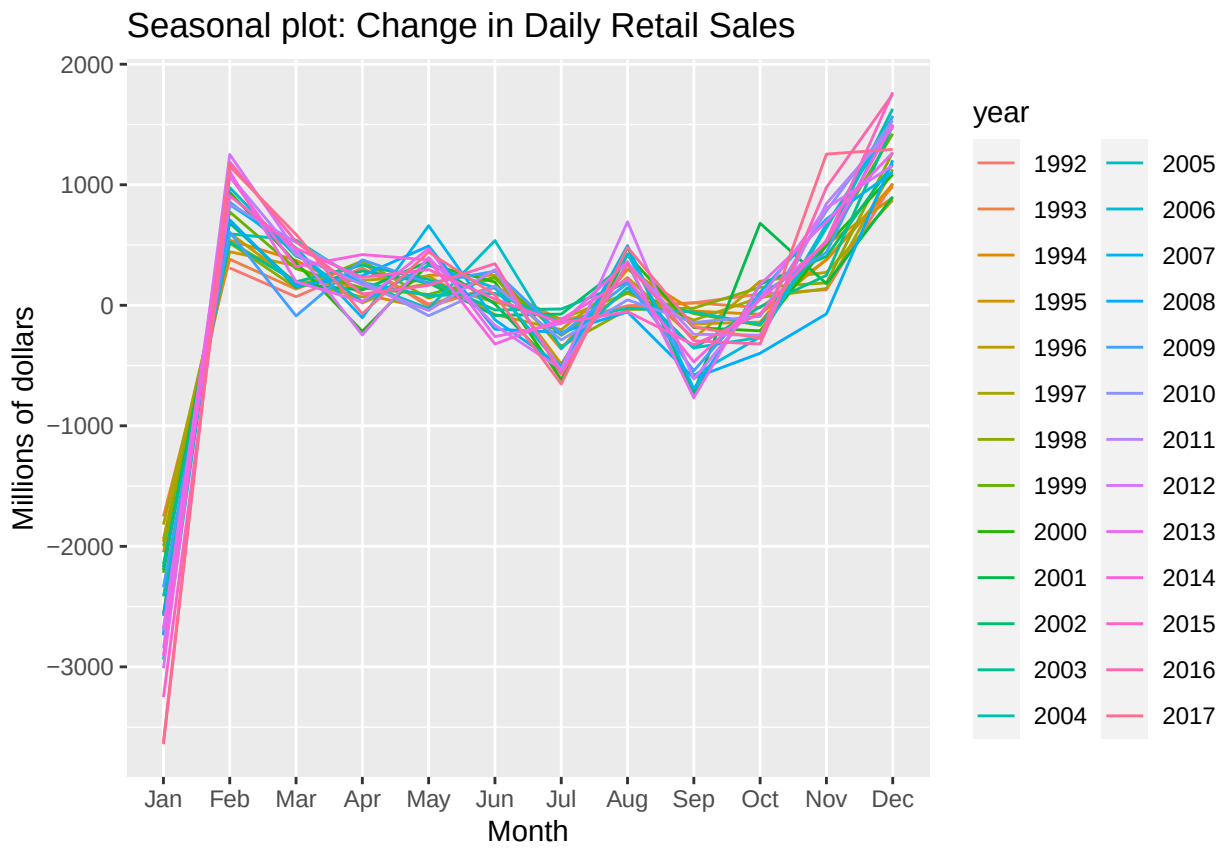
```
DY <- diff(Y)
```

Time plot of differenced data:



Series appears trend-stationary, but need to investigate seasonality.

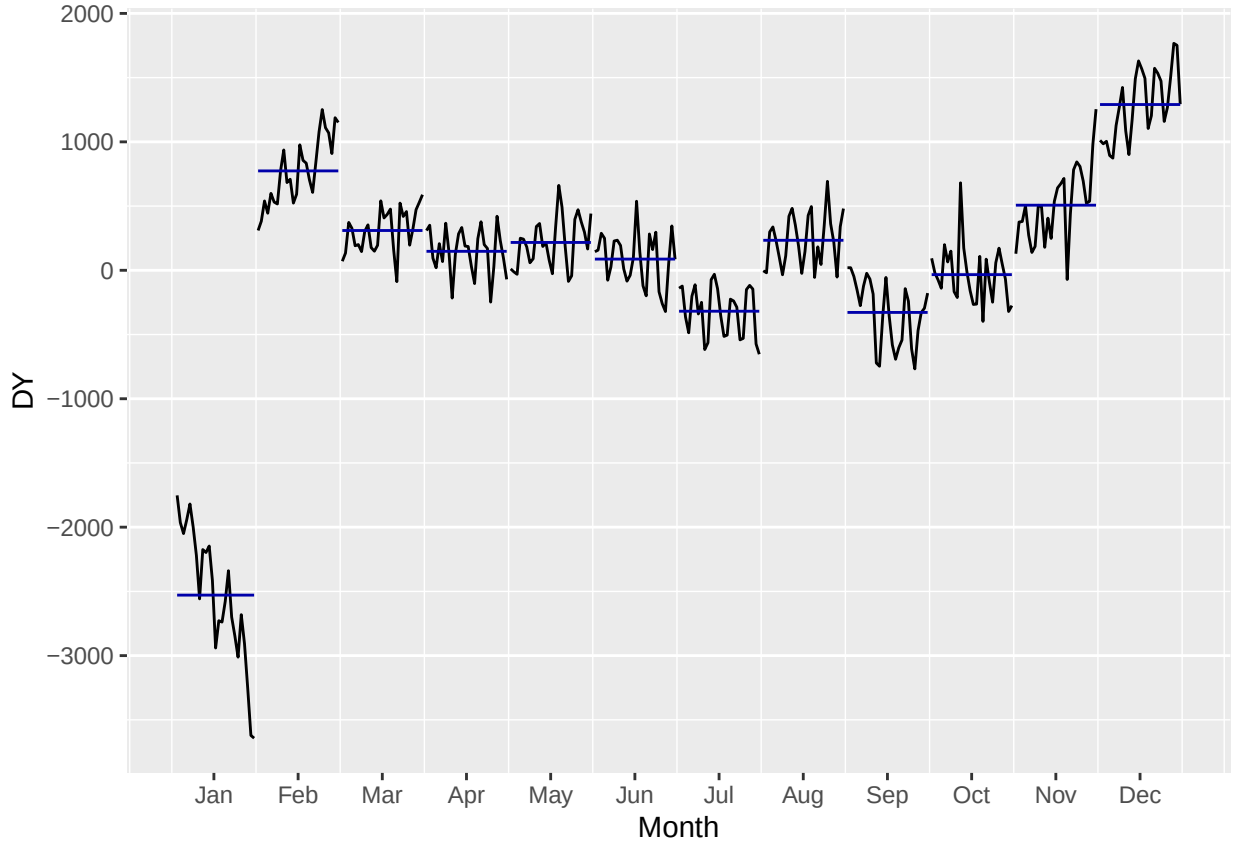
Lets now focus on seasonality. The next two plots are interesting and useful to analyse seasonality.



This function makes a series for each year and plot them together in a 12 months period to see the seasonality.

Apparently there is a drop between december and january, and a rise from november to december (christmas season).

Lets look at another seasonal plot, the subseries plot:



This function graphs each month of the series separately. Example: every January together (Ejemplo: todos los meses Enero de la serie, juntos) . Average: blue line.

So, we confirm the fluctuations we saw in the first plot is seasonality every december and january.

Forecast with various methods (considering seasonality).

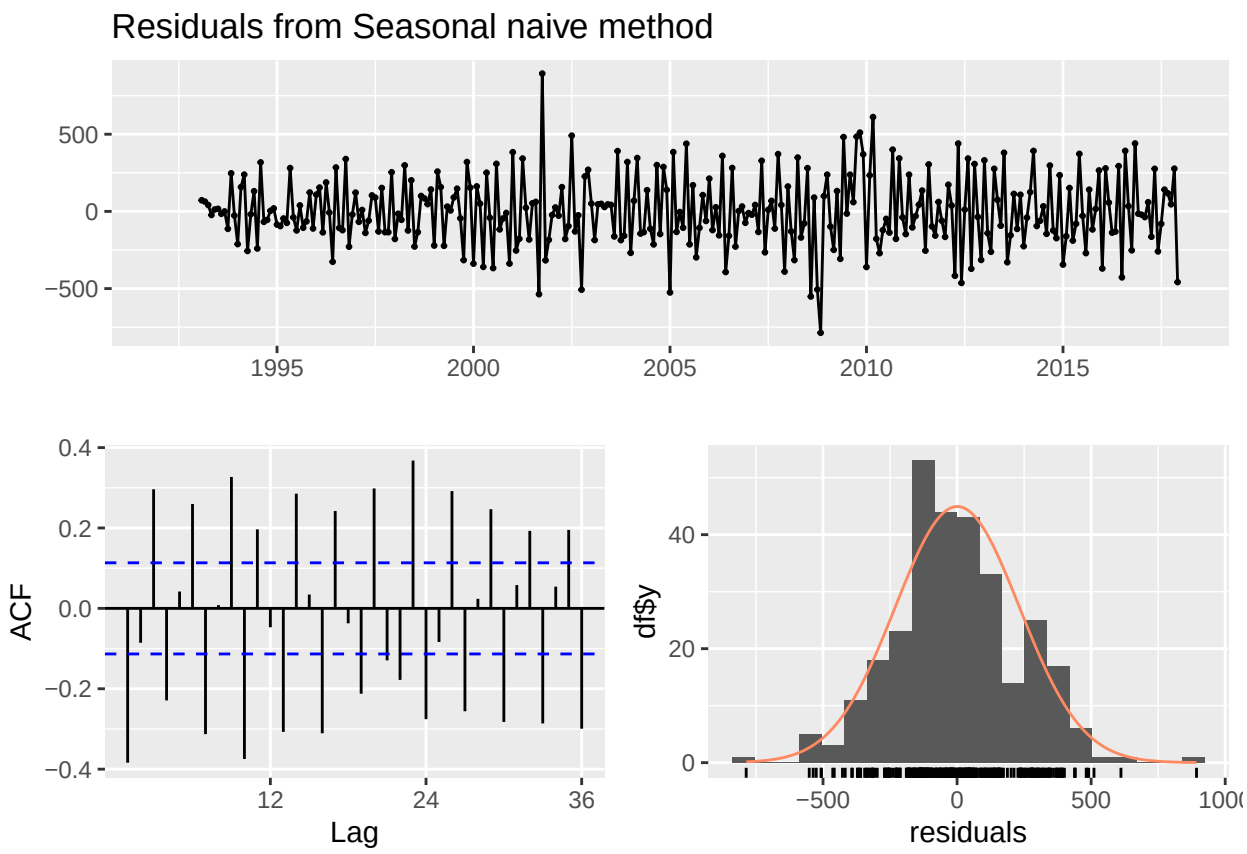
FIRST METHOD: Use a Benchmark method to forecast. (there are many)

So, lets use the Seasonal naive method as our benchmark: y_t = y_{t-s} + e_t

Using the differenced series (1st difference):

```
##
## Forecast method: Seasonal naive method
##
## Model Information:
## Call: snaive(y = DY)
##
## Residual sd: 231.4508
##
## Error measures:
##                       ME     RMSE      MAE      MPE     MAPE MASE        ACF1
## Training set 0.9090517 231.4508 181.9523 27.85462 105.8591    1 -0.3838154
##
## Forecasts:
```

```
##          Point Forecast       Lo 80       Hi 80       Lo 95       Hi 95
## Jan 2018    -3642.61290 -3939.22898 -3345.99682 -4096.24805 -3188.97775
## Feb 2018     1151.01843   854.40235  1447.63451   697.38328  1604.65358
## Mar 2018      588.36867   291.75259   884.98475   134.73352  1042.00382
## Apr 2018      -70.55914  -367.17522   226.05694  -524.19429   383.07601
## May 2018      441.65591   145.03983   738.27199   -11.97924   895.29106
## Jun 2018       85.87742  -210.73866   382.49350  -367.75773   539.51257
## Jul 2018     -653.61935  -950.23543  -357.00327 -1107.25450  -199.98420
## Aug 2018      480.58064   183.96456   777.19672    26.94549   934.21579
## Sep 2018     -177.62796  -474.24404   118.98812  -631.26311   276.00719
## Oct 2018     -274.98494  -571.60102    21.63114  -728.62009   178.65021
## Nov 2018     1254.55161   957.93553  1551.16769   800.91646  1708.18676
## Dec 2018     1293.57742   996.96134  1590.19350   839.94227  1747.21257
## Jan 2019    -3642.61290 -4062.09138 -3223.13442 -4284.14988 -3001.07592
## Feb 2019     1151.01843   731.53995  1570.49691   509.48145  1792.55541
## Mar 2019      588.36867   168.89019  1007.84715   -53.16831  1229.90565
## Apr 2019      -70.55914  -490.03762   348.91934  -712.09612   570.97784
## May 2019      441.65591    22.17743   861.13439  -199.88107  1083.19289
## Jun 2019       85.87742  -333.60106   505.35590  -555.65956   727.41440
## Jul 2019     -653.61935 -1073.09783  -234.14087 -1295.15633   -12.08237
## Aug 2019      480.58064    61.10216   900.05912  -160.95634  1122.11762
## Sep 2019     -177.62796  -597.10644   241.85052  -819.16494   463.90902
## Oct 2019     -274.98494  -694.46342   144.49354  -916.52192   366.55204
## Nov 2019     1254.55161   835.07313  1674.03009   613.01463  1896.08859
## Dec 2019     1293.57742   874.09894  1713.05590   652.04044  1935.11440
```



```
##
##  Ljung-Box test
##
## data:  Residuals from Seasonal naive method
## Q* = 464.01, df = 24, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 24
```

We see residuals ACF are not ideal, lets look if we can find a better model

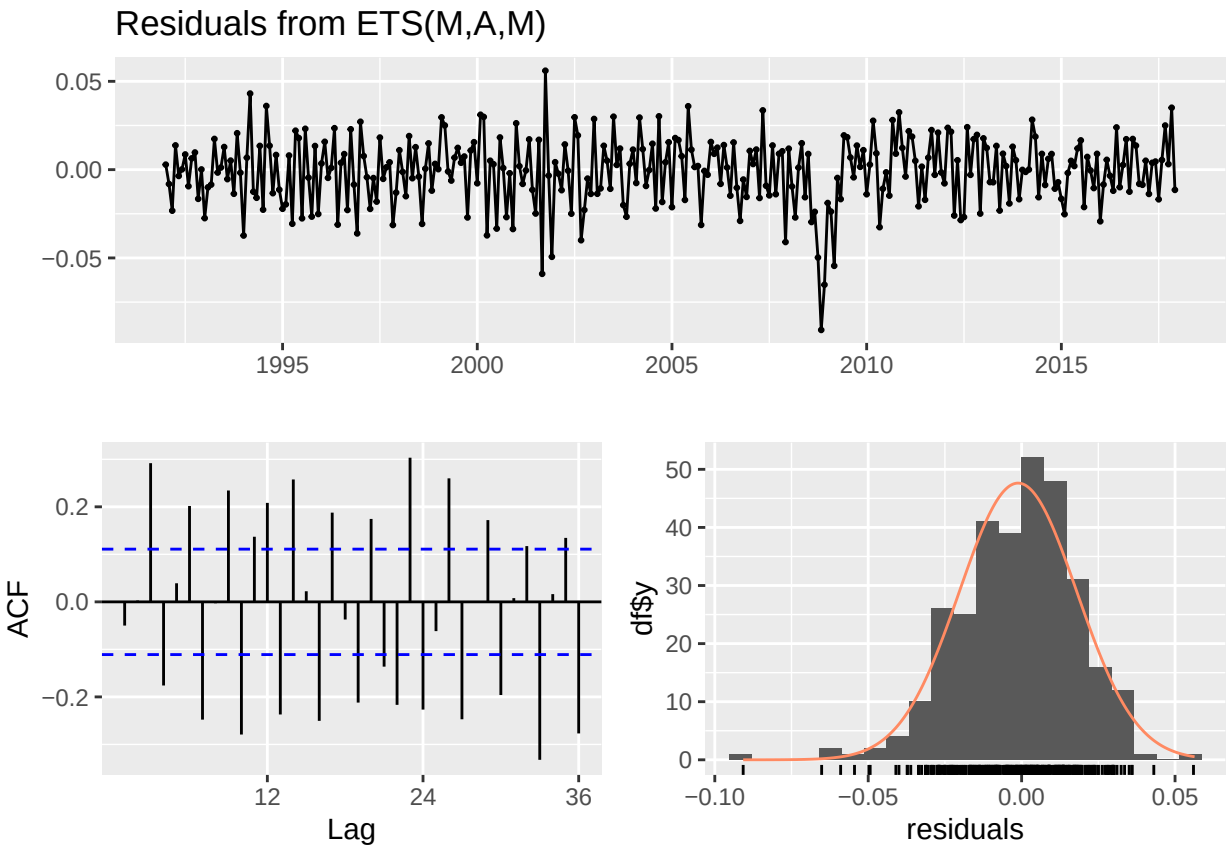
SECOND METHOD: Fit ETS method

Another type we can try is Exponential smoothing model.

They (exponential smoothing model) use weighted averages of past observations to forecast new values Exponential smoothing models are based on a description of the trend and seasonality in the data

We can use regular data (not the differenced one)

```
## ETS(M,A,M)
##
## Call:
##  ets(y = Y)
##
##   Smoothing parameters:
##     alpha = 0.4528
##     beta  = 2e-04
##     gamma = 0.2343
##
##   Initial states:
##     l = 4744.6674
##     b = 35.6917
##     s = 1.2033 1.0226 0.9822 0.9768 0.9912 0.985
##            1.029 1.009 1.003 0.9722 0.9462 0.8794
##
##   sigma:  0.0197
##
##      AIC     AICc      BIC
## 5040.721 5042.803 5104.352
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -12.10585 186.3035 140.0386 -0.1532637 1.503134 0.3074385
##                      ACF1
## Training set 0.009306847
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,A,M)
## Q* = 303.62, df = 8, p-value < 2.2e-16
##
## Model df: 16.   Total lags used: 24
```

Apparently it did not improve, still correlation left in the data and residuals look almost the same But it fits better, since the Residual SD is much smaller.

Lets do another type of model:

THIRD METHOD: Fit an ARIMA Model

Exponential smoothing and ARIMA models are the two most widely used approaches to time series forecasting.

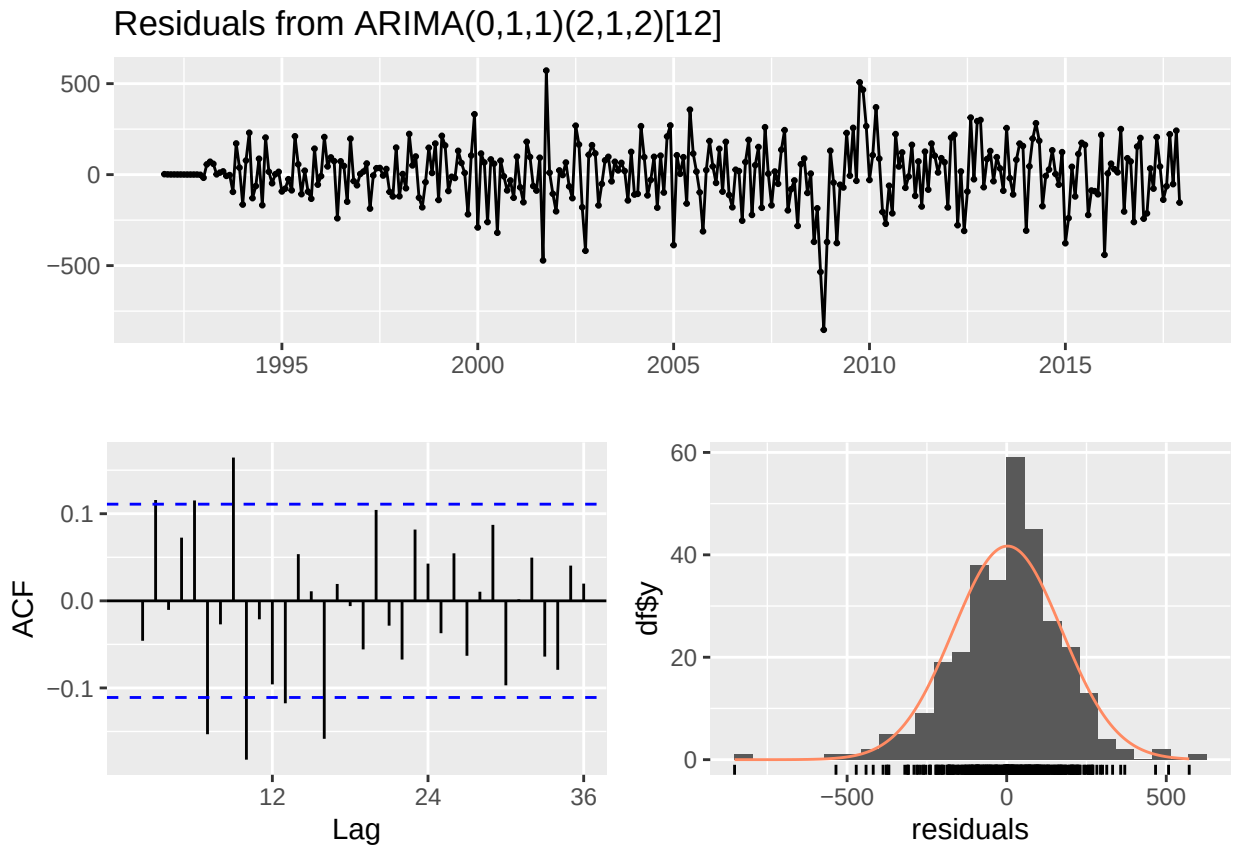
- ARIMA models aim to describe the autocorrelation in the data.
- Needs to be stationary data

So lets use differenced data, there is seasonality.

We can use regular data and tell it to take 1st difference (d=1) and 1st seasonality difference (D=1).

- "stepwise = FALSE" finds the model that fits better and try every single possibility, so it could take a lot of time
- "aproxximation = FALSE" will show the exactly result, not an approximation
- "Trace" will print all the models

```
## Series: Y
## ARIMA(0,1,1)(2,1,2)[12]
##
## Coefficients:
##           ma1    sar1     sar2     sma1    sma2
##       -0.3545  0.8436  -0.7189  -1.1980  0.6266
## s.e.   0.0580  0.0612   0.0826   0.0762  0.0837
##
## sigma^2 = 30589:  log likelihood = -1973.05
## AIC=3958.1   AICc=3958.38   BIC=3980.3
##
## Training set error measures:
##                     ME     RMSE      MAE          MPE     MAPE      MASE
## Training set 2.501669 169.7773 126.3304 -0.002525628 1.352898 0.2773436
##                       ACF1
## Training set -0.0009506685
```




```
##
##  Ljung-Box test
##
```

```
## data:  Residuals from ARIMA(0,1,1)(2,1,2)[12]
## Q* = 64.685, df = 19, p-value = 6.868e-07
##
## Model df: 5.   Total lags used: 24
```

```
#SD = sqrt(sigma^2)
sqrt(30589)
```

```
## [1] 174.8971
```

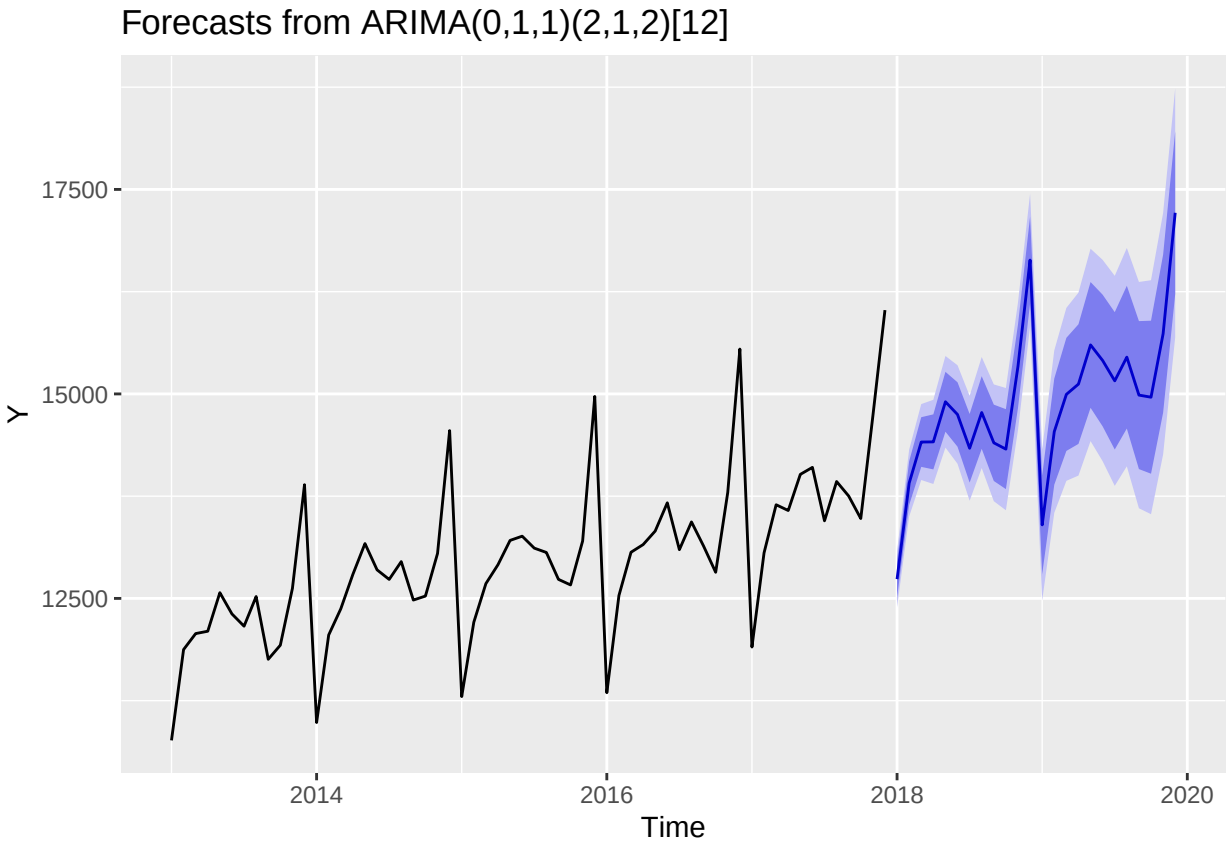
This model fits better, there is not much autorrelation in residuals (ACF), but is not perfect!

It means there is better model for this time series, but lets use this model.

FORECAST

Forecast with ARIMA Model

```
fcst <- forecast(fit_arima, h=24)
autoplot(fcst, include = 60)
```



```
print(summary(fcst))
```

```
##
## Forecast method: ARIMA(0,1,1)(2,1,2)[12]
##
## Model Information:
## Series: Y
## ARIMA(0,1,1)(2,1,2)[12]
##
## Coefficients:
##          ma1    sar1     sar2     sma1    sma2
##       -0.3545  0.8436  -0.7189  -1.1980  0.6266
## s.e.   0.0580  0.0612   0.0826   0.0762  0.0837
##
## sigma^2 = 30589:  log likelihood = -1973.05
## AIC=3958.1   AICc=3958.38   BIC=3980.3
##
## Error measures:
##                     ME     RMSE      MAE          MPE     MAPE      MASE
## Training set 2.501669 169.7773 126.3304 -0.002525628 1.352898 0.2773436
##                       ACF1
## Training set -0.0009506685
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2018       12735.82 12511.68 12959.96 12393.03 13078.61
## Feb 2018       13909.29 13642.50 14176.07 13501.27 14317.30
## Mar 2018       14411.74 14108.24 14715.23 13947.58 14875.90
## Apr 2018       14413.70 14077.47 14749.92 13899.49 14927.91
## May 2018       14903.98 14537.95 15270.02 14344.18 15463.79
## Jun 2018       14748.42 14354.83 15142.02 14146.47 15350.37
## Jul 2018       14335.54 13916.20 14754.89 13694.21 14976.88
## Aug 2018       14772.59 14328.98 15216.20 14094.15 15451.03
## Sep 2018       14402.36 13935.75 14868.97 13688.74 15115.97
## Oct 2018       14324.98 13836.45 14813.50 13577.84 15072.12
## Nov 2018       15339.64 14830.14 15849.15 14560.42 16118.87
## Dec 2018       16636.27 16106.62 17165.93 15826.24 17446.31
## Jan 2019       13399.32 12795.77 14002.88 12476.26 14322.39
## Feb 2019       14538.64 13889.81 15187.47 13546.35 15530.94
## Mar 2019       14994.55 14303.41 15685.69 13937.55 16051.55
## Apr 2019       15120.03 14389.03 15851.03 14002.06 16238.00
## May 2019       15599.12 14830.32 16367.92 14423.34 16774.90
## Jun 2019       15408.33 14603.50 16213.16 14177.45 16639.21
## Jul 2019       15160.02 14320.71 15999.34 13876.41 16443.64
## Aug 2019       15449.56 14577.13 16321.99 14115.29 16783.83
## Sep 2019       14985.54 14081.20 15889.88 13602.47 16368.61
## Oct 2019       14959.88 14024.72 15895.04 13529.67 16390.08
## Nov 2019       15736.02 14771.02 16701.02 14260.18 17211.86
## Dec 2019       17214.65 16220.71 18208.59 15694.55 18734.75
```

It looks realistic - "h = 24" forecasted period (months) - function Include, means the number of months (historic data)