Exponential Smoothing

Word Count: 1200 words[1]

Exponential smoothing

Exponential smoothing (ES) is an alternative to the Box-Jenkins ARIMA family of models that were proposed in the late 50s by Brown, Holt and Winters. Just like the ARIMA models, it is a univariate forecasting method. The difference is that ARIMA models make predictions based on a weighted sum of past observations. While ES models also make predictions based on a weighted sum of past observations, the weights assigned to past observations decay exponentially with the highest weight being assigned to the most recent observation. This is where these models get their name from. This family of models is known for producing quick and reliable results for many different types of time series.

Simple Exponential Smoothing

The simplest form of ES methods is the one called Simple Exponential Smoothing (SES) which can be fitted to time series that do not exhibit seasonal patterns or trends, such as stationary time series. Of course, trend and seasonality are common for time series and thus more complex methods have been developed and are discussed in the next chapter.

The intuition behind the ES method is that recent past observations are more important in making predictions than observations that are more distant. Hence, larger weights are assigned to more recent observations and the weights decay exponentially as we move to older observations.

$$\widehat{Y}_{T+1|T} = aY_T + a(1 - a)Y_{T-1} + a(1 - a^2)Y_{T-2} + ...$$

where $\alpha \in [0,1]$ is called the smoothing factor or the smoothing parameter and controls the decay rate of the weights. As stated in the above equation, the prediction for time T+1 is the weighted sum of the data from t=1 up to t=T. For example, for α=0.2, the five most recent observations carry the weight of ~67.23% and when α=0.8, they carry ~99.97%. Hence, for smaller values of α, the more distant observations are given more weight compared to higher values of α when the weights are concentrated on the most recent observations.

It is often convenient to write the SES in component form. For the SES, there is only one component - l_t or level. More components are added as the ES model becomes more complicated, such as for seasonal time series or for time series with trend. The component form of ES models consists of the forecast equation and the smoothing equation for each component. For the SES we have:

Forecast equation: $\widehat{y}_{t+h|t} = l_t$

Smoothing equation: $l_t = ay_t + (1 - a)l_{t-1}$

l_t is called level or smoothed value of the time series at time t. The forecast equation gives the fitted values for h=1, whereas for t=T, it gives predictions beyond the data that have been observed and are included in the dataset.

---

1 Excluding the explanation of the R implementation, the Mathematical Formulas, the Bibliography and the Appendix

It is also worth noting that $\hat{y}_{T+h|T} = \hat{y}_{T+1|T} = l_T$ for h = 2, 3, … . This means that all future forecasts take the value of the level component at time t=T which will be suitable only for stationary time series.

Fitting and optimisation of SES

To start the process, initial values of α and l_0 are required. These can be determined either subjectively by the forecaster based on prior experience or estimated from the data. The SES model is fitted on the data based on the following formula:

$$\hat{y}_{t+1|t} = ay_t + (1 - a)\hat{y}_{t|t-1}$$

The process starts with $\hat{y}_{2|1} = ay_1 + (1 - a)l_0$ and proceeds for t=1,2,...,T-1. For t=T, it starts making predictions in the future.

In order to estimate the optimal parameter α and l_0, we need to minimise the Sum of Squared Errors:

$$\text{SSE} = \sum_{t=1}^{T} e_t = \sum_{t=1}^{T} (y_t - \hat{y}_{t|t-1})^2$$

As a final note, it is optimal to use SES when the generating process of the time series is a non-stationary infinite order MA or an ARIMA(0,1,1) but SES has also been shown to be robust to plenty of other models [3].

Trend method

In this chapter, an extended version of the SES is presented. This method allows for forecasting using time series that exhibit trends. This is why there is a trend equation in addition to the forecasting and level equations in the case of SES:

Forecast equation: $\hat{y}_{t+h|t} = l_t + hb_t$

Smoothing equation: $l_t = ay_t + (1 - a)(l_{t-1} + b_{t-1})$

Trend equation: $b_t = \beta^*(l_t - l_{t-1}) + (1 - \beta^*)b_{t-1}$

where α is the smoothing parameter, l_t is the estimate of the level at time t, b_t is the estimate of the trend at time t and β^* is the smoothing parameter of the trend. Both α and $\beta^* \in [0,1]$. Smaller values of β^* correspond to small changes in the estimated trend. As in the case of the SES, l_t is a weighted average of the observation at time t and the one-step forecast for time t, $l_{t-1} - b_{t-1}$. b_t is a weighted average of the estimated trend, $l_t - l_{t-1}$ and the previous estimate of the trend, b_{t-1}.

It should also be noted that, unlike in the case of the SES, the future predictions do not take the same value but the h-step forecast is equal to l_T plus h times b_T. Hence, the h-step forecast is a linear function of h since l_T and b_T are the same.

Damped trend
The previous model makes the unrealistic assumption that the trend remains constant over time. This makes for very poor long-term predictions. As a result of introducing another parameter φ, which dampens the trend toward a constant value, this method has proven to be very successful, especially for forecasts further in the future [4].

Holt and Winters method

Holt and Winters extended the trend method of ES by adding another equation to capture seasonality in the data. The seasonal component s_t is introduced along with the parameters γ and m which denotes the seasonality (i.e. m=12 for monthly data).
There exist two variations of this method. First, there is the additive method in which the seasonal component is expressed in terms of the scale of the data. This method is used when there are not significant differences in the variations between seasons. Second, there is the multiplicative method where the seasonal component is expressed as percentage.it is used when the variation within each season is changing in proportion to the level.

In the case of the additive method, we subtract the seasonal component from the data to deseasonalize the time series, whereas in the case of the multiplicative method, we divide the data by the seasonal component.

Forecast equation: $\hat{y}_{t+h|t} = l_t + hb_t + s_{t+h-m(k+1)}$
Smoothing equation: $l_t = a(y_t - s_{t-m}) + (1 - a)(l_{t-1} + b_{t-1})$
Trend equation: $b_t = \beta^*(l_t - l_{t-1}) + (1 - \beta^*)b_{t-1}$
Seasonal equation: $s_t = \gamma(y_t - l_{t-1} - b_{t-1}) + (1 - \gamma)s_{t-m}$

where k is the integer part of (h-1)/m which adjusts the index of the seasonal component so that only the final year seasonal indices are used for forecasting. The level is again a weighted sum between the deseasonalised data and the deseasonalised forecast of t-1, while the seasonal equation is the weighted sum of the current seasonal index and the seasonal index of the previous year. Also, the parameter γ ∈ [0,1-α].

Similarly, the multiplicative method is given by the equations:

Forecast equation: $\hat{y}_{t+h|t} = (l_t + hb_t)s_{t+h-m(k+1)}$
Smoothing equation: $l_t = a\frac{y_t}{s_{t-m}} + (1 - a)(l_{t-1} + b_{t-1})$
Trend equation: $b_t = \beta^*(l_t - l_{t-1}) + (1 - \beta^*)b_{t-1}$
Seasonal equation: $s_t = \gamma\frac{y_t}{(l_{t-1}+b_{t-1})} + (1 - \gamma)s_{t-m}$

In both cases, the smoothing parameters are estimated by minimising the RMSE on the training data. As is the case with the ES with trend method, there exists a damped method for seasonality - both for the additive and the multiplicative method.

Summary

The Exponential Smoothing family gives a variety of different methods for forecasting with different sets of data. The results obtained by these, rather simple, statistical methods are usually reliable and are used in various applications. For example, during the M4 forecasting competition, the best performing model was a hybrid approach that included Holt-Winters multiplicative method and the Recurrent Neural Network. The ES method was used in many of the higher performing models, most of the time combined with other ML models This suggests that, although it is a rather old method, it is very useful until this day.

Implementation of fitting SES models in R

We will start by obtaining a time series of the price of Google's stock for a period of 1000 time points. The first 900 will be used as a training set, i.e. to fit the ES model and the last 100 as a validation set to optimise the hyper-parameters. First we look at the plot of the time series and observe that there's a trend Fig. 1. So we difference the time series to produce a stationary one.

Since the data appear to be stationary, we can fit an SES model with smoothing parameter $\alpha=0.3$. In order to assess its predictions on the validation set, we also need to difference the validation set. Then, we can evaluate the predictions through various metrics Table 1 where we get RMSE of 8.1558 on the test set. We might be able to get better accuracy by tuning the α parameter, so we calculate the RMSE on the validation set for $\alpha=0.01,...,0.99$. The smaller the value, the better the RMSE, so we next fit an SES model with $\alpha=0.01$. This produces RMSE of 8.0831 Table 2. Finally, we plot the predictions along with the difference validation set and the 80% and 95% predictions intervals, and observe that the prediction interval is much narrower than the one from the first model.

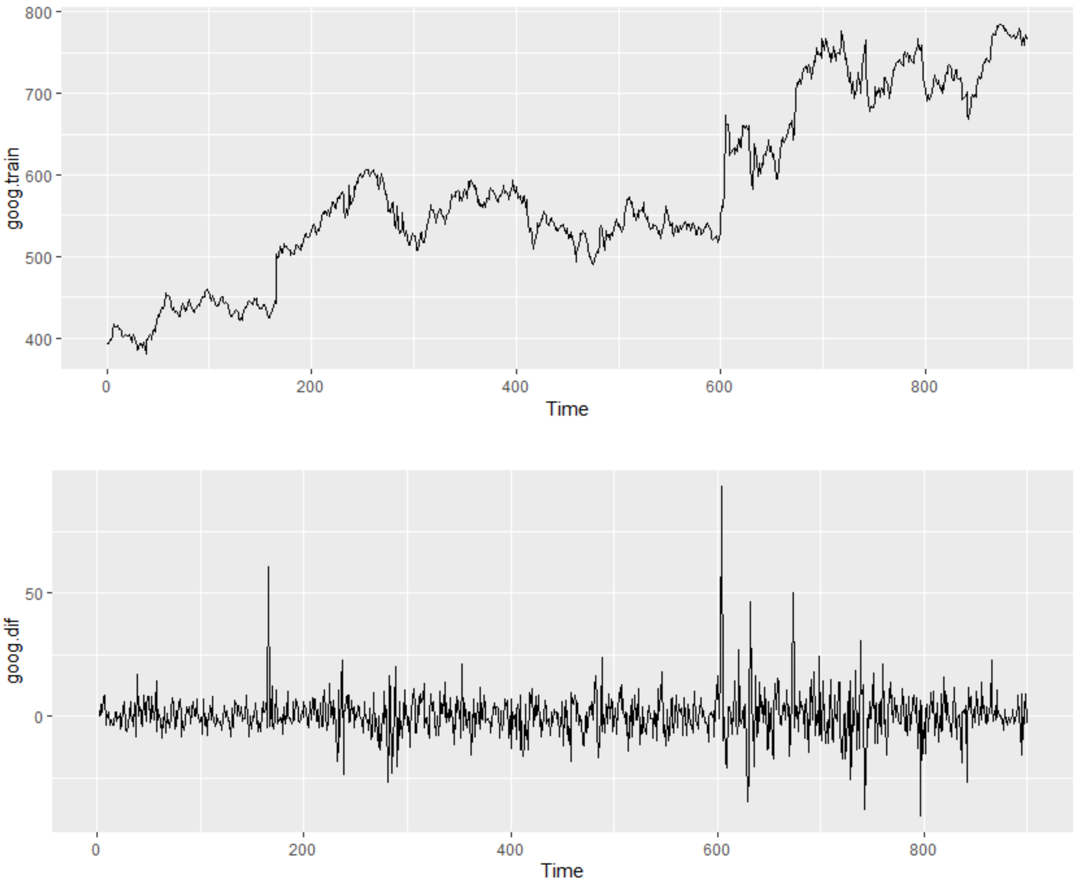

Fig. 1 Plot of the original time series and its first difference

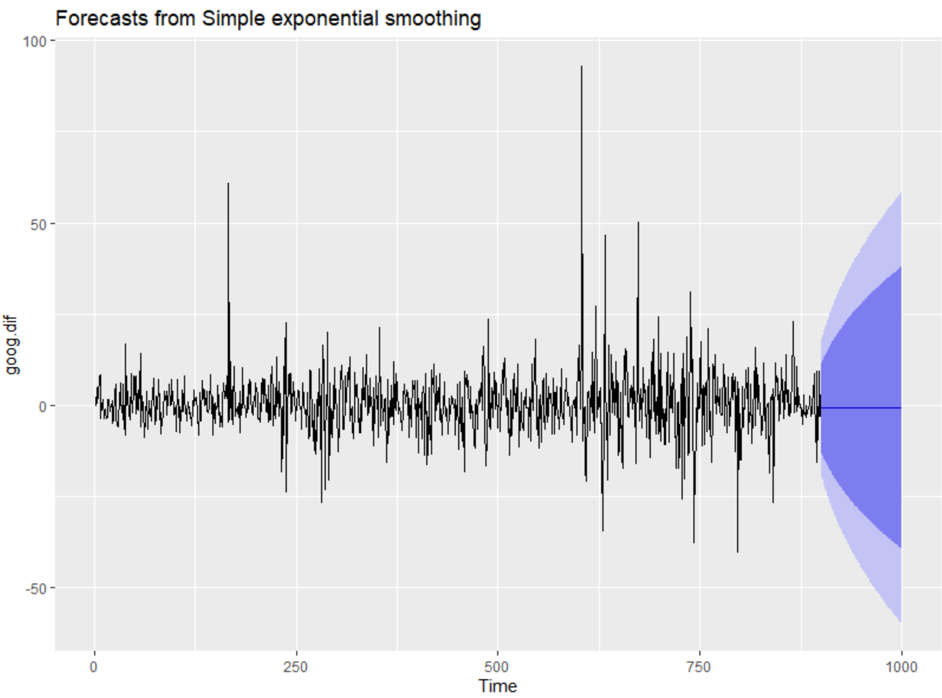

Fig. 2 Predictions of the SES model with 80% and 95% prediction interval

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	-0.0101515	9.571124	6.604688	88.50833	304.6519	0.7815623	-0.1047378	NA
Test set	1.0858739	8.155805	6.137003	112.00388	195.9062	0.7262191	0.1227814	0.9889647

Table 1. Evaluation metrics of SES model with α=0.30

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	-0.01139031	8.829941	5.865211	99.32230	126.7409	0.6940567	0.02886557	NA
Test set	-0.00412292	8.083196	6.015444	92.14942	154.1342	0.7118344	0.12278141	1.006324

Table 2. Evaluation metrics of the SES model with α=0.01

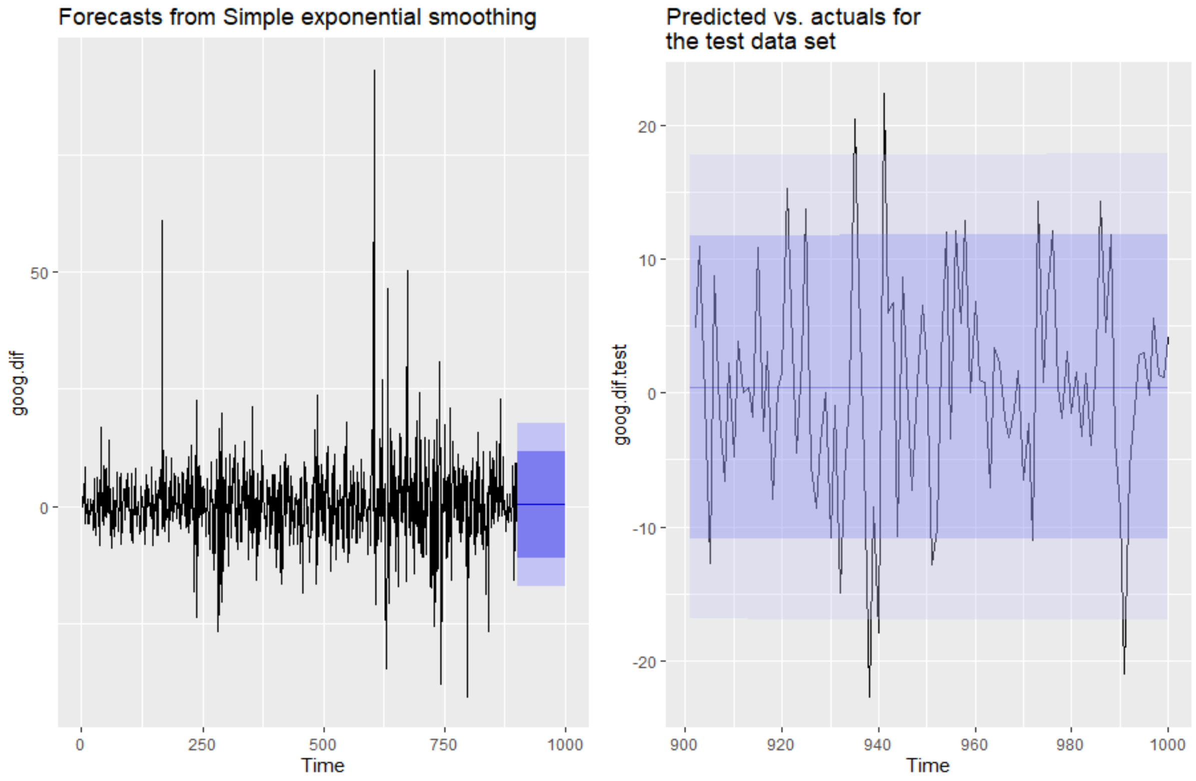

Fig 4. Plot of the training data with the predictions and the differenced validation set with the predictions and the 80% and 95% prediction interval

Bibliography

[1] CHRISTOPHER CHATFIELD, HAIPENG XING, The analysis of time series : an introduction with R, Boca Raton, FL : CRC Press: Seventh edition, 2019

[2] Hyndman, Rob J. & Athanasopoulos, George. & OTexts.com, issuing body. (2014). Forecasting : principles and practice. [Heathmont?, Victoria] : OTexts.com

[3] Chatfield, Chris, Anne B. Koehler, J. Keith Ord, and Ralph D. Snyder. "A New Look at Models for Exponential Smoothing." Journal of the Royal Statistical Society. Series D (The Statistician) 50, no. 2 (2001): 147-59.

[4] Gardner Jr., E.S. and McKenzie, E. (1985) Forecasting Trends in Time Series. Management Science, 31, 1237-1246.

[5] Spyros Makridakis; Evangelos Spiliotis and Vassilios Assimakopoulos, (2020), The M4 Competition: 100,000 time series and 61 forecasting methods, International Journal of Forecasting, 36, (1), 54-74

[6] Slawek Smyl, Jai Ranganathan, Andrea Pasqua, 2018, Uber Engineering, https://eng.uber.com/m4-forecasting-competition/

[7] Bradley Boehmke, UC Business Analytics R Programming: Exponential Smoothing, http://uc-r.github.io/ts_exp_smoothing

Appendix

```
# loading the required packages
library(tidyverse)
library(fpp2)

# create training and testing set
# of the Google stock data
goog.train <- window(goog, end = 900)
goog.test <- window(goog, start = 901)

#plot of time series
m<-autoplot(goog.train)
# removing the trend
goog.dif <- diff(goog.train)
m1<-autoplot(goog.dif)

gridExtra::grid.arrange(m ,m1, nrow = 2)

# applying SES on the filtered data
ses.goog.dif <- ses(goog.dif, alpha = .3, h = 100)
autoplot(ses.goog.dif)

library(formattable)
formattable(as.data.frame(accuracy(ses.goog.dif, goog.dif.test)))

# comparing our model
alpha <- seq(.01, .99, by = .01)
RMSE <- NA

for(i in seq_along(alpha)) {

  fit <- ses(goog.dif, alpha = alpha[i],
             h = 100)
  RMSE[i] <- sqrt(fit$model$mse)
}
# convert to a data frame and
# identify min alpha value
alpha.fit <- data_frame(alpha, RMSE)
alpha.min <- filter(alpha.fit,RMSE == min(RMSE))

# plot RMSE vs. alpha
ggplot(alpha.fit, aes(alpha, RMSE)) +
  geom_line() +
  geom_point(data = alpha.min, aes(alpha, RMSE), size = 2, color = "red")

# refit model with alpha = .05
ses.goog.opt <- ses(goog.dif, alpha = .01, h = 100)

# performance eval
formattable(as.data.frame(accuracy(ses.goog.opt, goog.dif.test)))

# plotting results
p1 <- autoplot(ses.goog.opt) +
  theme(legend.position = "bottom")
p2 <- autoplot(goog.dif.test) +
  autolayer(ses.goog.opt, alpha = .3) +
  ggtitle("Predicted vs. actuals for the test data set")

gridExtra::grid.arrange(p1, p2, nrow = 1)
```