Exponential smoothing in the **dataIrony** package

Søren Højsgaard

February 28, 2019

Contents

1 Introduction

2 Exponential smoothing

2.1 Single exponential smoothing

Given is a time series $\{y_1, y_2, \dots, y_T\}$ recorded at equidistant time points. Single exponential smoothing (SES) of data results in a new time series $\{S_1, S_2, \dots, S_T\}$:

$$S_t = \alpha y_t + (1-\alpha) S_{t-1}, \quad S_1 = y_1$$

where $0 < \alpha < 1$.

Given data up to time t, the natural forecast h time steps ahead is

$$\hat{y}_{t+h|t} = S_t$$

so the one–step–ahead forecast error is $e_t = y_t - \hat{y}_{t|t-1} = y_t - S_{t-1}$.

Single exponential smoothing works well in the sense of producing reasonable predictions if there is no clear trend in data, i.e. if $y_t \approx a$ for all t.

Notice: An alternative form is:

$$S_t = S_{t-1} + \alpha(y_t - S_{t-1}) = S_{t-1} + \alpha e_t$$

2.2 Double exponential smoothing

Single exponential smoothing does not work well (when it comes to producing reliable predictions) when there is a trend in data, i.e. if $y_t \approx a + bt$. Single exponential smoothing will be biased in the sense that

$$y_t - S_t = b\frac{\beta}{\alpha} \text{ for } t \to \infty \text{ where } \beta = 1 - \alpha$$

Hence $S_t \approx y_t - b\frac{\beta}{\alpha}$ and $y_t \approx S_t + b\frac{\beta}{\alpha}$ for large t.

If $y_t \approx a + bt$ then $S_t \approx (a - b\frac{\beta}{\alpha}) + bt = \tilde{a} + bt$. Therefore, if we smooth S_t (i.e. smooth data twice) we get (by the same argument) that

$$S_t - S_t^{[2]} \approx b\frac{\beta}{\alpha} \text{ and } S_t^{[2]} = (a - 2b\frac{\beta}{\alpha}) + bt \text{ for } t \to \infty$$

From these considerations we therefore have

$$b = \{S_t - S_t^{[2]}\}\frac{\alpha}{\beta} \quad (1)$$

$$y_t = S_t - b\frac{\beta}{\alpha} = S_t + (S_t - S_t^{[2]}) = 2S_t t - S_t^{[2]} \quad (2)$$

Hence natural estimates of levels and slopes become

$$\hat{y}_t = S_t + \{S_t - S_t^{[2]}\} = 2S_t - S_t^{[2]} \quad (3)$$

$$\hat{b}_t = \{S_t - S_t^{[2]}\}\frac{\alpha}{\beta} \quad (4)$$

Similarly, the forecasts become

$$\hat{y}_{t+h|t} = \hat{y}_t + \hat{b}_t h \quad (5)$$

3 Handling non-equidistant measurements

First consider equidistant data. For single exponential smoothing, the forecast error is $e_t = y_t - \hat{y}_{t|t-1} = y_t - S_{t-1}$. Hence

$$S_t = S_{t-1} + \alpha e_t$$

Hence, α is the weight attributed to the forecast error for recordings that are one time unit apart.

For non–equidistant data we let t_j denote the time of the jth measurement and we have accordingly $S_{t_j} = S_{t_{j-1}} + \alpha e_{t_j}$. Intuition suggests that the weight attributed to the forecast error should become smaller when $d_j = t_j - t_{j-1}$ becomes larger. Somewhat arbitrarily we handle non–equidistant measurements by

$$S_{t_j} = S_{t_{j-1}} + w(\alpha, d_j)e_{t_j}$$

where the weight $w()$ is given by

$$w(\alpha, d) = 1 - (1 - \alpha)^d$$

4 Parameter estimation

For single exponential smoothing we note that minimizing $\sum(y_t - \hat{y}_{t|t})^2$ will lead to $\alpha = 1$ and $S_t = y_t$ so that no smoothing takes place. Since double exponential smoothing is simple single exponential smoothing repeated twice the same phenomenon will appear for double exponential smoothing.

Instead we minimize (by default) the one step ahead forecast error $\sum(y_t - \hat{y}_{t|t-1})^2$.

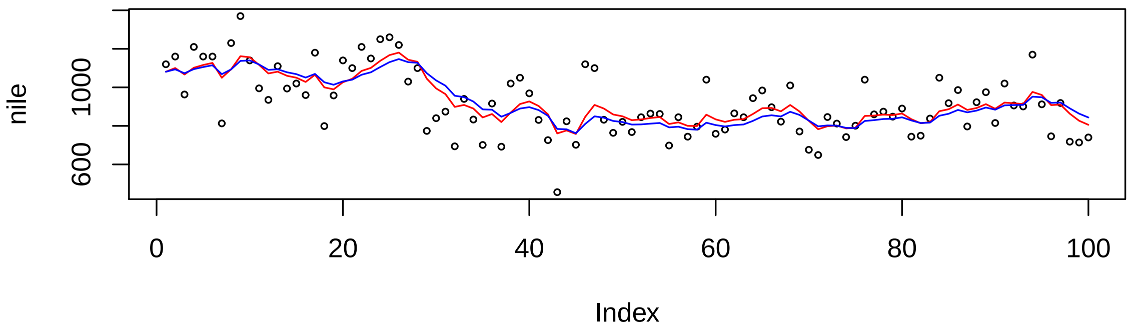
5 Example: Nile data

```
nile <- as.numeric(Nile) # a numeric vector
head(nile)

## [1] 1120 1160  963 1210 1160 1160
```

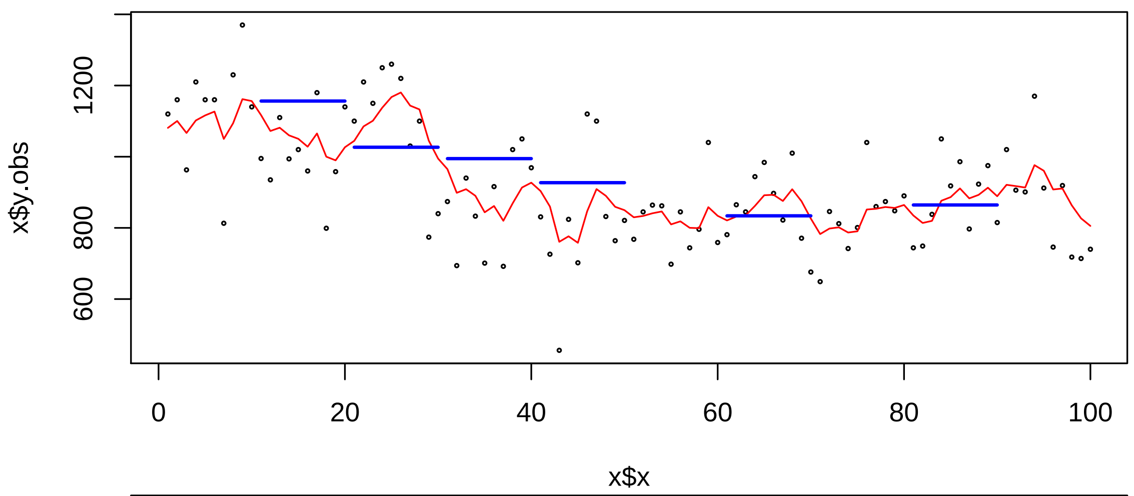
Create single and double exponential smoothing objects. When α is not specified, the parameter is estimated by by minimizing the 1–step forecasting error.

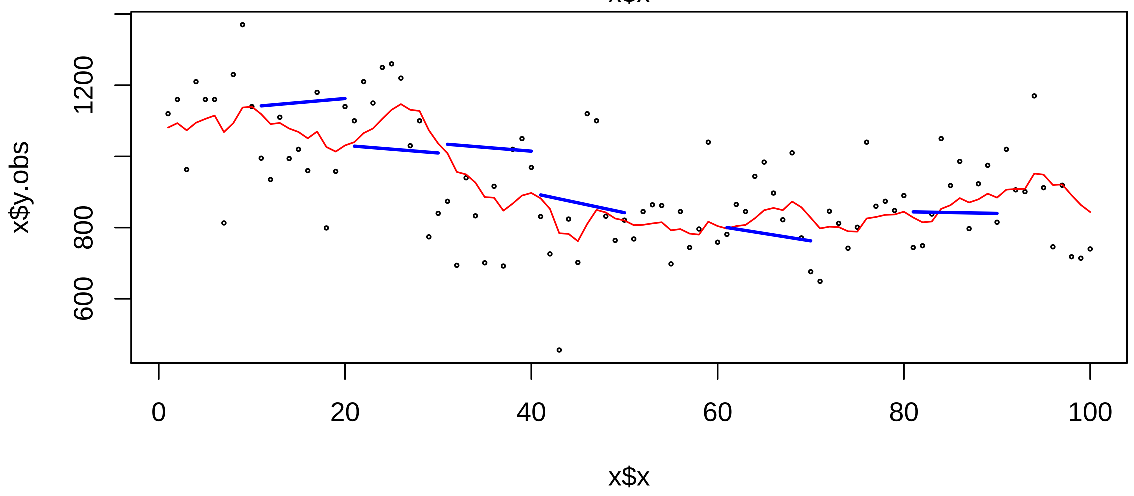
```
ses1 <- ses(nile)
des1 <- des(nile)
ses1

## List of 2
##  $ cls  : chr ".SES"
##  $ alpha: num 0.245

des1

## List of 2
##  $ cls  : chr ".DES"
##  $ alpha: num 0.0823
```

```
plot(nile, cex=.5)
lines(ses1, col="red")
lines(des1, col="blue")
```



```
par(mfrow=c(2,1))
plot(ses1, cex=.3)
z <- lapply(c(10, 20, 30, 40, 60, 80),
       function(a)
           lines(forecast(ses1, at=a, h=1:10), col="blue", lwd=2))

plot(des1, cex=.3)
z <- lapply(c(10, 20, 30, 40, 60, 80),
       function(a)
           lines(forecast(des1, at=a, h=1:10), col="blue", lwd=2))
```





6 Example: JohnsonJohnson

This dataset gives the quarterly earnings (dollars) per Johnson & Johnson share in the period 1960-80. On the log–scale, data is approximately linear, see Fig. 1.

```
y <- log10(as.numeric(JohnsonJohnson))
par(mfrow=c(1,2))
plot(JohnsonJohnson); plot(y, cex=.5)
```

The bias in SES is clear if a small value of α is chosen, whereas DES quickly captures the structure in data, see Fig. 2.

```
ses1 <- ses(y, alpha=.1, fit=FALSE)
des1 <- des(y, alpha=.1, fit=FALSE)
ses1

## List of 2
##  $ cls  : chr ".SES"
##  $ alpha: num 0.1

des1
```

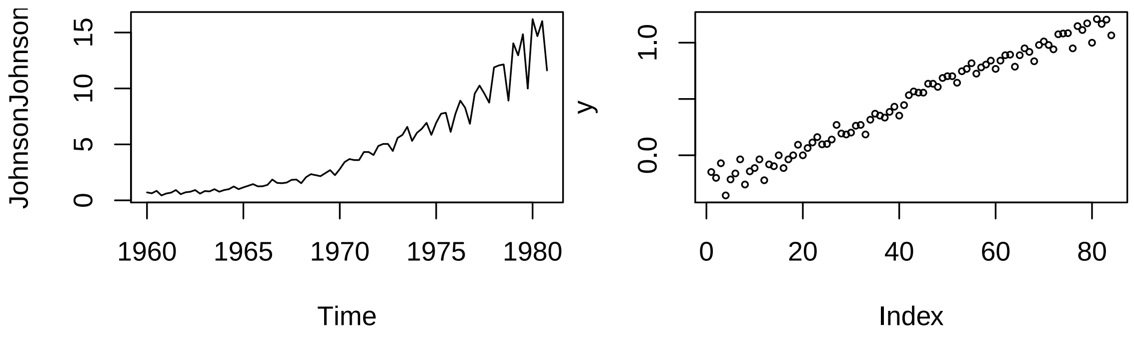


Figure 1: bla

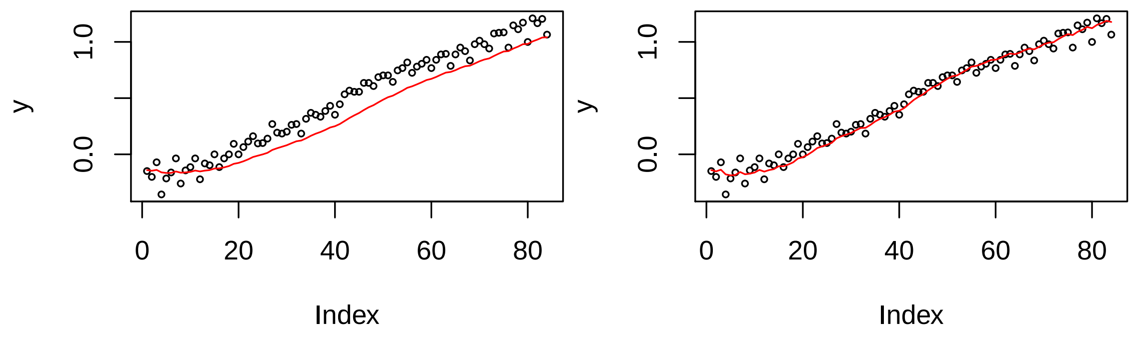


Figure 2: bla

```
## List of 2
##  $ cls  : chr ".DES"
##  $ alpha: num 0.1
```

When fitting α to data, the smoothed values fit better to data, but the predictions of SES are bad because of the linear trend in data, cfr. Fig. 3.

```
ses1 <- ses(y)
des1 <- des(y)
ses1

## List of 2
##  $ cls  : chr ".SES"
##  $ alpha: num 0.502

des1

## List of 2
##  $ cls  : chr ".DES"
##  $ alpha: num 0.16
```

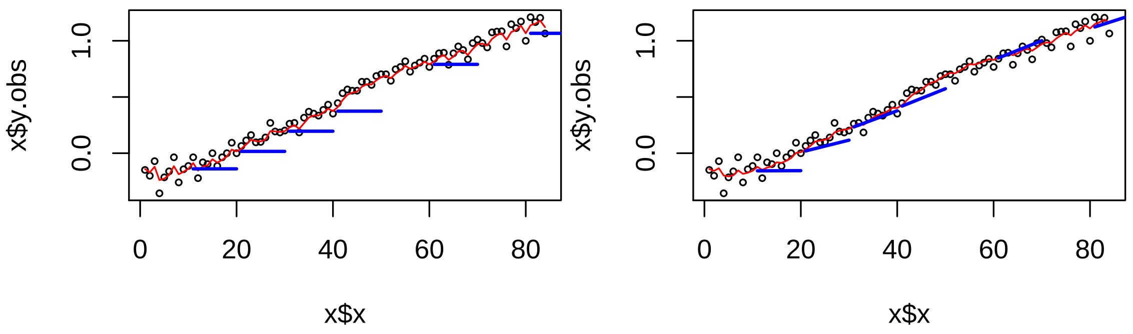


Figure 3: bla