Exponential Smoothing with Trend and Seasonality

Here we go! The big enchilada! Triple Exponential Smoothing!

There is an obvious periodic or seasonal component to the *AirPassengers* data set introduced in the last lecture. This is true for many data sets (think back to the Mauna Loa CO2 data set). We have already seen how to take a smoothed average in order to use past values in a forecast, and we can also accommodate a rising and falling data set by making a guess on current trend via current system value and past trend values. We will do something similar to bring seasonality into our forecasting. We'll also allow ourselves to look *h* steps into the future, instead of just one step as we've been doing so far.

We start with the "additive form" of the equations (i.e. we start with seasonal differences which are additive.)

- For example, if you believe your seasonal differences are *additive*, you might adjust your projected sales figures for canoes for the month of June based on, say, your January sales figures by adding a constant amount to your January. That is, you might take January sales data and add an extra 10,000 canoes to obtain the June projection.

- If you feel the seasonal differences are *multiplicative*, you might instead adjust your projected sales figures for canoes for the month of June based on your January sales figures by multiplying your January data by some constant amount. That is, you might take January sales data and multiply by three to obtain the June projection.

Additive seasonality doesn't care about the overall levels, whereas multiplicative seasonality does. A nice website to explore these differences is available from Nikolaos Kourentzes:

http://www.forsoc.net/2014/11/11/can-you-identify-additive-and-multiplicative-seasonality/

It asks the germane question "Can you identify additive and multiplicative seasonality?" and gives you some practice. We will discuss this some more in the context of Air Passenger Miles.

Developing Triple Exponential Smoothing Formulas with Additive Seasonality

The equations for your forecast with ***additive seasonality*** will look like (don't worry about that strange looking last term just yet)

$$\hat{x}_{n+h} = \text{level}_n + h \cdot \text{trend}_n + \text{seasonal}_{n+h-m}$$

We hope you won't mind that, once we develop a one-step-ahead term called $trend_n$, we move into the future h steps in the simplest possible way by multiplying $trend_n$ by the number of time lags h we wish to look into the future (i.e. distance travelled is the size of your step multiplied by the number of steps).

Multiplicative seasonality will look much the same, except that we will *multiply*

$$\hat{x}_{n+h} = (\text{level}_n + h \cdot \text{trend}_n) \cdot \text{seasonal}_{n+h-m}$$

Common Format

Our level, trend, and seasonal terms will all look like

$$\text{greek letter} \cdot this + (1 - \text{greek letter}) \cdot that$$

We will peer h steps into the future for a time series with seasonality of length m.

- h = number of steps into the future for forecast
- m = length of the season in your data

You can break the forecast down like this.

1. Smooth the Level

To get started, develop the level term similarly to what we did before, but now take a weighted average of the *seasonally adjusted level* $x_n - s_{n-m}$ (we say we are *deseasonalizing*) with the non-seasonal forecast $level_{n-1} + trend_{n-1}$.

$$level_n = \alpha(x_n - \text{seasonal}_{n-m}) + (1 - \alpha)\,(level_{n-1} + trend_{n-1})$$

2. **Smooth the Trend**

   For the trend term, we keep it just as before

$$trend_n = \beta \cdot (level_n - level_{n-1}) + (1 - \beta) \cdot trend_{n-1}$$

3. **Smooth the Season**

   Finally, for the seasonal term we bring in a new parameter, γ and smooth over past seasons

$$seasonal_n = \gamma \cdot (x_n - level_n) + (1 - \gamma) seasonal_{n-\mathrm{m}}$$

4. **Update the Forecast**

   For additive seasonality we use

$$\hat{x}_{\mathrm{n+h}} = level_n + h \cdot trend_n + seasonal_{n+h-m}$$

   And for multiplicative seasonality we modify very slightly

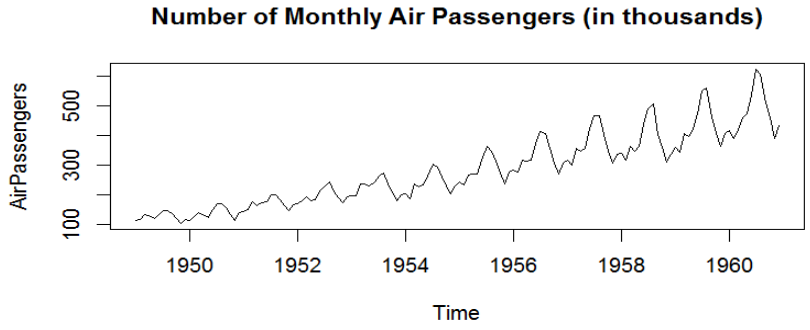
$$\hat{x}_{\mathrm{n+h}} = (level_n + h \cdot trend_n) \cdot seasonal_{n+h-m}$$

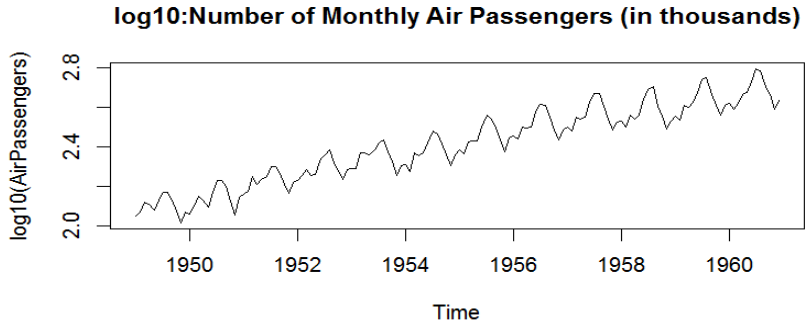
Your software will find optimal values of α and β and γ.

Airline Data

Recall the *AirPassengers* data set describing monthly totals of international airline passengers during the years 1949 to 1960.

Source: Box, G. E. P., Jenkins, G. M. and Reinsel, G. C. (1976) *Time Series Analysis, Forecasting and Control*. Third Edition. Holden-Day. Series G.





A plot of the time series reveals strong seasonality as well as trend. The data set also seems to be heteroscedastic, with increasing variability in later years. We transformed our data set and instead we considering $\log_{10}(\text{number of passengers})$. This gives us a data set with apparently additive seasonality and a calmer looking plot.

We saw that SES essentially led us to naïve forecasting.

$$\hat{x}_{n+1} = \alpha x_n$$

We obtained a quick quality measure,

AirPassengers.SES = HoltWinters(log10(AirPassengers), beta=FALSE, gamma=FALSE)

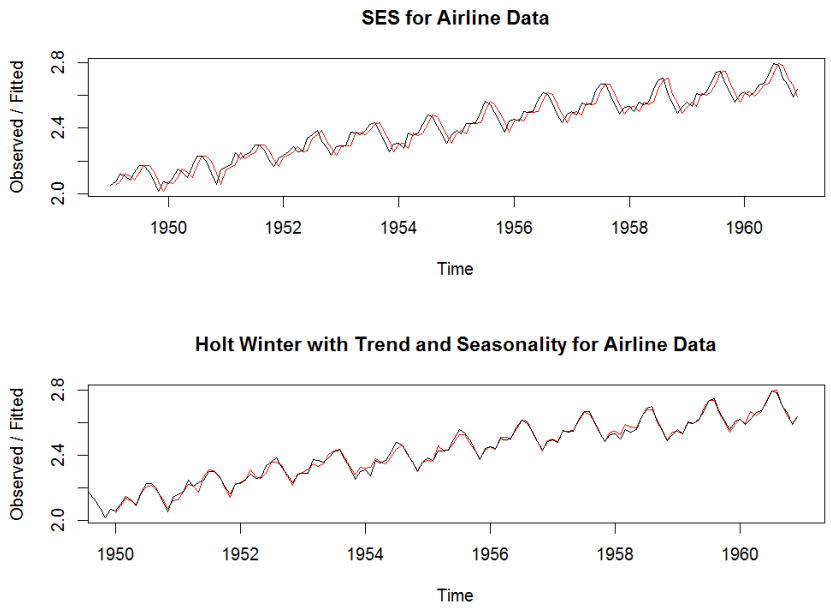
AirPassengers.SES$SSE # 0.3065102

Let's see if we can reduce the error. The call will be very simple! Additive seasonality is the default. (Standard R documentation tells us that additive is the default since it is the first possibility listed for the seasonal argument: *seasonal = c("additive", "multiplicative")*

AirPassengers.HW = HoltWinters(log10(AirPassengers))

Our error certainly decreases and we have forecasts that hug the curve:

AirPassengers.HW$SSE *# 0.0383026*



The output gives us our parameters and we can build our forecast:

Results:

Smoothing parameters:

alpha:	*AirPassengers.hw$alpha*	*#returns 0.326612*
beta :	*AirPassengers.hw$beta*	*#returns 0.005744246*
gamma:	*AirPassengers.hw$gamma*	*#returns 0.8207255*

AirPassengers.hw$coefficients

Coefficients:		*s1*	*-0.031790733*	*s7*	*0.127820295*
	[,1]	*s2*	*-0.061224237*	*s8*	*0.119893006*
a	*2.680598830*	*s3*	*-0.015941495*	*s9*	*0.038321663*
b	*0.003900787*	*s4*	*0.006307818*	*s10*	*-0.014181699*
		s5	*0.014138008*	*s11*	*-0.085995400*
		s6	*0.067260071*	*s12*	*-0.044672707*

Forecast for January 1961

$$\hat{x}_{n+h} = level_n + h \cdot trend_n + seasonal_{n+h-m}$$

$$\hat{x}_{144+1} = level_{144} + 1 \cdot trend_{144} + seasonal_{144+1-12}$$

$$\hat{x}_{144+1} = 2.680599 + 1 \cdot 0.003900787 + (-0.031790733)$$

$$\hat{x}_{144+1} = 2.652709$$

Forecast for August 1961

$$\hat{x}_{n+h} = level_n + h \cdot trend_n + seasonal_{n+h-m}$$

$$\hat{x}_{144+8} = level_{144} + 1 \cdot trend_{144} + seasonal_{144+8-12}$$

$$\hat{x}_{144+8} = 2.680599 + 8 \cdot 0.003900787 + (0.119893006)$$

$$\hat{x}_{144+1} = 2.831698$$

Can you build the forecast for December 1961? How about March 1962?

We've been eliding an issue and implicitly assuming with our subscript notation that we'd only look into the future by $h < m$ months. Using modular (clock) arithmetic we can set up a more complete (but less obvious) notation.

Since you are likely to use software for your forecasts this is not much of a problem. In a hand calculation example like we've been doing, if you are looking far into the future, just make sure you grab the correct seasonal coefficient for the month of interest. For March 1962:

$$\hat{x}_{144+15} = 2.680599 + 15 * 0.003900787 + (-0.015941495) = 2.723169$$

Rather than build the explicit forecast ourselves from the return values, let's make a quick call to the routine *forecast.HoltWinters ()* in the library *forecast.*

```
rm(list=ls(all=TRUE))
library("forecast")
AirPassengers.hw          <-    HoltWinters(log10(AirPassengers))
AirPassengers.forecast    <-    forecast.HoltWinters(AirPassengers.hw)
```

You can obtain point estimates and confidence intervals (shown here for 80% and 95%) for your forecasts. I'll include the forecasts for 1961 below. You can see the point estimates in our figure as a solid line, together with shading indicating the interval estimates. (Remember that these are transformed values!)

AirPassengers.forecast

		Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
Jan	1961	2.652709	2.630898	2.674520	2.619351	2.686066
Feb	1961	2.627176	2.604218	2.650134	2.592065	2.662287
Mar	1961	2.676360	2.652297	2.700422	2.639560	2.713160
Apr	1961	2.702510	2.677380	2.727640	2.664077	2.740942
May	1961	2.714241	2.688076	2.740406	2.674225	2.754257
Jun	1961	2.771264	2.744092	2.798436	2.729708	2.812820
Jul	1961	2.835725	2.807571	2.863878	2.792667	2.878782
Aug	1961	2.831698	2.802586	2.860811	2.787174	2.876222
Sep	1961	2.754028	2.723977	2.784079	2.708069	2.799987
Oct	1961	2.705425	2.674454	2.736396	2.658059	2.752791
Nov	1961	2.637512	2.605638	2.669386	2.588765	2.686259
Dec	1961	2.682736	2.649974	2.715497	2.632631	2.732840
Jan	1962	2.699518	2.661306	2.737731	2.641078	2.757959
Feb	1962	2.673986	2.635014	2.712957	2.614383	2.733588
Mar	1962	2.723169	2.683445	2.762894	2.662416	2.783923
Apr	1962	2.749319	2.708848	2.789790	2.687424	2.811214
May	1962	2.761050	2.719838	2.802262	2.698022	2.824078
Jun	1962	2.818073	2.776126	2.860020	2.753921	2.882226
Jul	1962	2.882534	2.839857	2.925211	2.817265	2.947803
Aug	1962	2.878508	2.835105	2.921910	2.812129	2.944886
Sep	1962	2.800837	2.756714	2.844960	2.733356	2.868318
Oct	1962	2.752234	2.707395	2.797074	2.683658	2.820811
Nov	1962	2.684322	2.638770	2.729873	2.614656	2.753987
Dec	1962	2.729545	2.683285	2.775805	2.658796	2.800294

And, of course we should produce a plot:

```
plot(AirPassengers.forecast,  xlim=c(1949, 1963))
```

The plot shows the point forecasts, together with shading regions indicating the appropriate confidence bands.