Chapter 8 Review

Anthony Tricarico

Table of contents

This chapter deals with exponential smoothing, a family of methods used to produce forecasts that attaches more importance (larger weight) to more recent observations.

> **!** Important
>
> Examples taken for this chapter come straight out of the book.
> Also, this is still a WIP.

1 Simple Exponential Smoothing

Using this method, the further away in the past one observation is the lower the weight attached to it. This is one of the simplest ways to use exponential smoothing and implies an exponential decay of the past observations. The only parameter needed by this method is α which ranges from 0 to 1 and controls how small the weights attached to the preceding observation get (larger α's implying a faster decay of the weights).

Using this technique, we are not taking into account any trend or seasonality patterns that might characterize the time series.

In the book there are alternative ways of representing mathematically the functional form of the simple exponential smoothing method but we won't delve too much into them here. Among them, I'd recommend getting familiar with the *component form* which is used extensively in later sections.

As it is usually the case with such methods, we can try and select the necessary parameters, or we can estimate them using the procedure below as explained in chapter 8:

```
library(fpp3)
```

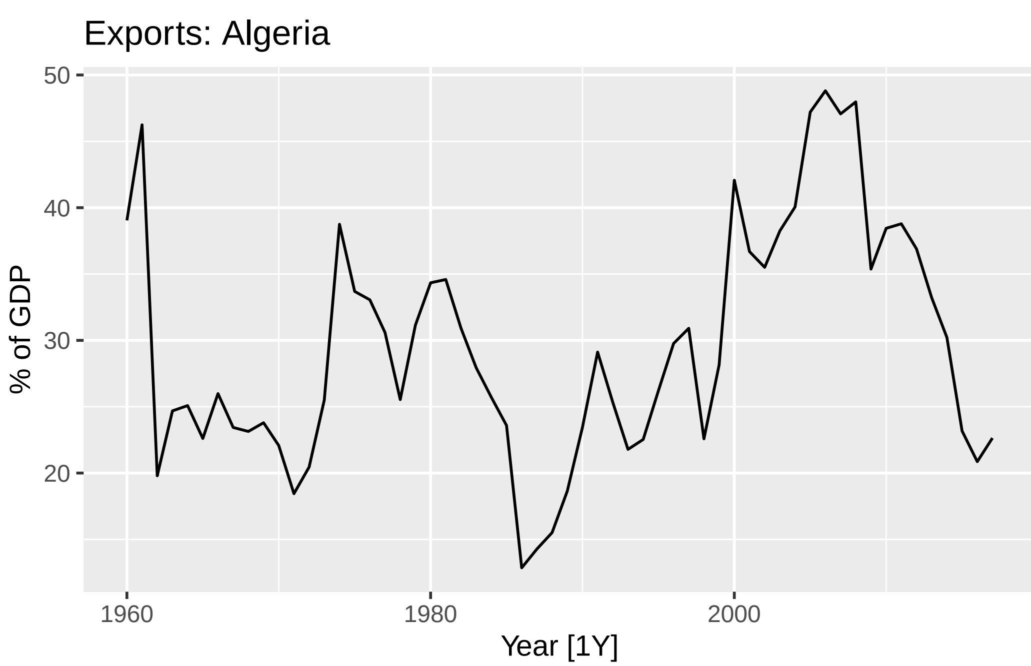
```
Registered S3 method overwritten by 'tsibble':
  method               from 
  as_tibble.grouped_df dplyr

-- Attaching packages -------------------------------------------- fpp3 1.0.1 --

v tibble      3.2.1     v tsibble     1.1.5
v dplyr       1.1.4     v tsibbledata 0.4.1
v tidyr       1.3.1     v feasts      0.4.1
v lubridate   1.9.3     v fable       0.4.1
v ggplot2     3.5.1     

-- Conflicts ------------------------------------------------- fpp3_conflicts --
x lubridate::date()    masks base::date()
x dplyr::filter()      masks stats::filter()
x tsibble::intersect() masks base::intersect()
x tsibble::interval()  masks lubridate::interval()
x dplyr::lag()         masks stats::lag()
x tsibble::setdiff()   masks base::setdiff()
x tsibble::union()     masks base::union()
```

```r
algeria_economy <- global_economy |>
  filter(Country == "Algeria")
algeria_economy |>
  autoplot(Exports) +
  labs(y = "% of GDP", title = "Exports: Algeria")
```



```
# Estimate parameters
fit <- algeria_economy |>
  model(ETS(Exports ~ error("A") + trend("N") + season("N")))
fc <- fit |>
  forecast(h = 5)

tidy(fit) %>%
  select(term, estimate)
```

```
# A tibble: 2 x 2
  term  estimate
  <chr>    <dbl>
1 alpha    0.840
2 l[0]    39.5
```

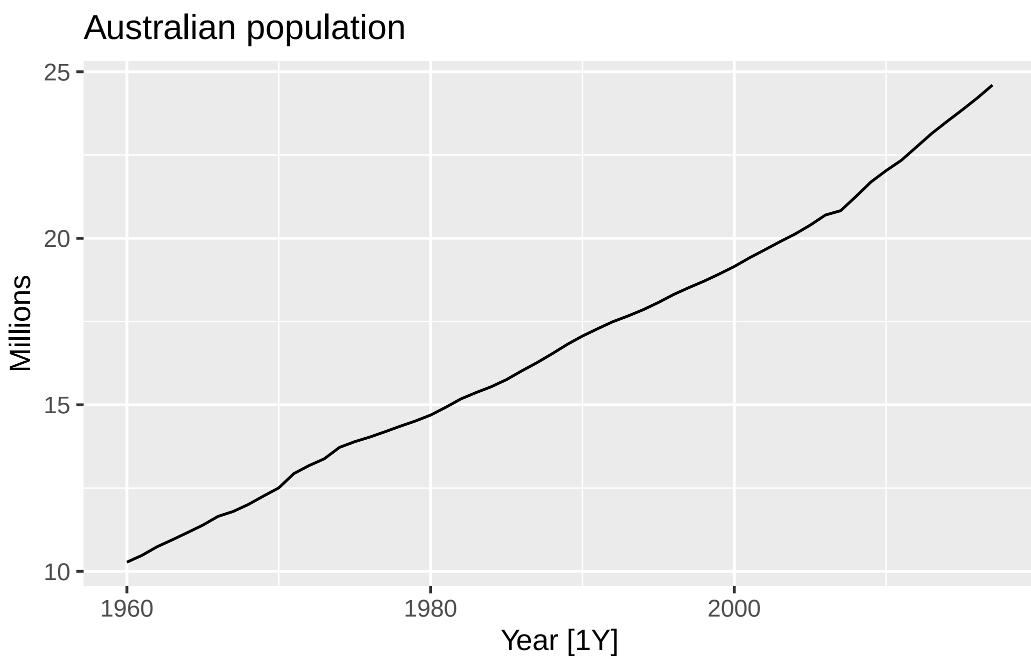
Here we see the estimated $\alpha = 0.84$ and $l_0 = 39.5$.

2 Holt's Linear Trend Method

One way to model time series which present a trend is to use Holt's method. Using this method we are not only estimating a decay parameter α but we are also estimating a β parameter to estimate by how much the trend component should decay as well. This would allow for a trending estimate that has now a linear functional form instead of a flat one.

```
aus_economy <- global_economy |>
  filter(Code == "AUS") |>
  mutate(Pop = Population / 1e6)

autoplot(aus_economy, Pop) +
  labs(y = "Millions", title = "Australian population")
```




```
fit <- aus_economy |>
  model(
    AAN = ETS(Pop ~ error("A") + trend("A") + season("N"))
  )
fc <- fit |> forecast(h = 10)

tidy(fit)
```

```
# A tibble: 4 x 4
  Country   .model term  estimate
  <fct>     <chr>  <chr>    <dbl>
1 Australia AAN    alpha    1.00
2 Australia AAN    beta     0.327
3 Australia AAN    l[0]    10.1
4 Australia AAN    b[0]     0.222
```

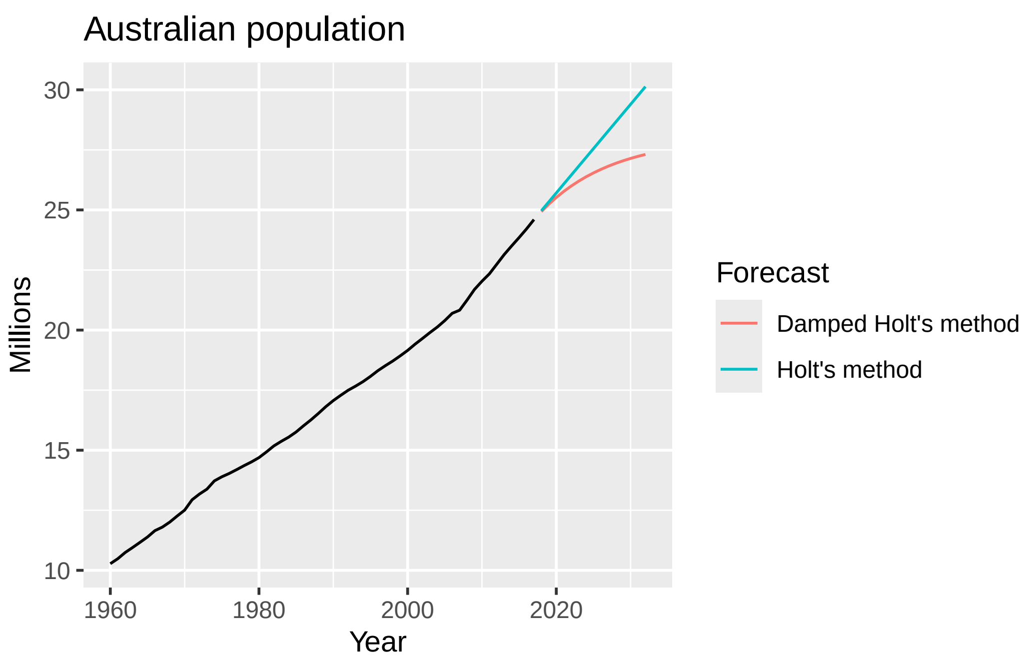
2.1 Damped trend methods

Since the method described above in Section 2 produces forecasts that grow (or decline) indefinitely in the future, some other methods were developed to alleviate the issue of over-forecasting. The damped trend methods allow for the trend component to change in a way that observations that are very far into the future converge to a constant value and stop growing, reducing the probability of over-forecasting.

```
aus_economy |>
  model(
    `Holt's method` = ETS(Pop ~ error("A") +
                          trend("A") + season("N")),
```

```
  `Damped Holt's method` = ETS(Pop ~ error("A") +
                         trend("Ad", phi = 0.9) + season("N"))        # (1)
  ) |>
  forecast(h = 15) |>
  autoplot(aus_economy, level = NULL) +
  labs(title = "Australian population",
       y = "Millions") +
  guides(colour = guide_legend(title = "Forecast"))
```

(1) The dampening parameter is specified by the argument `phi` inside of the `ETS` function. Remember that also `phi` must be in the interval [0,1]. Also, the short-hand `"Ad"` is now used to specify that we are using an *Additive Damped variant* in place of the standard Additive component. Moreover, when `phi` is not specified it will be estimated automatically as long as `"Ad"` has been specified before.

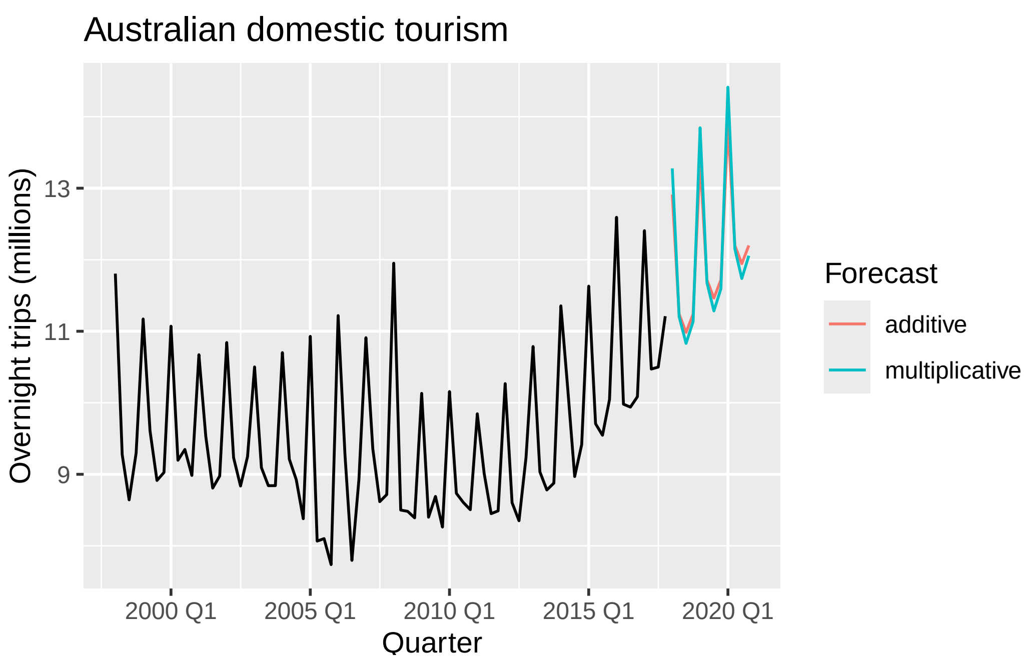


3 Holt-Winter's Method

Similarly to what seen so far, we now want to also account for seasonality in the time series other than its trend. Luckily, the method explained in this section does exactly that in a way that is similar to what described in the previous sections. Since now we would like to also estimate how the seasonal component varies, we assign it a weight γ that needs to be estimated. Using either the additive or the multiplicative formulation, we can estimate γ and start producing forecasts using this method as well.

```
aus_holidays <- tourism |>
  filter(Purpose == "Holiday") |>
  summarise(Trips = sum(Trips)/1000)
fit <- aus_holidays |>
  model(
```

```
    additive = ETS(Trips ~ error("A") + trend("A") +
                                                season("A")),
    multiplicative = ETS(Trips ~ error("M") + trend("A") +
                                                season("M"))
  )
fc <- fit |> forecast(h = "3 years")
fc |>
  autoplot(aus_holidays, level = NULL) +
  labs(title="Australian domestic tourism",
       y="Overnight trips (millions)") +
  guides(colour = guide_legend(title = "Forecast"))
```




```
tidy(fit)
```

```
# A tibble: 18 x 3
   .model         term   estimate
   <chr>          <chr>     <dbl>
 1 additive       alpha  0.262
 2 additive       beta   0.0431
 3 additive       gamma  0.000100
 4 additive       l[0]   9.79
 5 additive       b[0]   0.0211
 6 additive       s[0]  -0.534
 7 additive       s[-1] -0.670
 8 additive       s[-2] -0.294
 9 additive       s[-3]  1.50
10 multiplicative alpha  0.224
11 multiplicative beta   0.0304
```

```
12 multiplicative gamma  0.000100
13 multiplicative l[0]  10.0
14 multiplicative b[0]  -0.0114
15 multiplicative s[0]   0.943
16 multiplicative s[-1]  0.927
17 multiplicative s[-2]  0.969
18 multiplicative s[-3]  1.16
```

```
accuracy(fit) %>%                                                          ①
  select(.model, RMSE)
```

① Get in-sample accuracy

```
# A tibble: 2 x 2
  .model          RMSE
  <chr>          <dbl>
1 additive       0.417
2 multiplicative 0.412
```

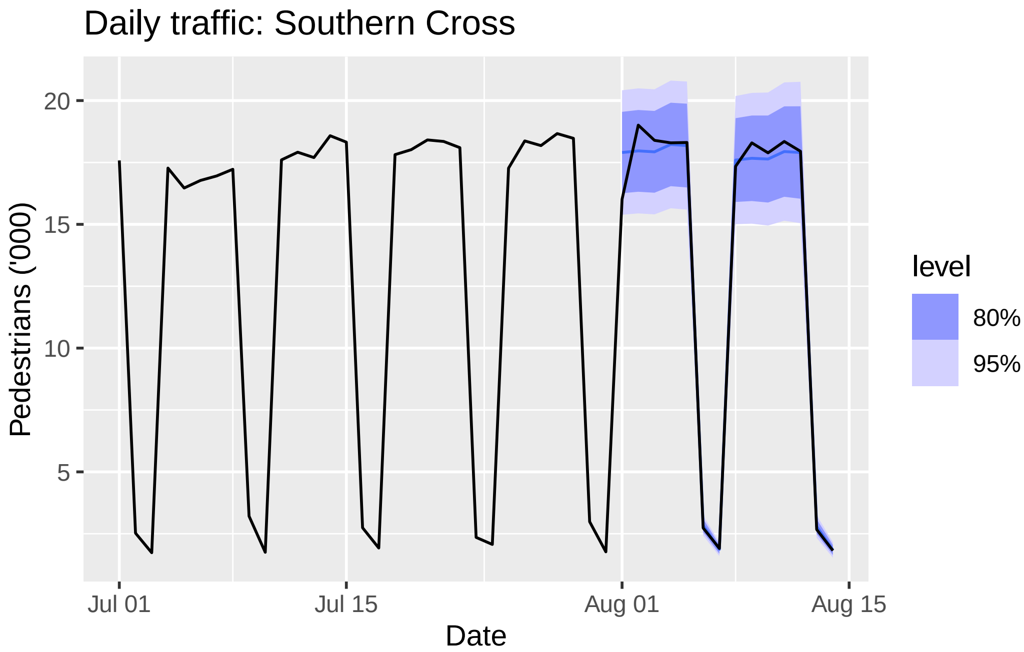
3.1 Holt-Winters' damped method

Similarly to the other methods, also here we can apply a dampening parameter to avoid over-forecasting in the future.

```
sth_cross_ped <- pedestrian |>
  filter(Date >= "2016-07-01",
         Sensor == "Southern Cross Station") |>
  index_by(Date) |>
  summarise(Count = sum(Count)/1000)

sth_cross_ped |>
  filter(Date <= "2016-07-31") |>
  model(
    hw = ETS(Count ~ error("M") + trend("Ad") + season("M"))              ①
  ) |>
  forecast(h = "2 weeks") |>
  autoplot(sth_cross_ped |> filter(Date <= "2016-08-14")) +
  labs(title = "Daily traffic: Southern Cross",
       y="Pedestrians ('000)")
```

① Notice the damped trend component (`"Ad"`) and the multiplicative seasonality component (`"M"`). This is usually the standard when we want to produce forecasts using the Holt-Winters' damped method.


Daily traffic: Southern Cross
Pedestrians ('000)
20
15
10
5
Jul 01
Jul 15
Aug 01
Aug 15
Date
level
80%
95%