Chapter 8: Exponential smoothing

Ankit Gupta

27/12/2023

Exponential smoothing was proposed in the late 1950s (Brown, 1959; Holt, 1957; Winters, 1960), and has motivated some of the most successful forecasting methods. Forecasts produced using exponential smoothing methods are weighted averages of past observations, with the weights decaying exponentially as the observations get older. In other words, the more recent the observation the higher the associated weight. This framework generates reliable forecasts quickly and for a wide range of time series, which is a great advantage and of major importance to applications in industry.

This chapter is divided into two parts. In the first part (Sections 8.1–8.4) we present the mechanics of the most important exponential smoothing methods, and their application in forecasting time series with various characteristics. This helps us develop an intuition to how these methods work. In this setting, selecting and using a forecasting method may appear to be somewhat ad hoc. The selection of the method is generally based on recognising key components of the time series (trend and seasonal) and the way in which these enter the smoothing method (e.g., in an additive, damped or multiplicative manner).

In the second part of the chapter (Sections 8.5–8.7) we present the statistical models that underlie exponential smoothing methods. These models generate identical point forecasts to the methods discussed in the first part of the chapter, but also generate prediction intervals. Furthermore, this statistical framework allows for genuine model selection between competing models.

8.1 Simple exponential smoothing

Simple Methods

Suppose we have a time series $y_1, y_2, ..., y_T$ and we want to generate a forecast for this time series.

Random walk forecasts This is a naive forecast. Here forecasts are given by what observed last.

$\hat{y}_{T+h|T} = y_T$

Average forecasts $\hat{y}_{T+h|T} = \frac{1}{T}\Sigma_{t=1}^{T} y_t$

So, thinking about the weights and thinking about the past observations, these two forecasts take two extremes into account. Average forecasts talk about the history we have seen with equal weights. The above two methods are extremes and we want the methods between them and also, most recent data should have more weight. We care more about recent history than the past history.

Now, we see what simple exponential smoothing does.

Forecast equation

$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + ...,$

where α is smoothing coefficient and $0 \leq \alpha \leq 1$.

So, here forecast for one step ahead is weighted average of $y_T, y_{T-1}, ...$ and here weights are decaying exponentially when we go in the past as $\alpha > \alpha(1-\alpha) > \alpha(1-\alpha)^2 > ...$

The higher the value of α, the quicker the decay is and the lower the α is, the slower the exponential decay is.

The traditional way of representing the forecast equation is following. Every exponential smoothing method has a forecast equation as well as at least one smoothing equation.

Component Form:

(1) *Forecast Equation:* $\hat{y}_{t+h|t} = l_t$

(2) *Smoothing Equation:* $l_t = \alpha y_t + (1-\alpha) l_{t-1}$

Here,

- l_t is the level (or the smoothed value) of the series at time t.
- $\hat{y}_{t+1|t} = \alpha y_t + (1-\alpha)\hat{y}_{t|t-1}$
- $\hat{y}_{T+h|T} = l_T,\ h = 2, 3, ...$

Iterate to get exponentially weighted moving average form.

$\hat{y}_{T+1|T} = \Sigma_{j=0}^{T-1} \alpha(1-\alpha)^j y_{T-j} + (1-\alpha)^T l_0$

This is the equivalent form of previous forecast equation of weighted average form.

Now, we need to choose best values for α and l_0

- It is similar to regression. Choose optimal parameters by minimizing SSE:

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

- Unlike regression there is no closed form solution. Use numerical optimization.
- l_0 is initial value and so when $\alpha = 0$ then we get $\hat{y} = l_0$ which can be taken as mean of all time series values.

Example: Algerian Exports

```
library(fpp3)
```

```
## -- Attaching packages --------------------------------------------- fpp3 0.5 --
```

```
## v tibble      3.2.1     v tsibble     1.1.3
## v dplyr       1.1.3     v tsibbledata 0.4.1
## v tidyr       1.3.0     v feasts      0.3.1
## v lubridate   1.9.3     v fable       0.3.3
## v ggplot2     3.4.4     v fabletools  0.3.4
```

```
## -- 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()
```

```
algeria_economy <- global_economy |>
  filter(Country=="Algeria")
fit <- algeria_economy |>
  model(ANN=ETS(Exports ~ error("A")+trend("N")+season("N")))
report(fit)
```

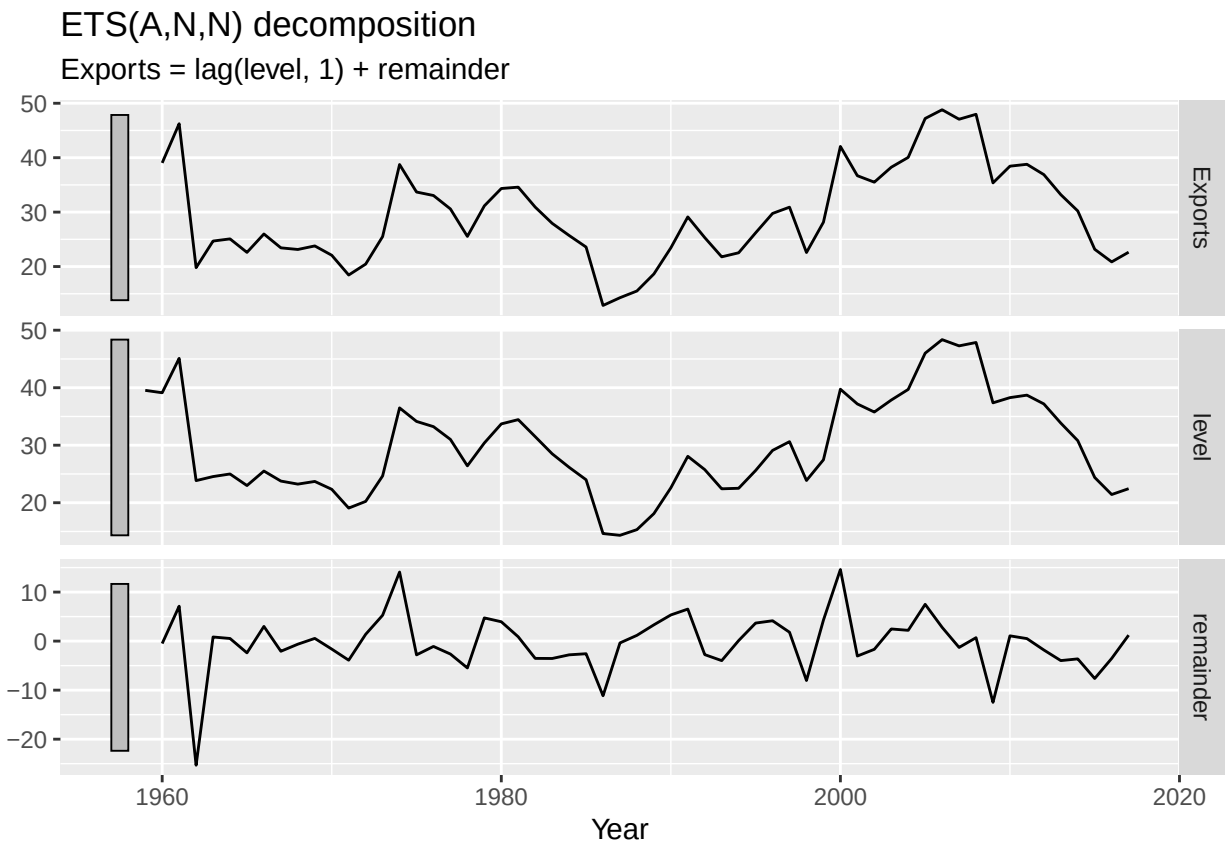
```
## Series: Exports
## Model: ETS(A,N,N)
##   Smoothing parameters:
##     alpha = 0.8399875
##
##   Initial states:
##     l[0]
##  39.539
##
##   sigma^2:  35.6301
##
##      AIC     AICc      BIC
## 446.7154 447.1599 452.8968
```

ETS stands for error,trend and seasonal. In simple exponential smoothing, we don't have the time and season component. Here, in error(), "A" stands for additive. In output, we get $\alpha = 0.83$ and initial value $l_0 = 39.539$

Now, we can take the above mable and plot its components.

```
components(fit) |> autoplot()
```

```
## Warning: Removed 1 row containing missing values (`geom_line()`).
```




```
components(fit) |>
  left_join(fitted(fit),by=c("Country",".model","Year"))
```

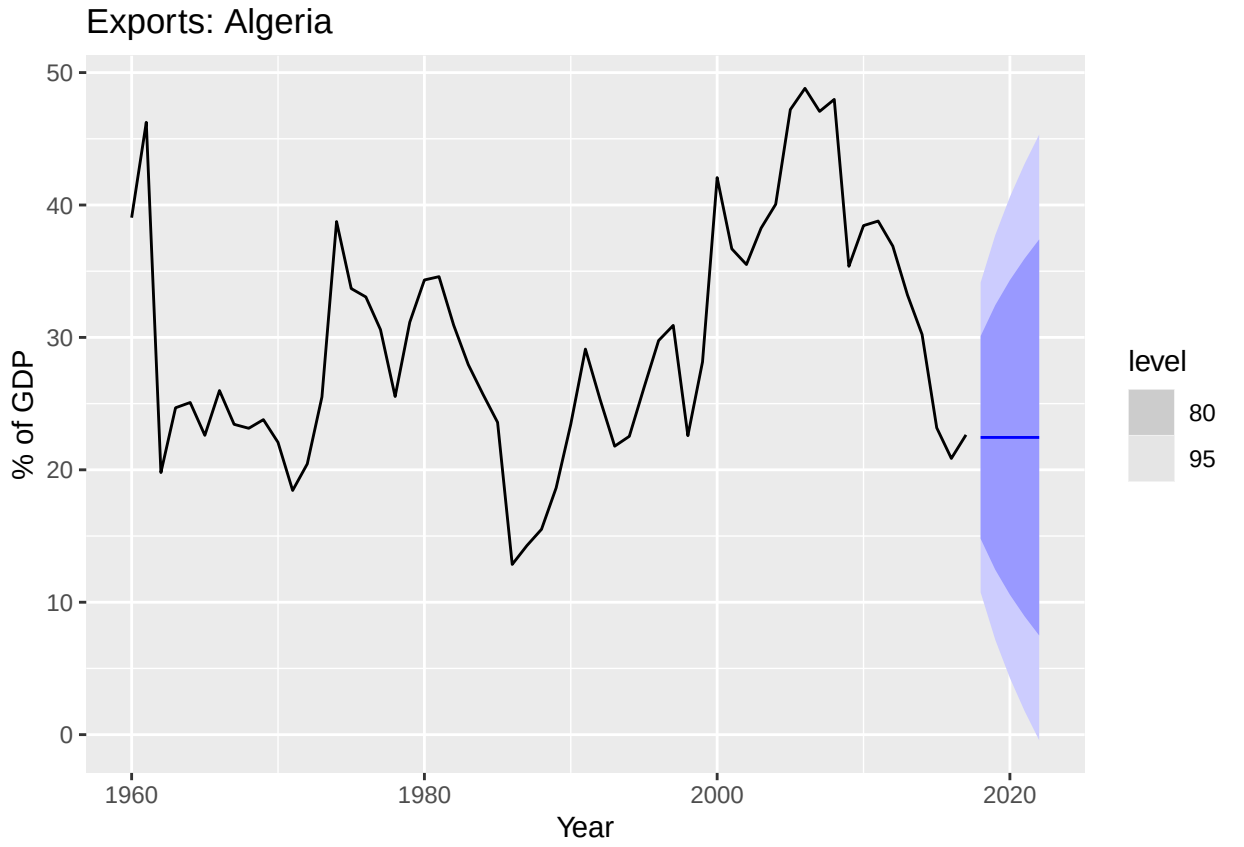
```
## # A dable: 59 x 7 [1Y]
```

```
## # Key:     Country, .model [1]
## # :       Exports = lag(level, 1) + remainder
##    Country .model  Year Exports level remainder .fitted
##    <fct>   <chr>  <dbl>   <dbl> <dbl>     <dbl>   <dbl>
##  1 Algeria ANN     1959    NA    39.5    NA        NA
##  2 Algeria ANN     1960    39.0  39.1    -0.496    39.5
##  3 Algeria ANN     1961    46.2  45.1     7.12     39.1
##  4 Algeria ANN     1962    19.8  23.8   -25.3      45.1
##  5 Algeria ANN     1963    24.7  24.6     0.841    23.8
##  6 Algeria ANN     1964    25.1  25.0     0.534    24.6
##  7 Algeria ANN     1965    22.6  23.0    -2.39     25.0
##  8 Algeria ANN     1966    26.0  25.5     3.00     23.0
##  9 Algeria ANN     1967    23.4  23.8    -2.07     25.5
## 10 Algeria ANN     1968    23.1  23.2    -0.630    23.8
## # i 49 more rows
```

Here, when you check the .fitted column and the level column, you will find initial value from level column is $l_0 = 39.5$ and so $\hat{y}_1 = l_0$ in fitted value column and similarly $l_1 = 39.1$ and so $\hat{y}_2 = l_1$

Now, to generate the forecast, we take the mable and pass it into the forecast() function as:

```
fit |>
  forecast(h=5) |>
  autoplot(algeria_economy) +
  labs(y= "% of GDP",title = "Exports: Algeria")
```

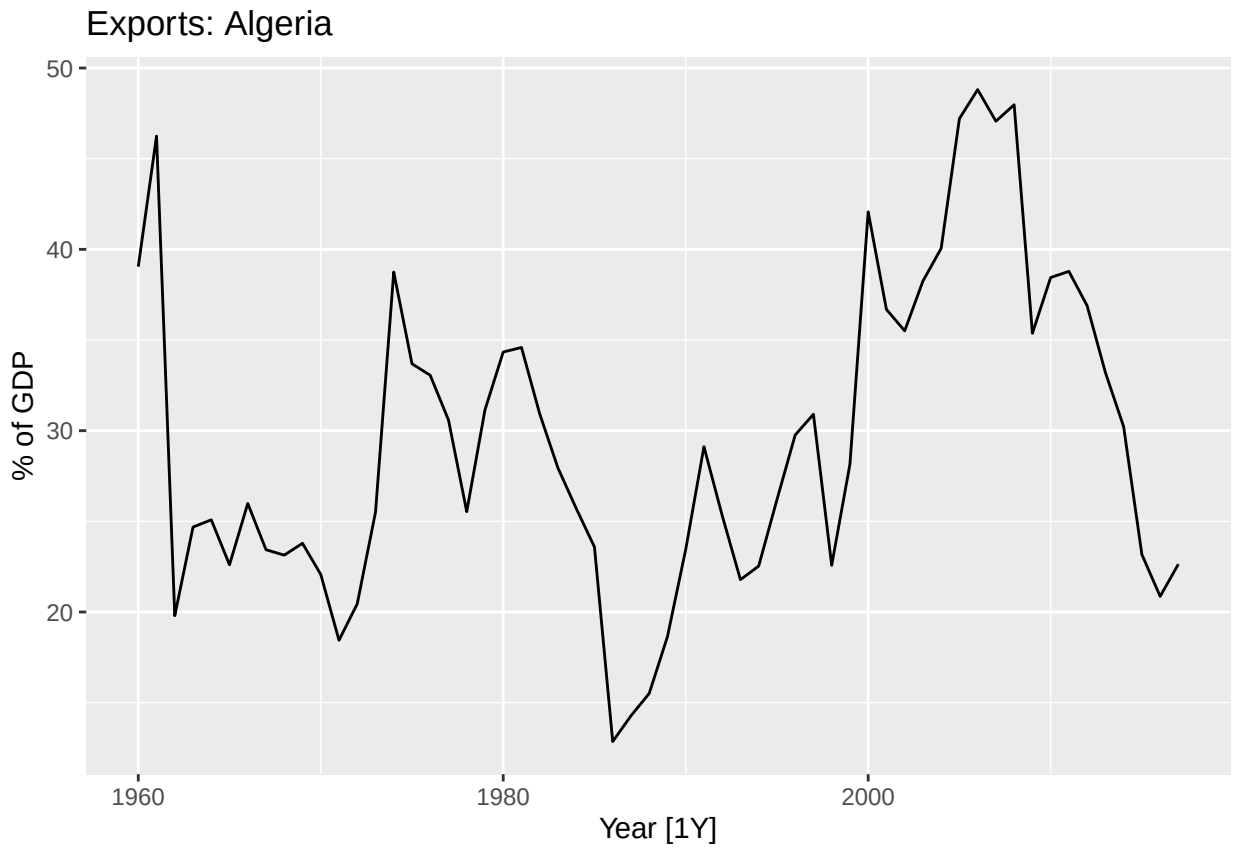


here, we are generating forecast for 5 steps ahead. Here, we are getting the line of point forecast from the previous weighted average formula.

Summary

The simplest of the exponentially smoothing methods is naturally called simple exponential smoothing (SES)15. This method is suitable for forecasting data with no clear trend or seasonal pattern. For example, the data in Figure 8.1 do not display any clear trending behaviour or any seasonality. (There is a decline in the last few years, which might suggest a trend. We will consider whether a trended method would be better for this series later in this chapter.) We have already considered the naïve and the average as possible methods for forecasting such data (Section 5.2).

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



Using the naïve method, all forecasts for the future are equal to the last observed value of the series,

$\hat{y}_{T+h|T} = y_T$

for $h = 1, 2, ...$ Hence, the naïve method assumes that the most recent observation is the only important one, and all previous observations provide no information for the future. This can be thought of as a weighted average where all of the weight is given to the last observation.

Using the average method, all future forecasts are equal to a simple average of the observed data,

$\hat{y}_{T+h|T} = \Sigma_{t=1}^{T} y_t$,

for $h = 1, 2, ...$ Hence, the average method assumes that all observations are of equal importance, and gives them equal weights when generating forecasts.

We often want something between these two extremes. For example, it may be sensible to attach larger weights to more recent observations than to observations from the distant past. This is exactly the concept behind simple exponential smoothing. Forecasts are calculated using weighted averages, where the weights decrease exponentially as observations come from further in the past — the smallest weights are associated with the oldest observations:

$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + ...$ (8.1)

where $0 \leq \alpha \leq 1$ is the smoothing parameter. The one-step-ahead forecast for time $T+1$ is a weighted average of all of the observations in the series $y_1, ..., y_T$. The rate at which the weights decrease is controlled by the parameter α.

For any α between 0 and 1, the weights attached to the observations decrease exponentially as we go back in time, hence the name "exponential smoothing". If
α is small (i.e., close to 0), more weight is given to observations from the more distant past. If α is large (i.e., close to 1), more weight is given to the more recent observations. For the extreme case where $\alpha = 1, \hat{y}_{T+1|T} = y_T$, and the forecasts are equal to the naïve forecasts.

We present two equivalent forms of simple exponential smoothing, each of which leads to the forecast Equation (8.1).

Weighted average form

The forecast at time $T+1$ is equal to a weighted average between the most recent observation y_T and the previous forecast $\hat{y}_{T|T-1}$:

$\hat{y}_{T+1|T} = \alpha y_T + (1-\alpha)\hat{y}_{T|T-1}$,

where $0 \leq \alpha \leq 1$ is the smoothing parameter. Similarly, we can write the fitted values as

$\hat{y}_{t+1|t} = \alpha y_t + (1-\alpha)\hat{y}_{t|t-1}$,

for $t = 1, ..., T$. (Recall that fitted values are simply one-step forecasts of the training data.)

The process has to start somewhere, so we let the first fitted value at time 1 be denoted by l_0 (which we will have to estimate). Then

$\hat{y}_{2|1} = \alpha y_1 + (1-\alpha)l_0$

$\hat{y}_{3|2} = \alpha y_2 + (1-\alpha)\hat{y}_{2|1}$

$\hat{y}_{4|3} = \alpha y_3 + (1-\alpha)\hat{y}_{3|2}$

till

$\hat{y}_{T+1|T} = \alpha y_T + (1-\alpha)\hat{y}_{T|T-1}$

Substituting each equation into the following equation, we obtain

$\hat{y}_{T+1|T} = \Sigma_{j=0}^{T-1}\alpha(1-\alpha)^j y_{T-j} + (1-\alpha)^T l_0$

The last term becomes tiny for large T . So, the weighted average form leads to the same forecast Equation (8.1).

Component form

An alternative representation is the component form. For simple exponential smoothing, the only component included is the level, l_t. (Other methods which are considered later in this chapter may also include a trend b_t and a seasonal component s_t.) Component form representations of exponential smoothing methods comprise a forecast equation and a smoothing equation for each of the components included in the method. The component form of simple exponential smoothing is given by:

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

- *Smoothing equation:* $l_t = \alpha y_t + (1-\alpha) l_{t-1}$,

where l_t is the level (or the smoothed value) of the series at time t. Setting $h = 1$ gives the fitted values, while setting $t = T$ gives the true forecasts beyond the training data.

The forecast equation shows that the forecast value at time t+1 is the estimated level at time t. The smoothing equation for the level (usually referred to as the level equation) gives the estimated level of the series at each period t. If we replace l_t with $\hat{y}_{t+1|t}$ and l_{t-1} with $\hat{y}_{t|t-1}$ in the smoothing equation, we will recover the weighted average form of simple exponential smoothing.

The component form of simple exponential smoothing is not particularly useful on its own, but it will be the easiest form to use when we start adding other components.

Flat forecasts

Simple exponential smoothing has a "flat" forecast function:

$\hat{y}_{T+h|T} = \hat{y}_{T+1|T} = l_T$, $h = 2, 3, ...$

That is, all forecasts take the same value, equal to the last level component. Remember that these forecasts will only be suitable if the time series has no trend or seasonal component.

Optimisation

The application of every exponential smoothing method requires the smoothing parameters and the initial values to be chosen. In particular, for simple exponential smoothing, we need to select the values of α and l_0. All forecasts can be computed from the data once we know those values. For the methods that follow there is usually more than one smoothing parameter and more than one initial component to be chosen.

In some cases, the smoothing parameters may be chosen in a subjective manner — the forecaster specifies the value of the smoothing parameters based on previous experience. However, a more reliable and objective way to obtain values for the unknown parameters is to estimate them from the observed data.

In Section 7.2, we estimated the coefficients of a regression model by minimising the sum of the squared residuals (usually known as SSE or "sum of squared errors"). Similarly, the unknown parameters and the initial values for any exponential smoothing method can be estimated by minimising the SSE. The residuals are specified as
$e_t = y_t - \hat{y}_{t|t-1}$ for $t = 1, ..., T$. Hence, we find the values of the unknown parameters and the initial values that minimise SSE.

Unlike the regression case (where we have formulas which return the values of the regression coefficients that minimise the SSE), this involves a non-linear minimisation problem, and we need to use an optimisation tool to solve it.

Example: Algerian exports

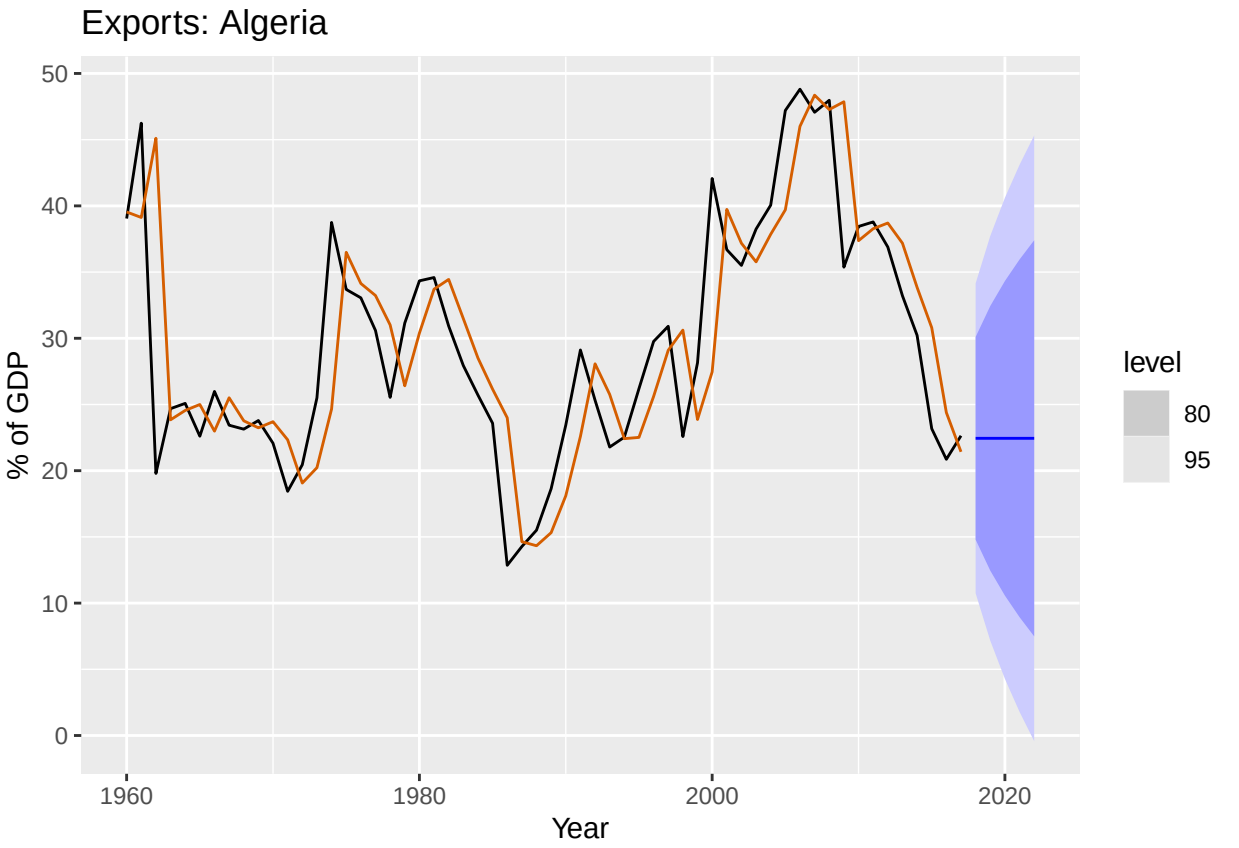
In this example, simple exponential smoothing is applied to forecast exports of goods and services from Algeria.

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

This gives parameter estimates $hat\alpha = 0.84$ and $\hat{l}_0 = 39.5$, obtained by minimising SSE over periods $t = 1, 2, ..., 58$, subject to the restriction that $0 \leq \alpha \leq 1$.

The black line in Figure 8.2 shows the data, which has a changing level over time.

```
fc |>
  autoplot(algeria_economy) +
  geom_line(aes(y = .fitted), col="#D55E00",
            data = augment(fit)) +
  labs(y="% of GDP", title="Exports: Algeria") +
  guides(colour = "none")
```



The forecasts for the period 2018–2022 are plotted in Figure 8.2. Also plotted are one-step-ahead fitted values alongside the data over the period 1960–2017. The large value of in this example is reflected in the large adjustment that takes place in the estimated level l_t at each time. A smaller value of would lead to smaller changes over time, and so the series of fitted values would be smoother.

The prediction intervals shown here are calculated using the methods described in Section 8.7. The prediction intervals show that there is considerable uncertainty in the future exports over the five-year forecast period. So interpreting the point forecasts without accounting for the large uncertainty can be very misleading.

8.2 Methods with trend

After simple exponential smoothing had been invented, people started developing alternatives that added various features. Most important of those was the addition of trend which was produced by Charles Holt in the 1950s.

His method can be written in a similar way to simple exponential smoothing with one extra equation.

Holt's linear trend

Component form

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

It says forecast for "h" steps ahead is equal to the last observed level l_t and b_t is the last observed slope estimate.

Level $l_t = \alpha y_t + (1-\alpha)(l_{t-1} + b_{t-1})$

In the level component, we have added b_{t-1} extra than the simple exponential smoothing.

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

Here, both two equations for level and trend, we are doing the weighted average and they are in convex combinations.

- Here, we have two smoothing parameters α and β^* ($0 , ^* 1 $) means both should simultaneously between 0 and 1.
- Here l_t is the level and β_t is the slope.
- l_t level: weighted average between y_t and one-step ahead forecast for time t, $(l_{t-1} + b_{t-1} = \hat{y}_{t|t-1})$
- b_t slope: weighted average of $l_t - l_{t-1}$ and b_{t-1}, current and previous estimate of slope.
- Choose $\alpha, \beta^*, l_0, b_0$ to minimise SSE.

When we are estimating this particular model, we need values of $\alpha, \beta^*, l_0, b_0$ where l_0 is initial level and b_0 is initial slope, so that we can start the iterative equations there.

Example: Australian population

```
aus_economy <- global_economy |>
  filter(Code == "AUS") |>
  mutate(Pop=Population/1e6)
fit <- aus_economy |>
  model(AAN = ETS(Pop ~ error("A") + trend("A") + season("N")))
report(fit)
```

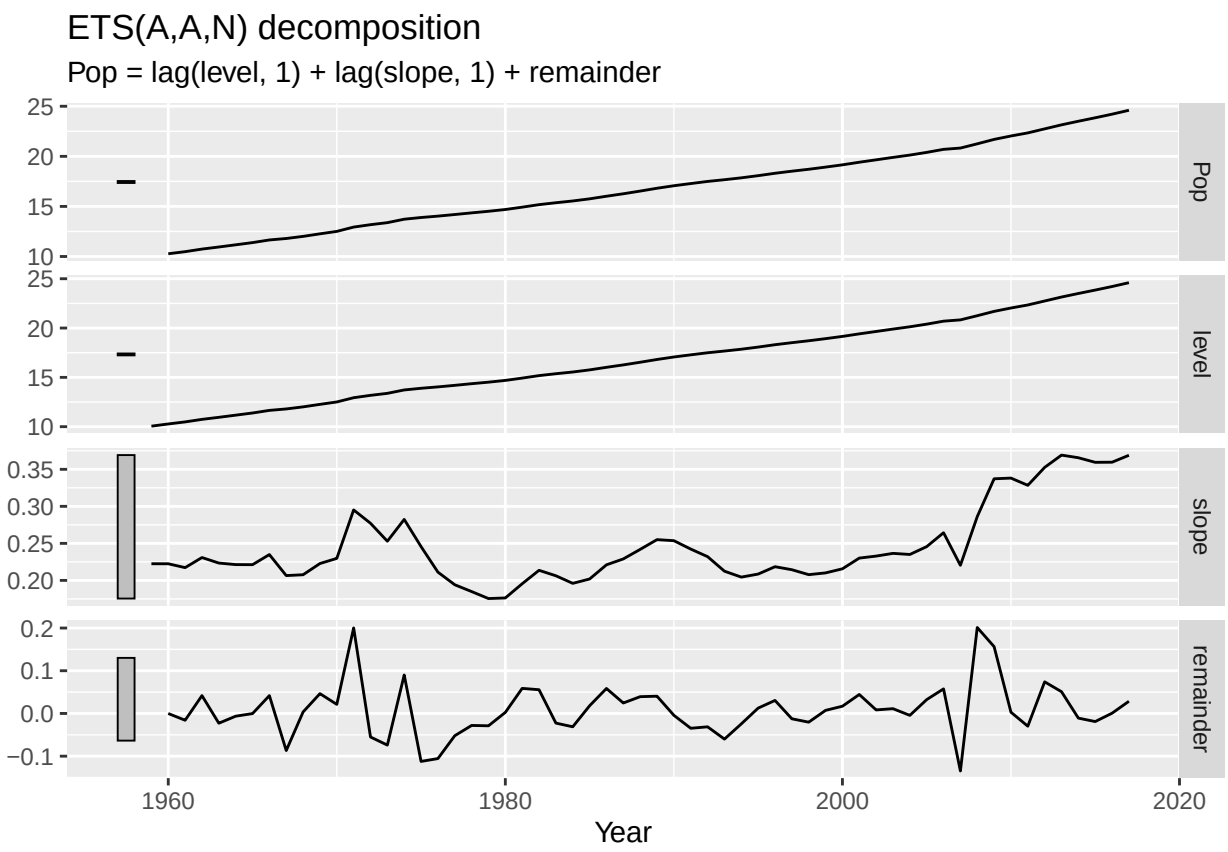
```
## Series: Pop
## Model: ETS(A,A,N)
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 0.3266366
##
##   Initial states:
##      l[0]      b[0]
##  10.05414 0.2224818
##
##   sigma^2:  0.0041
##
##       AIC      AICc       BIC
## -76.98569 -75.83184 -66.68347
```

The code for fitting this model is very similar to the code for fitting simple exponential smoothing, we just have to add trend as additive.

Here, first "A" is for additive error, second "A" for the trend and "N" is for seasonality. At this stage, we have not introduced the seasonality.

```
components(fit) |> autoplot()
```

```
## Warning: Removed 1 row containing missing values (`geom_line()`).
```



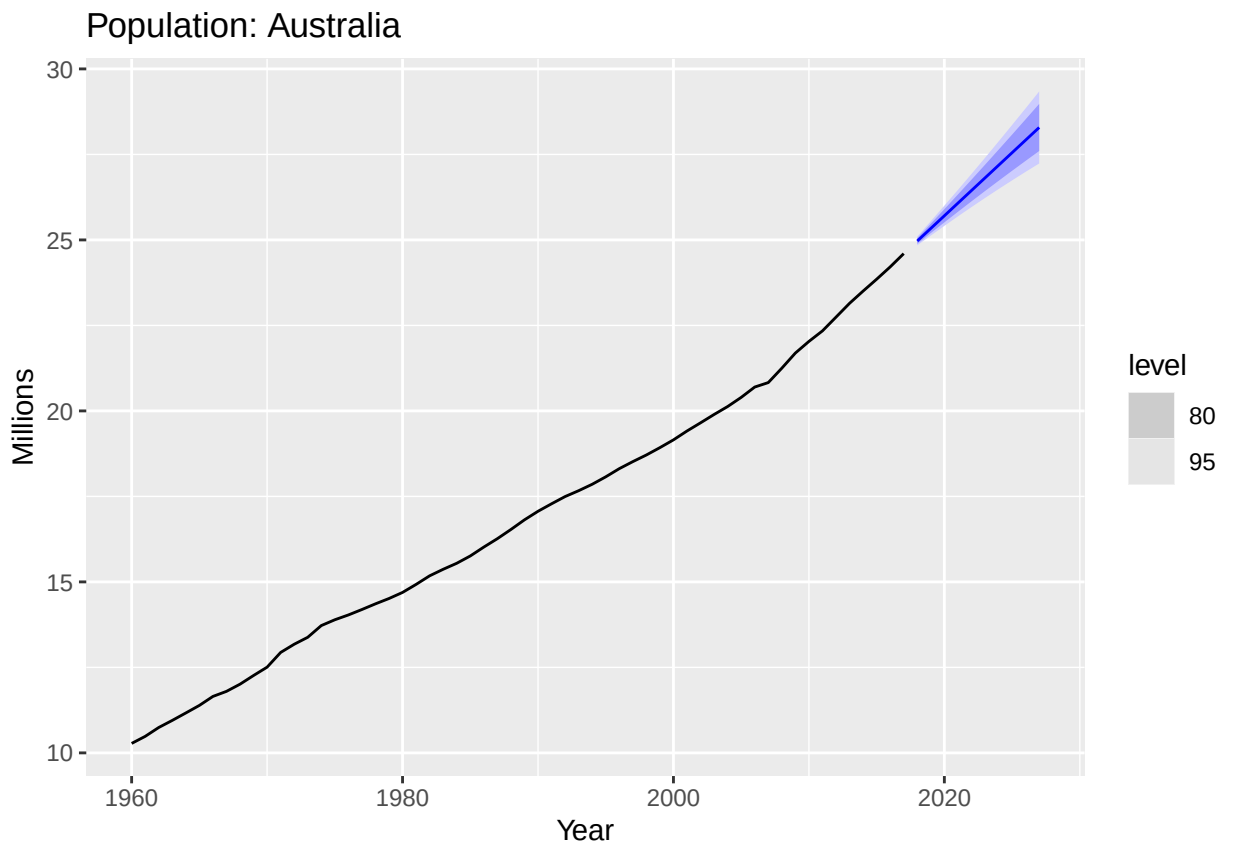
Here, level is very close to the data. so, slightly smoother. Slope is giving the estimate for the trend in the data. Slope is 0.2 at most of the time but reach at 0.35 when slope increases at the end and remainder part contains the noise in the data.

```
components(fit)|>
  left_join(fitted(fit),by = c("Country",".model","Year"))
```

```
## # A dable: 59 x 8 [1Y]
## # Key:     Country, .model [1]
## # :        Pop = lag(level, 1) + lag(slope, 1) + remainder
##    Country   .model  Year   Pop level slope remainder .fitted
##    <fct>     <chr>  <dbl> <dbl> <dbl> <dbl>     <dbl>   <dbl>
##  1 Australia AAN     1959  NA    10.1 0.222 NA           NA
##  2 Australia AAN     1960  10.3  10.3 0.222 -0.000145    10.3
##  3 Australia AAN     1961  10.5  10.5 0.217 -0.0159      10.5
##  4 Australia AAN     1962  10.7  10.7 0.231  0.0418      10.7
##  5 Australia AAN     1963  11.0  11.0 0.223 -0.0229      11.0
##  6 Australia AAN     1964  11.2  11.2 0.221 -0.00641     11.2
##  7 Australia AAN     1965  11.4  11.4 0.221 -0.000314    11.4
##  8 Australia AAN     1966  11.7  11.7 0.235  0.0418      11.6
##  9 Australia AAN     1967  11.8  11.8 0.206 -0.0869      11.9
## 10 Australia AAN     1968  12.0  12.0 0.208  0.00350     12.0
## # i 49 more rows
```

Here, the data is in "Pop" column. Here, $l_0 = 10.1$ and $b_0 = 0.222$

```
fit |>
  forecast(h=10) |>
  autoplot(aus_economy) +
  labs(y="Millions", title = "Population: Australia")
```



A little letter in time, another method is developed which is called "Damped Trend Method".

Damped Trend Method

In this method, the trend into the forecast period was not linear but actually flattening out. So, instead of going up at the same slope forever from the previous graph, in the damped trend version, it starts off at that slide but then it flattens out.

Component Form

$\hat{y}_{t+h|t} = l_t + (\phi + \phi^2 + ... + \phi^h)b_t$

Where, $l_t = \alpha y_t + (1-\alpha)(l_{t-1} + \phi b_{t-1})$

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

as h gets bigger then $\hat{y}_{t+h|t}$ gets constant that's why it flattens out to a horizontal line.

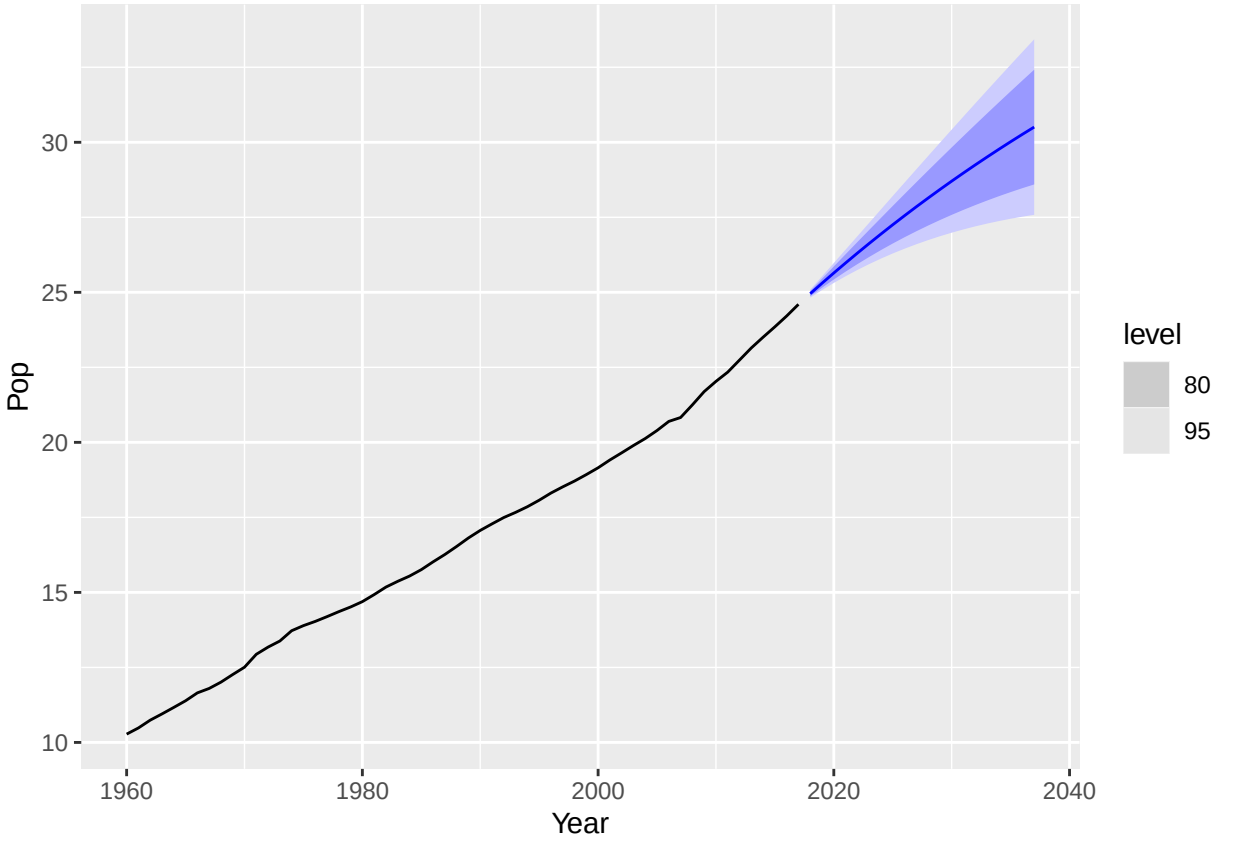
- Damping parameter $0 < \phi < 1$. It is usually constrained atleast 0.8 if you want a damped trend model otherwise it dampens too quickly and have numerical instabilities.
- If $\phi = 1$, it's identical to Holt's linear trend.

- As $h \to \infty$, $\hat{y}_{T+h|T} \to l_T + \frac{\phi}{1-\phi} b_T$
- Short-run forecasts trended, long-run forecasts constant. It can be quite useful in practice because often trends trends are not going to continue indefinitely.

Example:

Little modification in the previous code.

```
aus_economy |>
  model(holt=ETS(Pop ~ error("A") + trend("Ad")+ season("N"))) |>
  forecast(h=20) |>
  autoplot(aus_economy)
```



Here, "Ad" stands for the "Additive damped" means it is additive trend but we are damping it and we have taken a high forecast horizon i.e. $h = 20$.

Now, look at the three model which we have talked about in this chapter.

```
fit <- aus_economy |>
  filter(Year <= 2010) |>
  model(
    ses=ETS(Pop ~ error("A")+ trend("N")+season("N")),
    holt=ETS(Pop ~ error("A")+ trend("A")+season("N")),
    damped=ETS(Pop ~ error("A")+ trend("Ad")+season("N"))
  )
tidy(fit)
```

```
## # A tibble: 11 x 4
```

```
##    Country   .model term  estimate
##    <fct>     <chr>  <chr>    <dbl>
##  1 Australia ses    alpha    1.00
##  2 Australia ses    l[0]    10.3
##  3 Australia holt   alpha    1.00
##  4 Australia holt   beta     0.296
##  5 Australia holt   l[0]    10.1
##  6 Australia holt   b[0]     0.224
##  7 Australia damped alpha    1.00
##  8 Australia damped beta     0.402
##  9 Australia damped phi      0.980
## 10 Australia damped l[0]    10.0
## 11 Australia damped b[0]     0.246
```

```
accuracy(fit)
```

```
## # A tibble: 3 x 11
##   Country   .model .type       ME   RMSE    MAE    MPE  MAPE  MASE RMSSE    ACF1
##   <fct>     <chr>  <chr>    <dbl>  <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 Australia ses    Train~ 0.231   0.242  0.231  1.48   1.48  0.980 0.990  0.267
## 2 Australia holt   Train~ 0.00716 0.0646 0.0451 0.0336 0.289 0.192 0.264  0.0504
## 3 Australia damped Train~ 0.0157  0.0665 0.0466 0.0883 0.300 0.198 0.272 -0.0334
```

Here, "tidy()" function is used to get various coefficients and "accuracy()" to get various accuracy measures."ses" means simple exponential smoothing. Here, ses errors has much worse than other two models. Linear trend is doing the best here and damping didn't help at all.

Summary

Holt's linear trend method

Holt (1957) extended simple exponential smoothing to allow the forecasting of data with a trend. This method involves a forecast equation and two smoothing equations (one for the level and one for the trend):

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

Level equation $\ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1})$

Trend equation : $b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}$,

where ℓ_t denotes an estimate of the level of the series at time t, b_t denotes an estimate of the trend (slope) of the series at time t, is the smoothing parameter for the level, $0 \leq \alpha \leq 1$, and β^* is the smoothing parameter for the trend, $0 \leq \beta^* \leq 1$. (We denote this as β^* instead of for reasons that will be explained in Section 8.5.)

As with simple exponential smoothing, the level equation here shows that ℓ_t is a weighted average of observation y_t and the one-step-ahead training forecast for time t ,here given by $\ell_{t-1} + b_{t-1}$. The trend equation shows that b_t is a weighted average of the estimated trend at time t based on $\ell_t - \ell_{t-1}$ and b_{t-1} ,the previous estimate of the trend.

The forecast function is no longer flat but trending. The h -step-ahead forecast is equal to the last estimated level plus h times the last estimated trend value. Hence the forecasts are a linear function of h.

Example

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

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

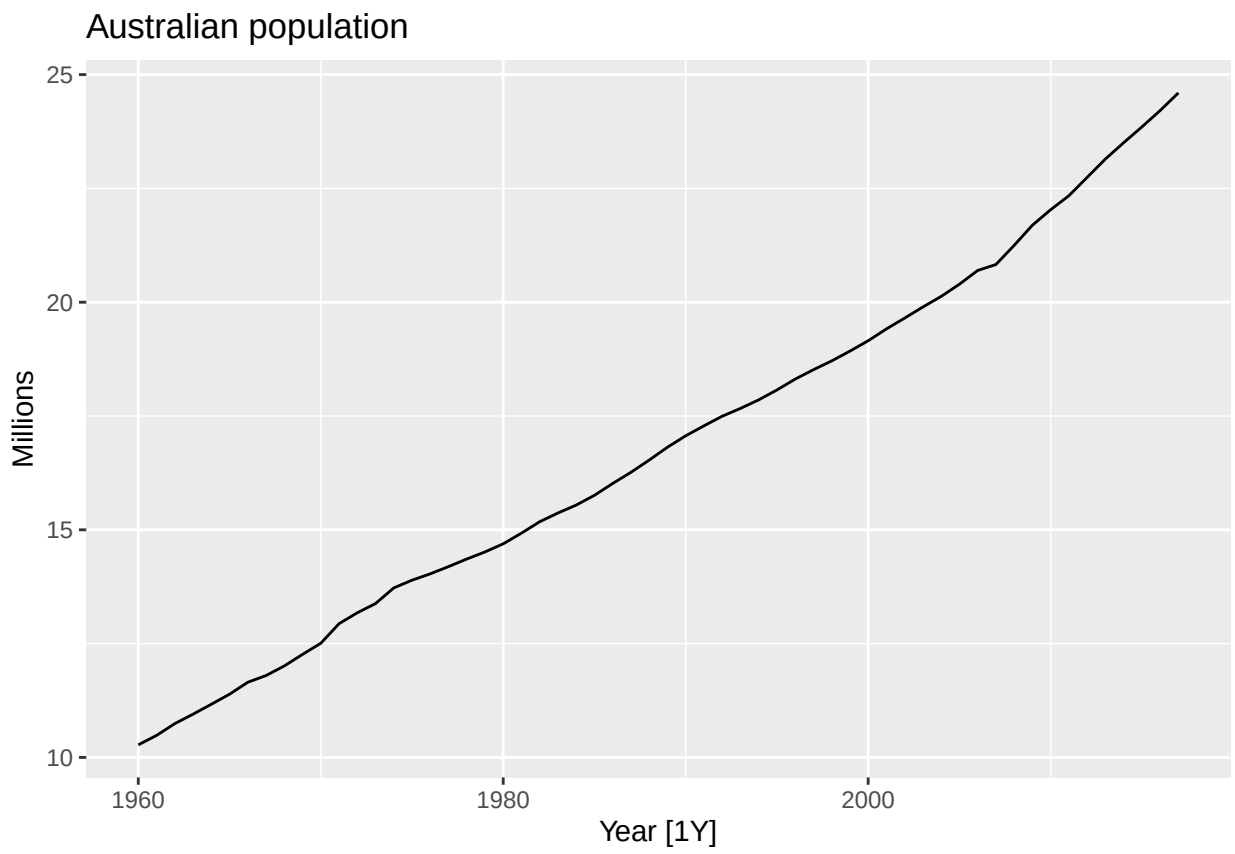


Figure 8.3 shows Australia's annual population from 1960 to 2017. We will apply Holt's method to this series. The smoothing parameters, and β^*, and the initial values ℓ_0 and b_0 are estimated by minimising the SSE for the one-step training errors as in Section 8.1.

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

The estimated smoothing coefficient for the level is $\hat{\alpha} = 0.9999$. The very high value shows that the level changes rapidly in order to capture the highly trended series. The estimated smoothing coefficient for the slope is $\hat{\beta}^*$=0.3267. This is relatively large suggesting that the trend also changes often (even if the changes are slight).

In Table 8.2 we use these values to demonstrate the application of Holt's method.

Damped trend methods

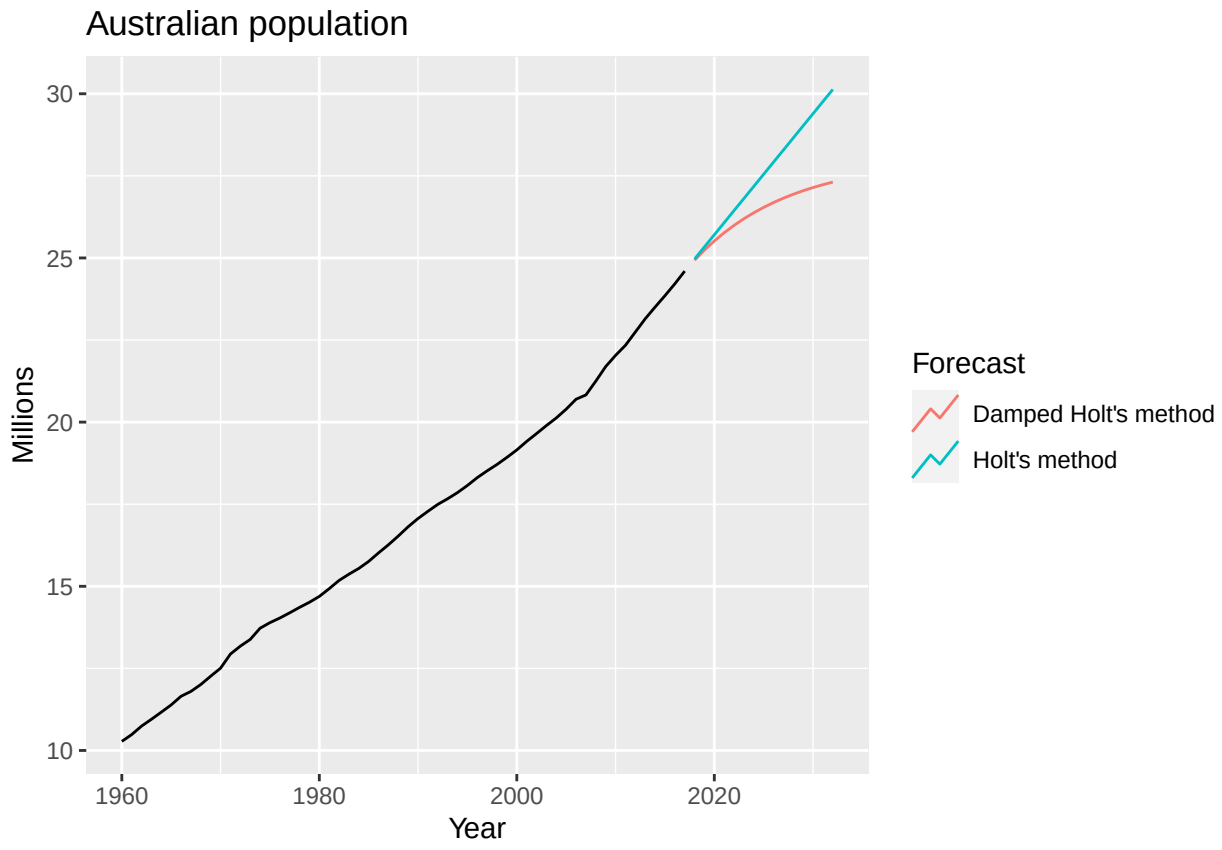
The forecasts generated by Holt's linear method display a constant trend (increasing or decreasing) indefinitely into the future. Empirical evidence indicates that these methods tend to over-forecast, especially for longer forecast horizons. Motivated by this observation, Gardner & McKenzie (1985) introduced a parameter that "dampens" the trend to a flat line some time in the future. Methods that include a damped trend have proven to be very successful, and are arguably the most popular individual methods when forecasts are

required automatically for many series.

In practice, is rarely less than 0.8 as the damping has a very strong effect for smaller values. Values of close to 1 will mean that a damped model is not able to be distinguished from a non-damped model. For these reasons, we usually restrict to a minimum of 0.8 and a maximum of 0.98.

Example

```
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"))
  ) |>
  forecast(h = 15) |>
  autoplot(aus_economy, level = NULL) +
  labs(title = "Australian population",
       y = "Millions") +
  guides(colour = guide_legend(title = "Forecast"))
```

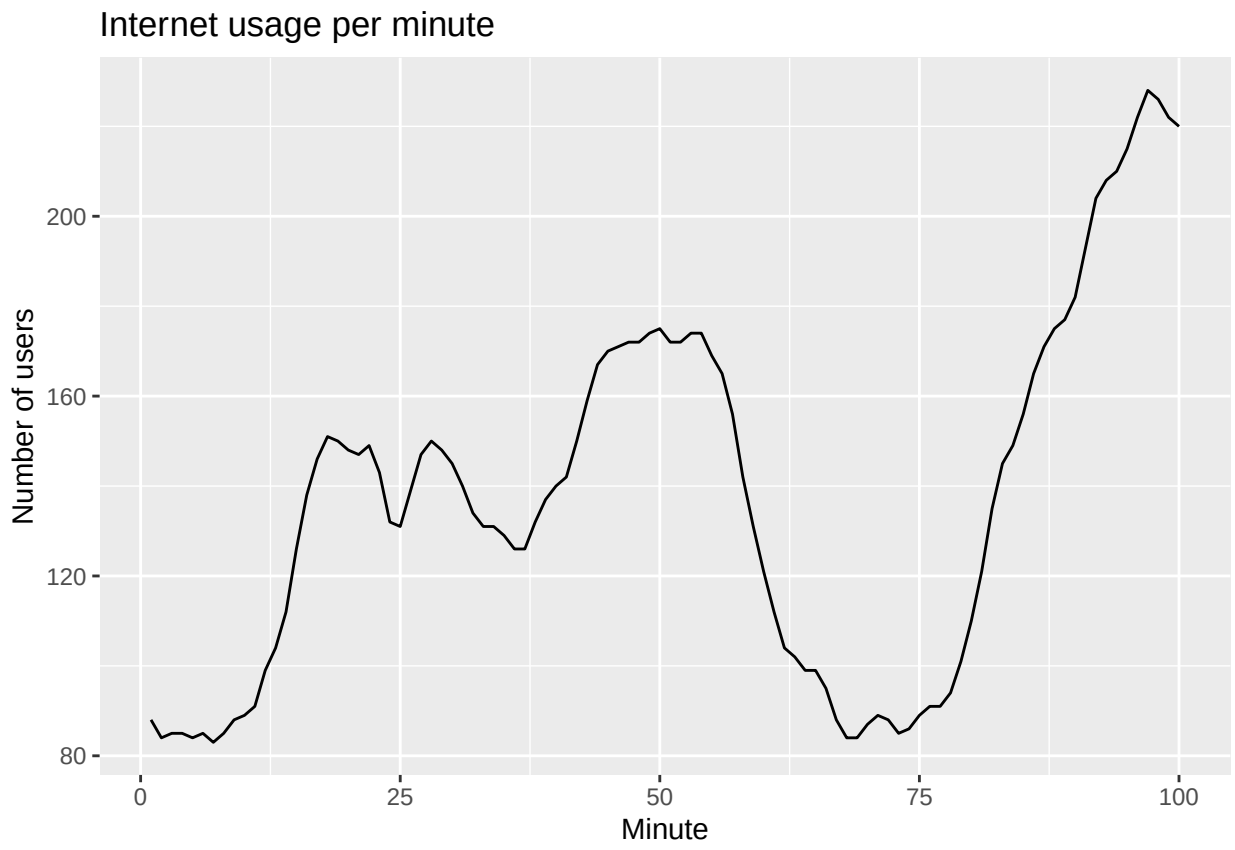


We have set the damping parameter to a relatively low number (=0.90) to exaggerate the effect of damping for comparison. Usually, we would estimate along with the other parameters. We have also used a rather large forecast horizon (h=15) to highlight the difference between a damped trend and a linear trend.

Example: Internet usage

In this example, we compare the forecasting performance of the three exponential smoothing methods that we have considered so far in forecasting the number of users connected to the internet via a server. The data is observed over 100 minutes and is shown in Figure 8.5.

```
www_usage <- as_tsibble(WWWusage)
www_usage |> autoplot(value) +
  labs(x="Minute", y="Number of users",
       title = "Internet usage per minute")
```



We will use time series cross-validation to compare the one-step forecast accuracy of the three methods.

```
www_usage |>
  stretch_tsibble(.init = 10) |>
  model(
    SES = ETS(value ~ error("A") + trend("N") + season("N")),
    Holt = ETS(value ~ error("A") + trend("A") + season("N")),
    Damped = ETS(value ~ error("A") + trend("Ad") +
                   season("N"))
  ) |>
  forecast(h = 1) |>
  accuracy(www_usage)
```

```
## Warning: The future dataset is incomplete, incomplete out-of-sample data will be treated as missing.
## 1 observation is missing at 101
```

```
## # A tibble: 3 x 10
##   .model .type     ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
```

```
##   <chr>  <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Damped Test  0.288   3.69  3.00 0.347  2.26 0.663 0.636 0.336
## 2 Holt   Test  0.0610  3.87  3.17 0.244  2.38 0.701 0.668 0.296
## 3 SES    Test  1.46    6.05  4.81 0.904  3.55 1.06  1.04  0.803
```

```
#> # A tibble: 3 × 10
#>   .model .type     ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
#>   <chr>  <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1 Damped Test  0.288   3.69  3.00 0.347  2.26 0.663 0.636 0.336
#> 2 Holt   Test  0.0610  3.87  3.17 0.244  2.38 0.701 0.668 0.296
#> 3 SES    Test  1.46    6.05  4.81 0.904  3.55 1.06  1.04  0.803
```

Damped Holt's method is best whether you compare MAE or RMSE values. So we will proceed with using the damped Holt's method and apply it to the whole data set to get forecasts for future minutes.

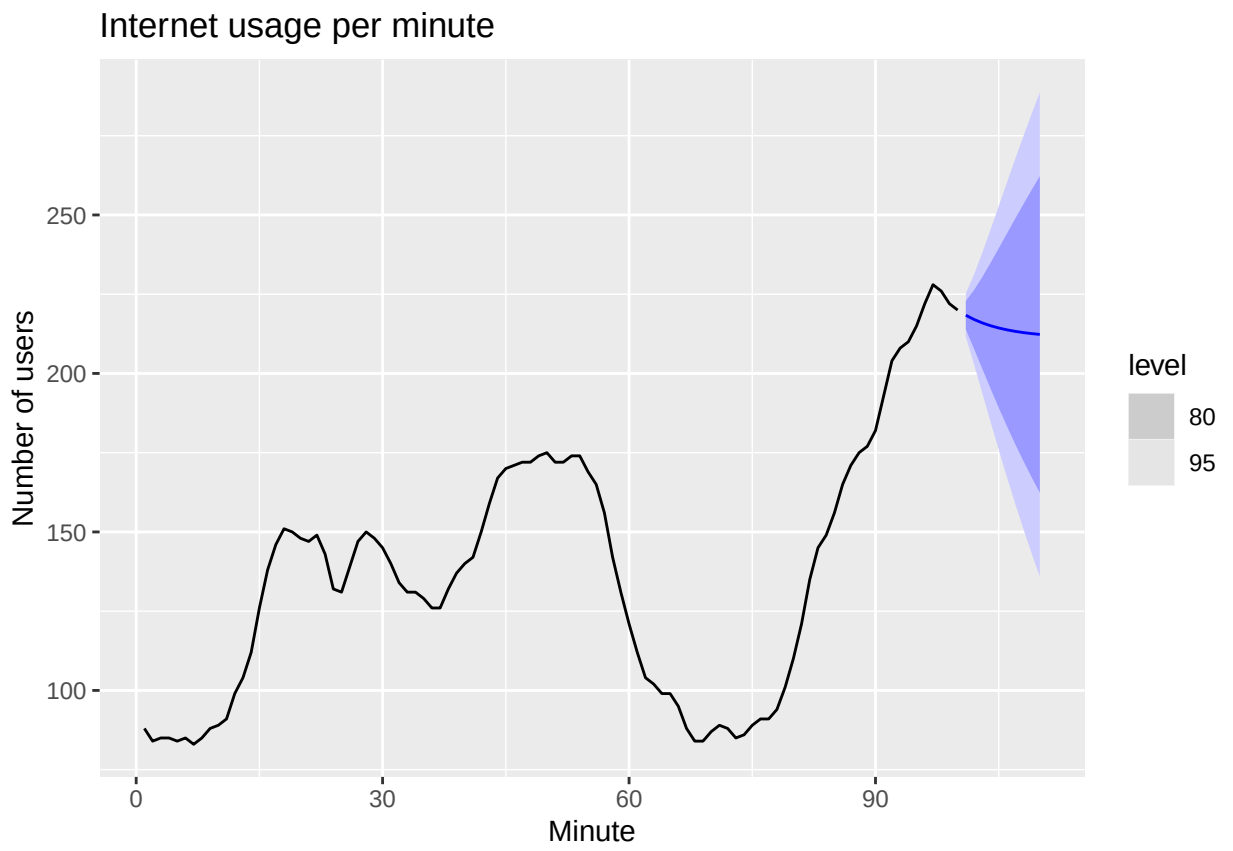
```
fit <- www_usage |>
  model(
    Damped = ETS(value ~ error("A") + trend("Ad") +
                   season("N"))
  )
# Estimated parameters:
tidy(fit)
```

```
## # A tibble: 5 x 3
##   .model term  estimate
##   <chr>  <chr>    <dbl>
## 1 Damped alpha   1.00
## 2 Damped beta    0.997
## 3 Damped phi     0.815
## 4 Damped l[0]   90.4
## 5 Damped b[0]   -0.0173
```

```
#> # A tibble: 5 × 3
#>   .model term  estimate
#>   <chr>  <chr>    <dbl>
#> 1 Damped alpha   1.00
#> 2 Damped beta    0.997
#> 3 Damped phi     0.815
#> 4 Damped l[0]   90.4
#> 5 Damped b[0]   -0.0173
```

The smoothing parameter for the slope is estimated to be almost one, indicating that the trend changes to mostly reflect the slope between the last two minutes of internet usage. The value of is very close to one, showing that the level reacts strongly to each new observation.

```
fit |>
  forecast(h = 10) |>
  autoplot(www_usage) +
  labs(x="Minute", y="Number of users",
       title = "Internet usage per minute")
```



The resulting forecasts look sensible with decreasing trend, which flattens out due to the low value of the damping parameter (0.815), and relatively wide prediction intervals reflecting the variation in the historical data. The prediction intervals are calculated using the methods described in Section 8.7.

In this example, the process of selecting a method was relatively easy as both MSE and MAE comparisons suggested the same method (damped Holt's). However, sometimes different accuracy measures will suggest different forecasting methods, and then a decision is required as to which forecasting method we prefer to use. As forecasting tasks can vary by many dimensions (length of forecast horizon, size of test set, forecast error measures, frequency of data, etc.), it is unlikely that one method will be better than all others for all forecasting scenarios. What we require from a forecasting method are consistently sensible forecasts, and these should be frequently evaluated against the task at hand.

8.3 Methods with seasonality

Charles Holt and his student Peter Winter extended Hott's linear trend method which is also account for seasonality.

Holt-Winter Additive Model

Component Form

$\hat{y}_{t+h|t} = l_t + hb_t + s_{t+h-m(k+1)}$

$l_t = \alpha(y_t - S_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1})$

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

$s_t = \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}$

- k = integer part of $\frac{h-1}{m}$. Ensures estimates from the final year are used for the forecasting.
- Parameter: $0 \leq \alpha \leq 1, 0 \leq \beta^* \leq 1, 0 \leq \gamma \leq 1-\alpha$ and m = period of seasonality (m=4 for for quarterly data).
- Seasonal component is usually expressed as

$s_t = \gamma^*(y_t - l_t) + (1-\gamma^*)s_{t-m}$.

- Substitute in for l_t :

$s_t = \gamma^*(1-\alpha)(y_t - l_{t-1} - b_{t-1}) + (1-\gamma^*(1-\alpha))s_{t-m}$

- We set $\gamma = \gamma^*(1-\alpha)$
- The usual parameter restriction is $0 \leq \gamma^* \leq 1$ which translates to $0 \leq \gamma \leq (1-\alpha)$
- When $\gamma = 0$ then no change in the seasonal component and when $\gamma = 1$ then it will be seasonal naive. Usually γ is very low i.e. $\gamma \leq 0.2$ and so seasonal component doesn't change immensely.

Now, an alternative to the Holt-Winter's Additive method is where level and trend interacting with a seasonal component in a multiplicative way.

Holt-Winter's Multiplicative Method

Seasonal variations change in proportion to the level of series.

Component Form

$\hat{y}_{t+h|t} = (l_t + hb_t)s_{t+h-m(k+1)}$

$l_t = \alpha\frac{y_t}{s_{t-m}} + (1-\alpha)(l_{t-1} + b_{t-1})$

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

$s_t = \gamma\frac{y_t}{(l_{t-1}+b_{t-1})} + (1-\gamma)s_{t-m}$

- k is an integer part of $\frac{h-1}{m}$
- Additive method: s_t in absolute terms – within each year $\Sigma_i s_i \approx 0$.
- Multiplicative method: s_t in relative terms – within each year $\Sigma_i s_i \approx m$.

Example

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

The estimated coefficients are

```
tidy(fit) |>
  spread(.model,estimate)
```

```
## # A tibble: 9 x 3
##   term     additive multiplicative
```

```
##   <chr>       <dbl>          <dbl>
## 1 alpha    0.236          0.186
## 2 b[0]   -37.4          -33.4
## 3 beta     0.0298         0.0248
## 4 gamma    0.000100       0.000100
## 5 l[0]  9899.          9853.
## 6 s[-1] -684.             0.926
## 7 s[-2] -290.             0.970
## 8 s[-3] 1512.             1.16
## 9 s[0]  -538.             0.943
```

Here, first column is Additive hot winters and second column is for multiplicative hot winters.

```
tidy(fit) |>
  spread(.model,estimate)
```

```
## # A tibble: 9 x 3
##   term     additive multiplicative
##   <chr>       <dbl>          <dbl>
## 1 alpha    0.236          0.186
## 2 b[0]   -37.4          -33.4
## 3 beta     0.0298         0.0248
## 4 gamma    0.000100       0.000100
## 5 l[0]  9899.          9853.
## 6 s[-1] -684.             0.926
## 7 s[-2] -290.             0.970
## 8 s[-3] 1512.             1.16
## 9 s[0]  -538.             0.943
```

Here, for additive model, s[0],... are absolute seasonal terms that sum to 0 and for multiplicative model, s[0],... are relative seasonal terms that sum to 1.

```
components(fit) |> filter(.model == 'additive') |>
  left_join(fitted(fit), by = c(".model","Quarter"))
```

```
## # A dable: 84 x 8 [1Q]
## # Key:     .model [1]
## # :        Trips = lag(level, 1) + lag(slope, 1) + lag(season, 4) + remainder
##    .model   Quarter  Trips level slope season remainder .fitted
##    <chr>      <qtr>  <dbl> <dbl> <dbl>  <dbl>     <dbl>   <dbl>
##  1 additive 1997 Q1    NA    NA   NA    1512.      NA       NA
##  2 additive 1997 Q2    NA    NA   NA    -290.      NA       NA
##  3 additive 1997 Q3    NA    NA   NA    -684.      NA       NA
##  4 additive 1997 Q4    NA  9899. -37.4  -538.      NA       NA
##  5 additive 1998 Q1 11806. 9964. -24.5  1512.     433.   11373.
##  6 additive 1998 Q2  9276. 9851. -35.6  -290.    -374.    9649.
##  7 additive 1998 Q3  8642. 9700. -50.2  -684.    -489.    9131.
##  8 additive 1998 Q4  9300. 9694. -44.6  -538.     188.    9111.
##  9 additive 1999 Q1 11172. 9652. -44.3  1512.      10.7  11161.
## 10 additive 1999 Q2  9608. 9676. -35.6  -290.     290.    9318.
## # i 74 more rows
```

Now for this quarterly data, first fitted value $11373 = l_t + b_t + s_{t-3} = 9899 + (-37.4) + (1512)$

```
components(fit) |> filter(.model == 'multiplicative') |>
  left_join(fitted(fit), by = c(".model","Quarter"))
```

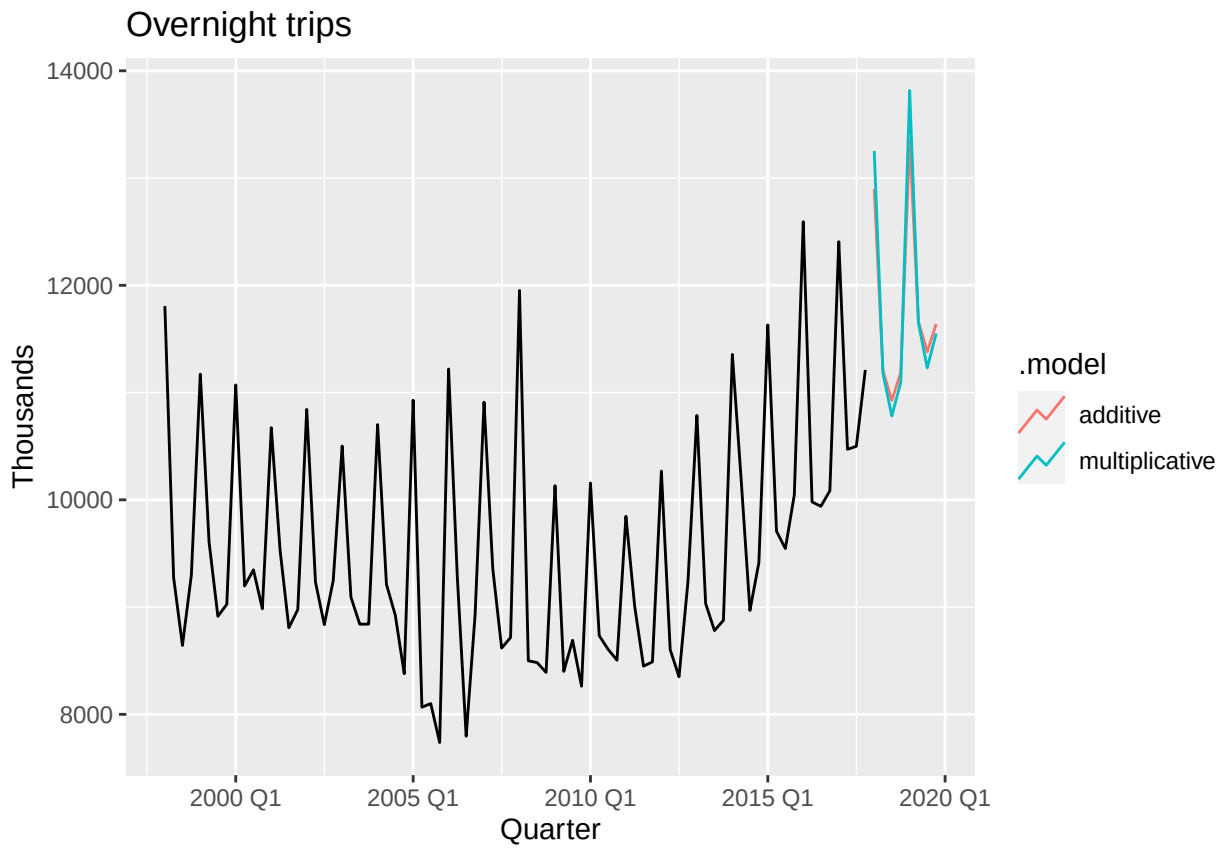
```
## # A dable: 84 x 8 [1Q]
```

```
## # Key:     .model [1]
## # :        Trips = lag(level, 1) + lag(slope, 1) + lag(season, 4) + remainder
##    .model         Quarter  Trips level slope season remainder .fitted
##    <chr>            <qtr>  <dbl> <dbl> <dbl>  <dbl>     <dbl>   <dbl>
##  1 multiplicative 1997 Q1    NA    NA   NA    1.16   NA           NA
##  2 multiplicative 1997 Q2    NA    NA   NA    0.970  NA           NA
##  3 multiplicative 1997 Q3    NA    NA   NA    0.926  NA           NA
##  4 multiplicative 1997 Q4    NA  9853. -33.4  0.943  NA           NA
##  5 multiplicative 1998 Q1 11806. 9883. -24.9  1.16    0.0348   11409.
##  6 multiplicative 1998 Q2  9276. 9803. -32.3  0.970  -0.0299    9562.
##  7 multiplicative 1998 Q3  8642. 9690. -43.0  0.926  -0.0444    9044.
##  8 multiplicative 1998 Q4  9300. 9688. -37.6  0.943   0.0227    9093.
##  9 multiplicative 1999 Q1 11172. 9644. -38.4  1.16   -0.00362  11213.
## 10 multiplicative 1999 Q2  9608. 9661. -31.0  0.970   0.0312    9317.
## # i 74 more rows
```

Here first fitted value $11409 = (l_t + b_t)s_{t-3} = (9853 + (-33.4)) \times 1.16$

Now, let's see some forecasts.

```
fc |>
  autoplot(aus_holidays, level = NULL)+
  labs(y="Thousands", title="Overnight trips")
```



Here, multiplicative method does a better job in sort of expanding the seasonality as it projects forward.

Now, the following tool that probably every forecaster should have in tool bag if they are forecasting seasonal data and they are using exponential smoothing.

This is a combination of having multiplicative hot winters seasonal method with a damped trend. Often the single most accurate forecasting method for seasonal data is following and it works extremely extremely well.

$\hat{y}_{t+h|t} = [l_t + (\phi + \phi^2 + ... + \phi^h)b_t]s_{t+h-m(k+1)}$

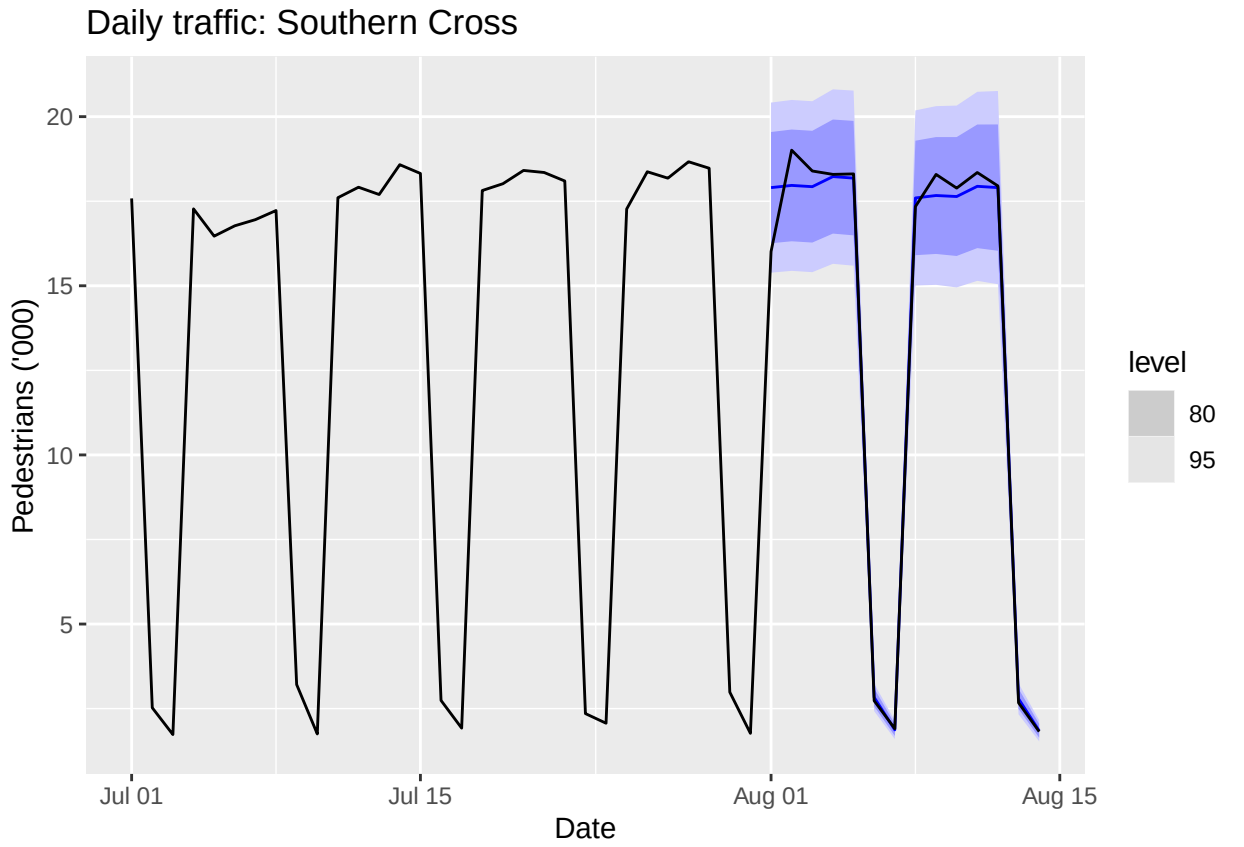
$l_t = \alpha(\frac{y_t}{s_{t-m}}) + (1-\alpha)(l_{t-1} + \phi b_{t-1})$

$b_t = \beta^*(l_t - l_{t-1}) + (1-\beta^*)\phi\beta_{t-1}$

$s_t = \gamma\frac{y_t}{l_{t-1}+\phi b_{t-1}} + (1-\gamma)s_{t-m}$

Example: Holt Winter's with daily data

```
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"))) |>
  forecast(h="2 weeks") |>
  autoplot(sth_cross_ped |> filter(Date <= "2016-08-14"))+
  labs(title = "Daily traffic: Southern Cross",y="Pedestrians ('000)")
```



Here blue lines are forecasts. For weekdays, it is having wider prediction interval and for weekends, it is

having narrower prediction interval.

Summary

Holt (1957) and Winters (1960) extended Holt's method to capture seasonality. The Holt-Winters seasonal method comprises the forecast equation and three smoothing equations — one for the level ℓ_t, one for the trend b_t, and one for the seasonal component s_t, with corresponding smoothing parameters , β^* and . We use m to denote the period of the seasonality, i.e., the number of seasons in a year. For example, for quarterly data m=4, and for monthly data m=12.

There are two variations to this method that differ in the nature of the seasonal component. The additive method is preferred when the seasonal variations are roughly constant through the series, while the multiplicative method is preferred when the seasonal variations are changing proportional to the level of the series. With the additive method, the seasonal component is expressed in absolute terms in the scale of the observed series, and in the level equation the series is seasonally adjusted by subtracting the seasonal component. Within each year, the seasonal component will add up to approximately zero. With the multiplicative method, the seasonal component is expressed in relative terms (percentages), and the series is seasonally adjusted by dividing through by the seasonal component. Within each year, the seasonal component will sum up to approximately m.

Holt-Winters' additive method

The component form for the additive method is:

$$\hat{y}_{t+h|t} = \ell_t + hb_t + s_{t+h-m(k+1)}$$

$$\ell_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1}) b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}$$

$$s_t = \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m},$$

where k is the integer part of $(h-1)/m$, which ensures that the estimates of the seasonal indices used for forecasting come from the final year of the sample. The level equation shows a weighted average between the seasonally adjusted observation $(y_t - s_{t-m})$ and the non-seasonal forecast $(\ell_{t-1} + b_{t-1})$ for time t. The trend equation is identical to Holt's linear method. The seasonal equation shows a weighted average between the current seasonal index, $(y_t - \ell_{t-1} - b_{t-1})$, and the seasonal index of the same season last year (i.e., m time periods ago).

The equation for the seasonal component is often expressed as $s_t = \gamma^*(y_t - \ell_t) + (1-\gamma^*)s_{t-m}$.

If we substitute ℓ_t from the smoothing equation for the level of the component form above, we get $s_t = \gamma^*(1-\alpha)(y_t - \ell_{t-1} - b_{t-1}) + [1-\gamma^*(1-\alpha)]s_{t-m}$, which is identical to the smoothing equation for the seasonal component we specify here, with $\gamma = \gamma^*(1-\alpha)$. The usual parameter restriction is $0 \leq \gamma^* \leq 1$, which translates to $0 \leq \gamma \leq 1-\alpha$.

Holt-Winters' multiplicative method

The component form for the multiplicative method is:

$$\hat{y}_{t+h|t} = (\ell_t + hb_t)s_{t+h-m(k+1)}$$

$$\ell_t = \alpha(y_t s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1})$$

$$b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}$$

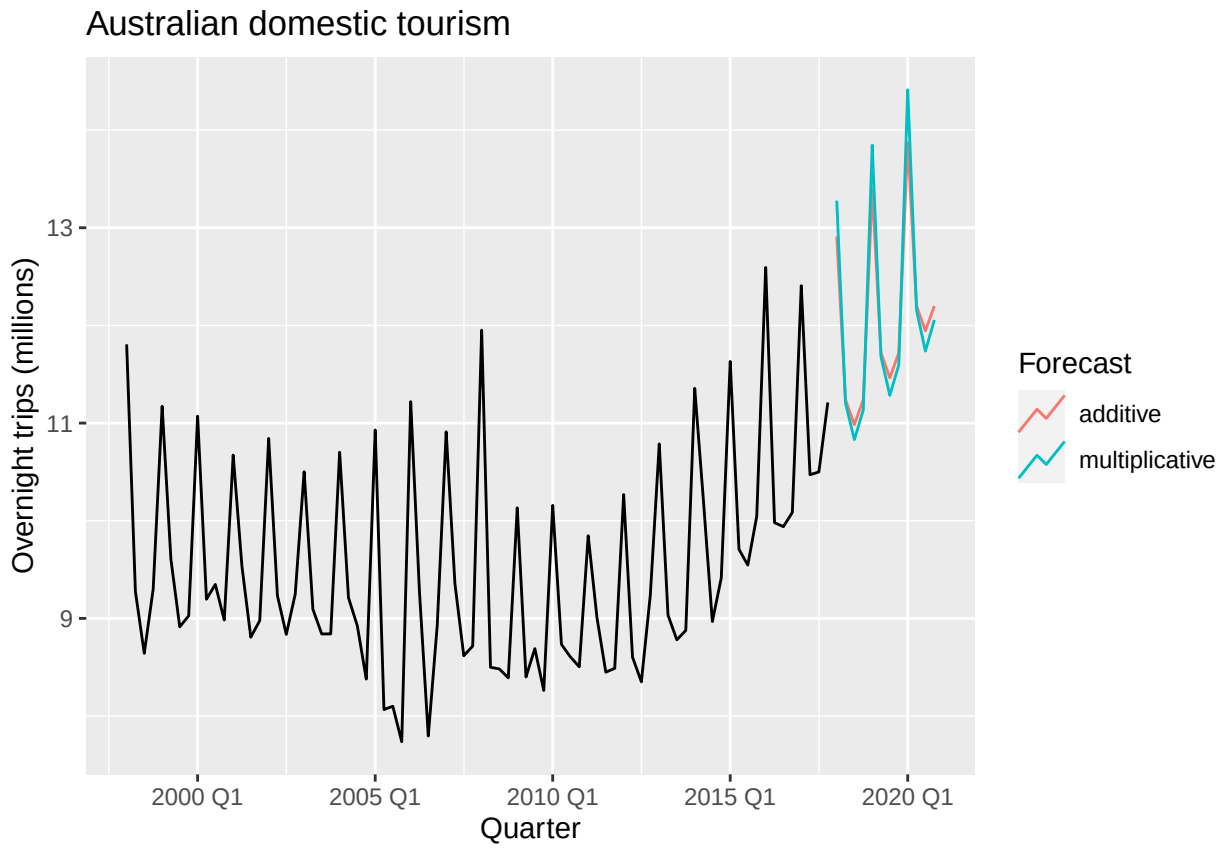
$$s_t = \gamma * (y_t/(\ell_{t-1} + b_{t-1})) + (1-\gamma)s_{t-m}.$$

Example: Domestic overnight trips in Australia

We apply Holt-Winters' method with both additive and multiplicative seasonality17 to forecast quarterly visitor nights in Australia spent by domestic tourists. Figure 8.7 shows the data from 1998–2017, and the

forecasts for 2018–2020. The data show an obvious seasonal pattern, with peaks observed in the March quarter of each year, corresponding to the Australian summer.

```
aus_holidays <- tourism |>
  filter(Purpose == "Holiday") |>
  summarise(Trips = sum(Trips)/1e3)
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"))
```



The applications of both methods (with additive and multiplicative seasonality) are presented in Tables 8.3 and 8.4 respectively. Because both methods have exactly the same number of parameters to estimate, we can compare the training RMSE from both models. In this case, the method with multiplicative seasonality fits the data slightly better.

The estimated components for both models are plotted in Figure 8.8. The small value of γ for the multiplicative model means that the seasonal component hardly changes over time. The small value of β^* means the

slope component hardly changes over time (compare the vertical scales of the slope and level components).

Holt-Winters' damped method

Damping is possible with both additive and multiplicative Holt-Winters' methods. A method that often provides accurate and robust forecasts for seasonal data is the Holt-Winters method with a damped trend and multiplicative seasonality:

$\hat{y}_{t+h|t} = [\ell_t + (\phi + \phi^2 + \cdots + \phi^h)b_t]s_{t+h-m(k+1)}$

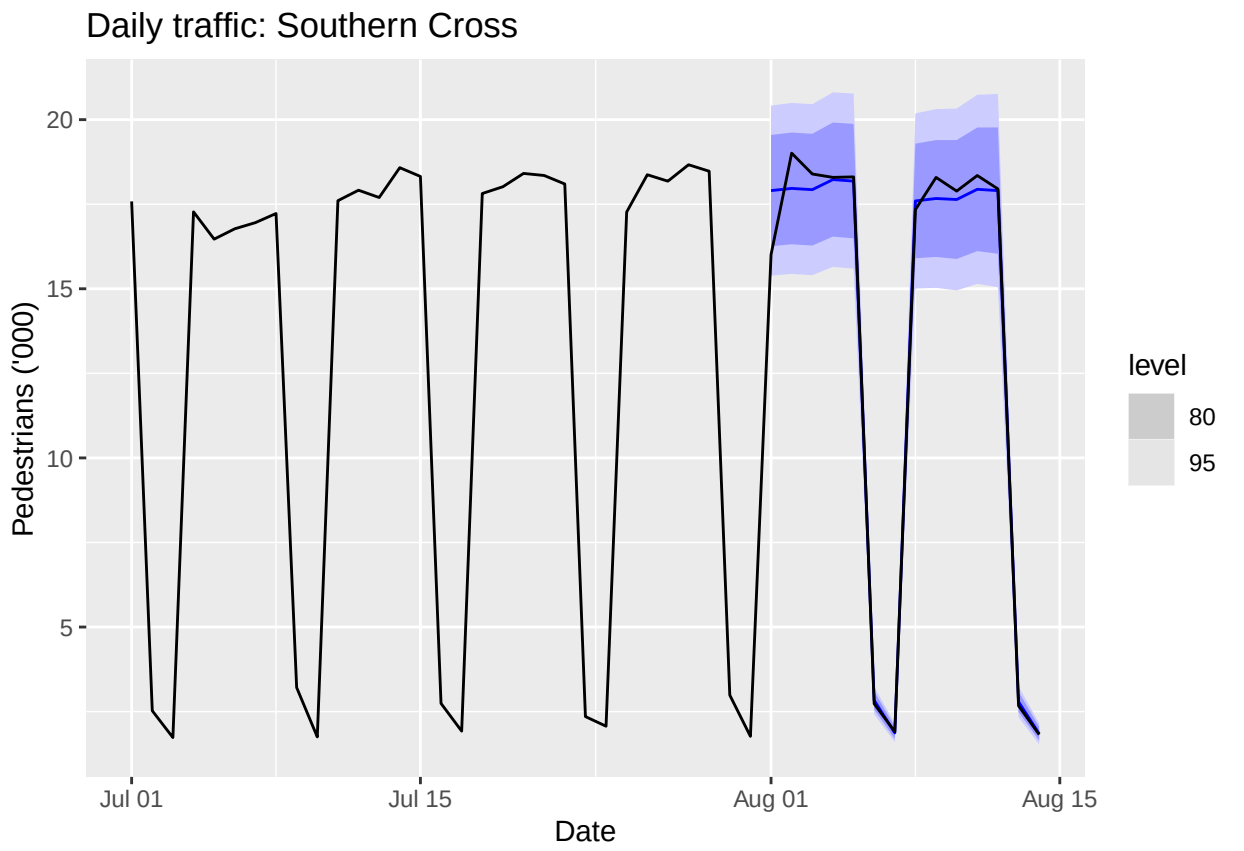
$\ell_t = \alpha(y_t/s_{t-m})) + (1-\alpha)(\ell_{t-1} + \phi b_{t-1})b_t = \beta^*(\ell_t - \ell_{t-1})) + (1-\beta^*)\phi b_{t-1}$

$s_t = \gamma * (y_t/(\ell_{t-1} + \phi b_{t-1})) + (1-\gamma)s_{t-m}.$

Example: Holt-Winters method with daily data

The Holt-Winters method can also be used for daily type of data, where the seasonal period is m=7, and the appropriate unit of time for h is in days. Here we forecast pedestrian traffic at a busy Melbourne train station in July 2016.

```
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"))
  ) |>
  forecast(h = "2 weeks") |>
  autoplot(sth_cross_ped |> filter(Date <= "2016-08-14")) +
  labs(title = "Daily traffic: Southern Cross",
       y="Pedestrians ('000)")
```



Clearly the model has identified the weekly seasonal pattern and the increasing trend at the end of the data, and the forecasts are a close match to the test data.

8.4 A taxonomy of exponential smoothing methods

Now, we see the whole landscape of methods which we got and how they are related to each other. We call this as a taxonomy of methods.

Check screenshot-189

We can combine various methods, for example: a method with no trend but with additive seasonality i.e. (N,A) method.

Here, (first_letter,second_letter) means (Trend,Seasonality).

There are also multiplicative trend methods but not recommended because they tend to give poor forecasts.

Check screenshot-190

———————————- Summary ———————————

Exponential smoothing methods are not restricted to those we have presented so far. By considering variations in the combinations of the trend and seasonal components, nine exponential smoothing methods are possible, listed in Table 8.5. Each method is labelled by a pair of letters (T,S) defining the type of 'Trend' and 'Seasonal' components. For example, (A,M) is the method with an additive trend and multiplicative seasonality; (A_d,N) is the method with damped trend and no seasonality; and so on.

This type of classification was first proposed by Pegels (1969), who also included a method with a multiplicative trend. It was later extended by Gardner (1985) to include methods with an additive damped trend and by J. W. Taylor (2003) to include methods with a multiplicative damped trend. We do not consider the

multiplicative trend methods in this book as they tend to produce poor forecasts. See Hyndman et al. (2008) for a more thorough discussion of all exponential smoothing methods.

Table 8.6 gives the recursive formulas for applying the nine exponential smoothing methods in Table 8.5. Each cell includes the forecast equation for generating h-step-ahead forecasts, and the smoothing equations for applying the method.

8.5 Innovations state space models for exponential smoothing

The methods which we have seen so far are called methods rather than models. They are simply algorithms that return the point forecast and although we fitted them using the model function and we showed the prediction intervals that we are looking at graphs.

Models will be used to show the prediction intervals. So, a model is a stochastic equation that includes some kind of random component. We are going to have 2 different models for every methods which we have talked about. The 2 models generate the same point forecast but could have quite different variances and therefore different forecast distributions.

SO WHEN WE talk about a model, it means a stochastic data generation process that can generate the whole forecast distribution. It also gives you tool for the model selection so that we can choose which model is more appropriate for the data.

Models:

- Generate some point forecasts but can also generate forecast distributions.
- A stochastic (or random) data generating process that can generate an entire forecast distribution.
- Allow for "proper" model selection.

ETS(A,N,N): SES with additive errors

Component Form

Forecast Equation

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

Smoothing Equation

$l_t = \alpha y_t + (1-\alpha) l_{t-1}$

Forecast error $e_t = y_t - \hat{y}_{t|t-1} = y_t - l_{t-1}$

Error Correction Form

$y_t = l_{t-1} + e_t$

$l_t = l_{t-1} + \alpha(y_t - l_{t-1})$

$l_t = l_{t-1} + \alpha(e_t)$

Specify probability distribution for e_t, we assume $e_t = \epsilon_t \; NID(0, \sigma^2)$.

Forecast error e_t has a normal distribution with constant variance and independent from time to time. "NID" means normal and independently distributed with mean 0 and variance σ^2. If we rewrite again:

Measurement Equation: $y_t = l_{t-1} + \epsilon_t$

State Equation: $l_t = l_{t-1} + \alpha\epsilon_t$

where $\epsilon_t \; NID(0, \sigma^2)$.

Properties We see what is forecast distribution or what is likelihood of it.

- It is called "innovations" or "single source of error state space model" because equations have the same error process, ϵ_t.
- Measurement equation: relationship between observation y_t and states like l_t or l_{t-1}.
- State equation(s): evolution of state(s) through time.

ETS(M,N,N): SES with multiplicative errors

- Specify relative errors(proportional error): $\epsilon_t = \frac{y_t - \hat{y}_{t|t-1}}{\hat{y}_{t|t-1}} \; NID(0, \sigma^2)$
- Substituting $\hat{y}_{t|t-1} = l_{t-1}$ in above relative error equation gives:

$y_t = l_{t-1} + l_{t-1}\epsilon_t$

$e_t = y_t - \hat{y}_{t|t-1} = l_{t-1}\epsilon_t$

Measurement Equation: $y_t = l_{t-1} + (1 + \epsilon_t)$

State Equation: $l_t = l_{t-1}(1 + \alpha\epsilon_t)$

So, we started with a same point forecast but we made a different assumption about what was the stochastic homoscedastic independent error is additive error or relative error.

- Models with additive and multiplicative errors with the same parameters generate the same point forecasts because α, l_0, b_0 are same but different prediction intervals because properties are different.

Specifying the model

$ETS(y\ error("A") + trend("N") + season("N"))$

$ETS(y\ error("M") + trend("N") + season("N"))$

These 2 represent a different likelihood functions.

By default, an optimal value for α and l_0 is used.

α can be chosen manually in trend().

$trend("N", alpha = 0.5)$

$trend("N", alpha_range = c(0.2, 0.8))$

We can do the same with different methods like damped method etc.

ETS(A,A,N)

Holt's linear method with additive errors.

- Assume $\epsilon_t = y_t - l_{t-1} - b_{t-1} \; NID(0, \sigma^2)$.
- Substituting into the error correction equations for Holt's linear method

$y_t = l_{t-1} + b_{t-1} + \epsilon_t$

$l_t = l_{t-1} + b_{t-1} + \alpha\epsilon_t$

$b_t = b_{t-1} + \hat{}* \; _t$

- For simplicity, set $\beta = \alpha\beta^*$.

ETS(M,A,N)

Holt's linear method with multiplicative errors

- Assume $\epsilon_t = \frac{y_t - (l_{t-1} + b_{t-1})}{(l_{t-1} + b_{t-1})}$
- Following a similar approach as above, the innovations state space model underlying Holt's linear method with multiplicative errors is specified as

$y_t = (l_{t-1} + b_{t-1})(1 + \epsilon_t)$

$l_t = (l_{t-1} + b_{t-1})(1 + \alpha\epsilon_t)$

$b_t = b_{t-1} + \beta(l_{t-1} + b_{t-1})\epsilon_t$

where again $\beta = \alpha\beta^*$ and $\epsilon_t \; NID(0, \sigma^2)$.

Specifying the model

$ETS(y\ error("A") + trend("A") + season("N"))$

$ETS(y\ error("M") + trend("A") + season("N"))$

These 2 represent a different likelihood functions.

By default, an optimal value for β and b_0 is used.

β can be chosen manually in trend().

$trend("A", beta = 0.004)$

$trend("A", beta_range = c(0, 0.1))$

ETS(A,A,A)

Holt-Winter's additive method with additive errors.

Forecast Equation: $\hat{y}_{t+h|t} = l_t + hb_t + s_{t+h-m(k+1)}$

Observation Equation: $y_t = l_{t-1} + b_{t-1} + s_{t-m} + \epsilon_t$

State Equations: $l_t = l_{t-1} + b_{t-1} + \alpha\epsilon_t$

$b_t = b_{t-1} + \beta\epsilon_t$

$s_t = s_{t-m} + \gamma\epsilon_t$

- *Forecast errors:* $\epsilon_t = y_t - \hat{y}_{t|t-1}$
- k is an integer part of $\frac{h-1}{m}$

ETS(M,A,M)

Holt-Winter's Multiplicative method with multiplicative errors.

Forecast Equation: $\hat{y}_{t+h|t} = (l_t + hb_t)s_{t+h-m(k+1)}$

Observation Equation: $y_t = (l_{t-1} + b_{t-1})s_{t-m}(1 + \epsilon_t)$

State Equations: $l_t = (l_{t-1} + b_{t-1})(1 + \alpha\epsilon_t)$

$b_t = b_{t-1} + \beta(l_{t-1} + b_{t-1})\epsilon_t$

$s_t = s_{t-m}(1 + \gamma\epsilon_t)$

- *Forecast errors:* $\epsilon_t = \frac{y_t - \hat{y}_{t|t-1}}{\hat{y}_{t|t-1}}$

- k is an integer part of $\frac{h-1}{m}$

Specifying the model

$ETS(y \sim error("A") + trend("A") + season("A"))$

$ETS(y \sim error("M") + trend("A") + season("M"))$

These 2 represent a different likelihood functions.

By default, an optimal value for γ and $s_0, s_{-1}, ..., s_{-m+1}$ are used.

γ can be chosen manually in season().

$season("A", gamma = 0.004)$

$season("A", gamma_range = c(0, 0.1))$

Check screenshots for 18 models and their equations: Screenshot-192,193,194

Summary -

In the rest of this chapter, we study the statistical models that underlie the exponential smoothing methods we have considered so far. The exponential smoothing methods presented in Table 8.6 are algorithms which generate point forecasts. The statistical models in this section generate the same point forecasts, but can also generate prediction (or forecast) intervals. A statistical model is a stochastic (or random) data generating process that can produce an entire forecast distribution. We will also describe how to use the model selection criteria introduced in Chapter 7 to choose the model in an objective manner.

Each model consists of a measurement equation that describes the observed data, and some state equations that describe how the unobserved components or states (level, trend, seasonal) change over time. Hence, these are referred to as state space models.

For each method there exist two models: one with additive errors and one with multiplicative errors. The point forecasts produced by the models are identical if they use the same smoothing parameter values. They will, however, generate different prediction intervals.

To distinguish between a model with additive errors and one with multiplicative errors (and also to distinguish the models from the methods), we add a third letter to the classification of Table 8.5. We label each state space model as ETS(, ,) for (Error, Trend, Seasonal). This label can also be thought of as ExponenTial Smoothing. Using the same notation as in Table 8.5, the possibilities for each component (or state) are: Error $= A, M$, Trend $= N, A, A_d$ and Seasonal $= N, A, M$.

ETS(A,N,N): simple exponential smoothing with additive errors

The training data errors lead to the adjustment of the estimated level throughout the smoothing process for $t = 1, ..., T$. For example, if the error at time t is negative, then $y_t < \hat{y}_{t|t-1}$ and so the level at time $t-1$ has been over-estimated. The new level ℓ_t is then the previous level ℓ_{t-1} adjusted downwards. The closer is to one, the "rougher" the estimate of the level (large adjustments take place). The smaller the , the "smoother" the level (small adjustments take place).

We can also write $y_t = \ell_{t-1} + e_t$, so that each observation can be represented by the previous level plus an error. To make this into an innovations state space model, all we need to do is specify the probability distribution for e_t. For a model with additive errors, we assume that residuals (the one-step training errors) e_t are normally distributed white noise with mean 0 and variance σ^2. A short-hand notation for this is $e_t = \varepsilon_t \sim NID(0, \sigma^2)$; NID stands for "normally and independently distributed".

We refer to (8.3) as the measurement (or observation) equation and (8.4) as the state (or transition) equation. These two equations, together with the statistical distribution of the errors, form a fully specified statistical model. Specifically, these constitute an innovations state space model underlying simple exponential smoothing.

The term "innovations" comes from the fact that all equations use the same random error process, ε_t. For the same reason, this formulation is also referred to as a "single source of error" model. There are alternative multiple source of error formulations which we do not present here.

The measurement equation shows the relationship between the observations and the unobserved states. In this case, observation y_t is a linear function of the level ℓ_{t-1}, the predictable part of y_t, and the error ε_t, the unpredictable part of y_t. For other innovations state space models, this relationship may be nonlinear.

The state equation shows the evolution of the state through time. The influence of the smoothing parameter is the same as for the methods discussed earlier. For example, governs the amount of change in successive levels: high values of allow rapid changes in the level; low values of lead to smooth changes. If =0, the level of the series does not change over time; if =1, the model reduces to a random walk model,$y_t = y_{t-1}+\varepsilon_t$. (See Section 9.1 for a discussion of this model.)

8.6 Estimation and model selection

Estimating ETS models

- Smoothing parameters $\alpha, \beta, \gamma, \phi(dampnening\ parameter)$ and the initial states $l_0, b_0, s_0, s_{-1}, ..., s_{-m+1}$ are estimated by maximizing the the "likelihood" (i.e. using maximum likelihood estimator)= the probability of the data arising from the specified model.
- For models with additive errors equivalent to minimising SSE.
- For models with multiplicative errors where we have heteroscadestic errors, *not* equivalent to minimising SSE.

Let $x_t = (l_t, b_t, s_t, s_{t-1}, ..., s_{t-m+1})$ and $\epsilon_t \sim_{iid} N(0, \sigma^2)$

Now, we can write all the models in this very general form:

Observation Equation: $y_t = h(x_{t-1}) + k(x_{t-1})\epsilon_t = \mu_t + e_t$

State Equation: $x_t = f(x_{t-1}) + g(x_{t-1})\epsilon_t$

Additive Errors

$k(x_{t-1}) = 1$ and $y_t = \mu_t + e_t$

Multiplicative Errors

$k(x_{t-1}) = \mu_t$ and $y_t = \mu_t(1 + e_t)$ and $\epsilon_t = \frac{y_t - \mu_t}{\mu_t}$ is relative error.

Now, we write the likelihood function as:

$L^*(\theta, x_0) = T \log(\Sigma_{t=1}^T \epsilon_t^2) + 2\Sigma_{t=1}^T \log|k(x_{t-1})|$

$= -2\log(likelihood) + constant$

Here maximizing the likelihood is equivalent to minimizing the L^* function. Since $k(x_{t-1}) = 1$ and so log value=0 and so we need to minimize $\Sigma_{t=1}^T \epsilon_t^2$

- Estimate parameters $\theta = (\alpha, \beta, \gamma, \phi)$ and initial states $x_0 = (l_0, b_0, s_0, s_0, s_{-1}, ..., s_{-m+1})$ by minimizing L^*

Usual region

- Traditional restrictions in the methods $0 < \alpha, \beta^*, \gamma^*, \phi < 1$ (equations interpreted as weighted averages)
- In models we set $\beta = \alpha\beta^*$ and $\gamma = (1 - \alpha)\gamma^*$
- Therefore, $0 < \alpha < 1, 0 < \beta < \alpha$ and $0 < \gamma < 1 - \alpha$
- $0.8 < \phi(dampening\ parameter) < 0.98$ to prevent the numerical difficulties.

Admissible Region

- To prevent observations in the distant past having a continuing effect on current forecasts.
- Usually (but not always) less restrictive than traditional region.
- For example, for ETS(A,N,N): traditional: $0 < \alpha < 1$ but admissible: $0 < \alpha < 2$

Model Selection

Akaike's Information Criterion

$AIC = -2\log(L) + 2k$

Where L is the likelihood and k is the number of parameters and initial states estimated in the model.

Corrected AIC

$AIC_c = AIC + \frac{2k(k+1)}{T-k-1}$

which is the AIC corrected (for small sample bias).

Bayesian Information Criterian

$BIC = AIC + k[\log(T) - 2]$

BIC actually does not focus on forecasting, it's got other optimality properties and so usually we don't use BIC and we use the other two.

AIC and cross validation

This is most significant result in this chapter.

Minimizing the AIC assuming Gaussian residuals is asymptotically equivalent to minimizing one-step time series cross validation MSE.

So, we mostly use AIC and the default setting in the ETS function is AIC_c in case we have small samples.

Automatic Forecasting

From Hyndman et al. (IJF, 2002):

- Apply each model that is appropriate to the data. Optimize parameters and initial values using MLE (or some other criterian)
- Select best method using AICc.
- Produce forecasts using best method.
- Obtain forecast intervals using underlying state space model.

Method performed very well in M3 competition.

Some unstable models

- Some of the combinations of (Error, Trend, Seasonal) can lead to numerical difficulties; see equations with *division by a state*
- These are: ETS(A,N,M), ETS(A,A,M), $ETS(A, A_d, M)$. and causes some instability.
- Models with *multiplicative errors* are useful for *strictly positive data,* but are not numerically stable with data containing zeros or negative values. In that case only the *six fully additive models* will be applied.

Example: National Populations

```
fit <- global_economy |>
  mutate(Pop = Population/1e6) |>
  model(ets=ETS(Pop))
```

```
## Warning: 1 error encountered for ets
## [1] ETS does not support missing values.
```

```
fit
```

```
## # A mable: 263 x 2
## # Key:     Country [263]
##    Country                      ets
##    <fct>                    <model>
##  1 Afghanistan         <ETS(A,A,N)>
##  2 Albania             <ETS(M,A,N)>
##  3 Algeria             <ETS(M,A,N)>
##  4 American Samoa      <ETS(M,A,N)>
##  5 Andorra             <ETS(M,A,N)>
##  6 Angola              <ETS(M,A,N)>
##  7 Antigua and Barbuda <ETS(M,A,N)>
##  8 Arab World          <ETS(M,A,N)>
##  9 Argentina           <ETS(A,A,N)>
## 10 Armenia             <ETS(M,A,N)>
## # i 253 more rows
```

Here, no seasonal component for this annual data. Sometimes process chooses additive errors and sometimes it chooses multiplicative errors.

```
fit |>
  forecast(h=5)
```

```
## # A fable: 1,315 x 5 [1Y]
## # Key:     Country, .model [263]
##    Country     .model  Year             Pop .mean
##    <fct>       <chr>  <dbl>          <dist> <dbl>
##  1 Afghanistan ets     2018    N(36, 0.012) 36.4
##  2 Afghanistan ets     2019    N(37, 0.059) 37.3
##  3 Afghanistan ets     2020     N(38, 0.16) 38.2
##  4 Afghanistan ets     2021     N(39, 0.35) 39.0
##  5 Afghanistan ets     2022     N(40, 0.64) 39.9
##  6 Albania     ets     2018 N(2.9, 0.00012)  2.87
##  7 Albania     ets     2019   N(2.9, 6e-04)  2.87
##  8 Albania     ets     2020  N(2.9, 0.0017)  2.87
##  9 Albania     ets     2021  N(2.9, 0.0036)  2.86
## 10 Albania     ets     2022  N(2.9, 0.0066)  2.86
## # i 1,305 more rows
```

Residuals

Response Residuals

$e_t = y_t - \hat{y}_{t|t-1}$

Innovation Residuals

Additive error model: $\hat{\epsilon}_t = y_t - \hat{y}_{t|t-1} = e_t$

Multiplicative error model: $\frac{y_t - \hat{y}_{t|t-1}}{\hat{y}_{t|t-1}} \neq e_t$

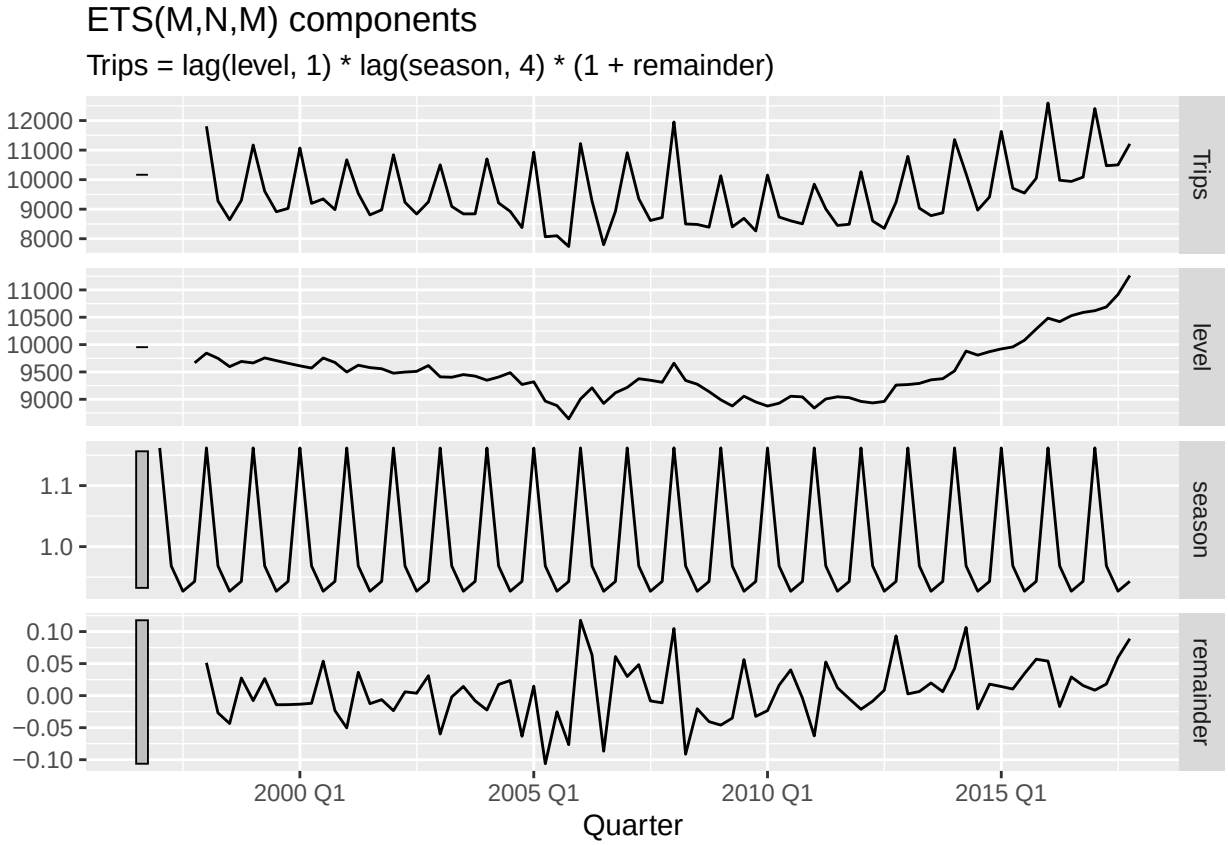
```
aus_holidays <- tourism |>
  filter(Purpose == "Holiday") |>
  summarise(Trips = sum(Trips))
fit <- aus_holidays |>
  model(ets = ETS(Trips)) |>
  report()
```

Example: Australian Holiday Tourism

```
## Series: Trips
## Model: ETS(M,N,M)
##   Smoothing parameters:
##     alpha = 0.3578226
##     gamma = 0.0009685565
##
##   Initial states:
##      l[0]      s[0]     s[-1]    s[-2]    s[-3]
##  9666.501 0.9430367 0.9268433 0.968352 1.161768
##
##   sigma^2:  0.0022
##
##      AIC     AICc      BIC
## 1331.372 1332.928 1348.046
```

```
components(fit) |> autoplot()+labs(title = "ETS(M,N,M) components")
```

```
## Warning: Removed 4 rows containing missing values (`geom_line()`).
```



Here, our seasonal component is of course multiplicative and remainder is relative error, it's not response residual. We can see the difference as:

```
residuals(fit)
```

```
## # A tsibble: 80 x 3 [1Q]
## # Key:       .model [1]
##    .model Quarter   .resid
##    <chr>    <qtr>    <dbl>
##  1 ets    1998 Q1  0.0513
##  2 ets    1998 Q2 -0.0269
##  3 ets    1998 Q3 -0.0435
##  4 ets    1998 Q4  0.0275
##  5 ets    1999 Q1 -0.00781
##  6 ets    1999 Q2  0.0266
##  7 ets    1999 Q3 -0.0142
##  8 ets    1999 Q4 -0.0140
##  9 ets    2000 Q1 -0.0134
## 10 ets    2000 Q2 -0.0120
## # i 70 more rows
```

```
residuals(fit,type="response")
```

```
## # A tsibble: 80 x 3 [1Q]
## # Key:       .model [1]
##    .model Quarter .resid
##    <chr>    <qtr>  <dbl>
##  1 ets    1998 Q1  576.
```

```
##  2 ets    1998 Q2 -257.
##  3 ets    1998 Q3 -393.
##  4 ets    1998 Q4  249.
##  5 ets    1999 Q1  -88.0
##  6 ets    1999 Q2  249.
##  7 ets    1999 Q3 -129.
##  8 ets    1999 Q4 -129.
##  9 ets    2000 Q1 -150.
## 10 ets    2000 Q2 -111.
## # i 70 more rows
```

```
fit |>
  augment()
```

```
## # A tsibble: 80 x 6 [1Q]
## # Key:       .model [1]
##    .model Quarter  Trips .fitted .resid   .innov
##    <chr>    <qtr>  <dbl>   <dbl>  <dbl>    <dbl>
##  1 ets    1998 Q1 11806.  11230.  576.   0.0513
##  2 ets    1998 Q2  9276.   9532. -257.  -0.0269
##  3 ets    1998 Q3  8642.   9036. -393.  -0.0435
##  4 ets    1998 Q4  9300.   9050.  249.   0.0275
##  5 ets    1999 Q1 11172.  11260.  -88.0 -0.00781
##  6 ets    1999 Q2  9608.   9358.  249.   0.0266
##  7 ets    1999 Q3  8914.   9042. -129.  -0.0142
##  8 ets    1999 Q4  9026.   9154. -129.  -0.0140
##  9 ets    2000 Q1 11071.  11221. -150.  -0.0134
## 10 ets    2000 Q2  9196.   9308. -111.  -0.0120
## # i 70 more rows
```

- Summary

Estimating ETS models

An alternative to estimating the parameters by minimising the sum of squared errors is to maximise the "likelihood". The likelihood is the probability of the data arising from the specified model. Thus, a large likelihood is associated with a good model. For an additive error model, maximising the likelihood (assuming normally distributed errors) gives the same results as minimising the sum of squared errors. However, different results will be obtained for multiplicative error models. In this section, we will estimate the smoothing parameters α, β, γ and ϕ, and the initial states $\ell_0, b_0, s_0, s_{-1}, ..., s_{-m+1}$, by maximising the likelihood.

The possible values that the smoothing parameters can take are restricted. Traditionally, the parameters have been constrained to lie between 0 and 1 so that the equations can be interpreted as weighted averages. That is, $0 < \alpha, \beta^*, \gamma^*, \phi < 1$. For the state space models, we have set $\beta = \alpha\beta^*$and $\gamma = (1 - \alpha)\gamma^*$.

Therefore, the traditional restrictions translate to $0 < \alpha < 1, 0 < \beta < \alpha$ and $0 < \gamma < 1 - \alpha$. In practice, the damping parameter ϕ is usually constrained further to prevent numerical difficulties in estimating the model. In the fable package, it is restricted so that $0.8 < \phi < 0.98$.

Another way to view the parameters is through a consideration of the mathematical properties of the state space models. The parameters are constrained in order to prevent observations in the distant past having a continuing effect on current forecasts. This leads to some admissibility constraints on the parameters, which are usually (but not always) less restrictive than the traditional constraints region (Hyndman et al., 2008, pp. 149–161). For example, for the ETS(A,N,N) model, the traditional parameter region is $0 < \alpha < 1$ but the admissible region is $0 < \alpha < 2$. For the ETS(A,A,N) model, the traditional parameter region is $0 < \alpha < 1$ and $0 < \beta < \alpha$ but the admissible region is $0 < \alpha < 2$ *and* $0 < \beta < 4 - 2\alpha$.

Model selection

A great advantage of the ETS statistical framework is that information criteria can be used for model selection. The AIC, AICc and BIC, introduced in Section 7.5, can be used here to determine which of the ETS models is most appropriate for a given time series.

For ETS models, Akaike's Information Criterion (AIC) is defined as

$AIC = -2log(L) + 2k,$

where L is the likelihood of the model and k is the total number of parameters and initial states that have been estimated (including the residual variance).

The AIC corrected for small sample bias (AICc) is defined as

$AICc = AIC + (2k(k+1))/(T-k-1),$

and the Bayesian Information Criterion (BIC) is

$BIC = AIC + k[log(T) - 2].$

Three of the combinations of (Error, Trend, Seasonal) can lead to numerical difficulties. Specifically, the models that can cause such instabilities are ETS(A,N,M), ETS(A,A,M), and ETS(A,A_d,M), due to division by values potentially close to zero in the state equations. We normally do not consider these particular combinations when selecting a model.

Models with multiplicative errors are useful when the data are strictly positive, but are not numerically stable when the data contain zeros or negative values. Therefore, multiplicative error models will not be considered if the time series is not strictly positive. In that case, only the six fully additive models will be applied.

Example: Domestic holiday tourist visitor nights in Australia

We now employ the ETS statistical framework to forecast Australian holiday tourism over the period 2016–2019. We let the ETS() function select the model by minimising the AICc.

```
aus_holidays <- tourism |>
  filter(Purpose == "Holiday") |>
  summarise(Trips = sum(Trips)/1e3)
fit <- aus_holidays |>
  model(ETS(Trips))
report(fit)
```

```
## Series: Trips
## Model: ETS(M,N,A)
##   Smoothing parameters:
##     alpha = 0.3484054
##     gamma = 0.0001000018
##
##   Initial states:
##      l[0]       s[0]      s[-1]      s[-2]    s[-3]
##  9.727072 -0.5376106 -0.6884343 -0.2933663 1.519411
##
##   sigma^2:  0.0022
##
##      AIC     AICc      BIC
## 226.2289 227.7845 242.9031
```

```
#> Series: Trips
#> Model: ETS(M,N,A)
```

```
#>   Smoothing parameters:
#>     alpha = 0.3484
#>     gamma = 1e-04
#>
#>   Initial states:
#>   l[0]    s[0]   s[-1]   s[-2] s[-3]
#>  9.727 -0.5376 -0.6884 -0.2934 1.519
#>
#>   sigma^2:  0.0022
#>
#>   AIC  AICc   BIC
#> 226.2 227.8 242.9
```

The model selected is ETS(M,N,A)

$y_t = (\ell_{t-1} + s_{t-m})(1 + \varepsilon_t)$

$\ell + t = \ell_{t-1} + \alpha(\ell_{t-1} + s_{t-m})\varepsilon_t$

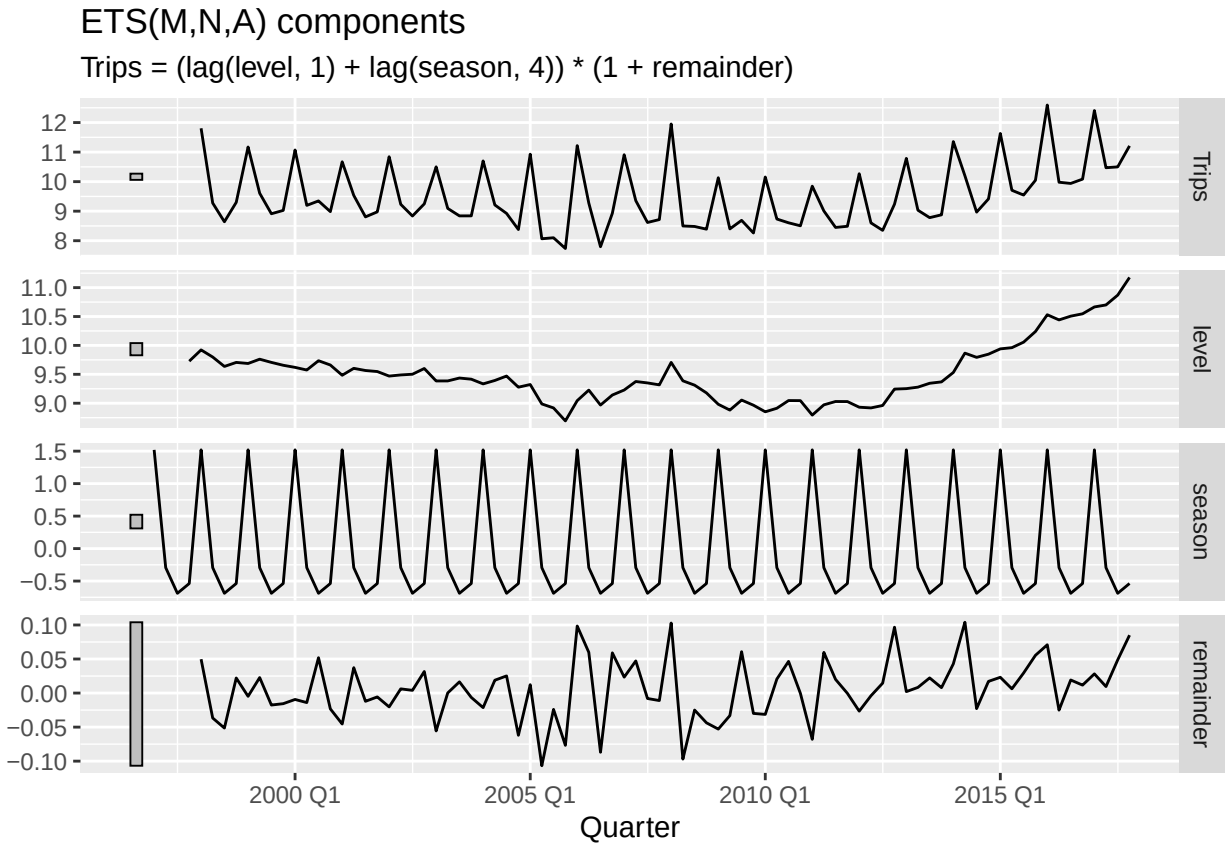
$s + t = s_{t-m} + \gamma(\ell_{t-1} + s_{t-m})\varepsilon_t$

The parameter estimates are $\hat{\alpha} = 0.3484$, and $\hat{\gamma} = 0.0001$. The output also returns the estimates for the initial states ℓ_0,s_0, s_{-1}, s_{-2} and s_{-3}.Compare these with the values obtained for the Holt-Winters method with additive seasonality presented in Table 8.3.

Figure 8.10 shows the states over time, while Figure 8.12 shows point forecasts and prediction intervals generated from the model. The small values of indicate that the seasonal states change very little over time.

```
components(fit) |>
  autoplot() +
  labs(title = "ETS(M,N,A) components")
```

```
## Warning: Removed 4 rows containing missing values (`geom_line()`).
```



Because this model has multiplicative errors, the innovation residuals are not equivalent to the regular residuals (i.e., the one-step training errors). The innovation residuals are given by $\hat{\varepsilon}_t$, while the regular residuals are defined as $y_t - \hat{y}_{t|t-1}$. We can obtain both using the augment() function. They are plotted in Figure 8.11.

8.7 Forecasting with ETS models

Traditional Point Forecasts: iterate the equations for $t = T+1, T+2, ..., T+h$ and set all stochastic error terms $\epsilon_t = 0$ for $t > T$

It works pretty well when the seasonality is additive but it is actually not the same as mean of the forecast distribution when the seasonality is multiplicative.

- Not the same as $E(y_{t+h}|x_t)$ unless seasonality is additive.
- fable uses $E(y_{t+h}|x_t)$.
- Point forecasts for ETS(A,* , *) are identical to ETS(M, * , *) if the parameters are the same. The only difference is whether the error is multiplicative or additive then we will end up with the same point forecasts otherwise we will have different point forecasts.

Example: ETS(A,A,N)

$y_{T+1} = l_T + b_T + \epsilon_{T+1}$

$\hat{y}_{T+1|T} = l_T + b_T$

$y_{T+2} = l_{T+1} + b_{T+1} + \epsilon_{T+2}$

$= (l_T + b_T + \alpha\epsilon_{T+1}) + (b_T + \beta\epsilon_{T+1}) + \epsilon_{T+2}$

$\hat{y}_{T+2|T} = l_T + 2b_T$

etc.

Here, ϵ_{T+1} is unknown and mean of it, is zero which is reflected in $\hat{y}_{T+1}$ and $\hat{y}_{T+2}$.

Example: ETS(M,A,N)

$y_{T+1} = (l_T + b_T)(1 + \epsilon_{T+1})$

$\hat{y}_{T+1|T} = l_T + b_T$

$y_{T+2} = (l_{T+1} + b_{T+1})(1 + \epsilon_{T+2})$

$= (l_T + b_T)(1 + \alpha\epsilon_{T+1}) + [(b_T + \beta(l_T + b_T)\epsilon_{T+1}](1 + \epsilon_{T+2})$

$\hat{y}_{T+2|T} = l_T + 2b_T$

etc.

Forecasting with ETS Models

Prediction Intervals

can only be generated from models.

- The prediction intervals will differ between models with additive and multiplicative errors.
- Exact formulae for some models.
- More general to simulate future sample paths, conditional on the last estimate of the states and to obtain prediction intervals from the percentiles of these simulated future paths.

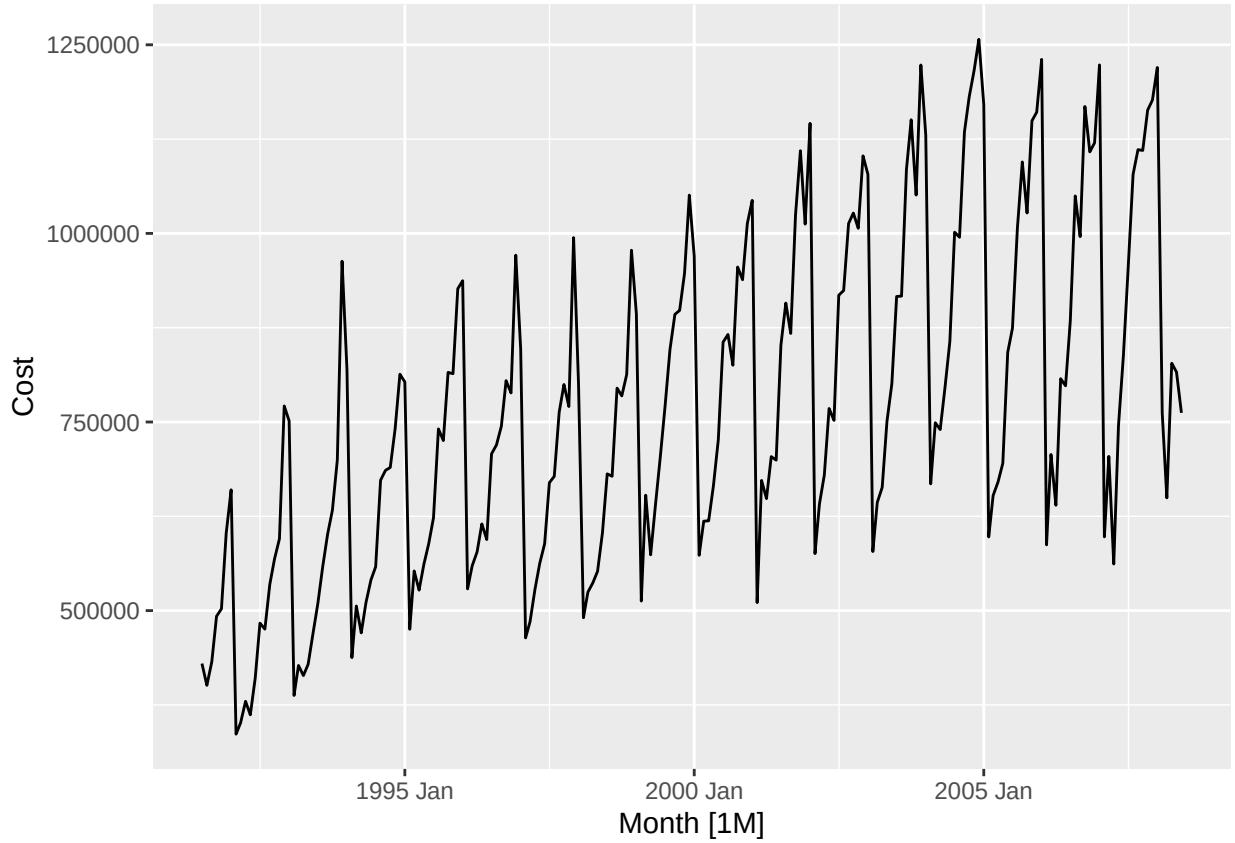
Prediction Intervals(PI)

PI for most ETS models: $\hat{y}_{T+h|T} \pm c\sigma_h$ where c depends on the coverage probability and σ_h is the forecast standard deviation.

Check screenshot for the exact formulas. Screenshot- 195

Example: Corticosteriod drug sales

```
h02 <- PBS |>
  filter(ATC2 == "H02") |>
  summarise(Cost = sum(Cost))
h02 |> autoplot(Cost)
```



Now, if we ask fable to give us an ETS model, it gives $ETS(M, A_d, M)$ as:

```
h02 |>
  model(ETS(Cost)) |>
  report()
```

```
## Series: Cost
## Model: ETS(M,Ad,M)
##   Smoothing parameters:
##     alpha = 0.3071016
##     beta  = 0.0001006793
##     gamma = 0.0001007181
##     phi   = 0.977528
##
##   Initial states:
##      l[0]    b[0]      s[0]     s[-1]     s[-2]     s[-3]     s[-4]    s[-5]
##  417268.7 8205.82 0.8716807 0.8259747 0.7562808 0.7733338 0.6872373 1.283821
##     s[-6]    s[-7]    s[-8]    s[-9]   s[-10]    s[-11]
##  1.324616 1.180067 1.163601 1.104801 1.047963 0.9806235
##
##   sigma^2:  0.0046
##
##      AIC     AICc      BIC
## 5515.212 5518.909 5574.938
```

Here, beta and gamma are almost zero that tells us something about the model that the slope is not changing much over the time, it;s hardly changing, so, we have got linear trend and seasonality is also not changing

much over time, so seasonality is roughly periodic for this model.

Here, we use AICc. σ^2 represents the variance of ϵ or variance of the relative errors.

Now, what if we had chosen a model.

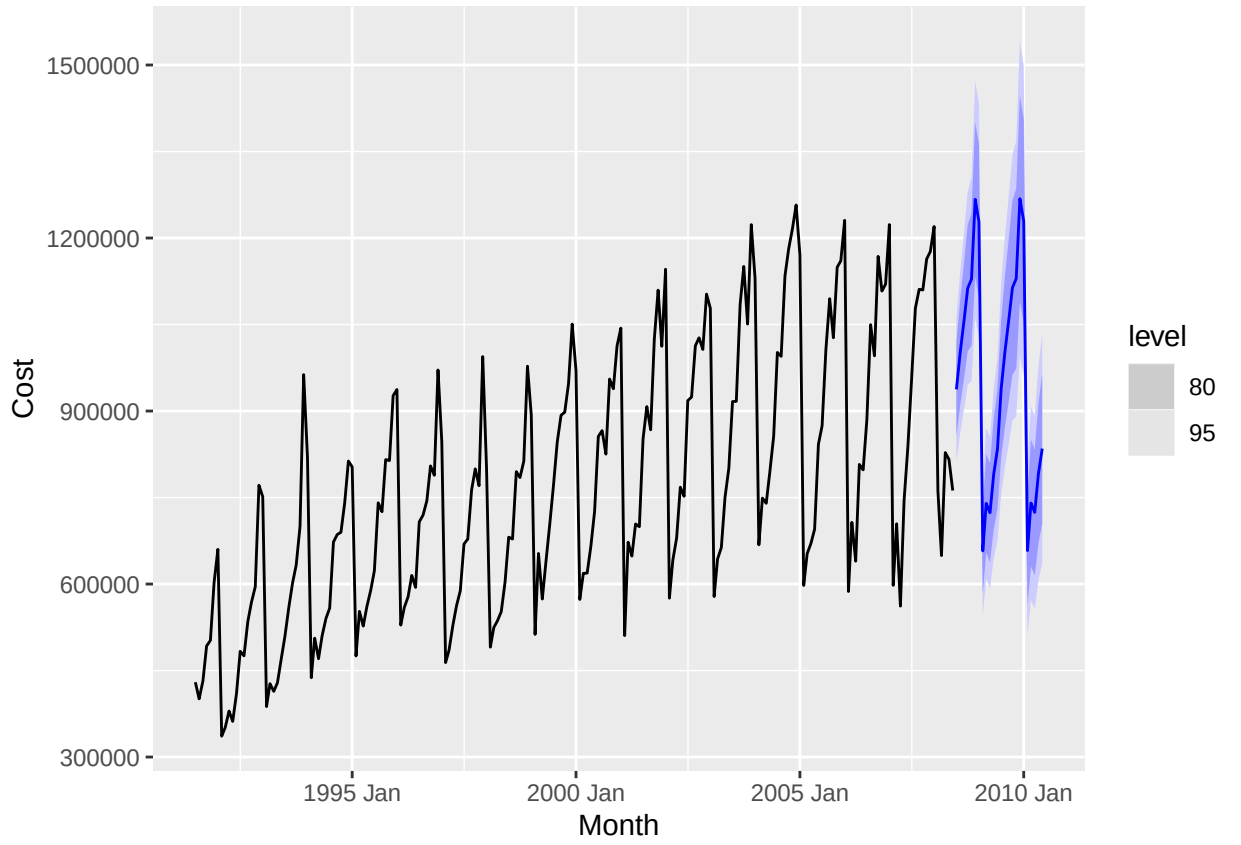
```
h02 |>
  model(ETS(Cost ~ error("A")+trend("A")+season("A"))) |>
  report()
```

```
## Series: Cost
## Model: ETS(A,A,A)
##   Smoothing parameters:
##     alpha = 0.1702163
##     beta  = 0.006310854
##     gamma = 0.4545987
##
##   Initial states:
##      l[0]     b[0]      s[0]     s[-1]     s[-2]     s[-3]     s[-4]    s[-5]
##  409705.9 9097.111 -99075.37 -136602.3 -191496.1 -174530.8 -241436.7 210643.8
##     s[-6]    s[-7]    s[-8]    s[-9]  s[-10]    s[-11]
##  244644.2 145368.2 130569.6 84457.69 39131.7 -11673.71
##
##   sigma^2:  3498869384
##
##      AIC     AICc      BIC
## 5585.278 5588.568 5641.686
```

Here, we still get a small beta because trend is close to linear but other things are different.Here, AICc is bigger than previous. So, Fable choses the model with lower AICc and it's giving the best model that's generally gives the best forecast.

Now, let's see the forecast with the automatically chosen model.

```
h02 |>
  model(ETS(Cost)) |>
  forecast() |>
  autoplot(h02)
```



Now, compare models on training set.

```
h02 |>
  model(
    auto = ETS(Cost),
    AAA = ETS(Cost ~ error("A")+trend("A")+season("A"))
  ) |>
  accuracy()
```

```
## # A tibble: 2 x 10
##   .model .type        ME   RMSE    MAE     MPE  MAPE  MASE RMSSE    ACF1
##   <chr>  <chr>     <dbl>  <dbl>  <dbl>   <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 auto   Training  2461. 51102. 38649. -0.0127  4.99 0.638 0.689 -0.0958
## 2 AAA    Training -5780. 56784. 43378. -1.30    6.05 0.716 0.766  0.0258
```

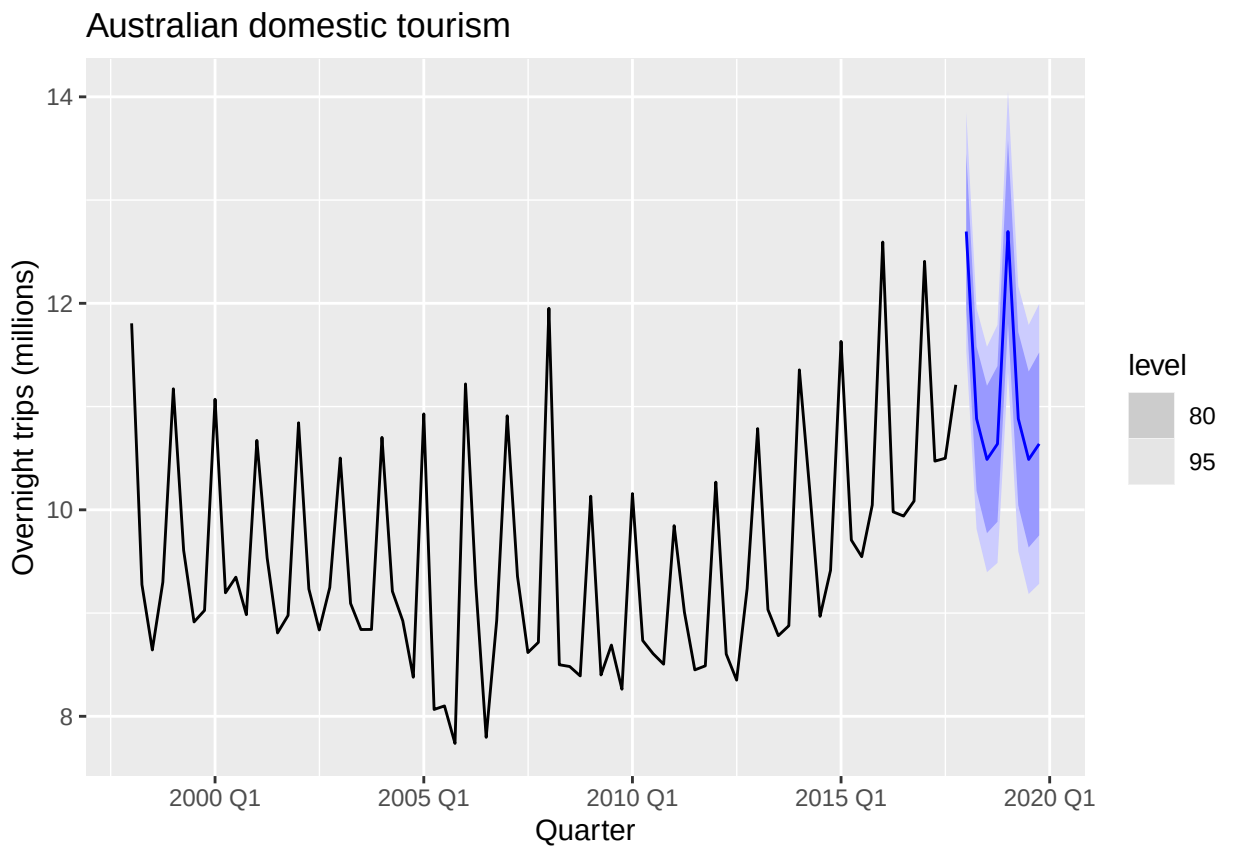
Here, RMSE shows that automatically chosen model is better fit than the AAA model.

Summary

The point forecasts obtained from the method and from the two models that underlie the method are identical (assuming that the same parameter values are used). ETS point forecasts constructed in this way are equal to the means of the forecast distributions, except for the models with multiplicative seasonality (Hyndman et al., 2008).

To obtain forecasts from an ETS model, we use the forecast() function from the fable package. This function will always return the means of the forecast distribution, even when they differ from these traditional point forecasts.

```
fit |>
  forecast(h = 8) |>
  autoplot(aus_holidays)+
  labs(title="Australian domestic tourism",
       y="Overnight trips (millions)")
```



Prediction Intervals

A big advantage of the statistical models is that prediction intervals can also be generated — something that cannot be done using the point forecasting methods alone. The prediction intervals will differ between models with additive and multiplicative methods.

For a few ETS models, there are no known formulas for prediction intervals. In these cases, the forecast() function uses simulated future sample paths and computes prediction intervals from the percentiles of these simulated future paths.