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

The simplest of the exponentially smoothing methods is naturally called **simple exponential smoothing** (SES)[1]. 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 = pd.read_csv("../data/algeria_exports.csv", parse_dates=["ds"])

plot_series(algeria_economy, xlabel="Year [1Y]", ylabel="% of GDP", title="Exports: Algeria")
```

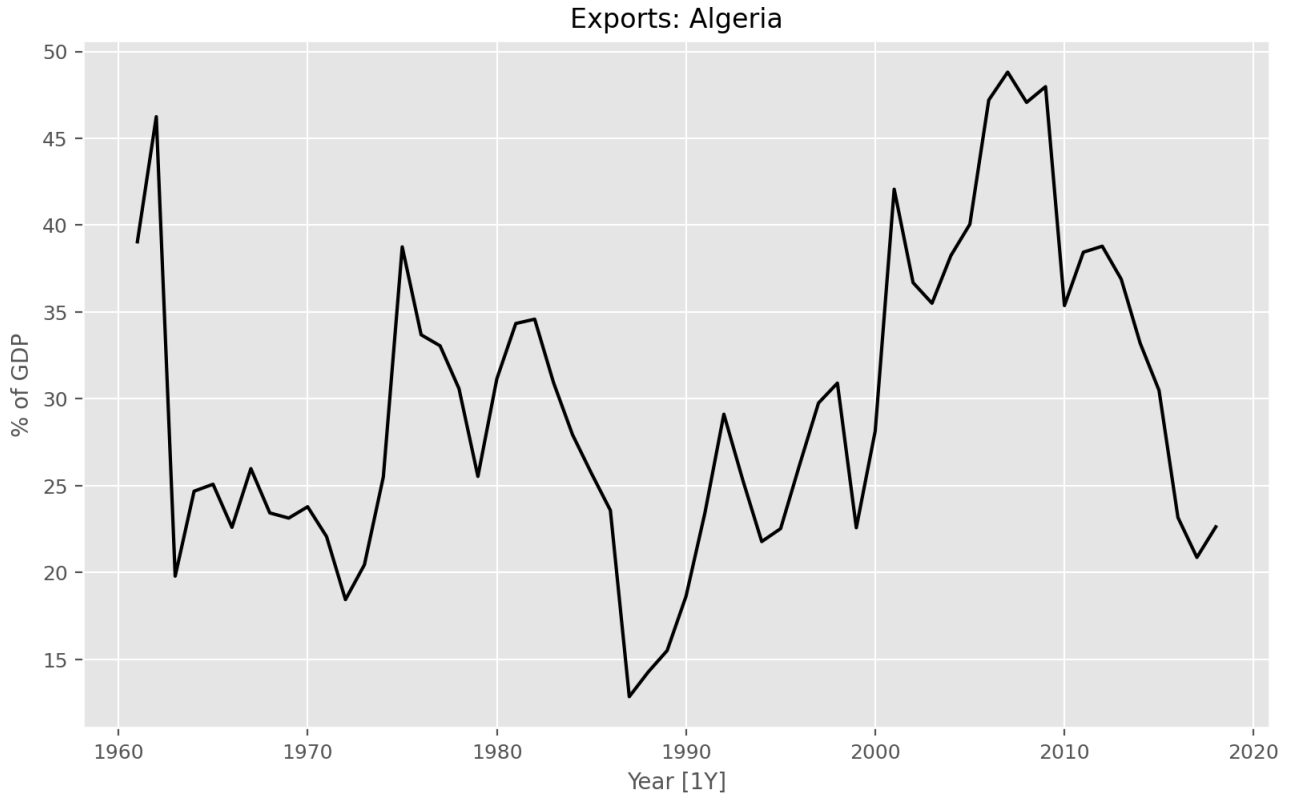

Figure 8.1: Exports of goods and services from Algeria from 1960 to 2017.

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,\dots. 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} = \frac1T \sum_{t=1}^T y_t, for h=1,2,\dots. 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}+ \cdots, \tag{8.1} where 0 \le \alpha \le 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,\dots,y_T. The rate at which the weights decrease is controlled by the parameter \alpha.

The table below shows the weights attached to observations for four different values of \alpha when forecasting using simple exponential smoothing. Note that the sum of the weights even for a small value of \alpha will be approximately one for any reasonable sample size.

	\alpha = 0.2	\alpha = 0.4	\alpha = 0.6	\alpha = 0.8
y_T	0.2000	0.4000	0.6000	0.8000
y_{T-1}	0.1600	0.2400	0.2400	0.1600
y_{T-2}	0.1280	0.1440	0.0960	0.0320
y_{T-3}	0.1024	0.0864	0.0384	0.0064
y_{T-4}	0.0819	0.0518	0.0154	0.0013
y_{T-5}	0.0655	0.0311	0.0061	0.0003

For any \alpha between 0 and 1, the weights attached to the observations decrease exponentially as we go back in time, hence the name “exponential smoothing”. If \alpha is small (i.e., close to 0), more weight is given to observations from the more distant past. If \alpha 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 \le \alpha \le 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,\dots,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 \ell_0 (which we will have to estimate). Then \begin{align*} \hat{y}_{2|1} &= \alpha y_1 + (1-\alpha) \ell_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}\\ \vdots\\ \hat{y}_{T|T-1} &= \alpha y_{T-1} + (1-\alpha) \hat{y}_{T-1|T-2}\\ \hat{y}_{T+1|T} &= \alpha y_T + (1-\alpha) \hat{y}_{T|T-1}. \end{align*} Substituting each equation into the following equation, we obtain \begin{align*} \hat{y}_{3|2} & = \alpha y_2 + (1-\alpha) \left[\alpha y_1 + (1-\alpha) \ell_0\right] \\ & = \alpha y_2 + \alpha(1-\alpha) y_1 + (1-\alpha)^2 \ell_0 \\ \hat{y}_{4|3} & = \alpha y_3 + (1-\alpha) [\alpha y_2 + \alpha(1-\alpha) y_1 + (1-\alpha)^2 \ell_0]\\ & = \alpha y_3 + \alpha(1-\alpha) y_2 + \alpha(1-\alpha)^2 y_1 + (1-\alpha)^3 \ell_0 \\ & ~~\vdots \\ \hat{y}_{T+1|T} & = \sum_{j=0}^{T-1} \alpha(1-\alpha)^j y_{T-j} + (1-\alpha)^T \ell_{0}. \end{align*} 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, \ell_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: \begin{align*} \text{Forecast equation} && \hat{y}_{t+h|t} & = \ell_{t}\\ \text{Smoothing equation} && \ell_{t} & = \alpha y_{t} + (1 - \alpha)\ell_{t-1}, \end{align*} where \ell_{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 \ell_t with \hat{y}_{t+1|t} and \ell_{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}=\ell_T, \qquad h=2,3,\dots. 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 \alpha and \ell_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,\dots,T. Hence, we find the values of the unknown parameters and the initial values that minimise \text{SSE}=\sum_{t=1}^T(y_t - \hat{y}_{t|t-1})^2=\sum_{t=1}^Te_t^2. \tag{8.2} 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. As we will explain later, simple exponential smoothing is a specific instance of the `AutoETS` model. You can also use `SimpleExponentialSmoothingOptimized`, which serves as a wrapper around `AutoETS` for this particular case.

```
sf = StatsForecast(
    models=[AutoETS(season_length=1, model="ANN", alias="SES")], freq="Y"
)
```

To generate the forecasts, you can use the `forecast` method from the `StatsForecast` class. Setting `fitted=True` will return the fitted values. These values can be accessed through the `forecast_fitted_values` method after executing the `forecast` method.

```
fc = sf.forecast(df=algeria_economy, h=5, level=[80, 95], fitted=True)
fc.head()
```

	unique_id	ds	SES	SES-lo-95	SES-lo-80	SES-hi-80	SES-hi-95
0	Algeria	2018-12-31	22.442	10.523	14.648	30.235	34.361
1	Algeria	2019-12-31	22.442	6.875	12.263	32.620	38.009
2	Algeria	2020-12-31	22.442	3.933	10.339	34.544	40.951
3	Algeria	2021-12-31	22.442	1.398	8.682	36.202	43.486
4	Algeria	2022-12-31	22.442	-0.863	7.204	37.680	45.746

```
fitted_vals = sf.forecast_fitted_values()
fitted_vals.head()
```

	unique_id	ds	y	SES	SES-lo-95	SES-lo-80	SES-hi-80	SES-hi-95
0	Algeria	1960-12-31	39.043	39.530	27.720	31.808	47.252	51.340
1	Algeria	1961-12-31	46.245	39.121	27.311	31.399	46.843	50.931
2	Algeria	1962-12-31	19.794	45.105	33.295	37.383	52.828	56.916
3	Algeria	1963-12-31	24.685	23.842	12.031	16.119	31.564	35.652
4	Algeria	1964-12-31	25.084	24.550	12.740	16.828	32.272	36.360

The `forecast` method of the `StatsForecast` class is optimized for speed and scalability, so it does not store any model parameters. If you want to store the model parameters, you can use the models directly. You first need to instantiate them, call the `fit` method, and then extract the optimal parameters, as shown below.

```
ses = AutoETS(season_length=1, model="ANN", alias="SES")
ses = ses.fit(y=algeria_economy["y"].values)

params = np.round(ses.model_["fit"][0],4)
print("Optimal parameters:")
print("alpha:", params[0])
print("Initial level:", params[1])
```

```
Optimal parameters:
alpha: 0.8401
Initial level: 39.53
```

This gives parameter estimates \hat\alpha=0.84 and \hat\ell_0=39.5, obtained by minimising SSE over periods t=1,2,\dots,58, subject to the restriction that 0\le\alpha\le1.

In Table 8.1 we demonstrate the calculation using these parameters. The second last column shows the estimated level for times t=0 to t=58; the last few rows of the last column show the forecasts for h=1 to 5-steps ahead.

Table 8.1: Forecasting goods and services exports from Algeria using simple exponential smoothing.

Year	Time	Observation	Level	Forecast
	t	y_t	l_t	\hat{y}_{t|t-1}
1959	0		39.54	
1960	1	39.04	39.12	39.54
1961	2	46.24	45.10	39.12
1962	3	19.79	23.84	45.10
1963	4	24.68	24.55	23.84
1964	5	25.08	25.00	24.55
1965	6	22.60	22.99	25.00
1966	7	25.99	25.51	22.99
1967	8	23.43	23.77	25.51
	□	□	□	□
2014	55	30.22	30.80	33.85
2015	56	23.17	24.39	30.80
2016	57	20.86	21.43	24.39
2017	58	22.64	22.44	21.43
	h			\hat{y}_{T+h|T}
2018	1			22.44
2019	2			22.44
2020	3			22.44
2021	4			22.44
2022	5			22.44

```
fitted_vals
```

	unique_id	ds	y	SES	SES-lo-95	SES-lo-80	SES-hi-80	SES-hi-95
0	Algeria	1960-12-31	39.043	39.530	27.720	31.808	47.252	51.340
1	Algeria	1961-12-31	46.245	39.121	27.311	31.399	46.843	50.931
2	Algeria	1962-12-31	19.794	45.105	33.295	37.383	52.828	56.916
3	Algeria	1963-12-31	24.685	23.842	12.031	16.119	31.564	35.652
4	Algeria	1964-12-31	25.084	24.550	12.740	16.828	32.272	36.360
5	Algeria	1965-12-31	22.604	24.999	13.188	17.276	32.721	36.809
6	Algeria	1966-12-31	25.986	22.987	11.177	15.265	30.709	34.797
7	Algeria	1967-12-31	23.434	25.507	13.696	17.784	33.229	37.317
8	Algeria	1968-12-31	23.136	23.766	11.956	16.043	31.488	35.576
9	Algeria	1969-12-31	23.789	23.236	11.426	15.514	30.959	35.047
10	Algeria	1970-12-31	22.073	23.700	11.890	15.978	31.423	35.511
11	Algeria	1971-12-31	18.443	22.333	10.523	14.611	30.055	34.143

	unique_id	ds	y	SES	SES-lo-95	SES-lo-80	SES-hi-80	SES-hi-95
12	Algeria	1972-12-31	20.450	19.065	7.254	11.342	26.787	30.875
13	Algeria	1973-12-31	25.504	20.228	8.418	12.506	27.950	32.038
14	Algeria	1974-12-31	38.749	24.660	12.850	16.938	32.382	36.470
15	Algeria	1975-12-31	33.689	36.496	24.686	28.774	44.218	48.306
16	Algeria	1976-12-31	33.055	34.138	22.328	26.416	41.860	45.948
17	Algeria	1977-12-31	30.587	33.228	21.418	25.506	40.950	45.038
18	Algeria	1978-12-31	25.536	31.009	19.199	23.287	38.731	42.819
19	Algeria	1979-12-31	31.148	26.411	14.601	18.689	34.133	38.221
20	Algeria	1980-12-31	34.338	30.391	18.581	22.668	38.113	42.201
21	Algeria	1981-12-31	34.587	33.707	21.897	25.985	41.429	45.517
22	Algeria	1982-12-31	30.925	34.447	22.636	26.724	42.169	46.257
23	Algeria	1983-12-31	27.942	31.488	19.678	23.766	39.210	43.298
24	Algeria	1984-12-31	25.710	28.509	16.699	20.787	36.231	40.319
25	Algeria	1985-12-31	23.584	26.158	14.347	18.435	33.880	37.968
26	Algeria	1986-12-31	12.855	23.996	12.185	16.273	31.718	35.806
27	Algeria	1987-12-31	14.272	14.636	2.826	6.914	22.359	26.447
28	Algeria	1988-12-31	15.508	14.331	2.520	6.608	22.053	26.141
29	Algeria	1989-12-31	18.639	15.320	3.509	7.597	23.042	27.130
30	Algeria	1990-12-31	23.444	18.108	6.298	10.386	25.831	29.919
31	Algeria	1991-12-31	29.118	22.590	10.780	14.868	30.313	34.401
32	Algeria	1992-12-31	25.320	28.074	16.264	20.352	35.796	39.884
33	Algeria	1993-12-31	21.784	25.760	13.950	18.038	33.482	37.570
34	Algeria	1994-12-31	22.531	22.420	10.610	14.697	30.142	34.230
35	Algeria	1995-12-31	26.195	22.513	10.703	14.791	30.235	34.323
36	Algeria	1996-12-31	29.760	25.606	13.796	17.884	33.328	37.416
37	Algeria	1997-12-31	30.906	29.096	17.286	21.374	36.818	40.906
38	Algeria	1998-12-31	22.578	30.617	18.807	22.895	38.339	42.427
39	Algeria	1999-12-31	28.150	23.864	12.054	16.142	31.586	35.674
40	Algeria	2000-12-31	42.070	27.465	15.654	19.742	35.187	39.275
41	Algeria	2001-12-31	36.689	39.734	27.924	32.012	47.456	51.544
42	Algeria	2002-12-31	35.505	37.176	25.366	29.454	44.899	48.986
43	Algeria	2003-12-31	38.249	35.772	23.962	28.050	43.494	47.582
44	Algeria	2004-12-31	40.053	37.853	26.043	30.130	45.575	49.663
45	Algeria	2005-12-31	47.205	39.701	27.891	31.979	47.424	51.512
46	Algeria	2006-12-31	48.811	46.005	34.195	38.283	53.727	57.815
47	Algeria	2007-12-31	47.068	48.362	36.552	40.640	56.084	60.172
48	Algeria	2008-12-31	47.973	47.275	35.465	39.553	54.997	59.085

	Algeria	2009-12-31	35.372	47.862	36.551	40.639	55.584	59.652
50	Algeria	2010-12-31	38.445	37.369	25.559	29.647	45.091	49.179
51	Algeria	2011-12-31	38.788	38.273	26.462	30.550	45.995	50.083
52	Algeria	2012-12-31	36.891	38.706	26.895	30.983	46.428	50.516
53	Algeria	2013-12-31	33.210	37.181	25.371	29.459	44.903	48.991
54	Algeria	2014-12-31	30.488	33.845	22.035	26.123	41.567	45.655
55	Algeria	2015-12-31	23.172	31.025	19.214	23.302	38.747	42.835
56	Algeria	2016-12-31	20.872	24.428	12.617	16.705	32.150	36.238
57	Algeria	2017-12-31	22.632	21.441	9.631	13.719	29.163	33.251

```
plot_series(
    algeria_economy,
    pd.concat([fc, fitted_vals.drop(["y", "SES-lo-95", "SES-hi-95", "SES-lo-80", "SES-hi-80"], axi

    level=[80, 95],
    xlabel="Year [1Y]",
    ylabel="% of GDP",
    title="Exports: Algeria",
    rm_legend=False,
)
```

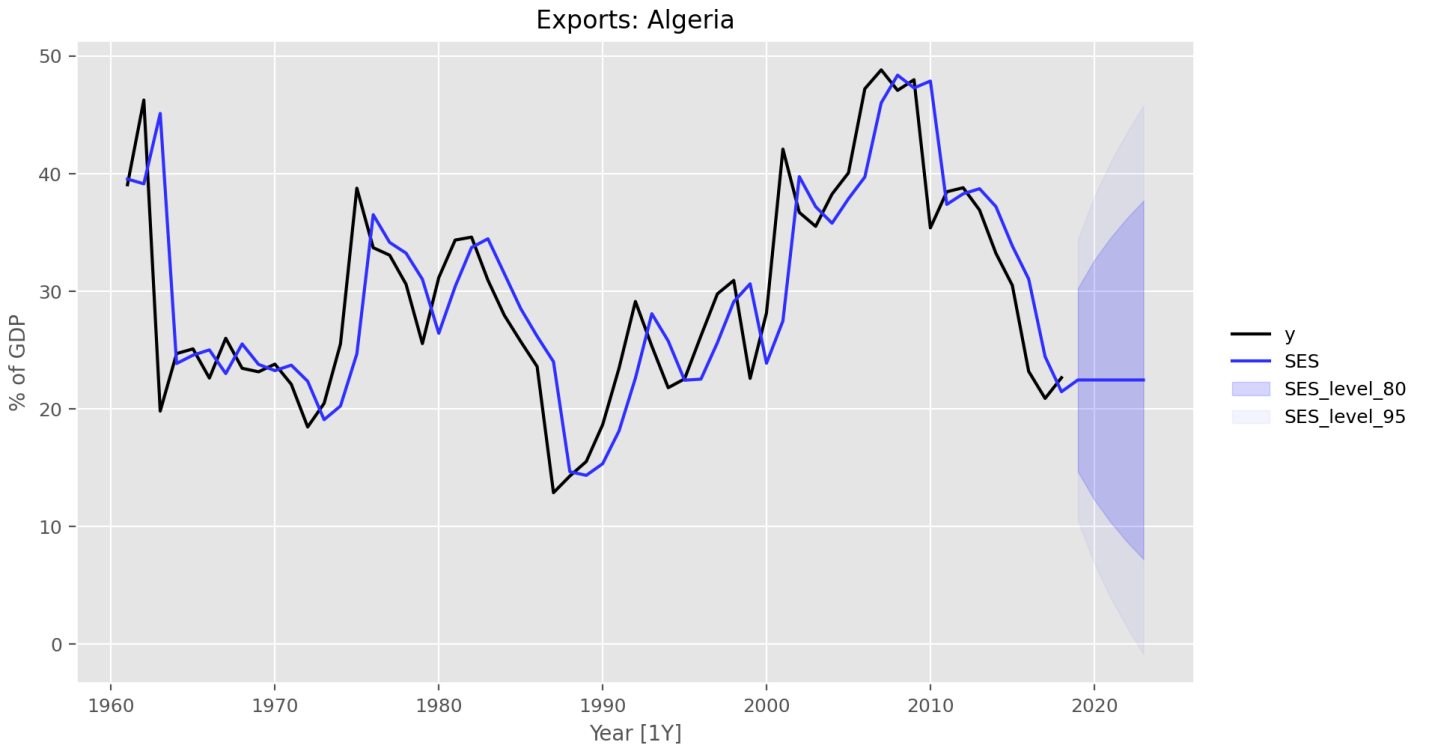

Figure 8.2: Simple exponential smoothing applied to exports from Algeria (1960–2017).

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 \alpha in this example is reflected in the large adjustment that takes place in the estimated level \ell_t at each time. A smaller value of \alpha 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

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): \begin{align*} \text{Forecast equation}&& \hat{y}_{t+h|t} &= \ell_{t} + hb_{t} \\ \text{Level equation} && \ell_{t} &= \alpha y_{t} + (1 - \alpha)(\ell_{t-1} + b_{t-1})\\ \text{Trend equation} && b_{t} &= \beta^*(\ell_{t} - \ell_{t-1}) + (1 -\beta^*)b_{t-1}, \end{align*} where \ell_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, \alpha is the smoothing parameter for the level, 0\le\alpha\le1, and \beta^* is the smoothing parameter for the trend, 0\le\beta^*\le1. (We denote this as \beta^* instead of \beta for reasons that will be explained in Section 8.5.)

As with simple exponential smoothing, the level equation here shows that \ell_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: Australian population

```
aus_economy = pd.read_csv("../data/aus_economy.csv", parse_dates=["ds"])
aus_economy["y"] = aus_economy["y"] / 1e6

plot_series(aus_economy, xlabel="Year [1Y]", ylabel="Millions", title="Australian population")
```

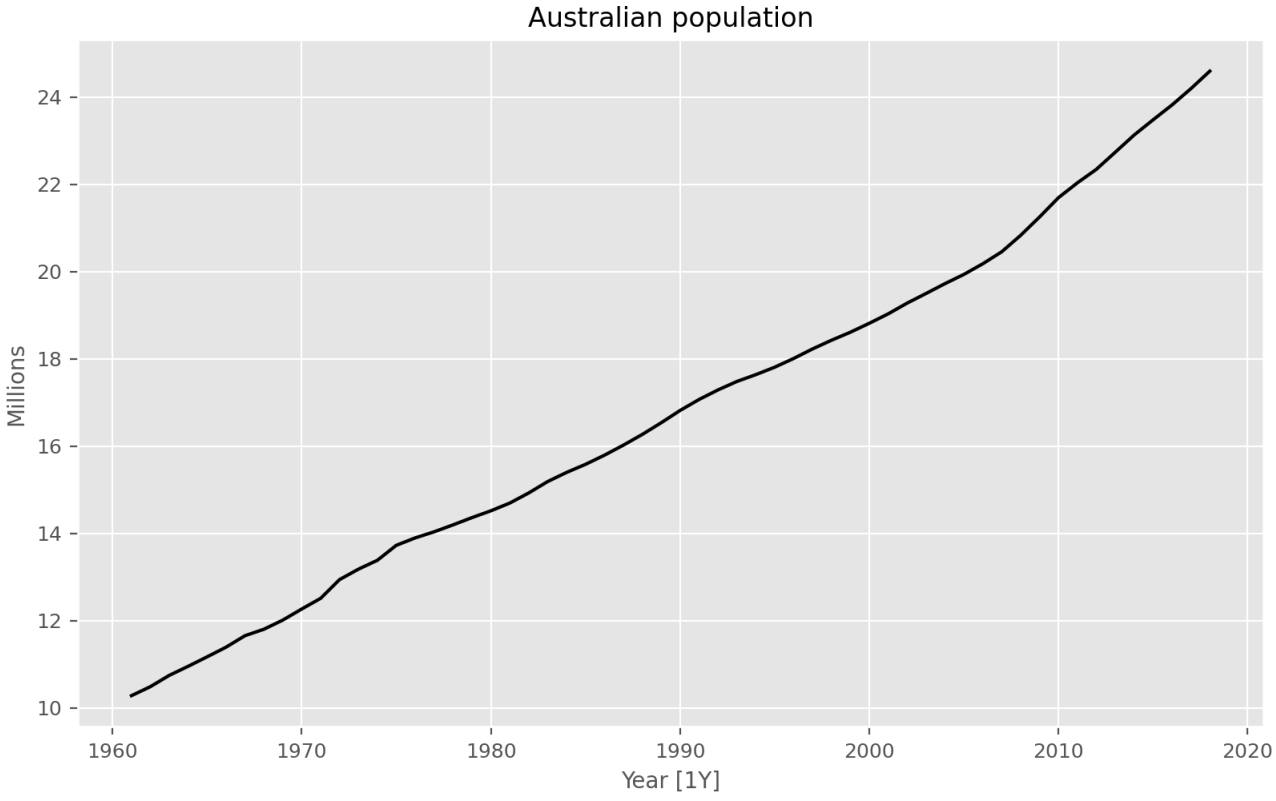

Figure 8.3: Australia's population, 1960-2017.

Figure 8.3 shows Australia's annual population from 1960 to 2017. We will apply Holt's method to this series. The smoothing parameters, \alpha and \beta^*, and the initial values \ell_0 and b_0 are estimated by minimising the SSE for the one-step training errors as in Section 8.1.

```
sf = StatsForecast(
    models=[AutoETS(season_length=1, model="AAN", alias="Holt")], freq="A"
)

fc = sf.forecast(df=aus_economy, h=10)
```

You can also use `Holt` from `statsforecast.models`, which is a wrapper for this particular case of the `AutoETS` class.

As before, to extract the optimal parameters, we first need to instantiate the `AutoETS` class and then call the `fit` method.

```
autoets = AutoETS(season_length=1, model="AAN", alias="Holt")
autoets = autoets.fit(y=aus_economy["y"].values)

params = np.round(autoets.model_["fit"][0], 4)
print("Optimal parameters:")
print("alpha:", params[0])
print("beta:", params[1])
print("Initial level:", params[2])
print("Initial trend:", params[3])
```

```
Optimal parameters:
alpha: 0.9999
beta: 0.2001
Initial level: 10.0509
Initial trend: 0.2264
```

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}^* = 0.20006. 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.

Table 8.2: Forecasting Australian annual population using Holt's linear trend method.

Year	Time	Observation	Level	Slope	Forecast
	t	y_t	l_t		\hat{y}_{t+1|t}
1959	0		10.05	0.22	
1960	1	10.27	10.27	0.22	10.50
1961	2	10.48	10.48	0.22	10.70
1962	3	10.74	10.74	0.23	10.97
1963	4	10.95	10.95	0.22	11.17
1964	5	11.16	11.16	0.22	11.39
1965	6	11.38	11.38	0.22	11.61
1966	7	11.65	11.65	0.23	11.88
	□	□	□	□	□
2014	55	23.47	23.47	0.35	23.52
2015	56	23.81	23.81	0.35	23.87
2016	57	24.19	24.19	0.35	24.21
2017	58	24.69	24.59	0.36	24.57
	h				\hat{y}_{T+h|T}
2018	1				24.95
2019	2				25.31
2020	3				25.67
2021	4				26.04
2022	5				26.40
2023	6				26.76
2024	7				27.12
2025	8				27.49
2026	9				27.85
2027	10				28.21

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 and 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 conjunction with the smoothing parameters \alpha and \beta^* (with values between 0 and 1 as in Holt's method), this method also includes a damping parameter 0<\phi<1: \begin{align*} \hat{y}_{t+h|t} &= \ell_{t} + (\phi+\phi^2 + \dots + \phi^{h})b_{t} \\ \ell_{t} &= \alpha y_{t} + (1 - \alpha)(\ell_{t-1} + \phi b_{t-1})\\ b_{t} &= \beta^*(\ell_{t} - \ell_{t-1}) + (1 -\beta^*)\phi b_{t-1}. \end{align*} If \phi=1, the method is identical to Holt's linear method. For values between 0 and 1, \phi dampens the trend so that it approaches a constant some time in the future. In fact, the forecasts converge to \ell_T+\phi b_T/(1-\phi) as h\rightarrow\infty for any value 0<\phi<1. This means that short-run forecasts are trended while long-run forecasts are constant.

In practice, \phi is rarely less than 0.8 as the damping has a very strong effect for smaller values. Values of \phi 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 \phi to a minimum of 0.8 and a maximum of 0.98.

Example: Australian Population (continued)

Figure 8.4 shows the forecasts for years 2018–2032 generated from Holt's linear trend method and the damped trend method.

```
sf = StatsForecast(
    models=[
        AutoETS(season_length=1, model="AAN", alias="Holt"),
        AutoETS(
            season_length=1,
            model="AAN",
            damped=True,
            phi=0.9,
            alias="Damped Holt's method",
        ),
    ],
    freq="Y",
)

fc = sf.forecast(df=aus_economy, h=15)

plot_series(aus_economy, fc, xlabel="Year [1Y]", ylabel="Millions", title="Australian population",
```

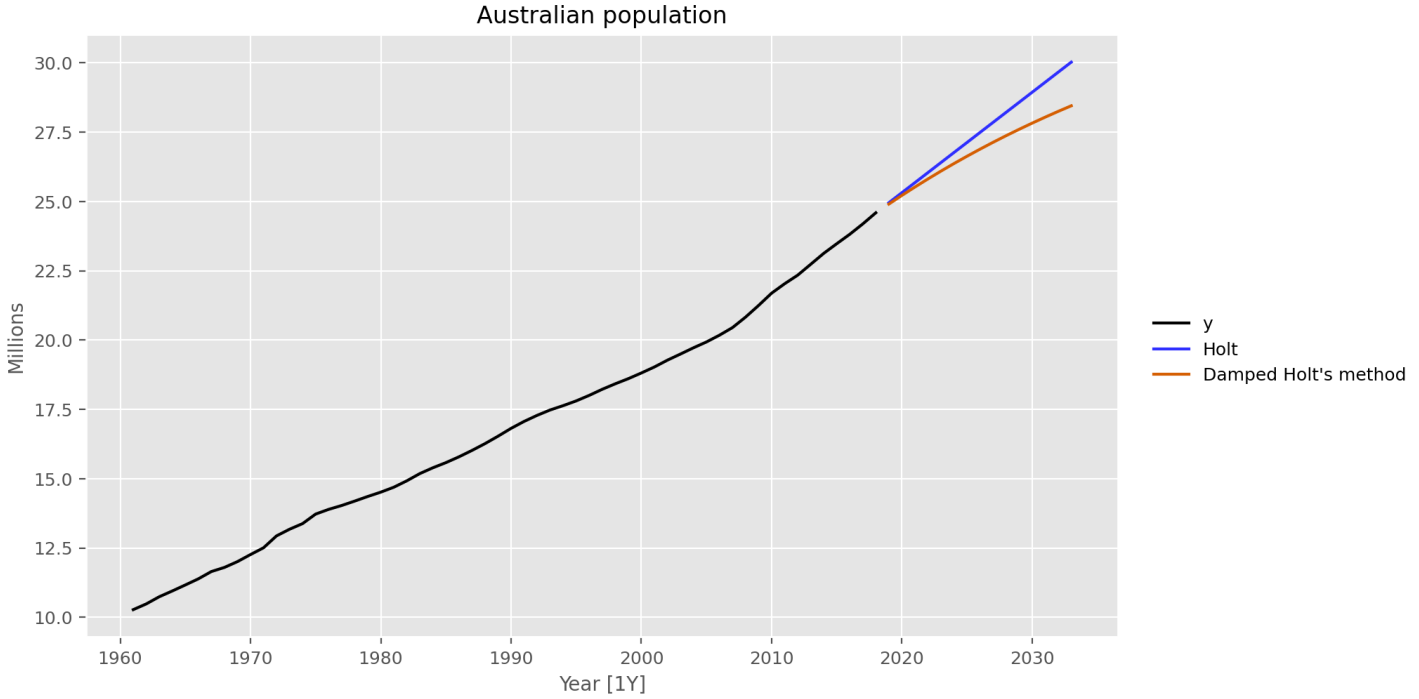

Figure 8.4: Forecasting annual Australian population (millions) over 2018-2032. For the damped trend method, \phi=0.90.

We have set the damping parameter to a relatively low number (\phi=0.90) to exaggerate the effect of damping for comparison. Usually, we would estimate \phi 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 = pd.read_csv("../data/www_usage.csv")

plot_series(www_usage, xlabel="Minute", ylabel="Number of users", title="Internet usage per minute
```

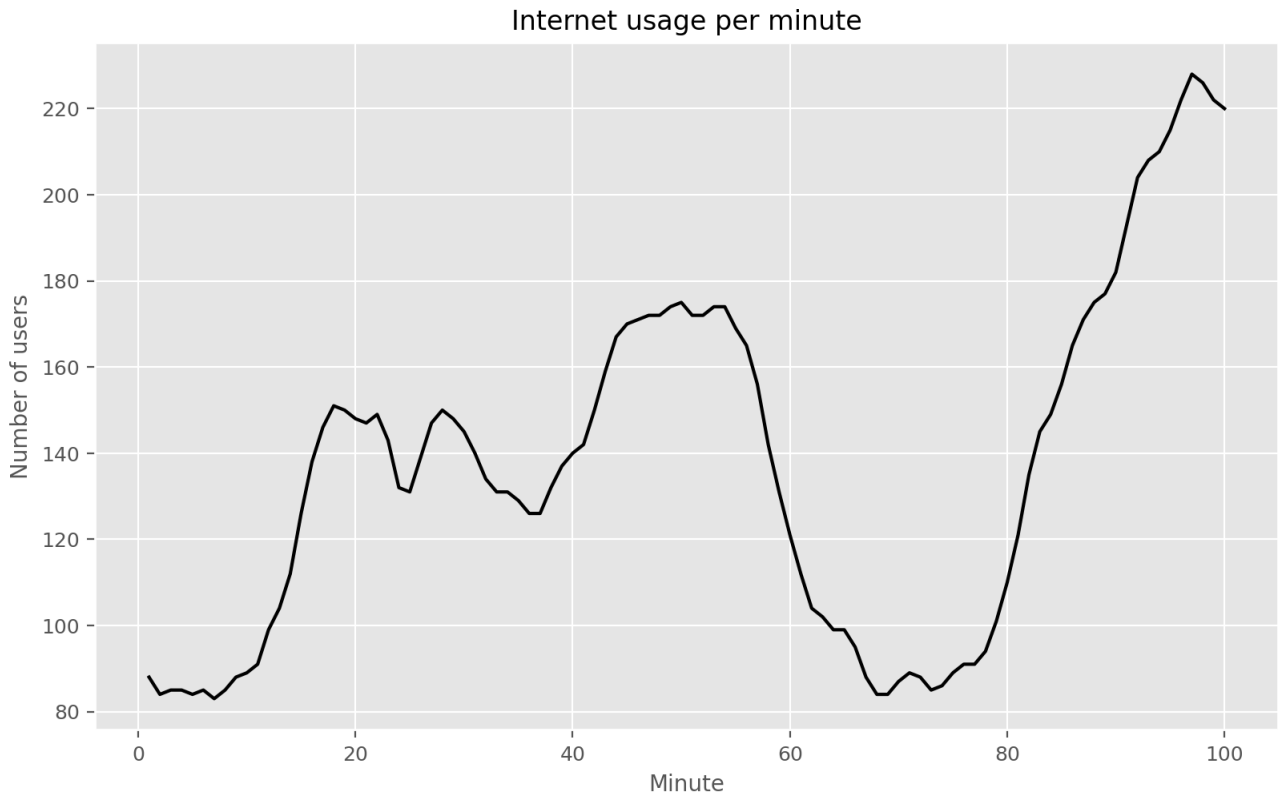

Figure 8.5: Users connected to the internet through a server.

We will use time series cross-validation to compare the one-step forecast accuracy of the three methods. We have set `step_size=1` to get the one-step forecasts and `n_windows=90` to cover most of the data.

```
sf = StatsForecast(
    models=[
        AutoETS(season_length=1, model="ANN", alias="SES"),
        AutoETS(season_length=1, model="AAN", alias="Holt"),
        AutoETS(season_length=1, model="AAN", damped=True, phi=0.9, alias="Damped"),
    ],
    freq=1,
)

cv = sf.cross_validation(df=www_usage, h=1, n_windows=90, step_size=1)

from utilsforecast.losses import mape as _mape

def mape(df, models, id_col = "unique_id", target_col = "y"):
    df_mape = _mape(df, models, id_col=id_col, target_col=target_col)
    df_mape.loc[:, df_mape.select_dtypes(include='number').columns] *= 100
    return df_mape

evaluate(
    cv.drop("cutoff", axis=1),
    metrics=[rmse, mae, mape, partial(mase, seasonality=1)],
    train_df=www_usage,
)
```

	unique_id	metric	SES	Holt	Damped
0	www_usage	rmse	6.072	5.086	5.037
1	www_usage	mae	4.849	4.181	4.057
2	www_usage	mape	3.588	3.164	3.058
3	www_usage	mase	1.072	0.924	0.896

Damped Holt's method is best whether you compare RMSE, MAE, MAPE, or MASE 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.

```
sf = StatsForecast(
    models=[AutoETS(season_length=1, model="AAN", damped=True, alias="Damped")], freq=1
)
```

To extract the parameters, we need to call the `AutoETS` method directly.

```
autoets = AutoETS(season_length=1, model="AAN", damped=True, alias="Damped")
autoets = autoets.fit(y=www_usage["y"].values)

params = np.round(autoets.model_["fit"][0],4)
print("Optimal parameters:")
print("Damped alpha:", params[0])
print("Damped beta:", params[1])
print("Damped phi:", params[2])
print("Initial level:", params[3])
print("Initial trend:", params[4])
```

```
Optimal parameters:
Damped alpha: 0.9999
Damped beta: 0.2
Damped phi: 0.9087
Initial level: 88.8688
Initial trend: -0.9231
```

The smoothing parameter for the slope is estimated to be 0.2, indicating that the trend updates smoothly instead of reacting strongly to immediate changes in the last minutes of internet usage. The value of \alpha is very close to one, indicating that the level component reacts strongly to each new observation.

```
forecast = sf.forecast(df=www_usage, h=10, level=[80, 95])
plot_series(www_usage, forecast, level=[80, 95], models=["Damped"], xlabel="Minute", ylabel="Numbe
```

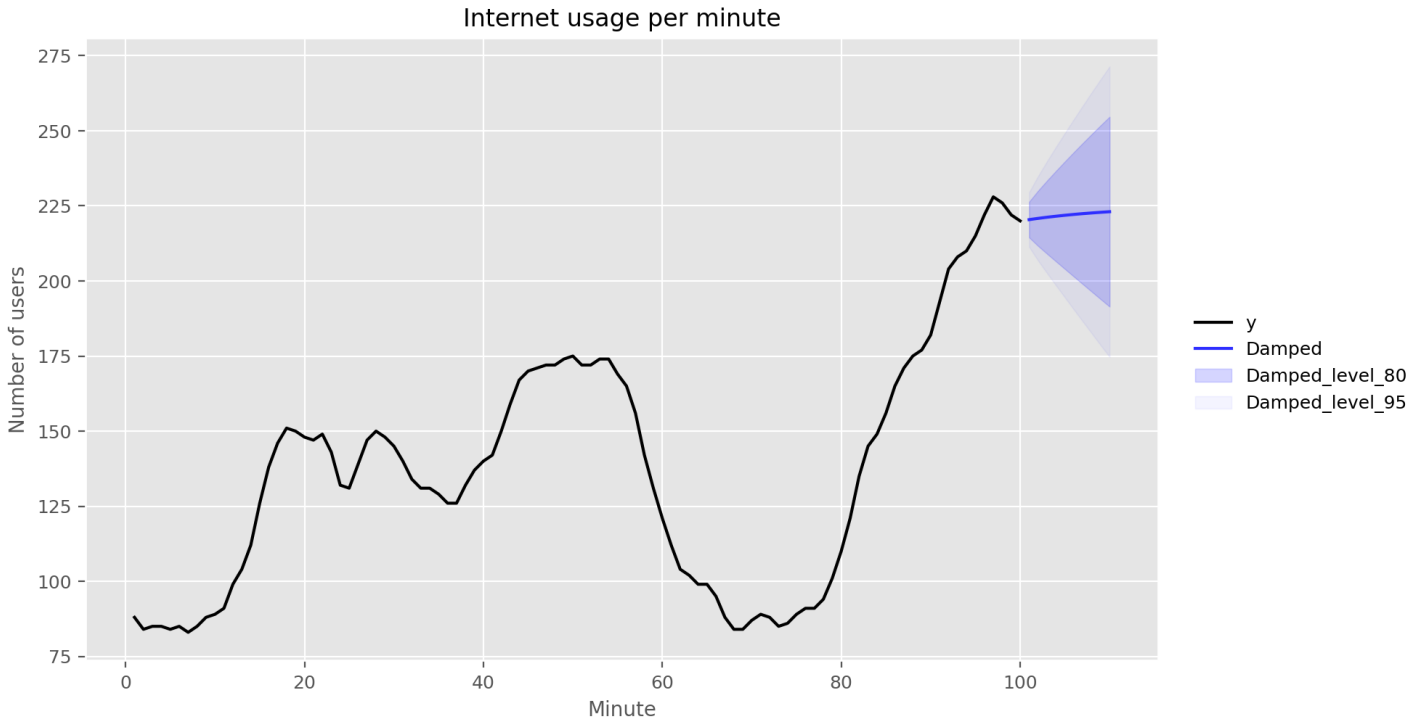

Figure 8.6: Forecasting internet usage: comparing forecasting performance of non-seasonal methods.

The resulting forecasts look sensible with a minimally increasing trend, 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 MAE and RMSE 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

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 \ell_t, one for the trend b_t, and one for the seasonal component s_t, with corresponding smoothing parameters \alpha, \beta^* and \gamma. 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: \begin{align*} \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}, \end{align*} 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 \ell_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\le\gamma^*\le1, which translates to

0\le\gamma\le 1-\alpha.

Holt-Winters' multiplicative method

The component form for the multiplicative method is: \begin{align*} \hat{y}_{t+h|t} &= (\ell_{t} + hb_{t})s_{t+h-m(k+1)} \\ \ell_{t} &= \alpha \frac{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 \frac{y_{t}}{(\ell_{t-1} + b_{t-1})} + (1 - \gamma)s_{t-m}. \end{align*}

Example: Domestic overnight trips in Australia

We apply Holt-Winters' method with both additive and multiplicative seasonality[2] 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.

```
tourism = pd.read_csv("../data/tourism.csv", parse_dates=["ds"])
aus_holidays = (
    tourism.query('Purpose == "Holiday"')
    .groupby(["Purpose", "ds"], as_index=False)["y"]
    .sum()
)
aus_holidays["y"] = aus_holidays["y"] / 1e3
aus_holidays.rename(columns={"Purpose": "unique_id"}, inplace=True)

sf = StatsForecast(
    models=[
        AutoETS(season_length=4, model="AAA", alias="additive"),
        AutoETS(season_length=4, model="MAM", alias="multiplicative"),
    ],
    freq="Q",
)
fc = sf.forecast(df=aus_holidays, h=12, fitted=True)
sf = sf.fit(aus_holidays)
plot_series(aus_holidays, fc, xlabel="Quarter", ylabel="Overnight trips (millions)", title="Austra
```

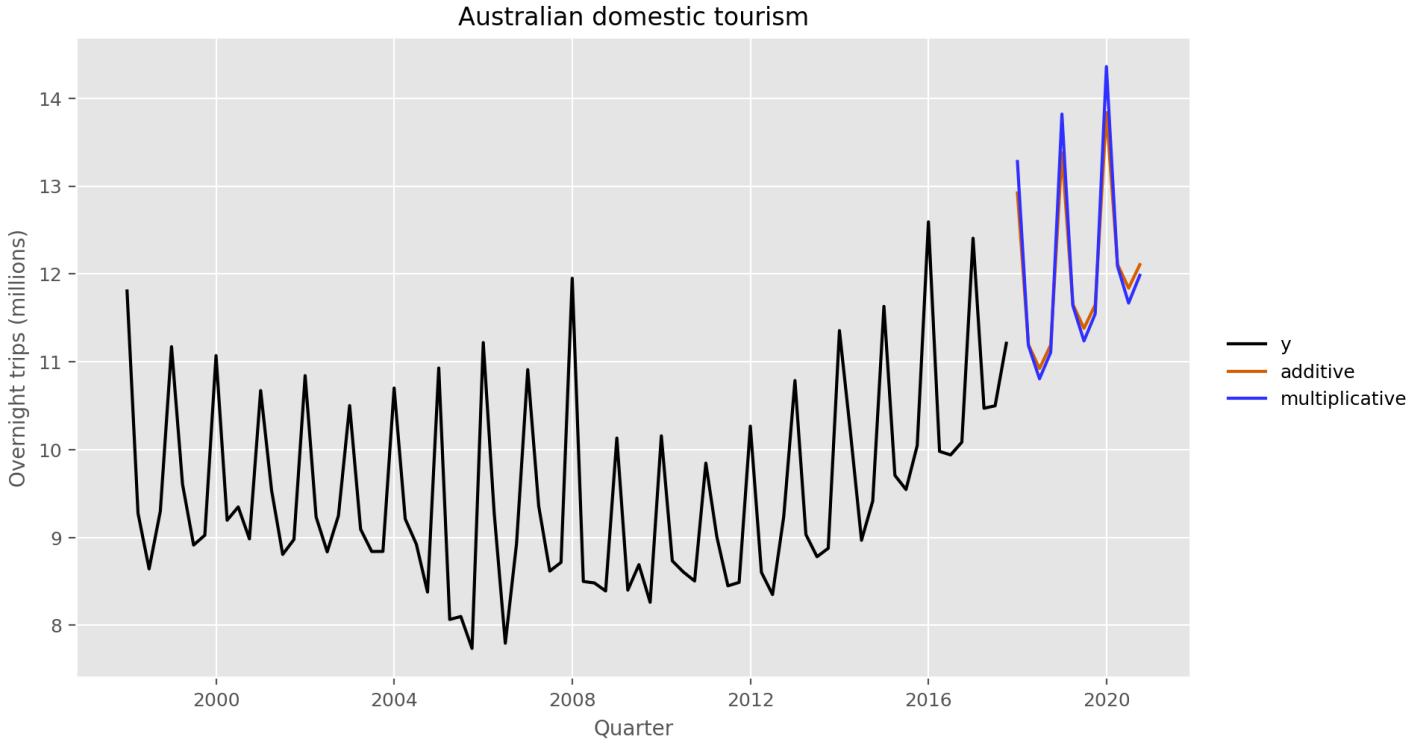

Figure 8.7: Forecasting domestic overnight trips in Australia using the Holt-Winters method with both additive and multiplicative seasonality.

You can also use `HoltWinters`, which is a wrapper for these particular cases of the `AutoETS` class. Simply specify the `error_type` as either `A` (additive) or `M` (multiplicative).

```
HoltWinters(season_length=4, error_type='additive', alias='additive')
HoltWinters(season_length=4, error_type='multiplicative', alias='multiplicative')
```

Table 8.3: Applying Holt-Winters' method with additive seasonality for forecasting domestic tourism in Australia. Notice that the additive seasonal component sums to approximately zero. The smoothing parameters are \alpha=0.2325, \beta^*=0.03, \gamma=0.0001 and \text{RMSE}=0.412.

Quarter	Time (t)	Observation	Level	Slope	Season	Forecast
	t	y_t	l_t	b_t	s_t	\hat{y}_{t+1|t}
1997 Q1	0				1.5	
1997 Q2	1				-0.3	
1997 Q3	2				-0.7	
1997 Q4	3		9.8	0.0	-0.5	
1998 Q1	4	11.8	9.9	0.0	1.5	11.3
1998 Q2	5	9.3	9.9	0.0	-0.3	9.6
1998 Q3	6	8.6	9.7	-0.0	-0.7	9.1
1998 Q4	7	9.3	9.8	0.0	-0.5	9.1
□	□	□	□	□	□	□
2017 Q1	80	12.4	10.9	0.1	1.5	12.5
2017 Q2	81	10.5	10.9	0.1	-0.3	10.5
2017 Q3	82	10.5	11.0	0.1	-0.7	10.1
2017 Q4	83	11.2	11.3	0.1	-0.5	10.4
	h					\hat{y}_{T+h|T}
2018 Q1	1					12.9
2018 Q2	2					11.2
2018 Q3	3					11.0
2018 Q4	4					11.2
2019 Q1	5					13.4
2019 Q2	6					11.7
2019 Q3	7					11.5
2019 Q4	8					11.7
2020 Q1	9					13.9
2020 Q2	10					12.2
2020 Q3	11					11.9
2020 Q4	12					12.2

Table 8.4: Applying Holt-Winters' method with multiplicative seasonality for forecasting domestic tourism in Australia. Notice that the multiplicative seasonal component sums to approximately m=4. The smoothing parameters are \alpha=0.239, \beta^*=0.0274, \gamma=0.0008 and \text{RMSE}=0.412.

Quarter	Time	Observation	Level	Slope	Season	Forecast
	t	y_t	l_t	b_t	s_t	\hat{y}_{t+1|t}
1997 Q1	0				1.2	
1997 Q2	1				1.0	
1997 Q3	2				0.9	
1997 Q4	3		10.0	-0.0	0.9	
1998 Q1	4	11.8	10.0	-0.0	1.2	11.3
1998 Q2	5	9.3	9.9	-0.0	1.0	9.5
1998 Q3	6	8.6	9.8	-0.0	0.9	9.0
1998 Q4	7	9.3	9.8	-0.0	0.9	9.0
□	□	□	□	□	□	□
2017 Q1	80	12.4	10.8	0.1	1.2	12.5
2017 Q2	81	10.5	10.9	0.1	1.0	10.5
2017 Q3	82	10.5	11.1	0.1	0.9	10.1
2017 Q4	83	11.2	11.3	0.1	0.9	10.4
	h					\hat{y}_{T+h|T}
2018 Q1	1					13.2
2018 Q2	2					11.1
2018 Q3	3					10.8
2018 Q4	4					11.1
2019 Q1	5					13.8
2019 Q2	6					11.6
2019 Q3	7					11.2
2019 Q4	8					11.5
2020 Q1	9					14.3
2020 Q2	10					12.0
2020 Q3	11					11.6
2020 Q4	12					11.9

The applications of both methods (with additive and multiplicative seasonality) are presented in Table 8.3 and Table 8.4 respectively. To compute the RMSE, we used the `evaluate` and `rmse` functions from `utilsforecast`. 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 additive method fits the data as well as the multiplicative method.

```
fitted_vals = sf.forecast_fitted_values()
evaluate(fitted_vals, metrics=[rmse])
```

	unique_id	metric	additive	multiplicative
0	Holiday	rmse	0.412	0.412

To extract the parameters, call the `AutoETS` method directly, specifying the desired model, either `AAA` or `MAM`.

```
autoets = AutoETS(season_length=4, model="AAA")
autoets = autoets.fit(y=aus_holidays["y"].values)

params = np.round(autoets.model_["fit"][0], 4)
print("Optimal parameters:")
print("alpha:", params[0])
print("beta:", params[1])
print("gamma:", params[2])
print("Initial level:", params[3])
print("Initial seasonal components:", params[4:len(params)])
```

```
Optimal parameters:
alpha: 0.2325
beta: 0.03
gamma: 0.0001
Initial level: 9.8372
Initial seasonal components: [-0.0238 -0.5386 -0.6916 -0.3051]
```

The estimated components for both models are plotted in Figure 8.8 using the `seasonal_decompose` function from `statsmodels`.

```
add_fitted_model = sf.fitted_[0, 0].model_
mult_fitted_model = sf.fitted_[0, 1].model_

fig, axes = plt.subplots(5, 2, figsize=(8, 8*14/12), sharex=True)
axes[0,0].plot(aus_holidays["y"].values)
axes[0,0].set_title("ETS(A,A,A) decomposition \n Additive seasonality")
axes[0,0].set_ylabel("Trips")

axes[1,0].plot(add_fitted_model["states"][:, 0])
axes[1,0].set_ylabel("Level")

axes[2,0].plot(add_fitted_model["states"][:, 1])
axes[2,0].set_ylabel("Slope")

axes[3,0].plot(add_fitted_model["states"][:, 2])
axes[3,0].set_ylabel("Season")

axes[4,0].plot(add_fitted_model["residuals"])
axes[4,0].set_ylabel("Residual")

axes[0,1].plot(aus_holidays["y"].values)
axes[0,1].set_title("ETS(M,A,M) decomposition \n Multiplicative seasonality")
axes[0,1].set_ylabel("Trips")


axes[1,1].plot(mult_fitted_model["states"][:, 0])
axes[1,1].set_ylabel("Level")

axes[2,1].plot(mult_fitted_model["states"][:, 1])
axes[2,1].set_ylabel("Slope")

axes[3,1].plot(mult_fitted_model["states"][:, 2])
axes[3,1].set_ylabel("Season")

axes[4,1].plot(mult_fitted_model["residuals"])
axes[4,1].set_ylabel("Residual")

for ax in axes.flat:
    ax.grid(True)
plt.show()
```

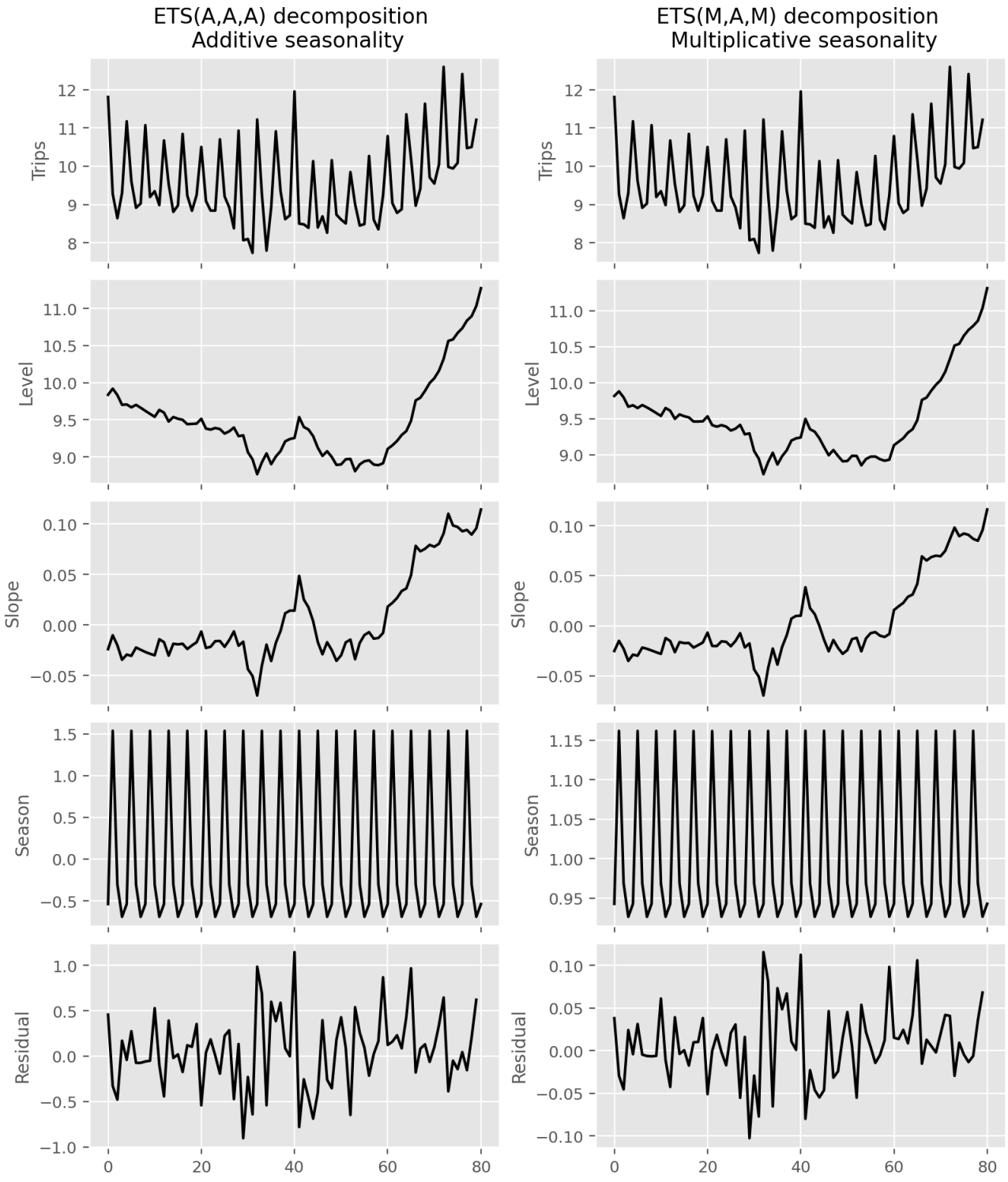

Figure 8.8: Estimated components for the Holt-Winters method with additive and multiplicative seasonal components.

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: \begin{align*} \hat{y}_{t+h|t} &= \left[\ell_{t} + (\phi+\phi^2 + \dots + \phi^{h})b_{t}\right]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 \frac{y_{t}}{(\ell_{t-1} + \phi b_{t-1})} + (1 - \gamma)s_{t-m}. \end{align*}

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.

```
pedestrian = pd.read_csv("../data/pedestrian.csv", parse_dates=["ds"])
sth_cross_ped = (
    pedestrian.query(
        'ds >= "2016-07-01" and ds <= "2016-07-31" and unique_id == "Southern Cross Station"'
    )
    .groupby(["unique_id", "ds"], as_index=False)["y"]
    .sum()
)
sth_cross_ped["y"] = sth_cross_ped["y"] / 1000

sf = StatsForecast(
    models=[AutoETS(season_length=7, model="MAM", damped=True, alias="HW Damped")],
    freq="D",
)

fc = sf.forecast(df=sth_cross_ped, h=14, level=[80, 95])
plot_series(sth_cross_ped, fc, level=[80, 95], xlabel="Date", ylabel="Pedestrians ('000)", title="
```

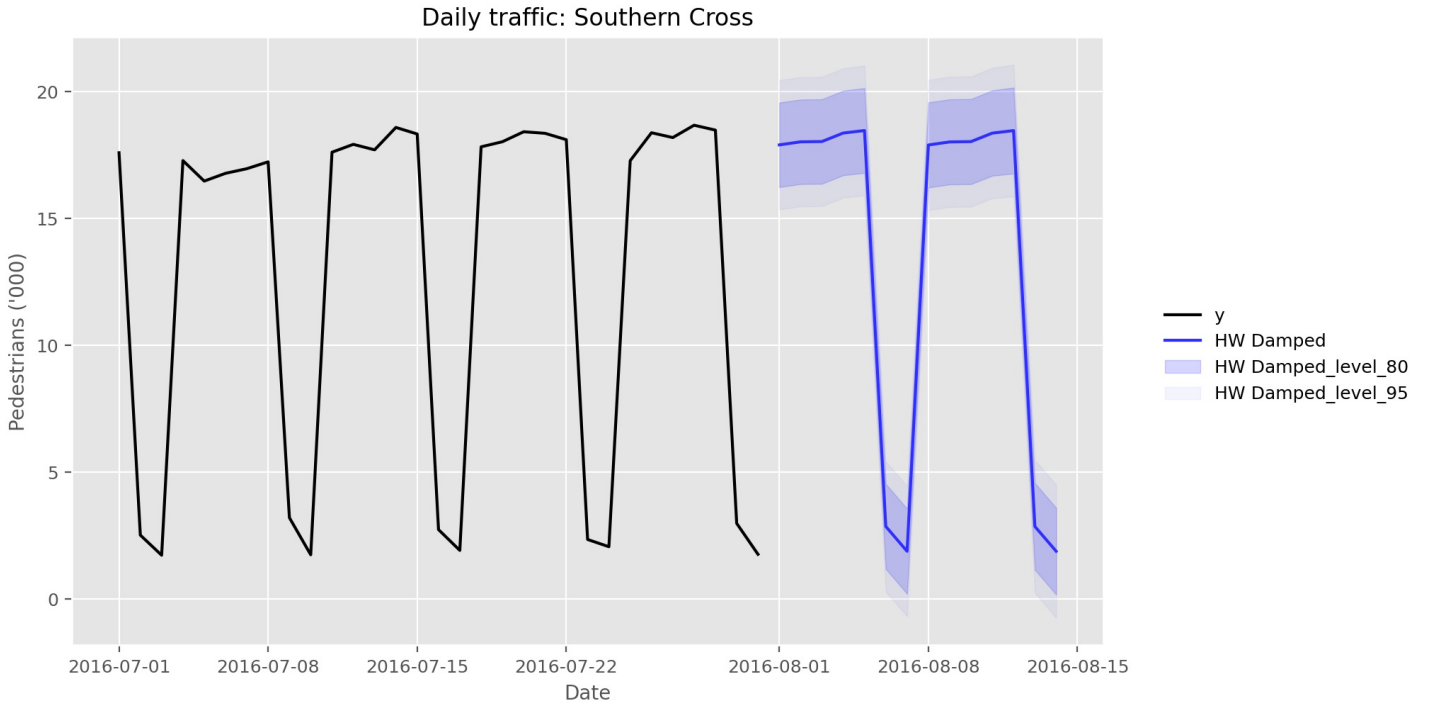

Figure 8.9: Forecasts of daily pedestrian traffic at the Southern Cross railway station, Melbourne.

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

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.

Table 8.5: A two-way classification of exponential smoothing methods.

Trend Component	Seasonal Component: N (None)	Seasonal Component: A (Additive)	Seasonal Component: M (Multiplicative)
N (None)	(N,N)	(N,A)	(N,M)
A (Additive)	(A,N)	(A,A)	(A,M)
A_d (Additive damped)	(A_d,N)	(A_d,A)	(A_d,M)

Some of these methods we have already seen using other names:

Table 8.6: Some common exponential smoothing methods.

Short hand	Method
(N,N)	Simple exponential smoothing
(A,N)	Holt's linear method
(A_d,N)	Additive damped trend method
(A,A)	Additive Holt-Winters' method
(A,M)	Multiplicative Holt-Winters' method
(A_d,A)	Holt-Winters' damped method

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 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 8.7 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.

Table 8.7: Formulas for recursive calculations and point forecasts. In each case, \ell_t denotes the series level at time t, b_t denotes the slope at time t, s_t denotes the seasonal component of the series at time t, and m denotes the number of seasons in a year; \alpha, \beta^*, \gamma and \phi are smoothing parameters, \phi_h = \phi+\phi^2+\dots+\phi^{h}, and k is the integer part of (h-1)/m."

Trend	Seasonal		
	N	A	M
N	$\hat{y}_{t+h\|t} = \ell_t$ $\ell_t = \alpha y_t + (1-\alpha)\ell_{t-1}$	$\hat{y}_{t+h\|t} = \ell_t + s_{t+h-m(k+1)}$ $\ell_t = \alpha(y_t - s_{t-m}) + (1-\alpha)\ell_{t-1}$ $s_t = \gamma(y_t - \ell_{t-1}) + (1-\gamma)s_{t-m}$	$\hat{y}_{t+h\|t} = \ell_t s_{t+h-m(k+1)}$ $\ell_t = \alpha(y_t/s_{t-m}) + (1-\alpha)\ell_{t-1}$ $s_t = \gamma(y_t/\ell_{t-1}) + (1-\gamma)s_{t-m}$
A	$\hat{y}_{t+h\|t} = \ell_t + hb_t$ $\ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1})$ $b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}$	$\hat{y}_{t+h\|t} = \ell_t + hb_t + s_{t+h-m(k+1)}$ $\ell_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1})$ $b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}$ $s_t = \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}$	$\hat{y}_{t+h\|t} = (\ell_t + hb_t)s_{t+h-m(k+1)}$ $\ell_t = \alpha(y_t/s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1})$ $b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}$ $s_t = \gamma(y_t/(\ell_{t-1} + b_{t-1})) + (1-\gamma)s_{t-m}$
A_d	$\hat{y}_{t+h\|t} = \ell_t + \phi_h b_t$ $\ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + \phi b_{t-1})$ $b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)\phi b_{t-1}$	$\hat{y}_{t+h\|t} = \ell_t + \phi_h b_t + s_{t+h-m(k+1)}$ $\ell_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + \phi b_{t-1})$ $b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)\phi b_{t-1}$ $s_t = \gamma(y_t - \ell_{t-1} - \phi b_{t-1}) + (1-\gamma)s_{t-m}$	$\hat{y}_{t+h\|t} = (\ell_t + \phi_h b_t)s_{t+h-m(k+1)}$ $\ell_t = \alpha(y_t/s_{t-m}) + (1-\alpha)(\ell_{t-1} + \phi b_{t-1})$ $b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)\phi b_{t-1}$ $s_t = \gamma(y_t/(\ell_{t-1} + \phi b_{t-1})) + (1-\gamma)s_{t-m}$

8.5 Innovations state space models for exponential smoothing

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 8.7 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(\cdot,\cdot,\cdot) 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: \text{Error} =\{A,M\}, \text{Trend}=\{N,A,A_d\} and \text{Seasonal} =\{N,A,M\}.

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

Recall the component form of simple exponential smoothing: \begin{align*} \text{Forecast equation} && \hat{y}_{t+1|t} & = \ell_{t}\\ \text{Smoothing equation} && \ell_{t} & = \alpha y_{t} + (1 - \alpha)\ell_{t-1}. \end{align*} If we re-arrange the smoothing equation for the level, we get the “error correction” form, \begin{align*} \ell_{t} %&= \alpha y_{t}+\ell_{t-1}-\alpha\ell_{t-1}\\ &= \ell_{t-1}+\alpha(y_{t}-\ell_{t-1})\\ &= \ell_{t-1}+\alpha e_{t}, \end{align*} where e_{t}=y_{t}-\ell_{t-1}=y_{t}-\hat{y}_{t|t-1} is the residual at time t.

The training data errors lead to the adjustment of the estimated level throughout the smoothing process for t=1,\dots,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 \ell_t is then the previous level \ell_{t-1} adjusted downwards. The closer \alpha is to one, the “rougher” the estimate of the level (large adjustments take place). The smaller the \alpha, 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 \sigma^2. A short-hand notation for this is e_t = \varepsilon_t\sim\text{NID}(0,\sigma^2); NID stands for “normally and independently distributed”.

Then the equations of the model can be written as \begin{align} y_t &= \ell_{t-1} + \varepsilon_t \\ \ell_t&=\ell_{t-1}+\alpha \varepsilon_t. \end{align} \tag{8.3} We refer to the first line of these equations as the *measurement* (or observation) equation and the second line 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, \varepsilon_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 \ell_{t-1}, the predictable part of y_t, and the error \varepsilon_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 \alpha is the same as for the methods discussed earlier. For example, \alpha governs the amount of change in successive levels: high values of \alpha allow rapid changes in the level; low values of \alpha lead to smooth changes. If \alpha=0, the level of the series does not change over time; if \alpha=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.)

ETS(M,N,N): simple exponential smoothing with multiplicative errors

In a similar fashion, we can specify models with multiplicative errors by writing the one-step-ahead training errors as relative errors \varepsilon_t = \frac{y_t-\hat{y}_{t|t-1}}{\hat{y}_{t|t-1}} where \varepsilon_t \sim \text{NID}(0,\sigma^2). Substituting \hat{y}_{t|t-1}=\ell_{t-1} gives y_t = \ell_{t-1}+\ell_{t-1}\varepsilon_t and e_t = y_t - \hat{y}_{t|t-1} = \ell_{t-1}\varepsilon_t.

Then we can write the multiplicative form of the state space model as \begin{align*} y_t&=\ell_{t-1}(1+\varepsilon_t)\\ \ell_t&=\ell_{t-1}(1+\alpha \varepsilon_t). \end{align*}

ETS(A,A,N): Holt's linear method with additive errors

For this model, we assume that the one-step-ahead training errors are given by \varepsilon_t=y_t-\ell_{t-1}-b_{t-1} \sim \text{NID}(0,\sigma^2). Substituting this into the error correction equations for Holt’s linear method we obtain \begin{align*} y_t&=\ell_{t-

1}+b_{t-1}+\varepsilon_t\\ \ell_t&=\ell_{t-1}+b_{t-1}+\alpha \varepsilon_t\\ b_t&=b_{t-1}+\beta \varepsilon_t, \end{align*} where for simplicity we have set \beta=\alpha \beta^*.

ETS(M,A,N): Holt's linear method with multiplicative errors

Specifying one-step-ahead training errors as relative errors such that \varepsilon_t=\frac{y_t-(\ell_{t-1}+b_{t-1})}{(\ell_{t-1}+b_{t-1})} and following an approach similar to that used above, the innovations state space model underlying Holt's linear method with multiplicative errors is specified as \begin{align*} y_t&=(\ell_{t-1}+b_{t-1})(1+\varepsilon_t)\\ \ell_t&=(\ell_{t-1}+b_{t-1})(1+\alpha \varepsilon_t)\\ b_t&=b_{t-1}+\beta(\ell_{t-1}+b_{t-1}) \varepsilon_t, \end{align*} where again \beta=\alpha \beta^* and \varepsilon_t \sim \text{NID}(0,\sigma^2).

Other ETS models

In a similar fashion, we can write an innovations state space model for each of the exponential smoothing methods of Table 8.7. Table 8.8 presents the equations for all of the models in the ETS framework.

Table 8.8: State space equations for each of the models in the ETS framework.

ADDITIVE ERROR MODELS

Trend	Seasonal: N	Seasonal: A	Seasonal: M
N	$y_t = \ell_{t-1} + \varepsilon_t$ $\ell_t = \ell_{t-1} + \alpha\varepsilon_t$	$y_t = \ell_{t-1} + s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1} + \alpha\varepsilon_t$ $s_t = s_{t-m} + \gamma\varepsilon_t$	$y_t = \ell_{t-1}s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1} + \alpha\varepsilon_t/s_{t-m}$ $s_t = s_{t-m} + \gamma\varepsilon_t/\ell_{t-1}$
A	$y_t = \ell_{t-1} + b_{t-1} + \varepsilon_t$ $\ell_t = \ell_{t-1} + b_{t-1} + \alpha\varepsilon_t$ $b_t = b_{t-1} + \beta\varepsilon_t$	$y_t = \ell_{t-1} + b_{t-1} + s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1} + b_{t-1} + \alpha\varepsilon_t$ $b_t = b_{t-1} + \beta\varepsilon_t$ $s_t = s_{t-m} + \gamma\varepsilon_t$	$y_t = (\ell_{t-1} + b_{t-1})s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1} + b_{t-1} + \alpha\varepsilon_t/s_{t-m}$ $b_t = b_{t-1} + \beta\varepsilon_t/s_{t-m}$ $s_t = s_{t-m} + \gamma\varepsilon_t/(\ell_{t-1} + b_{t-1})$
A_d	$y_t = \ell_{t-1} + \phi b_{t-1} + \varepsilon_t$ $\ell_t = \ell_{t-1} + \phi b_{t-1} + \alpha\varepsilon_t$ $b_t = \phi b_{t-1} + \beta\varepsilon_t$	$y_t = \ell_{t-1} + \phi b_{t-1} + s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1} + \phi b_{t-1} + \alpha\varepsilon_t$ $b_t = \phi b_{t-1} + \beta\varepsilon_t$ $s_t = s_{t-m} + \gamma\varepsilon_t$	$y_t = (\ell_{t-1} + \phi b_{t-1})s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1} + \phi b_{t-1} + \alpha\varepsilon_t/s_{t-m}$ $b_t = \phi b_{t-1} + \beta\varepsilon_t/s_{t-m}$ $s_t = s_{t-m} + \gamma\varepsilon_t/(\ell_{t-1} + \phi b_{t-1})$

MULTIPLICATIVE ERROR MODELS

Trend	Seasonal: N	Seasonal: A	Seasonal: M
N	$y_t = \ell_{t-1}(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1}(1 + \alpha\varepsilon_t)$	$y_t = (\ell_{t-1} + s_{t-m})(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1} + \alpha(\ell_{t-1} + s_{t-m})\varepsilon_t$ $s_t = s_{t-m} + \gamma(\ell_{t-1} + s_{t-m})\varepsilon_t$	$y_t = \ell_{t-1}s_{t-m}(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1}(1 + \alpha\varepsilon_t)$ $s_t = s_{t-m}(1 + \gamma\varepsilon_t)$
A	$y_t = (\ell_{t-1} + b_{t-1})(1 + \varepsilon_t)$ $\ell_t = (\ell_{t-1} + b_{t-1})(1 + \alpha\varepsilon_t)$ $b_t = b_{t-1} + \beta(\ell_{t-1} + b_{t-1})\varepsilon_t$	$y_t = (\ell_{t-1} + b_{t-1} + s_{t-m})(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1} + b_{t-1} + \alpha(\ell_{t-1} + b_{t-1} + s_{t-m})\varepsilon_t$ $b_t = b_{t-1} + \beta(\ell_{t-1} + b_{t-1} + s_{t-m})\varepsilon_t$ $s_t = s_{t-m} + \gamma(\ell_{t-1} + b_{t-1} + s_{t-m})\varepsilon_t$	$y_t = (\ell_{t-1} + b_{t-1})s_{t-m}(1 + \varepsilon_t)$ $\ell_t = (\ell_{t-1} + b_{t-1})(1 + \alpha\varepsilon_t)$ $b_t = b_{t-1} + \beta(\ell_{t-1} + b_{t-1})\varepsilon_t$ $s_t = s_{t-m}(1 + \gamma\varepsilon_t)$
A_d	$y_t = (\ell_{t-1} + \phi b_{t-1})(1 + \varepsilon_t)$ $\ell_t = (\ell_{t-1} + \phi b_{t-1})(1 + \alpha\varepsilon_t)$ $b_t = \phi b_{t-1} + \beta(\ell_{t-1} + \phi b_{t-1})\varepsilon_t$	$y_t = (\ell_{t-1} + \phi b_{t-1} + s_{t-m})(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1} + \phi b_{t-1} + \alpha(\ell_{t-1} + \phi b_{t-1} + s_{t-m})\varepsilon_t$ $b_t = \phi b_{t-1} + \beta(\ell_{t-1} + \phi b_{t-1} + s_{t-m})\varepsilon_t$ $s_t = s_{t-m} + \gamma(\ell_{t-1} + \phi b_{t-1} + s_{t-m})\varepsilon_t$	$y_t = (\ell_{t-1} + \phi b_{t-1})s_{t-m}(1 + \varepsilon_t)$ $\ell_t = (\ell_{t-1} + \phi b_{t-1})(1 + \alpha\varepsilon_t)$ $b_t = \phi b_{t-1} + \beta(\ell_{t-1} + \phi b_{t-1})\varepsilon_t$ $s_t = s_{t-m}(1 + \gamma\varepsilon_t)$

8.6 Estimation and model selection

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 \alpha, \beta, \gamma and \phi, and the initial states \ell_0, b_0, s_0,s_{-1},\dots,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 \phi is usually constrained further to prevent numerical difficulties in estimating the model. In the `StatsForecast` 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, 149–61). 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, AIC_{\text{c}} 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 \text{AIC} = -2\log(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 (AIC_\text{c}) is defined as \text{AIC}_{\text{c}} = \text{AIC} + \frac{2k(k+1)}{T-k-1}, and the Bayesian Information Criterion (BIC) is \text{BIC} = \text{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 `AutoETS()` function select the model by minimising the AICc (this is done by default).

```
autoets = AutoETS(season_length=4)
autoets = autoets.fit(y=aus_holidays["y"].values)
autoets.model_["method"]
```

```
'ETS(M,N,M)'
```

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

\begin{align*} y_t &= \ell_{t-1} s_{t-m} (1 + \epsilon_t) \\ \ell_t &= \ell_{t-1} (1 + \alpha \epsilon_t) \\ s_t &= s_{t-m} (1 + \gamma \epsilon_t) \\ \end{align*}

```
params = np.round(autoets.model_["fit"][0], 4)
print("Optimal parameters:")
print("alpha:", params[0])
print("gamma:", params[1])
print("Initial level for series:", params[2])
print("Initial seasonal components:", params[3:len(params)])
```

```
Optimal parameters:
alpha: 0.3612
gamma: 0.0001
Initial level for series: 9.7878
Initial seasonal components: [0.9432 0.9268 0.9683]
```

The parameter estimates are \hat\alpha= 0.3612 and \hat\gamma=0.0001. The output also returns the estimates for the initial states \ell_0 and 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 estimated components of the model, while Figure 8.12 shows point forecasts and prediction intervals generated from the model. The small values of \gamma indicate that the seasonal states change very little over time.

```
fig, axes = plt.subplots(4, 1, sharex=True)
axes[0].plot(aus_holidays["y"].values)
axes[0].set_title("ETS(M,N,M) decomposition \n\n Trips = (lag(level,1)+lag(season,4))*(1+residuals

axes[0].set_ylabel("Trips")
axes[1].plot(autoets.model_["states"][:,0])
axes[1].set_ylabel("Level")
axes[2].plot(autoets.model_["states"][:,1])
axes[2].set_ylabel("Season")
axes[3].plot(autoets.model_["residuals"])
axes[3].set_ylabel("Remainder")
plt.show()
```

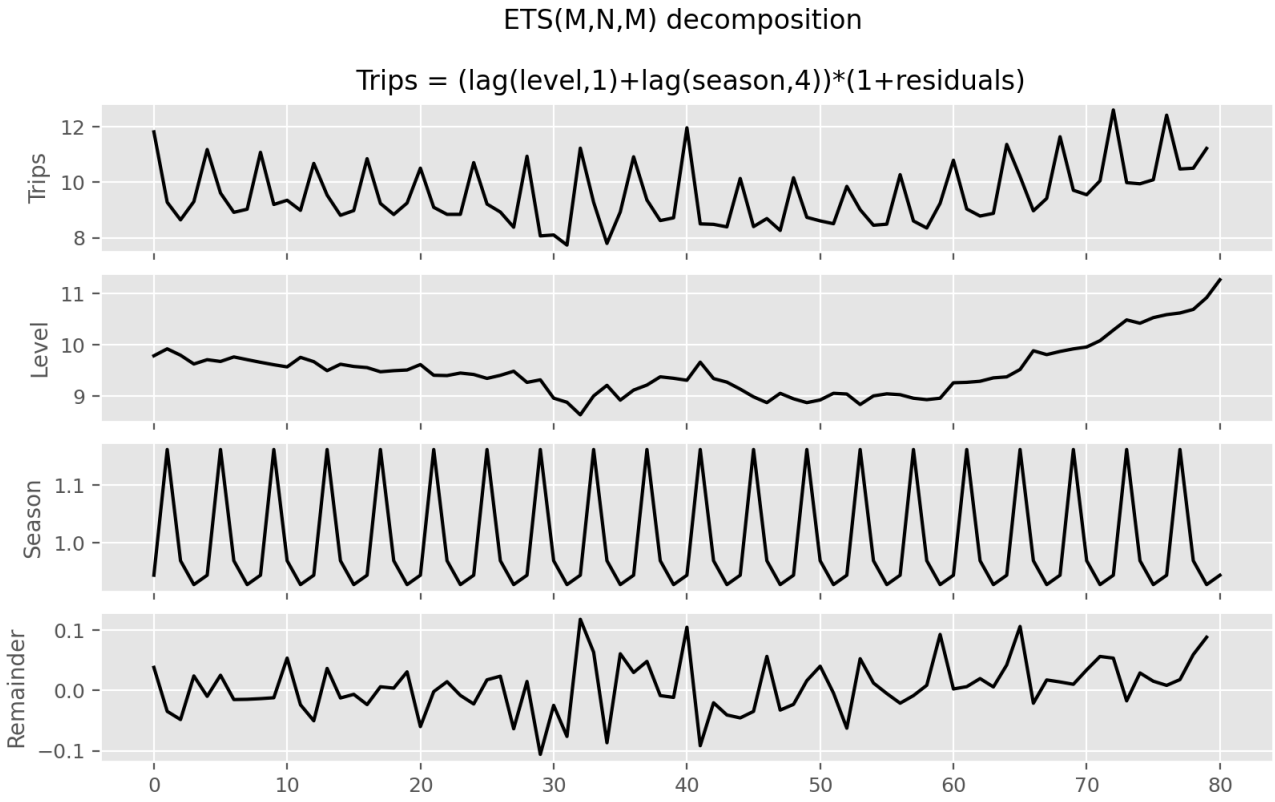

Figure 8.10: Graphical representation of the estimated states over time.

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{\epsilon}_t, while the regular residuals are defined as y_t-\hat{y}_{t|t-1}. They are plotted in Figure 8.11.

```
fig, axes = plt.subplots(2, 1, sharex=True)
axes[0].plot(autoets.model_["residuals"])
axes[0].set_ylabel("Innovation residuals")
axes[1].plot(autoets.model_["actual_residuals"])
axes[1].set_ylabel("Regular residuals")
axes[1].set_xlabel("Quarter")
plt.show()
```

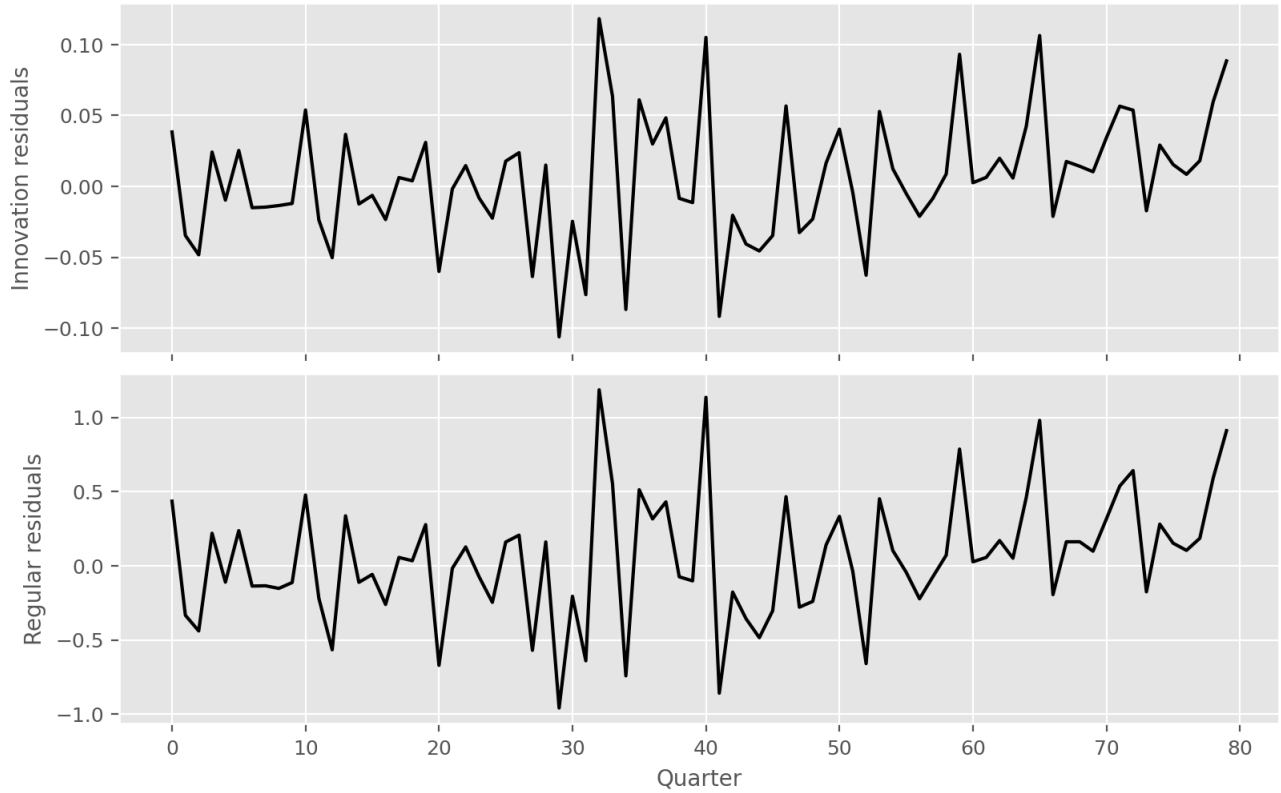

Figure 8.11: Residuals and one-step forecast errors from the ETS(M,N,M) model.

8.7 Forecasting with ETS models

Point forecasts can be obtained from the models by iterating the equations for t=T+1,\dots,T+h and setting all \varepsilon_t=0 for t>T.

For example, for model ETS(M,A,N), y_{T+1} = (\ell_T + b_T)(1+ \varepsilon_{T+1}). Therefore \hat{y}_{T+1|T}=\ell_{T}+b_{T}. Similarly, \begin{align*} y_{T+2} &= (\ell_{T+1} + b_{T+1})(1 + \varepsilon_{T+2})\\ &= \left[(\ell_T + b_T) (1+ \alpha\varepsilon_{T+1}) + b_T + \beta (\ell_T + b_T)\varepsilon_{T+1} \right] (1 + \varepsilon_{T+2}). \end{align*} Therefore, \hat{y}_{T+2|T}= \ell_{T}+2b_{T}, and so on. These forecasts are identical to the forecasts from Holt's linear method, and also to those from model ETS(A,A,N). Thus, 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` method from the `StatsForecast` package. This function will always return the means of the forecast distribution, even when they differ from these traditional point forecasts.

```
sf = StatsForecast(models=[AutoETS(season_length=4)], freq="Q")

fc = sf.forecast(df=aus_holidays, h=8, level=[80, 95])
plot_series(aus_holidays, fc, level=[80, 95], xlabel="Quarter", ylabel="Overnight trips (millions)")
```

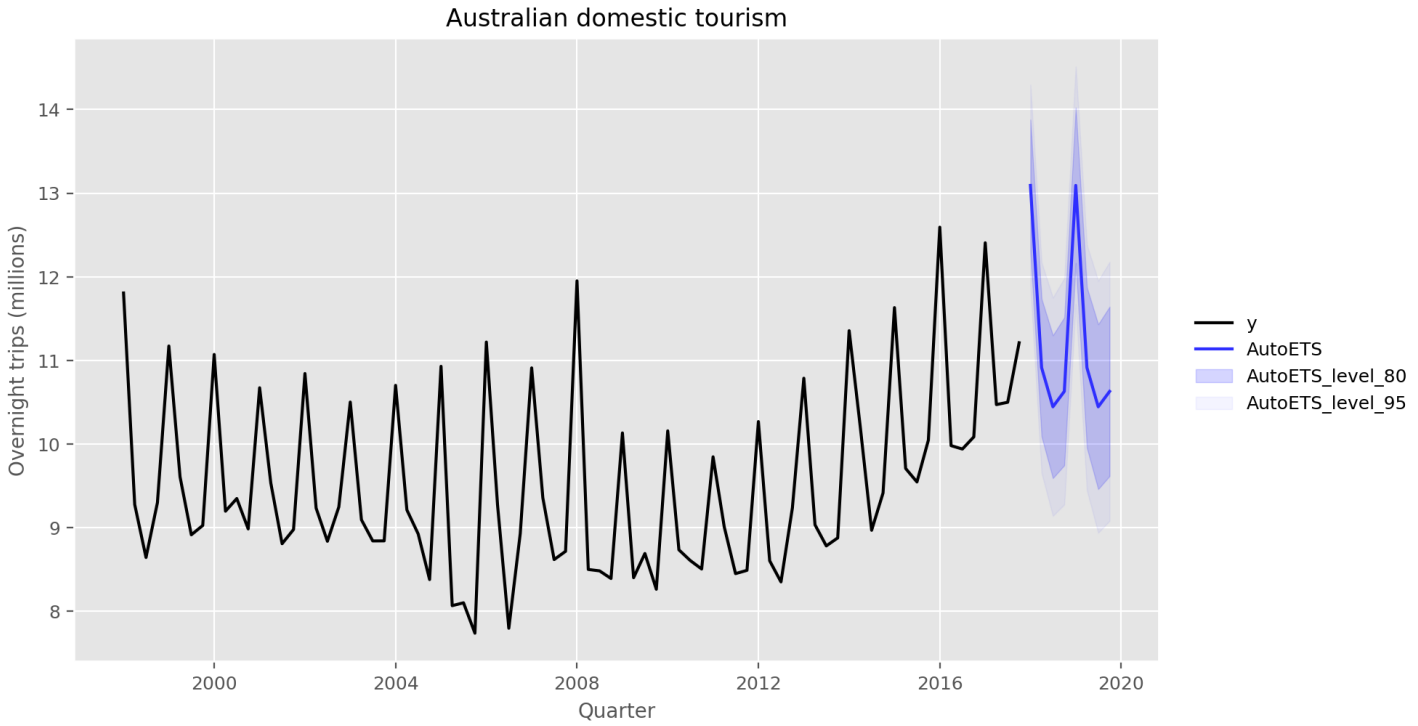

Figure 8.12: Forecasting Australian domestic overnight trips using an ETS(M,N,M) model.

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 most ETS models, a prediction interval can be written as \hat{y}_{T+h|T} \pm c \sigma_h where c depends on the coverage probability, and \sigma_h^2 is the forecast variance. Values for c were given in Table 5.1. For ETS models, formulas for \sigma_h^2 can be complicated; the details are given in Chapter 6 of Hyndman et al. (2008). In Table 8.9 we give the formulas for the additive ETS models, which are the simplest.

Table 8.9: Forecast variance expressions for each additive state space model, where \sigma^2 is the residual variance, m is the seasonal period, and k is the integer part of (h-1) /m (i.e., the number of complete years in the forecast period prior to time T+h).

Model	Forecast variance: σ_h^2
(A,N,N)	$\sigma_h^2 = \sigma^2(1+\alpha^2(h-1))$
(A,A,N)	$\sigma_h^2 = \sigma^2[1+(h-1)\{\alpha^2+\alpha\beta h+\frac{1}{6}\beta^2 h(2h-1)\}]$
(A,A_d,N)	$\sigma_h^2 = \sigma^2[1+\alpha^2(h-1)+\frac{\beta\phi h}{(1-\phi)^2}\{2\alpha(1-\phi)+\beta\phi\}-\frac{\beta\phi(1-\phi^h)}{(1-\phi)^2(1-\phi^2)}\{2\alpha(1-\phi^2)+\beta\phi(1+2\phi-\phi^h)\}]$
(A,N,A)	$\sigma_h^2 = \sigma^2[1+\alpha^2(h-1)+\gamma k(2\alpha+\gamma)]$
(A,A,A)	$\sigma_h^2 = \sigma^2[1+(h-1)\{\alpha^2+\alpha\beta h+\frac{1}{6}\beta^2 h(2h-1)\}+\gamma k\{2\alpha+\gamma+\beta m(k+1)\}]$
(A,A_d,A)	$\sigma_h^2 = \sigma^2\left[1+\alpha^2(h-1)+\gamma k(2\alpha+\gamma)+\frac{\beta\phi h}{(1-\phi)^2}\{2\alpha(1-\phi)+\beta\phi\}-\frac{\beta\phi(1-\phi^h)}{(1-\phi)^2(1-\phi^2)}\{2\alpha(1-\phi^2)+\beta\phi(1+2\phi-\phi^h)\}+\frac{2\beta\gamma\phi}{(1-\phi)(1-\phi^m)}\{k(1-\phi^m)-\phi^m(1-\phi^{mk})\}\right]$

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

8.8 Exercises

1. Consider the the number of pigs slaughtered in Victoria, available in the `aus_livestock` dataset.
    a. Use the `AutoETS()` function to estimate the equivalent model for simple exponential smoothing. Find the optimal values of \alpha and \ell_0, and generate forecasts for the next four months.

b. Compute a 95% prediction interval for the first forecast using \hat{y} \pm 1.96s where s is the standard deviation of the residuals. Compare your interval with the interval produced by `StatsForecast`.

2. Write your own function to implement simple exponential smoothing. The function should take arguments `y` (the time series), `alpha` (the smoothing parameter \alpha) and `level` (the initial level \ell_0). It should return the forecast of the next observation in the series. Does it give the same forecast as `SimpleExponentialSmoothing` from `StatsForecast`? Note that in `SimpleExponentialSmoothing`, you must specify the `alpha` parameter.

3. Modify your function from the previous exercise to return the sum of squared errors rather than the forecast of the next observation. Then find the optimal values of \alpha and \ell_0. Do you get the same values as the `AutoETS(model='ANN')` function?

4. Combine your previous two functions to produce a function that both finds the optimal values of \alpha and \ell_0, and produces a forecast of the next observation in the series.

5. Data set `global_economy` contains the annual Exports from many countries. Select one country to analyse.

   a. Plot the Exports series and discuss the main features of the data.
   b. Use an ETS(A,N,N) model to forecast the series, and plot the forecasts.
   c. Compute the RMSE values for the training data.
   d. Compare the results to those from an ETS(A,A,N) model. (Remember that the trended model is using one more parameter than the simpler model.) Discuss the merits of the two forecasting methods for this data set.
   e. Compare the forecasts from both methods. Which do you think is best?
   f. Calculate a 95% prediction interval for the first forecast for each model, using the RMSE values and assuming normal errors. Compare your intervals with those produced using `StatsForecast``.

6. Forecast the Chinese GDP from the `chinese_economy` data set using an ETS model. Note that the values for the GDP have already been transformed with a division by 10^{6}. for ease of use. Experiment with the various options in the `AutoETS()` function to see how much the forecasts change with damped trend, or with a Box-Cox transformation. Try to develop an intuition of what each is doing to the forecasts.

   [Hint: use a relatively large value of `h` when forecasting, so you can clearly see the differences between the various options when plotting the forecasts.]

7. Find an ETS model for the Gas data from `aus_production` and forecast the next few years. Why is multiplicative seasonality necessary here? Experiment with making the trend damped. Does it improve the forecasts?

8. Recall your retail time series data (from Exercise 7 in Section 2.10).

   a. Why is multiplicative seasonality necessary for this series?
   b. Apply Holt-Winters' multiplicative method to the data. Experiment with making the trend damped.
   c. Compare the RMSE of the one-step forecasts from the two methods. Which do you prefer?
   d. Check that the residuals from the best method look like white noise.
   e. Now find the test set RMSE, while training the model to the end of 2010. Can you beat the seasonal naïve approach from Exercise 7 in Section 5.11?

9. For the same retail data, try an STL decomposition applied to the Box-Cox transformed series, followed by ETS on the seasonally adjusted data. How does that compare with your best previous forecasts on the test set?

10. Compute the total domestic overnight trips across Australia from the `tourism` dataset.

    a. Plot the data and describe the main features of the series.
    b. Decompose the series using MSTL and obtain the seasonally adjusted data.
    c. Forecast the next two years of the series using an additive damped trend method applied to the seasonally adjusted data. (This can be specified using `damped=True`.)
    d. Forecast the next two years of the series using an appropriate model for Holt's linear method applied to the seasonally adjusted data (as before but without damped trend).
    e. Now use `AutoETS()` to choose a seasonal model for the data.
    f. Compare the RMSE of the ETS model with the RMSE of the models you obtained using MSTL decomposition. Which gives the better in-sample fits?
    g. Compare the forecasts from the three approaches? Which seems most reasonable?
    h. Check the residuals of your preferred model.

11. For this exercise use the quarterly number of arrivals to Australia from New Zealand, 1981 Q1 – 2012 Q3, from data set `aus_arrivals`.

    a. Make a time plot of your data and describe the main features of the series.
    b. Create a training set that withholds the last two years of available data. Forecast the test set using an appropriate model for Holt-Winters' multiplicative method.
    c. Why is multiplicative seasonality necessary here?
    d. Forecast the two-year test set using each of the following methods:

- an ETS model;
- an additive ETS model applied to a log transformed series;
- a seasonal naïve method;
- a MSTL decomposition applied to the log transformed data followed by an ETS model applied to the seasonally adjusted (transformed) data.

e. Which method gives the best forecasts? Does it pass the residual tests?
f. Compare the same four methods using time series cross-validation instead of using a training and test set. Do you come to the same conclusions?

12. a. Apply cross-validation techniques to produce 1 year ahead ETS and seasonal naïve forecasts for Portland cement production (from `aus_production`). Use a stretching data window with initial size of 5 years, and increment the window by one observation.
    b. Compute the MSE of the resulting 4-step-ahead errors. Comment on which forecasts are more accurate. Is this what you expected?

13. Compare `AutoETS()` and `SeasonalNaive()` from `StatsForecast` on the following five time series. Use a test set of three years to decide what gives the best forecasts.

    - Beer and bricks production from `aus_production`.
    - Cost of drug subsidies for diabetes (`ATC2 == "A10"`) and corticosteroids (`ATC2 == "H02"`) from `pbs`.
    - Total food retailing turnover for Australia from `aus_retail`.

14. a. Use `AutoETS()` to select an appropriate model for the following series: total number of trips across Australia using `tourism`, the closing prices for the four stocks in `gafa_stock`, and the lynx series in `pelt`. Does it always give good forecasts?

    b. Find an example where it does not work well. Can you figure out why?

15. Show that the point forecasts from an ETS(M,A,M) model are the same as those obtained using Holt-Winters' multiplicative method.

16. Show that the forecast variance for an ETS(A,N,N) model is given by \sigma^2\left[1+\alpha^2(h-1)\right].

17. Write down 95% prediction intervals for an ETS(A,N,N) model as a function of \ell_T, \alpha, h and \sigma, assuming normally distributed errors.

8.9 Further reading

- Two articles by Ev Gardner (Gardner 1985, 2006) provide a great overview of the history of exponential smoothing, and its many variations.
- A full book treatment of the subject providing the mathematical details is given by Hyndman et al. (2008).

8.10 Used modules and classes

StatsForecast

- `StatsForecast` class- Core forecasting engine for fitting and generating predictions
- `AutoETS` model - For automatic exponential smoothing, including SES and Holt's methods

UtilsForecast

- `evaluate` function - For evaluating forecast accuracy
- `plot_series` function - For visualizing time series data and forecasts
- `rmse`, `mae`, `mape`, `mase` metrics - For calculating forecast accuracy metrics

---

1. In some books it is called "single exponential smoothing". ↩

2. Our implementation uses maximum likelihood estimation as described in Section 8.6 while Holt and Winters originally minimized the sum of squared errors. For multiplicative seasonality, this will lead to slightly different parameter estimates. ↩

Chapter 9 ARIMA models →