Exploring Time Series Data

2

In Chapter 1 a variety of time series datasets were presented, and basic interpretations were made of the available information. It was shown how to extract time series data from various sources and how to import data into R. In the current chapter, the next step in the process will be considered; that is, it is now time to begin the early stages of analyzing time series data.

Analogous to the utilization of visual representations in traditional statistical analysis, plotting data will play a central role in the time series setting. As mentioned in Chapter 1, plots of time series can reveal information about the data that would not otherwise be obvious. While plotting the data is the important first step in any time series analysis, it is possible that such plots may not be sufficient to provide an enhanced understanding of valuable underlying information. Time series data often have patterns that are noisy. Another issue is that the data may be so dense that they appear as a compact cloud of points and may hide or mask any possible patterns that exist. In this chapter, strategies will be discussed that assist in understanding data and forecasting future values.

2.1 UNDERSTANDING AND VISUALIZING DATA

A first step in any statistical analysis is understanding the data. For example, it is informative to calculate summary statistics, such as the mean, median, standard deviation, range, quartiles, etc. While these statistics can be calculated for time series data, there are many methods and statistics that are designed specifically for understanding the way time series data evolve in time. In Chapter 1 we discussed data that were cyclic, trending, wandering, and seasonal or a combination of these types. We also noted that viewing data over small snippets of time is helpful to better visualize the finer details. The types of behavior discussed above are often accompanied by unexpected spikes and dips and general noisiness.

It is often the case that such noisy behavior tends to hide the underlying fundamental patterns in the time series. In this section we will discuss methods designed specifically for enhancing the understanding of the data. These methods include smoothing methods for "smoothing out" the anomalies that obscure the "big-picture". Also, given a time series that contains, say seasonal and trending behavior, it may be that the repetitive seasonal behavior is covering up some important trending movement in the data that can be seen better after removing the seasonal effects. Seasonal adjustment is a widely used method in time series analysis. For example, many of the datasets in the website FRED discussed in Chapter 1 are seasonally adjusted, or the website provides an option to download either seasonally adjusted or non-seasonally adjusted data.

2.1.1 Smoothing Time Series Data

Several methods exist for "smoothing out" the noisy (possibly unimportant) behavior from a time series so that an important underlying "signal" can be better understood. We begin by discussing the most basic of smoothing methods: the *centered moving average smoother*.

DOI: 10.1201/9781003089070-2

2.1.1.1 Smoothing Data Using a Centered Moving Average Smoother

The centered moving average smoother is a method for replacing data values in a time series with an average of data values surrounding (and including) that data point. For example, a centered moving average smoother of order three replaces a data value x_t at time t with $s_t = (x_{t-1} + x_t + x_{t+1})/3$. That is, the average value is assigned to the middle (2nd) time point. Therefore, a centered moving average smoother of order three cannot be assigned to the first or last time point of a time series. It follows that the higher the order, the more values will be missing at the beginning and end of the smoothed dataset. For a 3rd-order centered moving average, the averaging formula moves along the time series data set, considering three consecutive data values together until arriving at the last three time points.

Definition 2.1: Centered Moving Average Smoother
Let $x_t, t = 1,\ldots,n$ be a set of time series data. The centered moving average smoother is defined as follows:

Case* 1**: ***m is an odd number: Let $k = (m-1)/2$. For $k < t < n-k$, the smoothed data value, s_t, at time t is given by

$$s_t = \frac{1}{m}\sum_{i=t-k}^{t+k} x_i \tag{2.1}$$

Case* 2**: ***m is an even number: Let $k = m/2$. For $k < t < n-k$, the smoothed data value, s_t, at time t is given by

$$s_t = \frac{x_{t-k}}{2m} + \frac{1}{m}\sum_{i=t-k+1}^{t+k-1} x_i + \frac{x_{t+k}}{2m} \tag{2.2}$$

Examples:

(a) For a 5th-order centered moving average at times $2 < t < n-2$, we have
$$s_t = (x_{t-2} + x_{t-1} + x_t + x_{t+1} + x_{t+2})/5. \tag{2.3}$$

(b) A 4th-order moving average smoother at times $2 < t < n-2$, is given by
$$s_t = \frac{x_{t-2}}{8} + \frac{x_{t-1} + x_t + x_{t+1}}{4} + \frac{x_{t+2}}{8}$$

The centered moving average smoother has two basic uses:

(1) Smoothing designed to remove (potentially meaningless) fluctuations from the data.
(2) Removing cyclic behavior from seasonal or other cyclic data with fixed cycle lengths.

Example 2.1 shows the use of centered moving average smoothing for the purpose of detecting or better understanding underlying, fundamental signals in the data.

Example 2.1 Smoothing the Tesla and DFW Temperature Data
The Tesla stock prices from January 1, 2020 through April 30, 2021 are shown in Figure 1.19. We discussed the fact that there has been a steady increase until early 2021 at which time the price leveled off and decreased. However, there is considerable day-to-day fluctuation, especially in 2021. Figure 1.20(b) shows the DFW average annual temperatures from 1900 through 2020. There we see considerable year-to-year fluctuation but somewhat of an increase beginning in about 2000.

Figures 2.1(a) and (d) are plots of the Tesla and DFW temperature data, respectively. By use of the centered moving average smoother, the day-to-day fluctuations are smoothed out; note that the 8th order produces more smoothing than the 3rd order. The fundamental behavior including the leveling off and decline in early 2021 are more clearly seen by minimizing the noisy day-to-day changes. The effect of smoothing is more obvious in the DFW temperature data. Year-to-year fluctuation is quite dramatic in the original dataset in Figure 2.1(d). A 3rd-order smoothing helps some, but the 8th-order smoothing shown in Figure 2.1(f) clearly shows an almost level behavior from 1900 through about 1985. Since that time there has been a rise that may have leveled off in the last few years. Particularly noticeable in Figure 2.1(f) is that the smoothing has removed the extremes. Specifically, the extremely high temperatures in 2012, 2016, and 2017 are moderated by the lower temperatures in surrounding years.

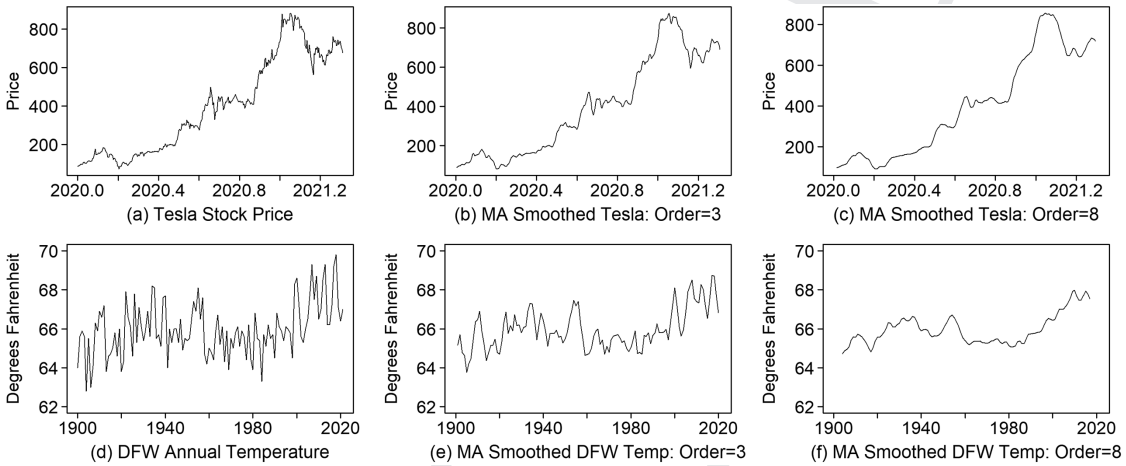


FIGURE 2.1 (a) Tesla stock prices, (b) and (c) data in (a) after applying 3rd- and 8th-order moving average smoothers, respectively. (d) DFW annual temperature data, (e) and (f) data in (d) after applying 3rd- and 8th-order moving average smoothers, respectively.

The ***tswge*** function **`ma.smooth.wge`** performs a centered moving average smoothing procedure on a set of data. There are two ways to present smoothed data, and we have used these two methods in Figures 2.1 and 2.2. Suppose **`x`** is a time series dataset; then the code

```
x.s=ma.smooth.wge(x,order=5,plot=TRUE)
```

plots the data in **`x`** and overlays it with a plot of the smoothed data (which resides in **`x.s$smooth`**) . The command

```
x.s=ma.smooth.wge(x,order=5,plot=FALSE)
```

simply performs the smoother and retains the smoothed data in **`x.s$smooth`**. This method was used in Figures 2.1(a)−(c) because the smoothed data were difficult to distinguish from the original data in an overlay plot. Figures 2.1(a)−(c) were obtained using the code below which performs centered moving average smoothers of orders 3 and 8 on the dataset **`tesla`**. [1]

```
data(tesla)
tesla.3=ma.smooth.wge(tesla,order=3,plot=FALSE)
tesla.8=ma.smooth.wge(tesla,order=8,plot=FALSE)
plot(tesla)
```

1 Code such as that given above often produces the plots without careful axis labeling and numbering. For example, **`tesla`** is a ***ts*** object (and the horizontal axis of a plot reflects the given dates). However, **`tesla.3$smooth`** is not a ***ts*** object. Creating a plot like Figure 2.1(b) would first require converting **`tesla.3$smooth`** to a ***ts*** file. Such steps will typically not be given in short clips of code throughout the book.

```
plot(tesla.3$smooth)
plot(tesla.8$smooth)
# the plots based on the tswge ts object dfw.yr are obtained analogously
```

Example 2.2 Smoothing a simulated cosine + trend + noise dataset

Figure 2.2(a) shows a cosine curve (c_t)with period 10, Figure 2.2(b) is a trend line $\ell_t = -2 + .05t$, and Figure 2.2(c) is the sum of the two forming the full underlying signal, $c_t + \ell_t$, which has the appearance of "ascending cosine curves". Figure 2.2(d) is a random noise sequence, z_t, and Figure 2.2(e) is the addition of the noise in Figure 2.2(d) to the underlying signal in Figure 2.2(c) to form the "simulated observed signal", x_t, in Figure 2.2(e) given by $x_t = c_t + \ell_t + z_t$.The "observed data" in Figure 2.2(e) are quite noisy and have the appearance of "noisy" data trending upward, with some hint of cyclic behavior. Figure 2.2(f) shows the results of applying a 5th-order centered moving average smoother to the data in Figure 2.2(e). The output from the 5th-order smoother, shown in Figure 2.2(f) superimposed on the data, does a good job of removing the noise and revealing the signal (cosine + trend line). The plots in Figures 2.2(d)−(f) are obtained using the code[2]

```
set.seed(6946)
t=1:60
cosine=cos(2*pi*t/10)
line=-2+.05*t
z=rnorm(n=60,sd=1)
x=cosine+line+z
plot(x)
ma.smooth.wge(x,order=5)
```

2.1.1.2 Other Methods Available for Smoothing Data

A topic closely related to smoothing is the more general topic of filtering, which will be discussed in more detail in Chapter 4. In that chapter, we will introduce the Butterworth filter which is a popular filtering technique among engineers and scientists. Other smoothing methods includes LOESS (locally estimated scatterplot smoothing), (Cleveland and Devlin (1988). In Section 4.3 we introduce the *first difference filter*. While the methodologies behind the filtering and smoothing techniques are sometimes different, they share the common goal of either identifying or removing an underlying signal. In Chapter 2 we will use filtering to remove seasonal components to isolate a trend. As will be discussed in the following chapters, smoothing (and more generally filtering) is also useful for transforming data to a form suitable for forecasting.

2.1.1.3 Moving Average Smoothing versus Aggregating

In Chapter 1 we noted that the new annual sunspot2.0 data are available from 1700 to the present and monthly from 1749 through the most recent full year.[3] The annual and monthly datasets are available in ***tswge*** as **`sunspot2.0`** and **`sunspot2.0.month`**, respectively. Figure 2.3(a) is a plot of **`sunspot2.0`** containing data for the 272 years between 1749 and 2020. Figure 2.3(b) is a plot of **`sunspot2.0.month`** containing 3264 monthly observations between 1749 and 2020. Specifically, **`sunspot2.0.month`** is a ***ts***

2 The Base R function **`rnorm`** generates *n* random normals. The default values are **`mean=0`** and **`sd=1`**. Also, note that **`plot=TRUE`** is the default option in **`ma.smooth.wge.`**

3 As of the writing of this book, the latest full year is 2020.

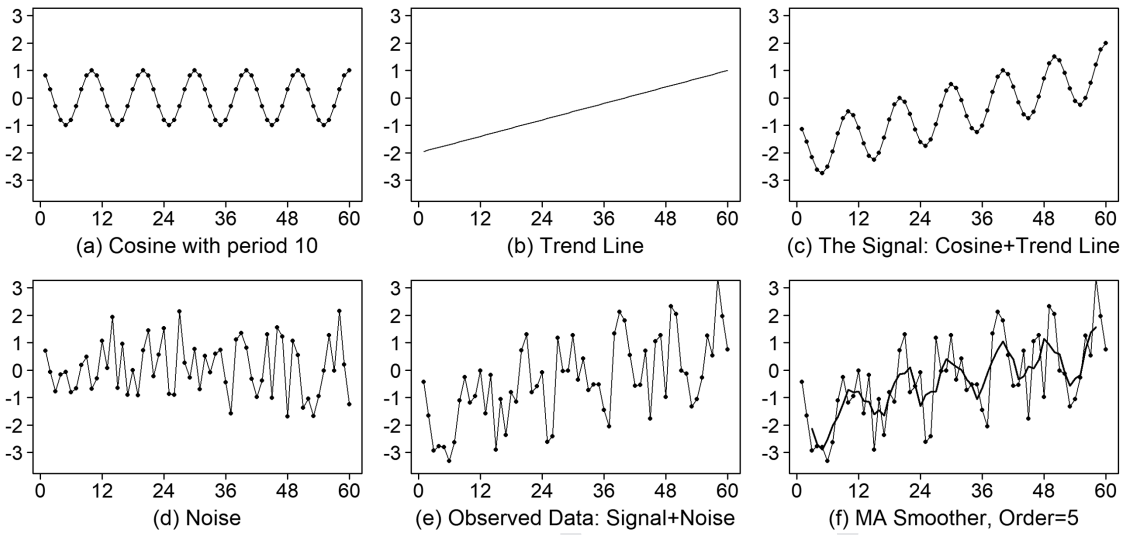


FIGURE 2.2 (a) Cosine curve with period 10, (b) the trend line, (c) cosine+line (the underlying signal), (d) the noise component, (e) signal+noise (the observed data), and (f) observed data in (e) with fifth order smoothing.

object with **`frequency=12`**. The data in Figure 2.3(a) are for the years 1749–2020 for which monthly data are available, and they are contained in the file

```
sunspot2.0.1749=window(sunspot2.0,start=1749,end=2020)
```

The annual data in **`sunspot2.0.1749`** can be calculated from the monthly data using a procedure called *aggregating*. The Base R function **`aggregate`** can be used to average the 12 monthly observations in each year, which results in one observation per year. For example, the command

```
ss.yr=aggregate(sunspot2.0.month,FUN=mean)
```

instructs R to use the sunspot monthly data, group it into subsets of 12 months (because **`frequency=12`** in the ***ts*** file) and summarize the information in each subset using the mean function (**`FUN=mean`**). Figure 2.3(c) is a plot of the aggregated data, and it consists of 272 data values almost identical to the data in **`sunspot2.0`** for the years 1749–2020.[4] Other summary statistics can be used. For more about aggregation methods see R help for the **`aggregate`** function. The annual data in **`ss.yr`** obtained using the **`aggregate`** command above are plotted in Figure 2.3(c). Note that **`ss.yr`** is a ***ts*** file, the first few lines of which are

```
Time Series:
Start = 1749
End = 2020
Frequency = 1
  [1]  134.875000 139.000000  79.441667  79.666667
  [5]   51.125000  20.358333  15.933333  16.983333
  [9]   54.041667  79.341667  89.941667 104.775000
```

4 **`ss.yr`** and **`sunspot2.0.1749`** differ slightly because ss.yr is simply the mean of the 12 monthly values while **`sunspot2.0.1749`** takes into account the number of days per month.

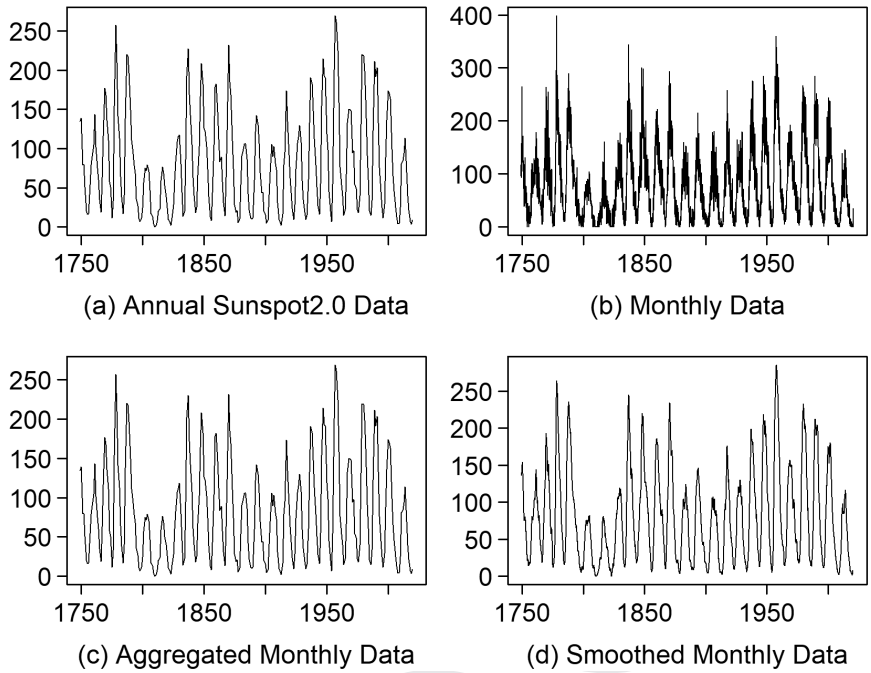


FIGURE 2.3 (a) Annual sunspot data for the years 1749–2020, (b) monthly data for the same time period, (c) annual data obtained by aggregating the monthly data, (d) smoothed monthly data using a 12th-order moving average smoother.

To create the plot in Figure 2.3(d) we applied a centered moving average smoother of order 12 to the monthly data in **`sunspot2.0.month`**. The plots in Figure 2.3 can be obtained using the following code.

```
data(sunspot2.0)
data(sunspot2.0.month)
sunspot2.0.12=ma.smooth.wge(sunspot2.0.month,order=12,plot=FALSE)
sunspot2.0.sm12=ts(sunspot2.0.12$smooth,start=c(1749,1),
frequency=12)
par(mfrow=c(2,2))
plotts.wge(sunspot2.0.1749)
plotts.wge(ss.yr)
plotts.wge(sunspot2.0.month)
plotts.wge(sunspot2.0.sm12)
```

Note: The aggregated data in Figure 2.3(c) and the smoothed data in Figure 2.3(d) appear to be very similar. However,

(i) Figure 2.3(c) is a plot of 272 points of annual data for years 1749 through 2020 obtained by aggregating the monthly data for each year into an annual value by averaging the monthly data.
(ii) Figure 2.3(d) is the result of a moving average smoother. It contains $3264 - 12 = 3252$ *monthly* data values.[5]

5 Note that six data values are "lost" at each end of the series because of the smoother. The rolling window calculates an "annualized value" for each consecutive 12 months.

2.1.1.4 Using Moving Average Smoothing for Estimating Trend in Data with Fixed Cycle Lengths

We discussed the centered moving average smoother in Section 2.1.1.1 and demonstrated that it can be used to "uncover" the underlying signal that was "covered up" by noise. It can also be used to remove cyclic content with fixed cycle length.

(1) Cosine+Line+Noise Data

Figure 2.2 illustrated the use of moving average smoothing to uncover a "trending cosine" curve of period 10 that is embedded in noise. Figure 2.4(a) shows the noisy data previously shown in Figure 2.2(e). Figure 2.4(b) shows the result of a 5th-order moving average smoother shown earlier in Figure 2.2(f). As mentioned previously, the 5th-order smoother does a good job of removing the noise from the data and revealing the underlying cosine+line signal. Figure 2.4(c) shows the effect of increasing the smoothing order to the *period length*, 10. The resulting smoothed data estimates the trend (in this case a line) and removes the cyclic component altogether. The following is an important point.

Key Point: If a centered moving average smoother of order k is applied to fixed period cyclic data with fixed cycle length *where k is the cycle length*, then the smoother removes the cyclic component and "reveals" (or estimates) the trend.

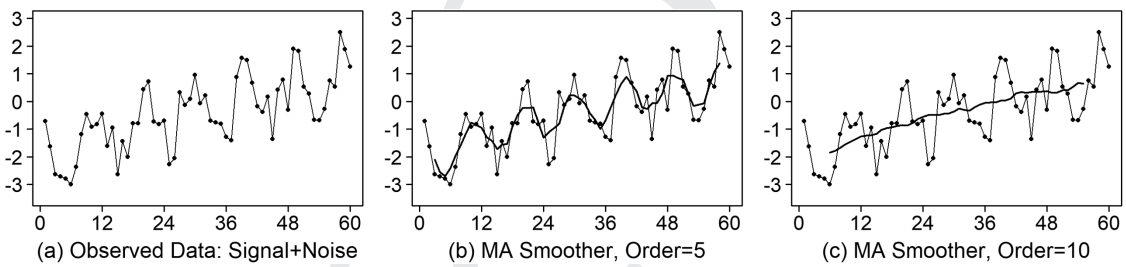


FIGURE 2.4 (a) Cosine+line+noise data, (b) and (c) observed data in (a) with 5th and 10th order smoothing, respectively.

(2) Air Passengers Data

The monthly **`AirPassengers`** data in Figure 2.5(a) were discussed in Chapter 1 where it was noted that there is a repetitive (seasonal but non-sinusoidal) behavior each year. The moving average smoothed data of order 12 in Figure 2.5(b) show no evidence of the within-year seasonal behavior, but instead it is a curve that tracks the trend in airline travel. For example, there were fairly "flat" spots around 1953 and 1958 where there was not much increase in the number of passengers over the previous year. These were both temporary in that air travel continued to increase after each of these "flat" spots. The point is that there are annual patterns that are more easily seen in Figure 2.5(b) than in the **`AirPassengers`** data (although close examination of Figure 2.5(a) shows the periods in question). Thus, the 12th-order moving average smoother behaved like the 10th-order moving average smoother on the period 10 sinusoidal data in Figure 2.4(c) in that it removed the cyclic behavior. It is interesting to note the effect of a 15th-order smoother on the **`AirPassengers`** data shown in Figure 2.5(c). The 15th order introduced some wiggle in the smoothed data reinforcing the fact that the cyclic component is most effectively removed when the order of the smoother is equal to the period length of the cyclic data.

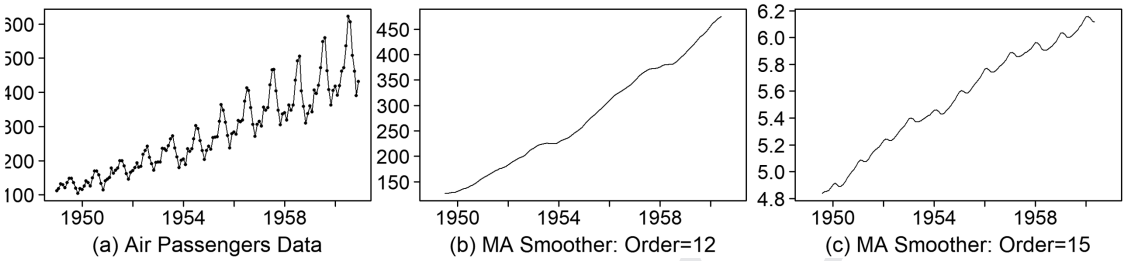


FIGURE 2.5 (a) Air Passengers data, (b) and (c) observed data in (a) with 12th and 15th -order smoothing, respectively.

Key Point: While a moving average smoother of order k applied to cyclic data with a fixed cycle length k will remove the cyclic behavior and retain the trend, smoothers of orders greater than k or less than k will retain some cyclic behavior.

QR 2.1 Smoothing Orders

(3) Texas Unemployment Data

Figure 2.6(a) displays the monthly unemployment rate in Texas for the years 2000−2019, obtained from the Texas Workforce Commission website https://twc.texas.gov. The data are in ***tswge ts*** file **`tx.unemp.unadj`**. The data show a seasonal behavior (which is not very clear in Figure 2.6(a)) but shows up better in the short time snippet in Figure 2.6(b). The seasonal pattern within a year appears to be that unemployment rates are lower in January through May than they are in the summer months, after which time they decline.

Figure 2.6(c) is a plot of the original unemployment data after applying a 12th-order moving average smoother. As previously discussed, the smoother removed the seasonal behavior and retained the "trend" which in this case was anything but linear. The trending behavior is important, and there is much to be learned from it. Unemployment rose to a peak of about 7.5% in 2004, dropped to about 4%, then increased rapidly to about 8% during the Great Recession, and stayed at that level from June 2009 through September 2011. The unemployment rate then consistently declined through 2019 when unemployment was at about 3.3%.[6] Figure 2.6(c) was obtained using the commands

```
data(tx.unemp.unadj)
tx.unemp.sm=ma.smooth.wge(ts.unemp.unadj,frequency=12, plot=FALSE)
tx.unemp.sm12=ts(tx.unemp.sm12,start=c(2000,1),frequency=12)
plotts.wge(ts.unemp.sm12,type='l')
```

6 Although data are available through 2020, the COVID pandemic caused very different and unpredictable unemployment behavior.

Key Point: A moving average smoother of order k applied to cyclic data with a fixed cycle length, k, will remove the cyclic behavior and retain the trend even when the trend follows a wandering path as in the Texas Unemployment data.

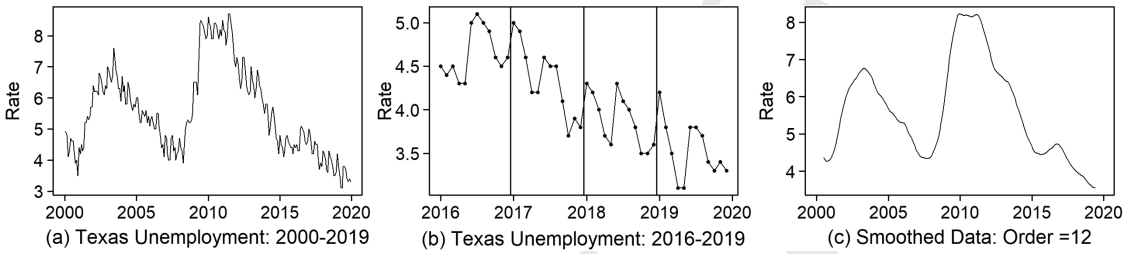


FIGURE 2.6 (a) Monthly Texas unemployment rates from 2000 through 2019 and (b) unemployment rates from 2016 through 2019, (c) original data after applying a 12th-order moving average smoother.

2.1.2 Decomposing Seasonal Data

We defined seasonal data in Chapter 1 as cyclic data that have fixed periods and having a pattern related to the calendar. The **`AirPassengers`** dataset in Figure 1.5 and Figure 2.5(a) is a classical seasonal dataset. The dataset not only has a within-year seasonal behavior but also has a (near linear) increasing trend. It is common practice to consider seasonal data, x_t, to have: (a) a within-year seasonal component, s_t, (b) a longer-term trend component, tr_t, and (c) a random noise component, z_t. For example, we have seen these behaviors in the Air Passengers and Texas Unemployment data. Time series analysts focus on two types of seasonal models:

Additive Seasonal Data
The data, x_t, at time t can be thought of as a sum given in (2.4)

$$x_t = s_t + tr_t + z_t. \tag{2.4}$$

Multiplicative Seasonal Data
The data, x_t, at time t can be expressed as the product given in (2.5)

$$x_t = s_t \times tr_t \times z_t. \tag{2.5}$$

Example 2.3 Air Passengers Data
To illustrate the difference between the data types that are best fit with an additive and with a multiplicative model, we use the **`AirPassengers`** dataset. As previously noted, the **`AirPassengers`** data, plotted in Figure 2.7(a) have a seasonal and trend component but also the within-year variability increases in time.

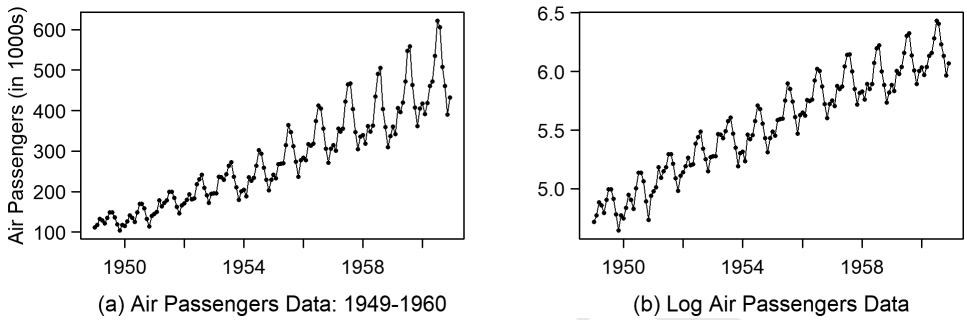


FIGURE 2.7 (a) Air Passengers data and (b) logarithm of Air Passengers data.

Datasets with this type of behavior are often modeled using multiplicative models. To eliminate the variability increase, analysts often take the logarithms of the data and use the "log data" for analysis. The R commands below create the l**ogAirPassengers** data which are plotted in Figure 2.7(b).

```
data(AirPassengers)
logAirPassengers=log(AirPassengers)
```

Recall that the **AirPassengers** data are in the form of a ***ts*** object, so **logAirPassengers** is also a ***ts*** object. The **logAirPassengers** data in Figure 2.7(b) show no increase in the within-year variability and are a classic example of data that are modeled using the additive model in (2.4). We will discuss the differences in modeling strategies for these two datasets in the following.

The additive and multiplicative decompositions follow similar implementation steps:

1. Estimate the trend component.
2. Remove the trend component which results in a dataset that is primarily made up of the seasonal fluctuations in the data.
3. Calculate an "average" within-year seasonal component.
4. Find the remaining noise.

We begin by discussing the additive model which is the more intuitive of the two.

Key Points

1. Data with increasing within-year variability, i.e. multiplicative data, are common in economics and other fields.
2. Analysts model multiplicative data in two ways:
   - Using the multiplicative model in (2.5)
   - Taking the logarithm of the data and modeling the log data with an additive model.

2.1.2.1 Decomposing Additive Models

In this section, we will obtain additive decompositions of the **logAirPassengers** and the Texas Unemployment data.

(1) Log Air Passengers Data

When data are analyzed using the additive model in (2.4), the assumption is made that the data are the sum of seasonal, trend, and random noise components. The analysis steps involved in decomposing the

logAirPassengers data are discussed. In practice, the components in (2.4) are estimated and an estimated model can be described as

$$x_t = \hat{s}_t + \hat{tr}_t + \hat{z}_t. \tag{2.6}$$

(a) Estimate the Trend Component

Figure 2.8 is a plot of the **logAirPassengers** dataset overlaid with the result of applying a 12th-order centered moving average smoother to the data. These plots were obtained using the commands:

```
data(AirPassengers)
logAirPassengers=log(AirPassengers)
logair.12=ma.smooth.wge(logAirPassengers,order=12)
logair.sm12=ts(logair.12$smooth,start=c(1949,1),frequency=12)
```

The **logair.12** object contains the data plotted in the overlaid trend in Figure 2.8. These data are listed below:

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1949	NA	NA	NA	NA	NA	NA	4.8373	4.8411	4.8466	4.8512	4.8545	4.8600
1950	4.8698	4.8814	4.8934	4.9043	4.9128	4.9237	4.9405	4.9574	4.9744	4.9919	5.0131	5.0338
1951	5.0478	5.0609	5.0738	5.0884	5.1069	5.1243	5.1383	5.1528	5.1637	5.1715	5.1784	5.1894
1952	5.2039	5.2181	5.2316	5.2437	5.2574	5.2707	5.2829	5.2921	5.3041	5.3233	5.3436	5.3574
1953	5.3677	5.3783	5.3884	5.3978	5.4038	5.4072	5.4104	5.4103	5.4084	5.4068	5.4062	5.4106
1954	5.4196	5.4283	5.4351	5.4422	5.4507	5.4611	5.4737	5.4897	5.5040	5.5164	5.5294	5.5427
1955	5.5579	5.5727	5.5875	5.6027	5.6167	5.6312	5.6459	5.6598	5.6742	5.6876	5.7008	5.7147
1956	5.7272	5.7389	5.7507	5.7607	5.7708	5.7804	5.7887	5.7965	5.8048	5.8141	5.8231	5.8327
1957	5.8427	5.8535	5.8649	5.8755	5.8857	5.8945	5.9016	5.9070	5.9100	5.9107	5.9116	5.9138
1958	5.9174	5.9229	5.9261	5.9276	5.9297	5.9305	5.9330	5.9384	5.9462	5.9564	5.9678	5.9773
1959	5.9853	5.9941	6.0040	6.0149	6.0266	6.0407	6.0545	6.0662	6.0731	6.0807	6.0919	6.1020
1960	6.1125	6.1212	6.1284	6.1374	6.1457	6.1515	NA	NA	NA	NA	NA	NA

In terms of the estimated model in (2.6), $\hat{tr}_t$ = **logair.sm12** is the near linear curve in Figure 2.8.

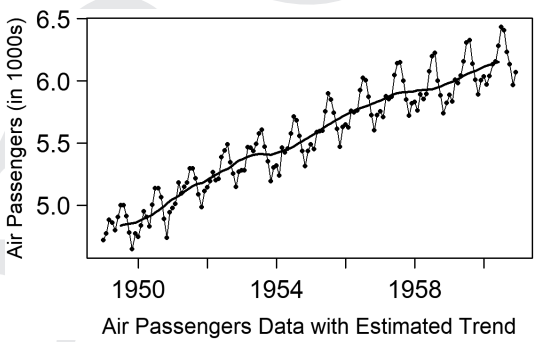


FIGURE 2.8 Log Air Passenger data overlaid with 12th-order centered moving average smoother.

(b) Remove the Trend Component from the Data

The next step is to subtract the estimated trend component from the data (**logAirPassengers**). This can be accomplished using the code:

```
seas.logair=logAirPassengers-logair.sm12
round(seas.logair,4) # rounding to 4 decimal places for listing below
```

The data in **seas.logair** are plotted in Figure 2.9(a) and the ***ts*** file is shown below:

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1949	NA	NA	NA	NA	NA	NA	0.1599	0.1561	0.0661	-0.0721	-0.2101	-0.0893
1950	-0.1249	-0.0451	0.0553	0.0010	-0.0844	0.0802	0.1953	0.1784	0.0882	-0.1016	-0.2769	-0.0922
1951	-0.0710	-0.0503	0.1080	0.0054	0.0406	0.0575	0.1550	0.1406	0.0512	-0.0839	-0.1948	-0.0774
1952	-0.0622	-0.0251	0.0311	-0.0452	-0.0479	0.1138	0.1552	0.1968	0.0383	-0.0711	-0.1961	-0.0896
1953	-0.0896	-0.1002	0.0754	0.0618	0.0299	0.0858	0.1656	0.1955	0.0597	-0.0549	-0.2133	-0.1073
1954	-0.1015	-0.1919	0.0245	-0.0173	0.0047	0.1148	0.2368	0.1905	0.0529	-0.0826	-0.2162	-0.1090
1955	-0.0689	-0.1217	-0.0002	-0.0080	-0.0182	0.1214	0.2512	0.1895	0.0688	-0.0745	-0.2327	-0.0871
1956	-0.0782	-0.1148	0.0082	-0.0145	-0.0088	0.1438	0.2347	0.2074	0.0673	-0.0905	-0.2210	-0.1091
1957	-0.0901	-0.1464	0.0101	-0.0233	-0.0135	0.1505	0.2405	0.2393	0.0914	-0.0614	-0.1913	-0.0967
1958	-0.0884	-0.1608	-0.0345	-0.0754	-0.0353	0.1449	0.2635	0.2862	0.0552	-0.0730	-0.2312	-0.1572
1959	-0.0992	-0.1593	0.0024	-0.0335	0.0137	0.1163	0.2518	0.2600	0.0646	-0.0719	-0.2003	-0.0981
1960	-0.0794	-0.1524	-0.0905	-0.0040	0.0112	0.1307	NA	NA	NA	NA	NA	NA

The year-to-year seasonal behavior is much easier to visualize in Figure 2.9(a) without the "interference" of the trend. Specifically, the pattern in each year is similar (high travel in summer, low in November, slightly up in December, lower in January, and so forth). However, there are year-to-year variations: air travel in November was unusually low in 1950 and then abnormally high for July and August of 1958.

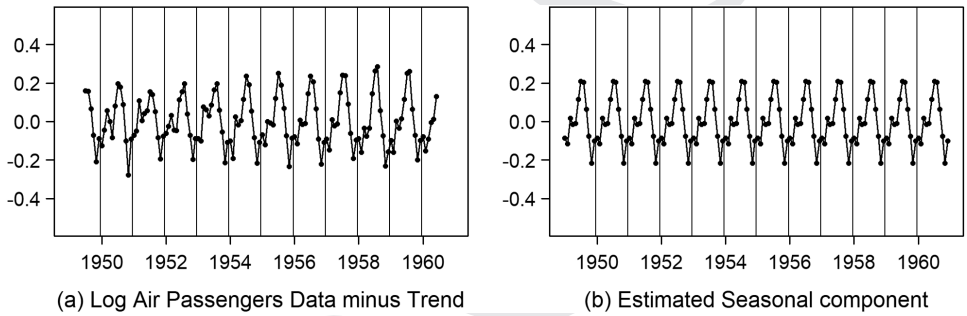


FIGURE 2.9 (a) Log Air Passengers data minus the trend component in Figure 2.8 and (b) estimated seasonal component.

(c) Calculate an "Average" Within-year Seasonal Component

Note that the seasonal component in the model (2.4) is an overall pattern that is the same from year to year. That is, $\{s_t, t = 1,\ldots,12\} = \{s_{t+12}, t = 1,\ldots,12\} = \{s_{t+2(12)}, t = 1,\ldots,12\} = \cdots$. The noise component, z_t, adjusts for the year-to-year variations from the overall seasonal pattern. The seasonal pattern, $s_t, t = 1,\ldots,12$, is estimated by averaging across years, and the estimated seasonal component, $\hat{s}_t$, (which is the same for each year) is plotted in Figure 2.9(b). The following code can be used to calculate the monthly means:

```
# convert the ts file to a vector
seas.logair.numeric=as.numeric(seas.logair) # convert ts file to a vector
# convert this vector to a matrix with ncol=number of years
seas.logair.matrix=matrix(seas.logair.numeric,ncol=12) #12 years and 12 months
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]
[1,]	NA	-0.1249	-0.0710	-0.0622	-0.0896	-0.1015	-0.0689	-0.0782	-0.0901	-0.0884	-0.0992	-0.0794
[2,]	NA	-0.0451	-0.0503	-0.0251	-0.1002	-0.1919	-0.1217	-0.1148	-0.1464	-0.1608	-0.1593	-0.1524
[3,]	NA	0.0553	0.1080	0.0311	0.0754	0.0245	-0.0002	0.0082	0.0101	-0.0345	0.0024	-0.0905
[4,]	NA	0.0010	0.0054	-0.0452	0.0618	-0.0173	-0.0080	-0.0145	-0.0233	-0.0754	-0.0335	-0.0040

```
 [5,]      NA -0.0844  0.0406 -0.0479  0.0299  0.0047 -0.0182 -0.0088 -0.0135 -0.0353  0.0137  0.0112
 [6,]      NA  0.0802  0.0575  0.1138  0.0858  0.1148  0.1214  0.1438  0.1505  0.1449  0.1163  0.1307
 [7,]  0.1599  0.1953  0.1550  0.1552  0.1656  0.2368  0.2512  0.2347  0.2405  0.2635  0.2518      NA
 [8,]  0.1561  0.1784  0.1406  0.1968  0.1955  0.1905  0.1895  0.2074  0.2393  0.2862  0.2600      NA
 [9,]  0.0661  0.0882  0.0512  0.0383  0.0597  0.0529  0.0688  0.0673  0.0914  0.0552  0.0646      NA
[10,] -0.0721 -0.1016 -0.0839 -0.0711 -0.0549 -0.0826 -0.0745 -0.0905 -0.0614 -0.0730 -0.0719      NA
[11,] -0.2101 -0.2769 -0.1948 -0.1961 -0.2133 -0.2162 -0.2327 -0.2210 -0.1913 -0.2312 -0.2003      NA
[12,] -0.0893 -0.0922 -0.0774 -0.0896 -0.1073 -0.1090 -0.0871 -0.1091 -0.0967 -0.1572 -0.0981      NA
```

Comparing **seas.logair.matrix** with **seas.logair** it can be seen that the rows and columns are reversed. The matrix with columns representing months as in the ***ts*** file is obtained by transposing the matrix.

```
seas.logair.matrix.t=t(seas.logair.matrix)
```

```
 [1,]      NA      NA      NA      NA      NA      NA  0.1599  0.1561  0.0661 -0.0721 -0.2101 -0.0893
 [2,] -0.1249 -0.0451  0.0553  0.0010 -0.0844  0.0802  0.1953  0.1784  0.0882 -0.1016 -0.2769 -0.0922
 [3,] -0.0710 -0.0503  0.1080  0.0054  0.0406  0.0575  0.1550  0.1406  0.0512 -0.0839 -0.1948 -0.0774
 [4,] -0.0622 -0.0251  0.0311 -0.0452 -0.0479  0.1138  0.1552  0.1968  0.0383 -0.0711 -0.1961 -0.0896
 [5,] -0.0896 -0.1002  0.0754  0.0618  0.0299  0.0858  0.1656  0.1955  0.0597 -0.0549 -0.2133 -0.1073
 [6,] -0.1015 -0.1919  0.0245 -0.0173  0.0047  0.1148  0.2368  0.1905  0.0529 -0.0826 -0.2162 -0.1090
 [7,] -0.0689 -0.1217 -0.0002 -0.0080 -0.0182  0.1214  0.2512  0.1895  0.0688 -0.0745 -0.2327 -0.0871
 [8,] -0.0782 -0.1148  0.0082 -0.0145 -0.0088  0.1438  0.2347  0.2074  0.0673 -0.0905 -0.2210 -0.1091
 [9,] -0.0901 -0.1464  0.0101 -0.0233 -0.0135  0.1505  0.2405  0.2393  0.0914 -0.0614 -0.1913 -0.0967
[10,] -0.0884 -0.1608 -0.0345 -0.0754 -0.0353  0.1449  0.2635  0.2862  0.0552 -0.0730 -0.2312 -0.1572
[11,] -0.0992 -0.1593  0.0024 -0.0335  0.0137  0.1163  0.2518  0.2600  0.0646 -0.0719 -0.2003 -0.0981
[12,] -0.0794 -0.1524 -0.0905 -0.0040  0.0112  0.1307      NA      NA      NA      NA      NA      NA
```

The following code creates the vector **months** containing the column means excluding the missing data.

```
months=colMeans(seas.logair.matrix.t, na.rm=TRUE)
round(months,4)
```

```
[1] -0.0867 -0.1153 0.0172 -0.0139 -0.0098 0.1145  0.2100 0.2036 0.0640 -0.0761 -0.2167 -0.1012
```

```
seas.means=rep(months,12) # replicates the 12 monthly means for each year (12)
seas.means=ts(seas.means,start=c(1949,1),frequency=12)
```

The ***ts*** file **seas.means** is plotted in Figure 2.9(b).

(d) Find the Remaining Noise Component

We have computed the following:

Trend estimate: **logair.sm12**
Seasonal component estimate: **seas.means**

The estimated noise in (2.6), $\hat{z}_t$, is calculated using the code:

```
logair.noise=logAirPassengers-logair.sm12-seas.means
```

PLOT OF DECOMPOSITION

Figure 2.10(d) is a plot of the **logair.noise** along with the other parts of the decomposition procedure:

(a) log **AirPassengers** data
(b) estimated trend component
(c) estimated seasonal component
(d) noise

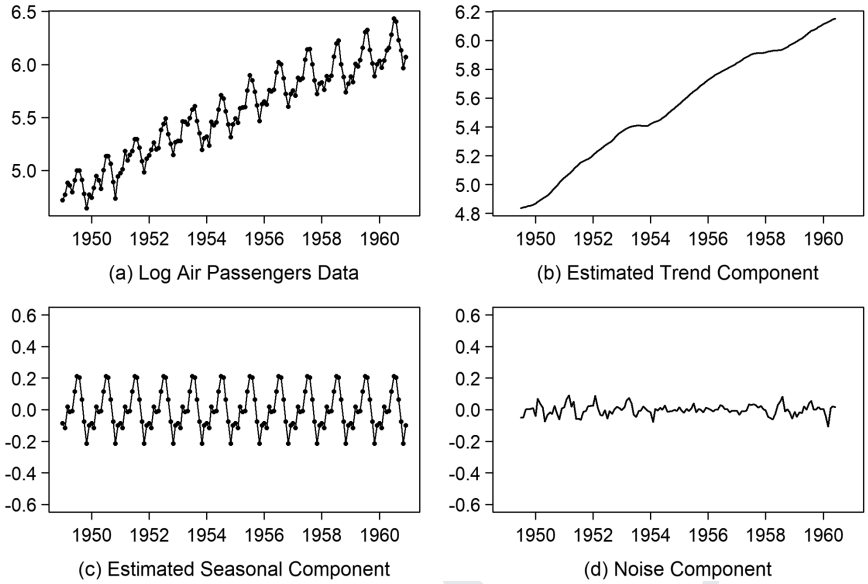


FIGURE 2.10 Additive Decomposition of Log Air Passengers Data.

(2) Texas Unemployment Data

We will repeat the steps used in the additive decomposition of the **`logAirPassengers`** data, but without as much detail.

(a) Estimate the Trend Component

Figure 2.11(a) shows the Texas Unemployment data (not seasonally adjusted) along with the trend estimate using a centered moving average smoother of order 12. This plot can be obtained using the commands:

```
data(tx.unemp.unadj)
tx.unemp.sm=ma.smooth.wge(tx.unemp.unadj,frequency=12,plot=FALSE)
tx.unemp.sm12=ts(tx.unemp.sm$smooth,start=c(2000,1), frequency=12)
plotts.wge(tx.unemp.sm12,type='l')
```

(b) Remove the Trend Component from the Data

Remove the estimated trend component from the data (**`tx.unemp.unadj`**) using the command

```
seas.tx.unemp=tx.unemp.unadj-tx.unemp.sm12
```

Figure 2.11(b) is a plot of **`seas.tx.unemp`** where it can be seen that there is considerable year-to-year variability in the seasonal behavior, but the general behavior is for the unemployment in summer months to be high. This was observed in the time snippet in Figure 2.6(b).

(c) Calculate an "Average" Within-year Seasonal Component

The estimated seasonal pattern, $\hat{s}_t, t = 1,\ldots,12$, is found by averaging across years, and the seasonal component consists of repeating this pattern for all years in the dataset. Figure 2.11(c) shows the estimated seasonal component. The "average" seasonal pattern is for high unemployment in the summer months with the best (lowest) unemployment in April and May.

```
seas.tx.unemp.numeric=as.numeric(seas.tx.unemp) #convert ts file to a vector
# convert this vector to a matrix with ncol=number of years
seas.tx.unemp.matrix=matrix(seas.tx.unemp.numeric,ncol=20) # 20 years
seas.tx.unemp.matrix.t=t(seas.tx.unemp.matrix) # transpose matrix
months=colMeans(seas.tx.unemp.matrix.t,na.rm=TRUE) # colMeans are monthly means
```

```
seas.means=rep(months,20)  # replicates the 12 monthly means for each year (20)
seas.means=ts(seas.means,start=c(2000,1),frequency=12)
```

The data plotted in Figure 2.11(c) are in the vector **seas.means.**

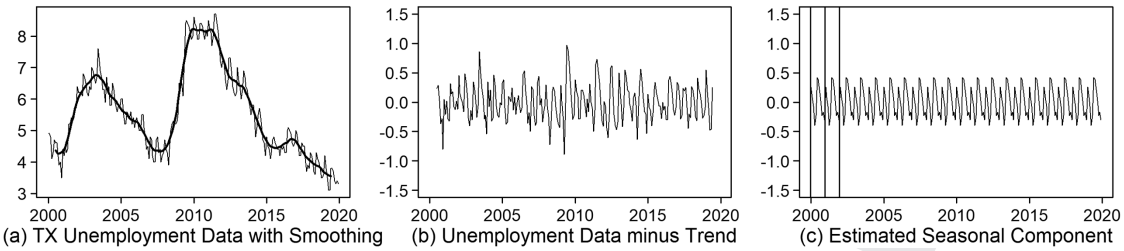


FIGURE 2.11 (a) Texas unemployment rates for 2000 through 2019 and showing moving average smoothed data of order 12, (b) Texas unemployment data minus trend, and (c) estimated seasonal component.

(d) Find the Remaining Noise Component
We have computed the following:

Trend estimate: **tx.unemp.sm12**
Seasonal component estimate: **seas.means**

The estimated noise in (2.6), $\hat{z}_t$, is calculated using the code:

```
tx.unemp.noise=tx.unemp.unadj-tx.unemp.sm12-seas.means
```

PLOT OF DECOMPOSITION

Figure 2.12(d), similar to that of Figure 2.10(d), is a plot of **tx.unemp.noise** along with the other parts of the decomposition procedure:

(a) Texas unemployment data
(b) estimated trend component
(c) estimated seasonal component
(d) noise

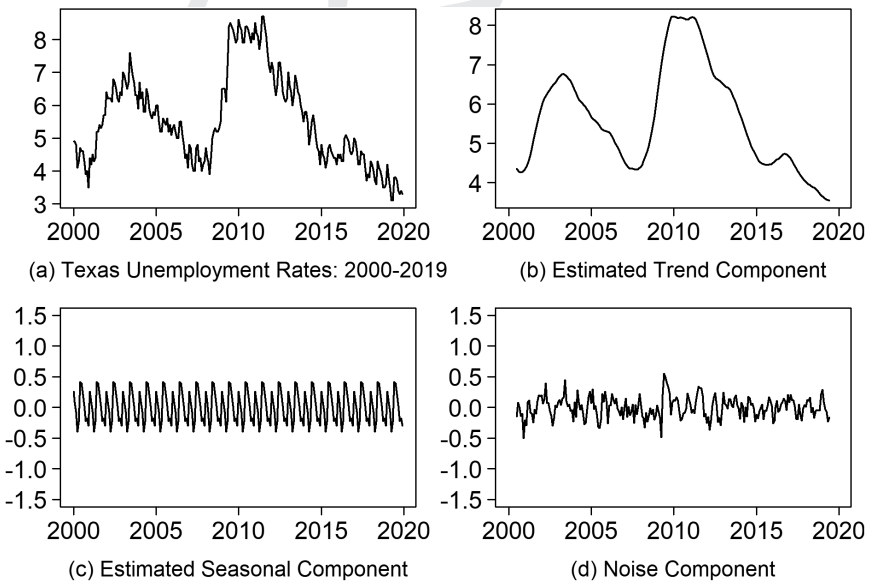


FIGURE 2.12 Additive Decomposition of Texas Unemployment Data.

2.1.2.2 Decomposing Multiplicative Models

We will obtain a multiplicative decomposition of the **AirPassengers** data shown in Figures 2.5(a) and 2.7(a). Recall that these data have a seasonal behavior that is increasing in time along with within-year variability that is also increasing in time. While such data can be modeled by taking the logarithm and then modeling the log data using an additive model, in this section we use a multiplicative model to analyze the "raw" **AirPassengers** data. When data are analyzed using multiplicative decomposition, the assumption is made that the data are the product of seasonal, trend, and noise components. The estimated model is

$$x_t = \hat{s}_t \times \widehat{tr}_t \times \hat{z}_t. \tag{2.7}$$

(1) Estimate the Trend Component

As with the additive model, the first step is to use a centered moving average smoother, again in this case of order 12. We previously computed and plotted the moving average smoother in Figure 2.5(b). The steps involved are as follows:

```
data(AirPassengers)
AirPass.sm12=ma.smooth.wge(AirPassengers,order=12)
AirPass.sm12=ts(AirPass.sm12$smooth,start=c(1949,1),frequency=12)
```

Recall that, in terms of the estimated model in (2.7), $\widehat{tr}_t$ = **AirPass.sm12**. This near linear curve is shown as part of the full decomposition in Figure 2.14(b).

(2) Remove the Trend Component from the Data

The next step is to remove the estimated trend component from the dataset, **AirPassengers**. This can be accomplished by *division* (instead of subtraction) using the code:

```
seas.AirPass=AirPassengers/AirPass.sm12
```

The year-to-year seasonal behavior is much easier to visualize in Figure 2.13(a) after removing the "interference" of the trend and increasing within-year variability. Note also that the within-year variability is not increasing. The increase in variability in the final model (2.7) is caused by the increasing trend being multiplied by the seasonal data in Figure 2.13(a). This plot has seasonal patterns similar to the additive data in Figure 2.9(a).

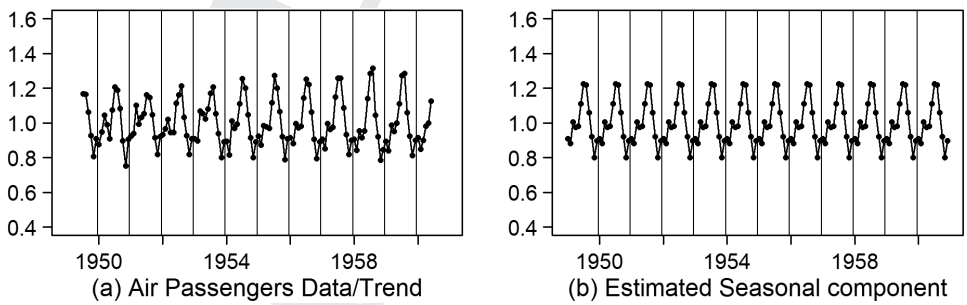


FIGURE 2.13 (a) Air Passenger data with trend component removed (by division) and (b) estimated seasonal component.

(3) Calculate an "Average" Within-year Seasonal Component.
As in the additive model, the seasonal component in the model (2.5) is an overall pattern that is assumed to be the same from year to year, and the noise component, z_t, adjusts for the year-to-year variations from the overall seasonal pattern. The estimated seasonal component, $\hat{s}_t$, (which is the same for each year) is plotted in Figure 2.13(b). Note the similarity between Figure 2.13(b) and Figure 2.9(b) which was the seasonal component for additive decomposition of the **log.air** data.

The following code can be used to calculate the monthly means:

```
seas.AirPass.numeric=as.numeric(seas.AirPass) #convert ts file to a vector
# convert this vector to a matrix with ncol=number of years
seas.AirPass.matrix=matrix(seas.AirPass.numeric,ncol=12) # 12 years
seas.AirPass.matrix.t=t(seas.AirPass.matrix) # transpose matrix
months=colMeans(seas.AirPass.matrix.t,na.rm=TRUE) # colMeans are monthly means
seas.means=rep(months,12)  # replicates the 12 monthly means for each year (12)
seas.means=ts(seas.means,start=c(1949,1),frequency=12)
```

The ***ts*** file **seas.means** is plotted in Figure 2.13(b).

(4) Find the Remaining Noise Component
We have computed the following:

Trend estimate: **AirPass.sm12**
Seasonal component estimate: **seas.means**

The estimated noise in (2.6), $\hat{z}_t$, is calculated using the code:

```
Air.Pass.noise=AirPassengers/(AirPass.sm12*seas.means)
```

Figure 2.14 is a plot of the component parts of the multiplicative decomposition of the **AirPassengers** data.[7]

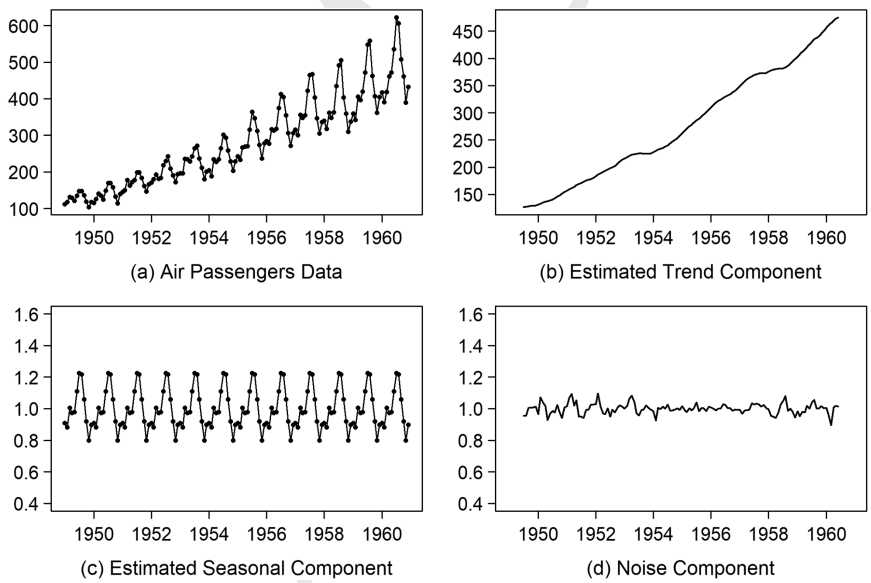


FIGURE 2.14 Multiplicative Decomposition of Air Passengers Data.

7 Base R function **decompose** can be used for time series decomposition. (See Appendix 2A)

2.1.3 Seasonal Adjustment

As you looked through the FRED website you probably noticed that many (even most) datasets are *seasonally adjusted*. Other government websites such as the Census (https://census.gov) and the Bureau of Labor Statistics (www.bls.gov/web/laus/laumstrk.htm) provide seasonally adjusted data. Seasonal adjustment techniques can be quite complex and have been widely studied and developed by statisticians at the Census Bureau and elsewhere. In this section we briefly discuss the basic ideas and goals of seasonal adjustment, go through a step-by-step rudimentary seasonal adjustment procedure, apply seasonal adjustment to the Texas Unemployment and Air Passengers data, and then discuss software that is available for up-to-date seasonal adjustment procedures.

The question is: "Why are so many economic-type datasets seasonally adjusted?" The Texas Unemployment data in Figure 2.12(a) contain a lot of information to be understood. There is year-to-year trending along with within-year seasonal patterns. These within-year seasonal patterns are fairly predictable, but they do have (possibly important) variations from year to year and month to month. One method of seeing the "big picture" is to perform a centered moving average smoother to obtain a trend estimate as shown in Figure 2.12(b). For example, the smoothed data in Figure 2.12(b) show the year-to-year changes such as the Great Recession which was followed by a consistent decline in unemployment until the COVID pandemic in 2020.[8] For this reason, the trend estimate obtained by moving average smoothing can be thought of as a type of "seasonal adjustment" because it removes the within-year seasonal changes and shows the overall behavior. However, notice in Figure 2.11(b) that within-year seasonal patterns differed quite a bit from year to year. If an important economic event occurred in May of some year, that information is smeared out in the trend estimate. For example, the patterns within the year 2009 were "more extreme" than those in other years. What is needed is a seasonal adjustment that removes the normal seasonal patterns, shows the overall trending behavior, but also retains month-to-month information that can highlight unusual activity.

2.1.3.1 Additive Seasonal Adjustment

Seasonal adjustments are related to the decompositions. If an additive decomposition is appropriate, then an additive seasonal adjustment is used. A similar pairing applies in the multiplicative case.

Because an additive decomposition was appropriate for the Texas Unemployment data, an additive-based seasonal adjustment will be used for this dataset. The most straightforward additive seasonal adjustment method that has some ability to accomplish the goals discussed in the preceding paragraph is to obtain the seasonally adjusted data, $\widehat{sa}_t$, using the formula

$$\widehat{sa}_t = x_t - \hat{s}_t, \tag{2.8}$$

which subtracts the seasonal component (shown in Figure 2.12(c)) from the data.

Figure 2.15(a) shows the seasonally adjusted Texas Unemployment data using this simple method. The command to create the seasonally adjusted data is

```
tx.unemp.adj.1 = tx.unemp.unadj - seas.means
```

Figure 2.15(a) shows the Texas Unemployment data superimposed with the trend estimate obtained using a centered moving average smoother of order 12. Figure 2.15(b) shows the seasonally adjusted data computed using (2.8). That is, it is a plot of **`tx.unemp.adj.1`**. This plot is similar to the estimated trend but with more detail concerning monthly changes that might be important. The website https://twc.texas.gov contains seasonally adjusted data, which is available in ***tswge*** data file **`tx.unemp.adj`**.The

8 As stated in a previous footnote, although data are available through 2020, the COVID pandemic caused very different and unpredictable unemployment behavior.

seasonal adjusted data on the https://twc.texas.gov website is much smoother than the data computed using (2.8). Finally, the mathematically sophisticated seasonal adjustment techniques developed at the Census Bureau are contained in the CRAN package ***seasonal***.[9] The function **`seas`** in the ***seasonal*** package performs the Census-Bureau seasonal adjustment. Although **`seas`** has numerous options, for purposes of this discussion we simply used **`seas`** with all default options. The following commands were used:

```
library(seasonal) # assuming that "seasonal" has been installed
census=seas(tx.unemp.unadj)
# the seasonally adjusted data are in the ts file census$data[,3]
```

Figure 2.15(d) is a plot of **`census$data[,3]`**. Examination of Figure 2.15 shows that the seasonal adjustment using **`seas`** with default options provides the same information as the one using (2.8).[10]

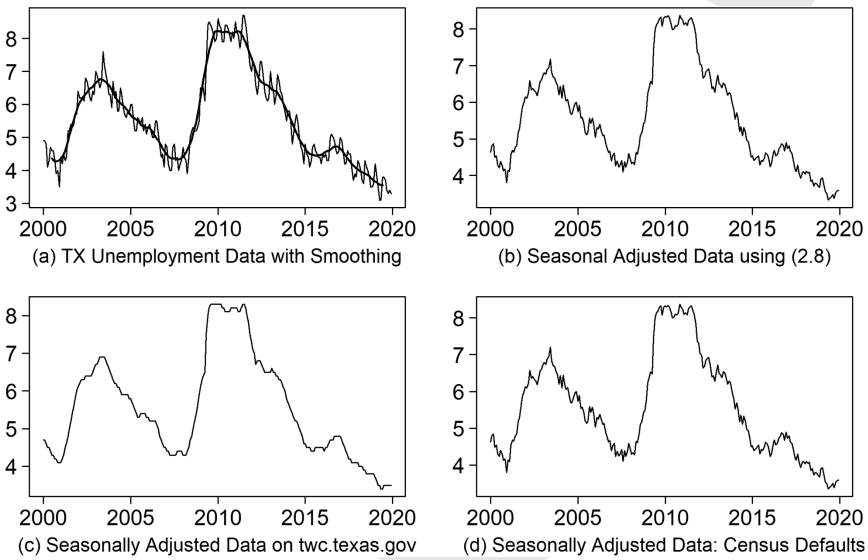


FIGURE 2.15 (a) Texas unadjusted unemployment data superimposed with the output from a 12th-order moving average smoother, (b) seasonally adjusted data using (2.8), (c) seasonally adjusted data from the twc.texas.gov website, and (d) seasonally adjusted data using the default options of the CRAN function seas.

2.1.3.2 Multiplicative Seasonal Adjustment

Because multiplicative decomposition was appropriate for the **`AirPassengers`** data, we will use multiplicative seasonal adjustment for this dataset. Analogous to the method used for the additive seasonal adjustment, the seasonally adjusted data uses the formula

$$\widehat{sa}_t = x_t / \hat{s}_t, \tag{2.9}$$

to *divide* the data by the seasonal component (shown in Figure 2.14(c)). The command to create the seasonally adjusted data is

9 The seasonal adjustment developed at the Census Bureau is referred to as X13-ARIMA-SEATS. Information about the Census Bureau seasonal adjustment can be found at www.census.gov/data/software/x13as.html

10 The ***seasonal*** package has a variety of seasonal adjustment methods including methods using seasonal ARIMA models. These models will be discussed in Chapter 7.

```
AirPassengers.adj = AirPassengers/seas.means
```

Figure 2.16(a) shows the **AirPassengers** data superimposed with the trend estimate obtained using a centered moving average smoother of order 12. Figure 2.16(b) is a plot of the seasonally adjusted data computed using (2.9). That is, it is a plot of **AirPassengers.adj.** Again, the seasonally adjusted data are similar to the estimated trend but with more detail concerning monthly changes. The following code uses the **seas** function with default options.

```
library(seasonal) # assuming that seasonal has been installed
census=seas(AirPassengers)
# the seasonally adjusted data are in the ts file census$data[,3]
```

Figure 2.16(c) is a plot of **census$data[,3]**. Examination of Figure 2.16(c) shows that the seasonal adjustment using **seas** is smoother and is less affected by the larger within-year variability increase in later years. Overall, the results are similar to those seen in Figure 2.16(b) with the seasonal effects removed.

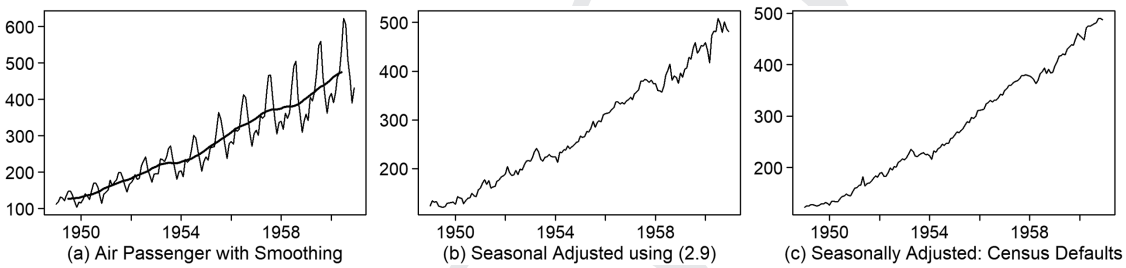


FIGURE 2.16 (a) Air Passengers data superimposed with the output from a 12th-order moving average smoother, (b) seasonally adjusted data using (2.9), (c) seasonally adjusted data using the default options of the function seas in the CRAN package *seasonal*.

In this section, we have shown how smoothing, decomposition, and seasonal adjustment are useful techniques to help the analyst understand and visualize data. These techniques have been illustrated using various examples. For both additive and multiplicative time series data, the separation of the original time series data into trend,[11] seasonality, and residual components enhances understanding of the individual components and the dataset as a whole. In many cases, unless a time series dataset is separated as such, important information is hidden and unrecognized.

2.2 FORECASTING

Obviously, a primary application of time series analysis is that of forecasting. It was previously stated that one of the purposes of smoothing is to forecast future values. Indeed, if evidence exists that previous patterns in a dataset will continue into the future, various techniques can be used to make forecasts. The techniques discussed in this chapter are "non-model" based approaches to forecasting, but later chapters will be devoted to "model-based" forecasting.

11 It should be emphasized that the "trend" recognized by the decomposition method is not necessarily a deterministic trend that can be assumed to continue indefinitely. Instead, it should be considered to be "the current trend". In later chapters we introduce a different methodology that also dissects (decomposes) time series data into various components and relevant details will be provided related to theory and useful application.

There are many situations for which businesses, organizations, and individuals are interested in the ability to make forecasts into the future. For example, a business owner may wish to forecast the future demand of a certain product so that the appropriate amount of inventory is ensured to be in stock. A city making decisions about infrastructure might need to forecast its population in ten years. A difficult problem but one of interest to many in the financial sector (and to most of us, actually) is forecasting the ebb and flow of the stock market and of individual stocks so that sound investing decisions can be made. Each of these examples illustrates the applicability and need for forecasting techniques. Without a better option, such forecasts are often made somewhat subjectively, based on past memory of similar events, rumor, or perhaps by intuitive but presumptuous calculations which provide educated guesses and estimates.

Fortunately, time series analysis techniques provide a mathematical, theory-based alternative if the underlying mathematical assumptions are appropriate. This results in algorithms that calculate forecasts along with corresponding prediction limits at a given level of confidence, analogous to calculating the sample mean plus or minus a margin of error in the non-time series, random sample setting. The typical scenario is that a forecasting model is developed which is then used to predict an outcome of interest. The forecast comprises parameter estimates which can be found using software in an effort to achieve optimal predictive ability. Notice that this is similar to linear regression methodology used for predicting a dependent variable. The difference is that linear regression modeling is based on independent errors and is thus inappropriate for time-dependent data. While some of the mathematical theory supporting time series methodology is quite complex, software is available that performs the heavy lifting. Also, it is convenient that a proposed time series forecasting model can often be "checked" (validated). In Chapters 6–11 these issues will be discussed in detail.

Key Point: We will use the terms *prediction* and *forecast* synonymously.

We will first introduce moving average forecasting, an intuitively simple method which can give reasonable 1-step ahead forecasts. A 1-step ahead forecast at time t is a forecast of x_{t+1} given data up to and including x_t. Exponential smoothing is a somewhat more sophisticated method of forecasting that also uses smoothing techniques. Finally, the Holt-Winters forecasting technique, an extension of exponential smoothing designed to account for trend and/or seasonality, is introduced.

Key Points

1. In later chapters, we will provide a comprehensive analysis of various "model-based" forecasting methodologies.
   - Model-based forecasting methods produce forecasts based on a particular mathematical model (including the autoregressive model) that has been fit to the data to describe the underlying physical process generating the data over time and involves certain distributional assumptions about residuals.[12]
2. The forecasting methods in this chapter are smoothing methods which are simply functions of the data that make few assumptions about the underlying physical process.

12 We will begin the discussion of model-based methods in Chapter 5.

2.2.1 Predictive Moving Average Smoother

In Section 2.1.1.1 we explained how the centered moving average smoother is used for visualizing a smoothed version of the data for purposes of "recovering" underlying signals or removing noise or seasonal effects. We can also use moving averages for prediction. Instead of the centered moving average discussed in Section 2.1.1.1, we will use the *predictive (rolling or trailing)* moving average for forecasting future values. If the data do not exhibit seasonality or trend, then to predict the value of the time series at time $t+1$, it makes sense to "predict" x_{t+1} to be the average of the previous k data values for some k. That is, letting $\tilde{x}_{t+1}$ denote the 1-step ahead prediction of x_{t+1} given data to time t, then a reasonable and very simple predictor would be $\tilde{x}_{t+1} = \left(x_t + x_{t-1} + \cdots + x_{t-k+1}\right)/k$. That is, the predictor of x_{t+1} is the average of the last k data values. Suppose, for example, that we want to use a 3-point moving average predictor for a data set of length $t = 20$. Note that the 1-step ahead prediction of x_4 using this predictor would be $\tilde{x}_4 = \left(x_3 + x_2 + x_1\right)/3$. Because the data set has 20 data values, there is a known value for x_4, so we can compare the predictor $\tilde{x}_4$ with the actual value, x_4, to evaluate the accuracy of the predictor. Using this procedure, we can predict the values x_4 up to x_n for which the prediction is $\tilde{x}_n = \left(x_{n-1} + x_{n-2} + x_{n-3}\right)/3$. Again, all of these 1-step ahead predictors can be compared with actual values to ascertain the quality of each prediction. Notice also, that we have observed the data values necessary to calculate $\tilde{x}_{n+1} = \left(x_n + x_{n-1} + x_{n-2}\right)/3$. In this case $\tilde{x}_{n+1}$ is a predictor of a future value which we presumably do not know. We can make an assessment about the quality of this prediction based on the predictions of $x_4, \ldots, x_n$, all of which can be "checked". These forecasts can be obtained using ***tswge*** function **`ma.pred.wge`**. The procedure is illustrated in the following example.

Key Point: A 1-step ahead forecast at time t is the forecast of x_{t+1} given data up to and including x_t.

Example 2.4 Moving Average Prediction of Monthly WTI Crude Prices 2018-2020

Figure 2.17(a) shows the monthly price of West Texas Intermediate crude oil for the years 2018−2020. During that time period the price varied dramatically from a low of $20 per /barrel to a high of about $70. The function **`ma.pred.wge`** below calculates 1-step ahead predictions using a fifth order moving average smoother. In the third command below the **`wtc36`** ***ts*** file is changed to a numeric vector so that the data will simply be indexed by 1−36 (months) for convenience.

```
data(wtcrude2020)
wtc36=window(wtcrude2020,start=c(2018,1))
wtc36=as.numeric(wtc36)
ma5=ma.pred.wge(wtc36,order=5,n.ahead=4)
```

The **`wtc36`** data and the 1-step ahead predictions, **`ma5$pred`**, are shown below (rounded to two decimal places)

```
round(wtc36,2)
 [1]  63.56  62.15 62.86  66.32  69.89  67.52 70.99 67.99  70.19  70.75 56.57 48.64
[13]  51.36  54.99 58.15  63.88  60.73  54.68 57.51 54.84  56.86  53.98 57.11 59.86
[25]  57.71  50.60 29.88  16.81  28.79  38.30 40.75 42.36  39.61  39.53 41.52 47.09

round(ma5$pred,2)
 [1]     NA    NA    NA    NA    NA 64.96 65.75 67.52 68.54 69.32 69.49 67.30
[13]  62.83 59.50 56.46 53.94 55.40 57.82 58.49 58.99 58.33 56.92 55.57 56.06
[25]  56.53 57.10 55.85 51.03 42.97 36.76 32.88 30.91 33.40 37.96 40.11 40.75
[37]  42.02 41.95 42.42 43.00
```

The **wtc36** data are plotted again in Figure 2.17(b) along with the moving average predictors listed above.

Notes

(a) Because the moving average predictor is fifth order, the data begin at $t = 6$.
(b) The prediction at $t = 6$, obtained using the first five data values is \$64.96 while the actual value at $t = 6$ was \$67.52. Additionally, at time $t = 12$, the predicted price was \$67.30 while the actual price was \$48.64, a drop in prices that was not anticipated by the moving average. In general, the moving average predictor lags behind dramatic changes in the behavior of the process.
(c) Predictions for $t = 6$ through $t = 36$ are 1-step ahead predictions.
(d) The **ma.pred.wge** command above specifies forecasts up to four steps *beyond* the end of the dataset. These use data only up to and including $t = 36$. Consequently, the prediction $\tilde{x}_{37}$ is the average of the last five data values, x_{32} through x_{36},which are observed. The forecast for time $t = 38$ is the average

$$\tilde{x}_{38} = \frac{x_{33} + x_{34} + x_{35} + x_{36} + \tilde{x}_{37}}{4}$$

The forecast $\tilde{x}_{38}$, which is based on data up to and including time $t = 36$, is called a 2-step ahead forecast. The predictions for $t = 38$ through $t = 40$ proceed in a similar manner, gradually increase, but will eventually level off.

Key Points

1. In general, a forecast of $x_{t+\ell}$ given data up to and including time t is called an ℓ-step ahead forecast.
2. Figure 2.17(b) shows one-step-ahead forecasts for times $t = 6$ through $n+1 = 41$.
3. Forecasts for $t = 42, 43$,and 44 are also shown. These forecasts are two, three, and four-step-ahead forecasts, respectively.
4. Note that the parameter **n.ahead** in the **ma.pred.wge** function specifies how many steps beyond the observed data we want to forecast.

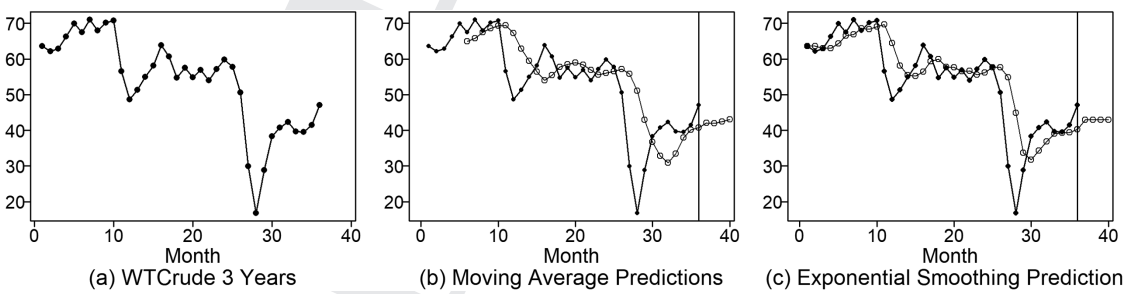


FIGURE 2.17 (a) Monthly price of West Texas Intermediate crude oil for the years 2018–2020, crude oil prices along with 1-step ahead predictors using a 5th-order smoother, and (c) one-step ahead predictions using exponential smoothing with $\alpha = .4$.

Key Points

1. The moving average predictions are based on "smoothing".
2. Thinking of the data value x_t as a sum $x_t = \mu_t + e_t$ (that is, a mean value plus a noise term), then at time t, instead of predicting the next *data value*, x_{t+1}, the prediction can be viewed as a predictor of the mean μ_{t+1}.

2.2.2 Exponential Smoothing

While moving average prediction is easy to conceptualize and calculate, it is unrealistic to assume that all preceding data values will have an equal amount of influence on future values. It is more intuitive to believe that in many cases more recent data should have more influence on future values than data in the more distant past because it is more reflective of the current state of reality. A smoothing method that takes this into consideration is called *exponential smoothing*. Exponential smoothing was first introduced by R.G. Brown (1956). To motivate the methodology, we again consider that each data value x_t in a time series comprises a mean value at each time point t and an independent error term with zero mean and constant variance. That is, $x_t = \mu_t + e_t$. For $0 \le \alpha \le 1$, the exponential smoothing recursion for $t = 1, 2, \ldots, n$ is[13]

$$u_{t+1} = \alpha x_t + (1-\alpha) u_t \tag{2.10}$$

The expression $\alpha x_t + (1-\alpha) u_t$ in (2.10) is a linear combination of the current value x_t along with u_t, which is the estimate of μ_t based on data up to but not including time t. "Standing at time t" the weighted average given in (2.10) is the predictor of μ_{t+1}. Note that the estimate u_t is heavily dependent on the smoothing parameter α which ranges from 0 to 1. The closer α is to 1, the more weight is given to most recent data while the closer α is to 0, the less weight is given to most recent data. Of course, the optimal value of α will depend on the particular application of the dataset and can be tuned accordingly.

Because the above formula is recursive, each new estimate u_t is dependent on the previous estimate u_{t-1}, which is dependent on the previous estimate u_{t-2}, and so on. This formula does not immediately appear to be "exponential", so how does the method get its name?

Letting $t = 1, 2, 3, 4$, we see that the recursive formula yields the following:

$$\begin{aligned}
u_1 &= x_1 \\
u_2 &= \alpha x_1 + (1-\alpha) u_1 \\
&= \alpha x_1 + (1-\alpha) x_1 \\
&= x_1 \\
u_3 &= \alpha x_2 + (1-\alpha) u_2 \\
&= \alpha x_2 + (1-\alpha) x_1 \\
u_4 &= \alpha x_3 + (1-\alpha) u_3 \\
&= \alpha x_3 + (1-\alpha)\left[\alpha x_2 + (1-\alpha) x_1\right]
\end{aligned}$$

QR 2.2 Exponential Smoothing Derivation

13 It is typical practice to set $u_1 = x_1$ to initiate the recursion.

$$= \alpha x_3 + \alpha(1-\alpha)x_2 + (1-\alpha)^2 x_1$$

$$u_5 = \alpha x_4 + (1-\alpha)u_4$$

$$= \alpha x_4 + (1-\alpha)\left[\alpha x_3 + \alpha(1-\alpha)x_2 + (1-\alpha)^2 x_1\right]$$

$$= \alpha x_4 + \alpha(1-\alpha)x_3 + \alpha(1-\alpha)^2 x_2 + (1-\alpha)^3 x_1$$

and in general,

$$u_k = \sum_{j=1}^{k-2} \alpha(1-\alpha)^{j-1} x_{k-j} + (1-\alpha)^{k-2} x_1. \tag{2.11}$$

It is clear from (2.11) that since α is between zero and one, the recursive formula gives less weight to earlier observations and more weight to the most recent observations, and that the magnitude of the weighting is exponential.

The ***tswge*** function **`exp.smooth.wge(x,alpha,n.ahead)`** performs exponential smoothing where **`x`** is the dataset, **`alpha`** is the value for α, and **`n.ahead`** is the number of steps beyond the end of the dataset you want to forecast (default=0). If you do not specify a value for **`alpha`**, the function will "pick" one based on a minimization of the one--step-ahead forecast errors. Because the goal is to forecast the mean level, μ_t, and the μ_ts are likely to be smoother than the data, minimizing one-step-ahead forecast errors may not be the best solution. Values of **`alpha`** between .2 and .4 are often used.

Example 2.4 (Revisited)

Figure 2.17(c) shows the **`wtc36`** data along with the u_t values. The figure shows one-step-ahead forecasts using exponential smoothing with $\alpha = .4$. The resulting predictors are similar to those in Figure 2.17(b) obtained using a 5th-order moving average predictor. Using the file **`wtc36`** that was created in Example 2.4, Figure 2.17(c) can be obtained using the command

```
wtc36.exp=exp.smooth.wge(wtc36,alpha=.4,n.ahead=4)
```

2.2.2.1 Forecasting with Exponential Smoothing beyond the Observed Dataset

A comment about exponential smoothing forecasts is in order. Recalling that for t within the time frame of the data, u_{t+1} is an estimate of the mean μ_{t+1} given data up to and including time t.When $t = n$, the end of the dataset, then u_{n+1} can be calculated using data up to and including $t = n$. That is, the prediction equation gives an estimate of μ_{n+1}. Because there is no assumed trend or seasonal effect in the data, exponential smoothing prediction beyond the end of the data is based on the assumption that, based on what we know at time $t = n$, the "best" estimate of $\mu_{n+k} = u_{n+1}$ for all $k \geq 1$. Figure 2.17(c) is a plot of forecasts for times $n+1,\ldots,n+4$. Note that the prediction for $t = n+1$ is slightly different from the prediction for time $t = n$. However, the predictions for times $t = n+2, n+3$, and $n+4$ are all the same as the prediction for $t = n+1$.

While these forecasts are unexciting and not appealing intuitively, the "conservative" nature of the forecasts can be the "best thing to do" with randomly wandering data like the West Texas crude oil prices. That is, predicting an upward trend to continue may not be a good idea if the trend is not persistent. Having said that, these predictions have a tendency to track poorly with upward or downward trends even when visual inspection, and possibly other information, suggest that the trend is continuing.[14] If a time series

14 The exponential smoothing forecasts are the optimal forecasts for data from "ARIMA(0,1,1)" models, which will be defined in Chapter 7. In future chapters we will make model identification decisions before using a particular model for prediction.

dataset exhibits trend or seasonality, simple exponential smoothing should not be used as a forecasting model. The Holt-Winters forecasting approach, to be discussed in Section 2.2.3, is designed to account for trend and seasonality.

Example 2.5 DFW Annual Temperature Data

Dataset **`dfw.yr`** contains the annual temperature data for the Dallas-Ft. Worth area for the years 1900 through 2020. As you can see, there is considerable year-to-year variability. Figures 2.18(a) and 2.18(b) show the 7th-order moving average predictor and the exponential smoothing predictor using $\alpha = .2$. These plots provide a better view of the fact that the predictors are predicting mean values. These smoothing-based predictors show that there seems to be a recent increase in temperatures.

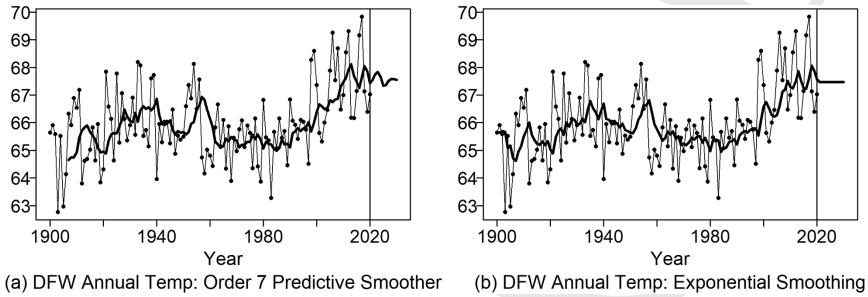


FIGURE 2.18 (a) DFW Annual Temperature data along with 7th-order moving average predictor and (b) DFW data with exponential smoothing predictor with $\alpha = 2$.

The plots were obtained using the code

```
data(dfw.yr)
exp.smooth.wge(dfw.yr,alpha=.2,n.ahead=10)
```

QR 2.3 Exponential Smoothing Example

Notice the forecasts beyond the end of the series. The moving average forecasts show some pattern but will eventually become horizontal while the exponential smoothing forecasts are horizontal from the beginning since all forecasts are equal to the initial forecast for $n+1$.

2.2.3 Holt-Winters Forecasting

The Holt-Winters approach is a technique developed by economists Holt (1957) and Winters (1960). This method was designed to generalize exponential smoothing to account for trend and seasonality. As mentioned above, this method is much more adept at tracking time series with trend and/or seasonality than is simple exponential smoothing. As with decomposition and seasonal adjustment, there are additive and multiplicative versions of Holt-Winters forecasting.

2.2.3.1 Additive Holt-Winters Equations

In this section we consider prediction when an additive model (see (2.4)) is appropriate. The formulas for Holt-Winters forecasting are generalizations of exponential smoothing, and the Holt-Winters equations are:

$$\begin{aligned} u_t &= \alpha(x_t - s_{t-m}) + (1-\alpha)(u_{t-1} + v_{t-1}) \\ v_t &= \beta(u_t - u_{t-1}) + (1-\beta)v_{t-1} \\ s_t &= \gamma(x_t - u_{t-1}) + (1-\gamma)s_{t-m} \end{aligned} \tag{2.12}$$

where $0 < \alpha \le 1$, $0 \le \beta \le 1$, and $0 < \gamma \le 1 - \alpha$, and where m is the period length (for example, frequency in a *ts* file). For monthly data, $m = 12$, and for quarterly data, $m = 4$. The u_ts are related to simple exponential smoothing and provide a baseline. The v_ts and s_ts relate to trend and seasonal effects, respectively.

For times $t = m+1,\ldots,n$, the 1-step ahead forecasts, $\hat{x}_t$, for the mean at time t,are given by

$$\hat{x}_t = u_{t-1} + v_{t-1} + s_{t-m.}$$

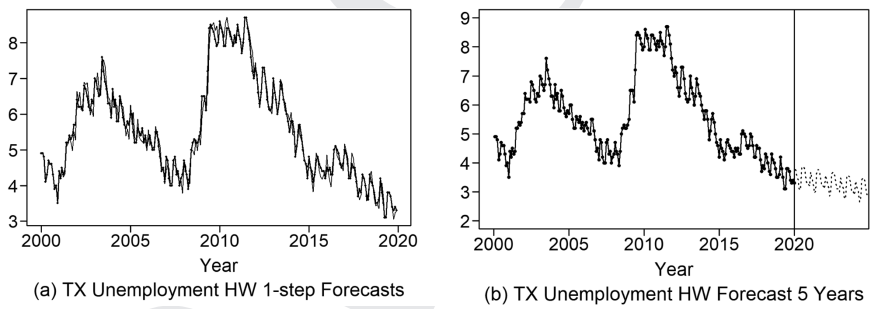
Forecasts for $x_{n+\ell}, \ell = 1,\ldots,K$ (that is, up to K steps beyond the end of the observed data), are given recursively by

$$\hat{x}_{n+\ell|n} = u_n + mv_n + s_{n+\ell-m\ell'.}$$

where $\ell' = [((\ell-1)/m)]+1$ with $[((\ell-1)/m)]$ denoting the greatest integer less than or equal to $(\ell-1)/m$. Here, α, β, and γ are smoothing parameters, and can be obtained using R commands. Notice that each of the three Holt-Winters equations follow the general exponential smoothing form. The only reasonable method for implementing the Holt-Winters computations is by using a computer. R has several packages that contain functions for performing the calculations. We will use the function **`HoltWinters`** which is in Base R.[15]

Example 2.6 Texas Unemployment Rates

In this example we will apply Holt-Winters forecasting to the non-seasonally adjusted Texas Unemployment data shown in Figure 2.12(a). We earlier obtained an "additive" decomposition and performed additive seasonal adjustment on this dataset. In this example additive Holt-Winters forecasts will be obtained. Figure 2.19(a) shows the data overlayed by the 1-step ahead Holt-Winters forecasts. The two curves essentially lie on top of each other and are indistinguishable. Figure 2.19(b) shows forecasts for five years beyond the end of the dataset. There it can be seen that the seasonal and trending behavior are forecast to continue. For more information see Holt (1957) and Winters (1960).



(a) TX Unemployment HW 1-step Forecasts

(b) TX Unemployment HW Forecast 5 Years

FIGURE 2.19 (a) Texas Unemployment data with 1-step ahead Holt-Winters forecasts and (b) Texas Unemployment data for 2000–2019 and Holt-Winters forecasts for the next five years.

2.2.3.2 Multiplicative Holt-Winters Equations

As the term implies, the multiplicative Holt-Winters equations are applicable to data for which the multiplicative model in (2.5) is appropriate. In this case, the Holt-Winters equations are:

$$u_t = \alpha\left(x_t / s_{t-m}\right) + (1-\alpha)\left(u_{t-1} + v_{t-1}\right)$$

15 The values of α, β, and γ are chosen by the **`HoltWinters`** function and are provided as output.

$$v_t = \beta(u_t - u_{t-1}) + (1-\beta)v_{t-1} \tag{2.13}$$
$$s_t = \gamma(x_t / u_{t-1}) + (1-\gamma)s_{t-m}$$

where $0 < \alpha \le 1, 0 \le \beta \le 1$, and $0 < \gamma \le 1 - \alpha$, and where again, m is the frequency.

Predictions for x_t values are given by

$$\hat{x}_t = (u_{t-1} + v_{t-1})s_{t-m.}$$

Forecasts for $x_{n+\ell}, \ell = 1,\ldots,K$ (that is, up to K steps beyond the end of the observed data) are given recursively by

$$\hat{x}_{n+\ell|n} = (u_n + \ell v_n)s_{n+\ell-m\ell'.} \tag{2.14}$$

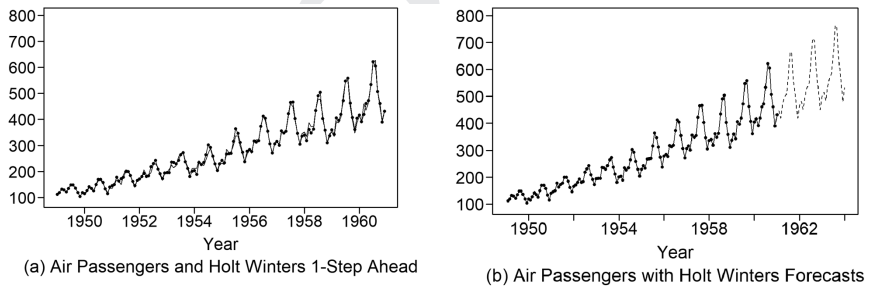
where $\ell' = [((\ell - 1)/m)] + 1$.

Example 2.7 Air Passengers data

Consider once again the **AirPassengers** data, first plotted in this chapter in Figure 2.7(a). Because later time points reveal increasing magnitudes of the cyclic peaks and troughs, the time series is multiplicative rather than additive. Figure 2.20(a) shows the one-step-ahead Holt-Winters forecasts for years 1950–1960 (superimposed on actual data) given by the formula using the estimated smoothing parameters and coefficients above. The one-step-ahead forecasts (solid line) are quite accurate and are barely distinguishable from the data (points on the line). Figure 2.20(b) shows the **AirPassengers** data along with Holt-Winters forecasts for the next three years (dotted line). These forecasts appear to accurately extend the preceding pattern. The following code produces the plots in Figure 2.20.

```
# Figure 2.20(a)
ap.hw=HoltWinters(AirPassengers,seasonal="mult")
plot(ap.hw)
# Figure 2.20(b)
ap.pred=predict(ap.hw,n.ahead=36)
plot(ap.hw,ap.pred,lty=1:2)
```

QR 2.4 Holt Winters Example



(a) Air Passengers and Holt Winters 1-Step Ahead

(b) Air Passengers with Holt Winters Forecasts

FIGURE 2.20 (a) Air Passengers data with 1-step ahead Holt-Winters predictions and (b) Air Passengers data with Holt-Winters forecasts for years 1961–1963.

2.2.4 Assessing the Accuracy of Forecasts

The forecasts in Figures 2.19(b) and 2.20(b) "look" like they are good predictors. However, the "acid" test is to compare forecasts with actual values. Consider the **AirPassengers** data, which are for the 144 months from January 1949 through December 1960. (And we do not know the actual values for the following years.) Instead of forecasting 36 months "beyond the known" data, a common method for

assessing accuracy in this situation is, for example, to base the forecasts on the first nine years (108 months) and "forecast" the next 36 months. In this example, the first nine years would be considered to be the dataset from which forecasts were produced. The forecasts can then be compared with the known values that were "held out". Common terminology is to refer to the first nine years in this example as the *training data* and the last three years as the *test data*. The code

```
AP9=window(AirPassengers,start=c(1949,1),end=c(1957,12))
AP9.hw=HoltWinters(AP9,seasonal="mult")
AP9.pred=predict(AP9.hw,n.ahead=36)
plot(AirPassengers,type='l')
points(AP9.pred,type='l',lty=2)
```

produces Figure 2.21(a), which gives a visual comparison between the forecasts (dashed lines) and actual values for the last three years.

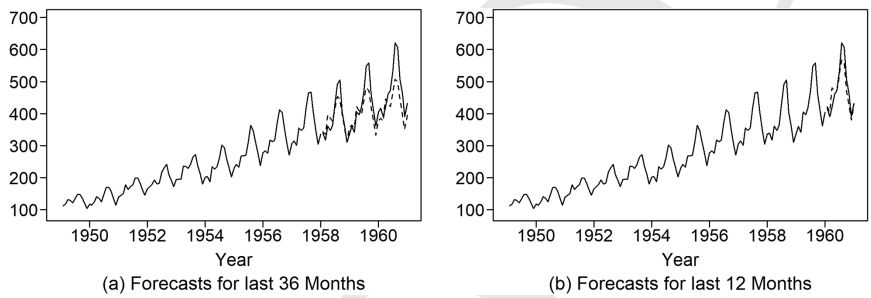


FIGURE 2.21 (a) Forecasts and actual values for the last 36 months of the Air Passenger data and (b) same as (a) for the last 12 months.

The forecasts and actual values for months 109–144 are given below:

```
Actual=window(AirPassengers,start=c(1958,1))
Actual
```

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1958	340	318	362	348	363	435	491	505	404	359	310	337
1959	360	342	406	396	420	472	548	559	463	407	362	405
1960	417	391	419	461	472	535	622	606	508	461	390	432

```
Pred=AP9.pred
Round(Pred,0)
```

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1958	344	336	398	386	376	419	454	447	398	353	314	357
1959	365	357	422	409	399	444	480	473	421	374	332	377
1960	386	377	446	432	421	469	507	500	444	395	351	398

Creating the R vector **Error** we obtain the following:

```
Error=Pred-Actual
round(Error,0)
```

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1958	4	18	36	38	13	-16	-37	-58	-6	-6	4	20
1959	5	15	16	13	-21	-28	-68	-86	-42	-33	-30	-28
1960	-31	-14	27	-29	-51	-66	-115	-106	-64	-66	-39	-34

Some differences are positive and some are negative. A common method of obtaining an "overall" quantification of the quality of the predictions is to find the mean square error (MSE), which eliminates the issue

with signs. Letting x_t and f_t represent the actual and predicted values at time t,respectively, then the MSE for the forecasts above is

$$MSE = \frac{1}{36}\sum_{t=1}^{36}\left(f_t - x_t\right)^2. \tag{2.15}$$

The MSE in this case can be found using the R command

```
MSE=mean(Error^2)
```

which is 2010.37. This value is in squared units and is not intuitively meaningful. For this reason, it is useful to take the square root of the MSE, which is called the *root mean square error* (RMSE). The RMSE is in the units of the problem. For the forecasts above, the RMSE is given by

```
RMSE=sqrt(MSE)
```

which is 44.84. Examining the differences, this is sort of an average amount by which forecasts differ from true values. Figure 2.21(b) shows the forecasts for the last year (12 months). Visually the forecasts look quite good, and in this case the RMSE is 31.98.

In this book we will use the RMSE to assess forecast performance and to compare forecasts from different models or methods. Another measure of forecast quality that has intuitive appeal is the mean absolute deviation, which is the average of the absolute values of the errors. Specifically,

$$MAD = \frac{1}{36}\sum_{t=1}^{36}\left|f_t - x_t\right|. \tag{2.16}$$

In Problem 2.5 you are asked to find the MAD for the above forecasts.

2.3 CONCLUDING REMARKS

In this chapter some preliminary time series data analysis techniques were introduced and illustrated. It was shown that applying various smoothing techniques can make time series data easier to visualize, interpret, and analyze. A thorough description of the decomposition of a time series dataset was presented, and the resulting applications related to decomposition were considered. It is common for a data analyst to encounter seasonally adjusted data; this term and related methods were defined and illustrated using examples. Finally, one of the most important applications of time series data analysis is that of forecasting. While in the chapter we introduced non model-based forecasting concepts using smoothing, we will consider a more theoretical approach in Chapters 5–10 when we study model-based forecasting. In Chapter 11 we will return to non model-based methods in the discussion of neural network analysis of time series data.

APPENDIX 2A

TSWGE FUNCTIONS

(a) **`exp.smooth.wge(x,alpha=NULL,n.ahead=0,plot=TRUE)`:** Computes exponential smoothing forecasts

x is a vector containing the time series
alpha is the α parameter.
n.ahead is the number of steps beyond the end of the dataset that you want to forecast (default=0)
plot=TRUE (default) produces a plot of the data overlayed with the 1-step ahead forecasts.
if **n.ahead** > 0, the forecasts beyond the dataset are plotted with dashed line.

Output

alpha
forecasts: The exponential smoothing 1-step ahead forecasts for $t = 1,\ldots,n$ and forecasts beyond n, if **n.ahead>0**.

Note: You can specify **alpha,** or if you leave **alpha** out of the call statement, the function will pick α by minimizing 1-step ahead forecast errors. This is not usually the goal of exponential smoothing; selecting values of **alpha** somewhere in the range from .2 to .4 is common practice.

(b) **ma.pred.wge(x,order=3,k.ahead=1,plot=TRUE):** computes a predictive (rolling) moving average on the data in **x**

x is a vector containing the time series
order is the order of the predictive moving average (default=3)
plot=TRUE (default) produces a plot of the data along with the predictions.

Example

```
data(AirPassengers)
ma.pred.wge(AirPassengers,order=5,k.ahead=20)
```

provides 1-step ahead predictions for the data up to x_{n+1} where n is the length of the **AirPassengers** data. Also, using the **AirPassengers** data, the function extends the predictions to x_{n+20}. The data and the predictions will be plotted.

(c) **ma.smooth.wge(x,order=3,plot=TRUE):** computes a centered moving average smoother on the data in **x**
x is a vector containing the time series
order is the order of the centered moving average smoother (default=3)
plot=TRUE (default) produces a plot of the data along with the smoothed version

Example

```
data(AirPassengers)
ma.smooth.wge(AirPassengers,order=5)
```

PROBLEMS

1. Use a 5th-order moving average smoother to smooth the following. In each case plot the smoothed data as an overlay on the original data:
   (a) **bitcoin**
   (b) **wtcrude2020**
2. Using the ***tswge ts*** object **dfw.mon** do the following:
   (a) Use the Base R window function to create a ***ts*** object consisting of the monthly temperatures from January 2000 through December 2020. Name the file **dfw.mon.2000**.

(b) Apply a 5th-order moving average smoother to the data in **dfw.mon.2000**. Plot the monthly data and overlay the smoothed data. Describe the appearance of the smoothed data.
(c) Apply an additive decomposition to **dfw.mon.2000** using the techniques in Section 2.1.2.1 and plot the four plots as in Figure 2.10.
(d) Base R has a function called **decompose** that performs an additive or multiplicative decompostion. Issue the following commands:

```
dfw.decomp=decompose(dfw.mon.2000,type='additive')
plot(dfw.decomp)
Compare the plots with those you produced in (c).
```

3. Using the **decompose** function produce a plot corresponding to the multiplicative decomposition shown in Figure 2.14 for the **AirPassengers** data. Use the command

```
decompose(AirPassengers,type='multiplicative')
```

4. ***tswge ts*** object **us.retail** is a listing of the quarterly US retail sales (in $millions) from the fourth quarter of 1999 through the second quarter of 2021.
   (a) Plot the data.
   (b) Discuss behavior of the time series.
   (c) Decompose the data (additive or multiplicative – you decide).
   (d) These data are not seasonally adjusted. Seasonally adjust the data using either (2.8) or (2.9) and plot your results. Does the seasonal adjustment highlight behavior not noticed in the non-seasonally adjusted data?
5. Use MAD to measure the accuracy of the **AirPassengers** forecasts obtained in Section 2.2.4.
6. On each of the following datasets use the strategy in Section 2.2.3 to find Holt-Winters forecasts for the last k data values in the dataset where you choose k. In each case
   (i) Fit an *additive model* to the data and use it to forecast the last k values.
   (ii) Fit a *multiplicative model* to the data and use it to forecast the last k values.
   (iii) For each dataset calculate the RMSEs for your forecasts and use this as a measure to assess which forecasting technique (additive or multiplicative) performs best.
      (a) **AirPassengers** data
      (b) log **AirPassengers** data
      (c) **lynx** data
      (d) log **lynx** data
      (e) **tx.unempl.unadj**
      (f) **us.retail**

tswge Datasets Introduced in Chapter 2

tx.unemp.unadj – Monthly Texas Unemployment Rates from 2000 through 2019 (not seasonally adjusted)

tx.unemp.adj – Monthly Texas Seasonally Adjusted Unemployment Rates from 2000 through 2019

us.retail – quarterly US retail sales (in $millions) from the fourth quarter of 1999 through the second quarter of 2021