Time series decomposition (cont) and ETS (Intro)

Forecasting: principles and practice

book by Rob Hyndman and George Athanasopoulos
slides by Peter Fuleky, updated for the 2nd edition by by Joseph Alba

Moving averages

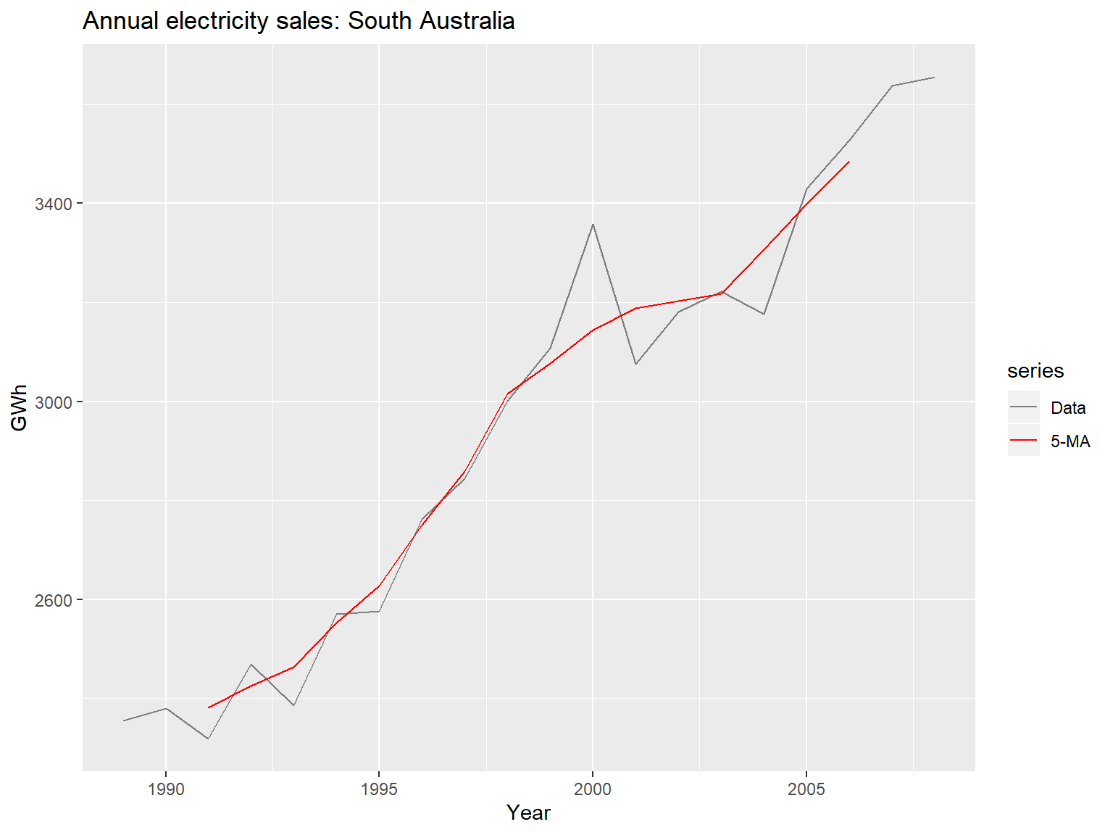
The classical method of time series decomposition originated in the 1920s. It still forms the basis of time series methods, and so it is important to understand how it works. The first step in a classical decomposition is to use a moving average method to estimate the trend-cycle.

A moving average of order m can be written as

$$\hat{T}_t = \frac{1}{m} \sum_{j=-k}^{k} y_{t+j},$$

where $m = 2k + 1$. At time t it is obtained by averaging values of the time series within k periods of t. We call this an m-MA meaning a moving average of order m. The moving average eliminates some of the randomness, giving a smooth result.

```r
autoplot(elecsales, series="Data") +
  forecast::autolayer(ma(elecsales,5), series="5-MA") +
  xlab("Year") + ylab("GWh") +
  ggtitle("Annual electricity sales: South Australia") +
  scale_colour_manual(values=c("Data"="grey50","5-MA"="red"),
                      breaks=c("Data","5-MA"))
```

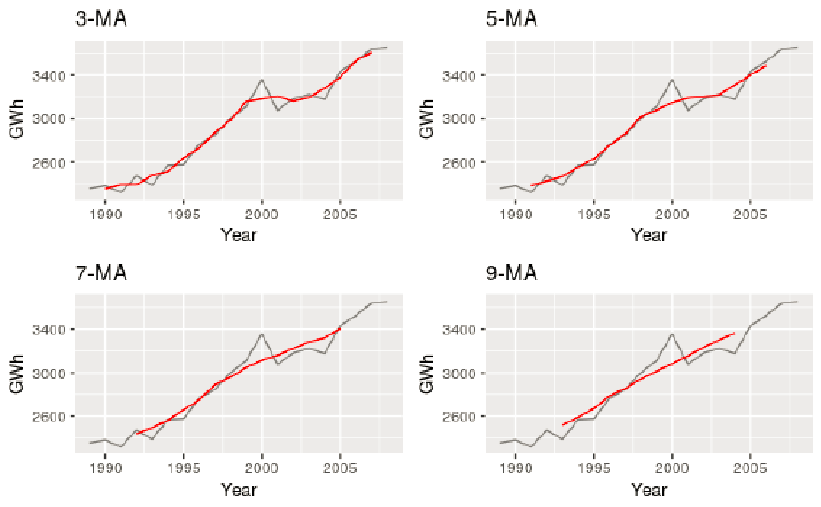

Annual electricity sales: South Australia
GWh
3400
3000
2600
1990
1995
2000
2005
Year
series
Data
5-MA

The data are shown in the next slide use the following commands (change rbind to cbind and rownames to colnames if you wish to reproduce Table 6.1):

```
ma5 <- ma(elecsales, 5)
table.out <- rbind(elecsales, ma5)
rownames(table.out) <- c("sales(GWh)", "5-MA")
table.out
```

```
##                [,1]    [,2]    [,3]     [,4]     [,5]     [,6]    [,7]
## sales(GWh) 2354.34 2379.71 2318.52 2468.990 2386.090 2569.470 2575.72
## 5-MA            NA      NA 2381.53 2424.556 2463.758 2552.598 2627.70
##                 [,8]     [,9]    [,10]  [,11]   [,12]  [,13]   [,14]
## sales(GWh) 2762.720 2844.500 3000.700 3108.1 3357.50 3075.7 3180.60
## 5-MA       2750.622 2858.348 3014.704 3077.3 3144.52 3188.7 3202.32
##               [,15]    [,16]    [,17]    [,18]   [,19] [,20]
## sales(GWh) 3221.60 3176.200 3430.600 3527.480 3637.89  3655
## 5-MA       3216.94 3307.296 3398.754 3485.434      NA    NA
```

The order of the moving average determines the smoothness of the trend-cycle estimate. In general, a larger order means a smoother curve.



Effect of the order of the moving average on smoothness

Note: Simple moving averages such as these are usually of an odd order (e.g., 3, 5, 7, etc.) This is so they are symmetric: in a moving average of order $m = 2k + 1$, there are earlier observations, later observations and the middle observation that are averaged. But if it was even, it would no longer be symmetric.

Moving averages of moving averages

It is possible to apply a moving average to a moving average. One reason for doing this is to make an even-order moving average symmetric. `

For example, we might take a moving average of order 4, and then apply another moving average of order 2 to the results.

```
beer2 <- window(ausbeer,start=1992)
ma4 <- ma(beer2, order=4, centre=FALSE)# this cmd gives non-centred ma4
ma2x4 <- ma(beer2, order=4, centre=TRUE)# default. gives ma2X4
```

Looking at the data:

```
table.out <- cbind(beer2, ma4, ma2x4)
colnames(table.out) <- c("data", "4-MA", "2x4-MA")
table.out
```

```
##         data   4-MA  2x4-MA
## 1992 Q1  443     NA      NA
## 1992 Q2  410 451.25      NA
## 1992 Q3  420 448.75 450.000
## 1992 Q4  532 451.50 450.125
## 1993 Q1  433 449.00 450.250
## 1993 Q2  421 444.00 446.500
## 1993 Q3  410 448.00 446.000
## 1993 Q4  512 438.00 443.000
## 1994 Q1  449 441.25 439.625
## 1994 Q2  381 446.00 443.625
## 1994 Q3  423 440.25 443.125
## 1994 Q4  531 447.00 443.625
## 1995 Q1  426 445.25 446.125
## 1995 Q2  408 442.50 443.875
## 1995 Q3  416 438.25 440.375
## 1995 Q4  520 435.75 437.000
## 1996 Q1  409 431.25 433.500
## 1996 Q2  398 428.00 429.625
## 1996 Q3  398 433.75 430.875
## 1996 Q4  507 433.75 433.750
## 1997 Q1  432 435.75 434.750
## 1997 Q2  398 440.50 438.125
## 1997 Q3  406 439.50 440.000
## 1997 Q4  526 439.25 439.375
## 1998 Q1  428 438.50 438.875
## 1998 Q2  397 436.25 437.375
## 1998 Q3  403 438.00 437.125
## 1998 Q4  517 434.50 436.250
## 1999 Q1  435 439.75 437.125
## 1999 Q2  383 440.75 440.250
## 1999 Q3  424 437.25 439.000
## 1999 Q4  521 442.00 439.625
## 2000 Q1  421 439.50 440.750
## 2000 Q2  402 434.25 436.875
## 2000 Q3  414 441.75 438.000
## 2000 Q4  500 436.25 439.000
## 2001 Q1  451 436.75 436.500
## 2001 Q2  380 434.75 435.750
## 2001 Q3  416 429.00 431.875
## 2001 Q4  492 436.00 432.500
## 2002 Q1  428 433.50 434.750
## 2002 Q2  408 437.00 435.250
## 2002 Q3  406 438.75 437.875
## 2002 Q4  506 431.75 435.250
## 2003 Q1  435 435.50 433.625
## 2003 Q2  380 431.50 433.500
## 2003 Q3  421 431.50 431.500
## 2003 Q4  490 434.00 432.750
## 2004 Q1  435 431.75 432.875
## 2004 Q2  390 422.75 427.250
## 2004 Q3  412 418.00 420.375
## 2004 Q4  454 421.25 419.625
## 2005 Q1  416 420.25 420.750
## 2005 Q2  403 427.25 423.750
## 2005 Q3  408 432.75 430.000
## 2005 Q4  482 428.50 430.625
## 2006 Q1  438 427.75 428.125
## 2006 Q2  386 430.00 428.875
## 2006 Q3  405 427.25 428.625
## 2006 Q4  491 426.50 426.875
## 2007 Q1  427 423.75 425.125
## 2007 Q2  383 419.25 421.500
## 2007 Q3  394 417.50 418.375
## 2007 Q4  473 419.25 418.375
## 2008 Q1  420 423.25 421.250
## 2008 Q2  390 427.00 425.125
## 2008 Q3  410 425.75 426.375
## 2008 Q4  488 427.75 426.750
## 2009 Q1  415 430.00 428.875
## 2009 Q2  398 430.00 430.000
## 2009 Q3  419 429.75 429.875
## 2009 Q4  488 423.75 426.750
## 2010 Q1  414     NA      NA
## 2010 Q2  374     NA      NA
```

The notation 2×4-MA in the last column means a 4-MA followed by a 2-MA.

The values in the last column are obtained by taking a moving average of order 2 of the values in the previous column. For example, the first two values in the 4-MA row are 451.2=(443+410+420+532)/4 and 448.8=(410+420+532+433)/4. The first value in the 2×4-MA row is the average of these two: 450.0=(451.2+448.8)/2.

When a 2-MA follows a moving average of even order (such as 4), it is called a centered moving average of order 4. This is because the results are now symmetric. (note that this is implemented automatically in ma() by using the option centre=true). This is the default if option centre is omitted.

To see that this is the case, we can write the 2×4-MA as follows:

$$\begin{aligned}\hat{T}_t &= \frac{1}{2}\Big[\frac{1}{4}(y_{t-2} + y_{t-1} + y_t + y_{t+1}) \\ &\quad + \frac{1}{4}(y_{t-1} + y_t + y_{t+1} + y_{t+2})\Big] \\ &= \frac{1}{8}y_{t-2} + \frac{1}{4}y_{t-1} + \frac{1}{4}y_t + \frac{1}{4}y_{t+1} + \frac{1}{8}y_{t+2}.\end{aligned}$$

It is now a weighted average of observations, but it is symmetric.

In general, an even order MA should be followed by an even order MA to make it symmetric. Similarly, an odd order MA should be followed by an odd order MA (e.g. 3×3-MA).

Estimating the trend-cycle with seasonal data

The most common use of centered moving averages is in estimating the trend-cycle from seasonal data. 2×4-MA:

$$\hat{T}_t = \frac{1}{8}y_{t-2} + \frac{1}{4}y_{t-1} + \frac{1}{4}y_t + \frac{1}{4}y_{t+1} + \frac{1}{8}y_{t+2}.$$

When applied to quarterly data, each quarter of the year is given equal weight as the first and last terms apply to the same quarter in consecutive years.

Consequently, the seasonal variation will be averaged out and the resulting values of $\hat{T}_t$ will have little or no seasonal variation remaining. A similar effect would be obtained using a 2×8-MA or a 2×12-MA.

In general, a $2 \times m$-MA is equivalent to a weighted moving average of order $m+1$ with all observations taking weight $1/m$ except for the first and last terms which take weights $1/(2m)$ (more on this later).

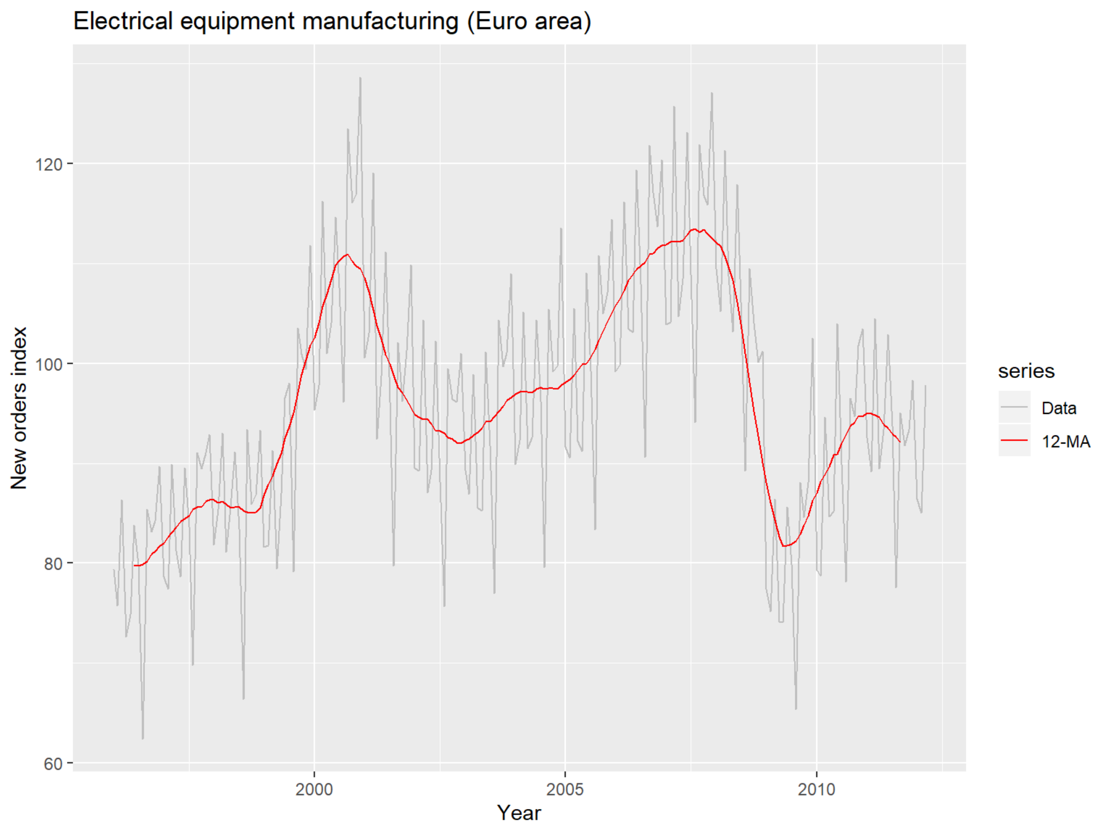
So if the seasonal period is even and of order m, use a $2 \times m$-MA to estimate the trend-cycle. If the seasonal period is odd and of order m, use a m-MA to estimate the trend cycle.

For monthly data, we have an even-order MA so we centre it by taking 2×12-MA which could be used to estimate the trend-cycle of monthly data

For daily data, we have an odd-order 7-MA which could be used to estimate the trend-cycle of daily data.

Other choices for the order of the MA will usually result in trend-cycle estimates being contaminated by the seasonality in the data.

```r
autoplot(elecequip, series="Data") +
  autolayer(ma(elecequip, 12, centre = FALSE), series="12-MA") +
  xlab("Year") + ylab("New orders index") +
  ggtitle("Electrical equipment manufacturing (Euro area)") +
  scale_colour_manual(values=c("Data"="grey","12-MA"="red"),
                      breaks=c("Data","12-MA"))
```



Trend-cycle is contaminated by some seasonality

Weighted moving averages

Combinations of moving averages result in weighted moving averages. For example, the 2×4-MA discussed above is equivalent to a weighted 5-MA with weights given by $[\frac{1}{8}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8}]$. In general, a weighted m-MA can be written as

$$\hat{T}_t = \sum_{j=-k}^{k} a_j y_{t+j},$$

where $k = (m-1)/2$ and the weights are given by $[a_{-k}, \dots, a_k]$. It is important that the weights all sum to one and that they are symmetric so that $a_j = a_{-j}$.

The simple m-MA is a special case where all the weights are equal to $1/m$. A major advantage of weighted moving averages is that they yield a smoother estimate of the trend-cycle. Instead of observations entering and leaving the calculation at full weight, their weights are slowly increased and then slowly decreased resulting in a smoother curve.

Name	a0	a1	a2	a3	a4	a5	a6	a7	a8
3-MA	.333	.333							
5 MA	.200	.200	.200						
2x12-MA	.083	.083	.083	.083	.083	.083	.042		
3x3-MA	.333	.222	.111						
3x5-MA	.200	.200	.133	.067					

The weighted moving average above is a linear filter that converts the observed time series into an estimate of the trend $\hat{T}_t$ through a linear operation (see above formula).

Other filters such as the Spencer weighted moving average and Henderson weighted moving average that leave higher order polynomials untouched (i.e. Henderson W MA preserves the cubic polynomial while Spencer WMA preserves the quadratic polynomial). These filters are typically incorporated in the seasonal adjustment packages available in R (eg., Henderson's WMA is used in the X-11 family of seasonal packages to be discussed later in this lecture). More on this next week ..

Classical decomposition

The classical decomposition method originated in the 1920s.

There are two forms of classical decomposition: an additive decomposition and a multiplicative decomposition.

In classical decomposition, we assume the seasonal component is constant from year to year. The m values are sometimes called the seasonal indices: (e.g., m=4 for quarterly data, m=12 for monthly data, m=7 for daily data with a weekly pattern).

Additive decomposition

1. If m is an even number, compute the trend-cycle component using a $2 \times m$-MA to obtain $\hat{T}_t$. If m is an odd number, compute the trend-cycle component using an m-MA to obtain $\hat{T}_t$.
2. Calculate the detrended series: $y_t - \hat{T}_t$.
3. To estimate the seasonal component for each month, average the detrended values for a particular month. For example, with monthly data, the seasonal component for March is the average of all the detrended March values in the data. These seasonal indexes are then adjusted to ensure that they add to m. The seasonal component is obtained by stringing together all the seasonal indices (for monthly data- Jan to Dec) for each year of data. This gives $\hat{S}_t$.

4. The remainder component is calculated by subtracting the estimated seasonal and trend-cycle components: $\hat{R}_t = y_t - \hat{T}_t - \hat{S}_t$.

Multiplicative decomposition

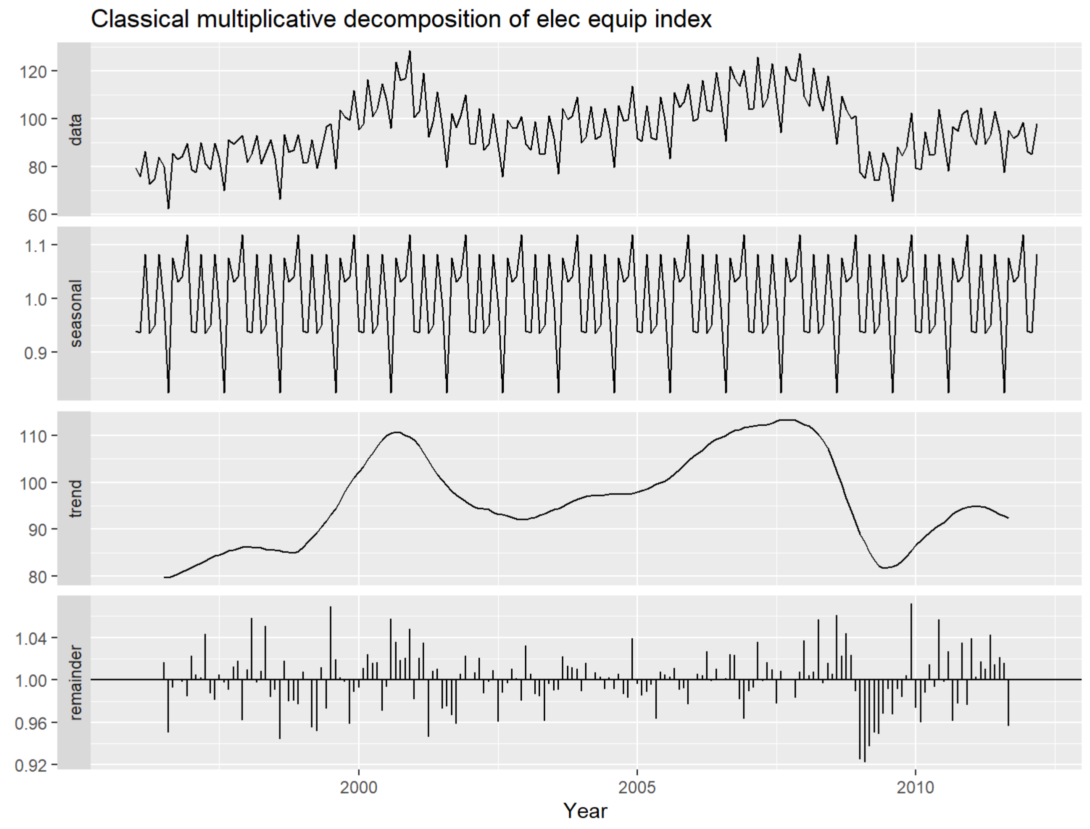
A classical multiplicative decomposition is very similar except the subtractions are replaced by divisions. Steps 1 and 3 are as described for additive decomposition.

Step 2: Calculate the detrended series: $y_t/\hat{T}_t$.

Step 4: The remainder component is calculated by dividing out the estimated seasonal and trend-cycle components: $\hat{R}_t = y_t/(\hat{T}_t\hat{S}_t)$.

We can plot the classical multiplicative decomposition using the pipe operator (%>%) below

```
elecequip %>% decompose(type="multiplicative") %>%
autoplot() + xlab("Year") +
ggtitle("Classical multiplicative decomposition of elec equip index")
```


Classical multiplicative decomposition of elec equip index
data
seasonal
trend
remainder
120
100
80
60
1.1
1.0
0.9
110
100
90
80
1.04
1.00
0.96
0.92
2000
2005
2010
Year

Step 3 See example in Excel

Comments on classical decomposition

The estimate of the trend is unavailable for the first few and last few observations. For example, if m=12, there is no trend estimate for the first six and last six observations. Consequently, there is also no estimate of the remainder component for the same time periods.

Classical decomposition methods assume that the seasonal component repeats from year to year. For many series, this is a reasonable assumption, but for some longer series it is not. The classical decomposition methods are unable to capture these seasonal changes over time.

Occasionally, the values of the time series in a small number of periods may be particularly unusual. The classical method is not robust to these kinds of unusual values.

X-11 decomposition

The X-11 method is based on classical decomposition, but with many extra steps and features to overcome the drawbacks of classical decomposition.

- In particular, the trend-cycle estimates are available for all observations including the end points, and the seasonal component is allowed to vary slowly over time.
- It also has methods to handle trading-day variation, holiday effects and known-predictor effects. X-11 handles both additive and multiplicative decomposition and is robust to outliers and level shifts in time series.

The details of the X11 method are described in Dagum and Bianconcini (2016).

A short description could be found at the ABS wesite (http://www.abs.gov.au/websitedbs/d3310114.nsf/4a256353001af3ed4b2562bb00121564/c890aa8e65957397ca256ce10018c9d8!OpenDocument)

The X11 method is available using the seas function from the seasonal package for R

As noted from the ABS document, statistical offices of other countries such as US, UK and New Zealand use X12-ARIMA/ X13-ARIMA SEATS.

For your interest:

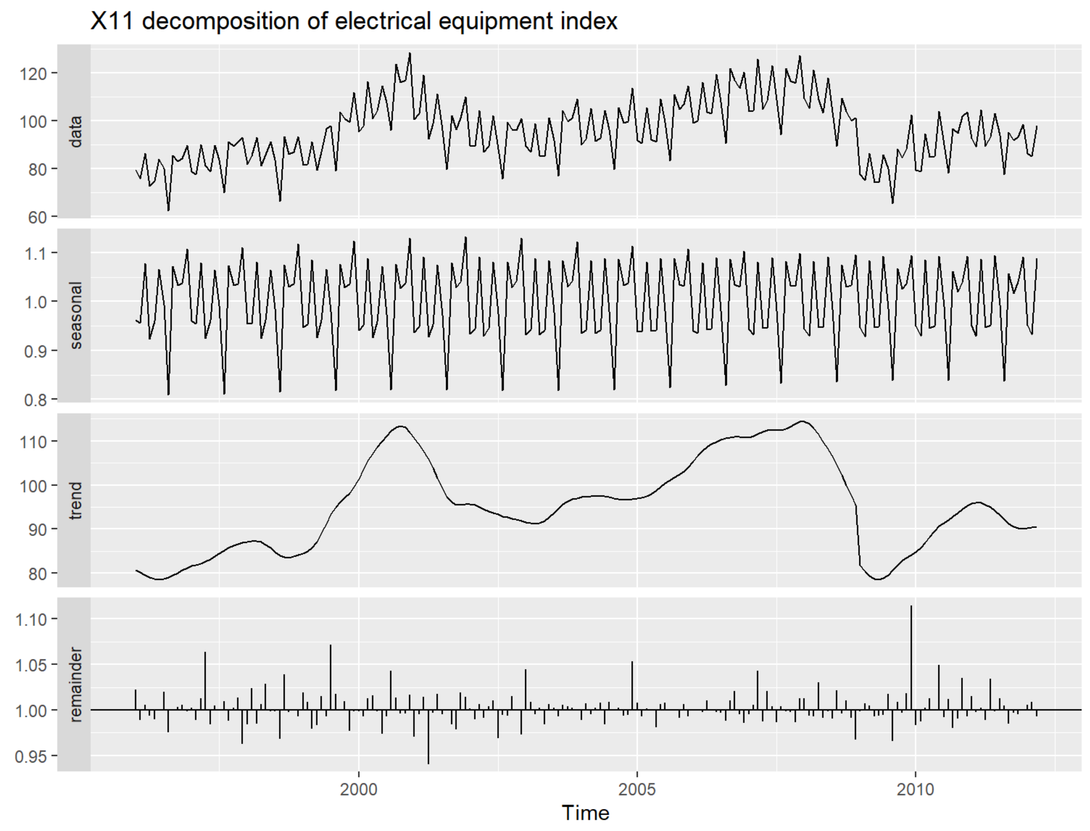
X12 ARIMA / X13 ARIMA SEATS may be installed by using 'install' and typing in 'X12'

Information on the X12 package could be accessed from (https://cran.r-project.org/web/packages/x12/x12.pdf)

Singapore also uses X12 ARIMA (https://www.singstat.gov.sg/-/media/files/publications/reference/ip-e32.pdf)

Example: X11 decomposition on electric equipment index

```r
library(fpp2)
library(seasonal)
fit2<-seas(elecequip, x11="")
  autoplot(fit2) +
  ggtitle("X11 decomposition of electrical equipment index")
```

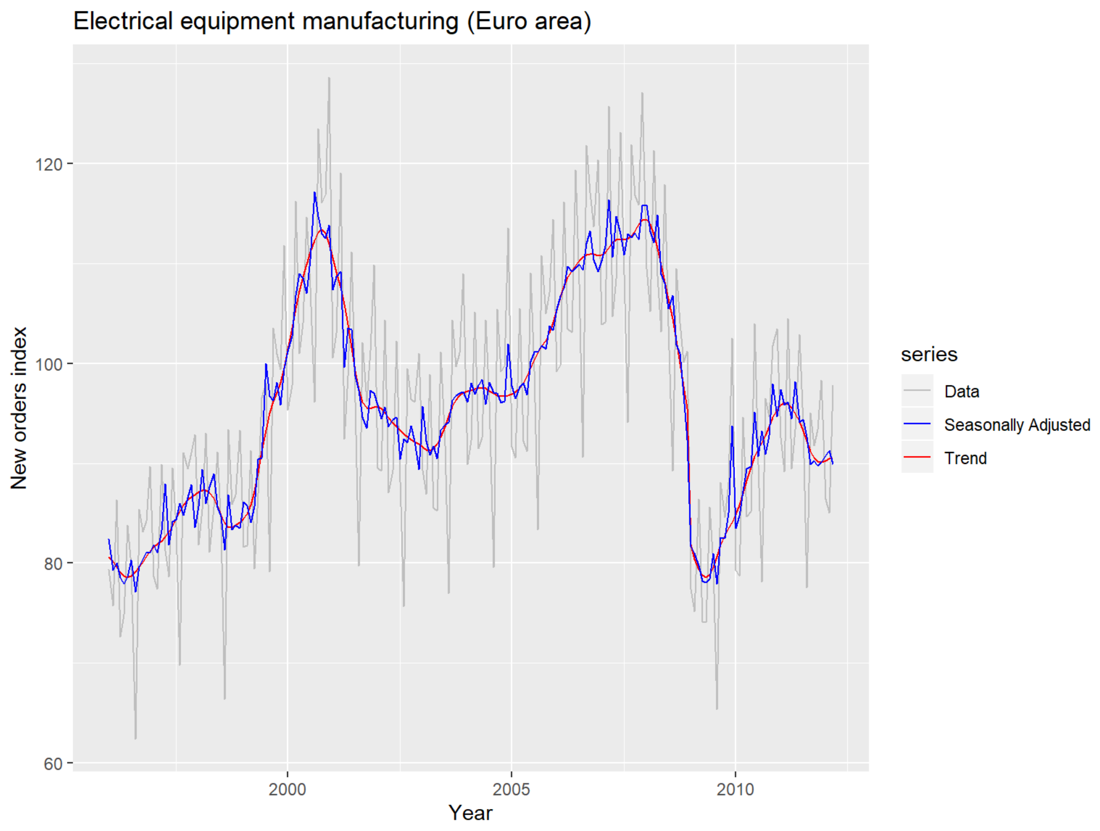

X11 decomposition of electrical equipment index
data
seasonal
trend
remainder
Time
2000
2005
2010

Compare this decomposition with the STL decomposition shown below and the classical decomposition shown above. The X11 trend-cycle has captured the sudden fall in the data in early 2009 better than either of the other two methods, and the unusual observation at the end of 2009 is now more clearly seen in the remainder component.

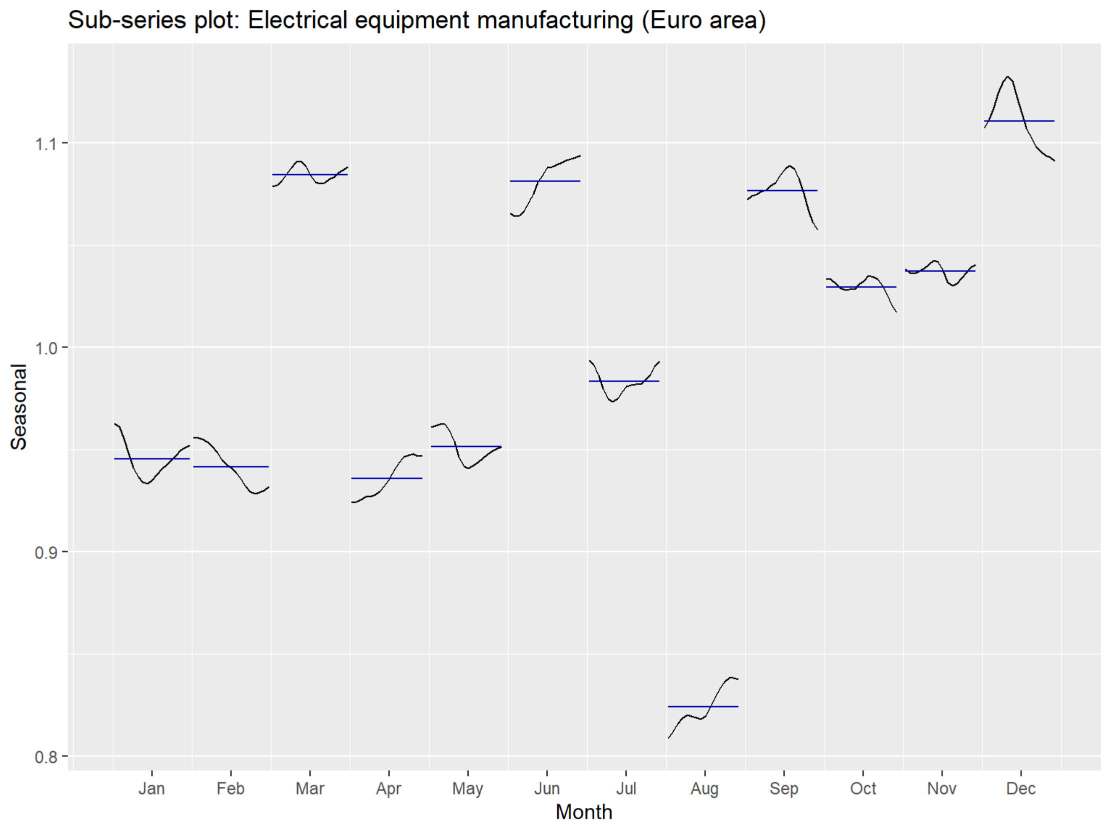
Given the output from the seas function, seasonal() will extract the seasonal component, trendcycle() will extract the trend-cycle component, remainder() will extract the remainder component, and seasadj() will compute the seasonally adjusted time series.

For example, Figure in the next slide shows the trend-cycle component and the seasonally adjusted data, along with the original data.

```r
autoplot(elecequip, series="Data") +
  forecast::autolayer(trendcycle(fit2), series="Trend") +
  forecast::autolayer(seasadj(fit2), series="Seasonally Adjusted") +
  xlab("Year") + ylab("New orders index") +
  ggtitle("Electrical equipment manufacturing (Euro area)") +
  scale_colour_manual(values=c("gray","blue","red"),
                     breaks=c("Data","Seasonally Adjusted","Trend"))
```


Electrical equipment manufacturing (Euro area)
New orders index
120
100
80
60
2000
2005
2010
Year
series
Data
Seasonally Adjusted
Trend

```
ggsubseriesplot(seasonal(fit2)) + ylab("Seasonal") +
ggtitle("Sub-series plot: Electrical equipment manufacturing (Euro area)")
```



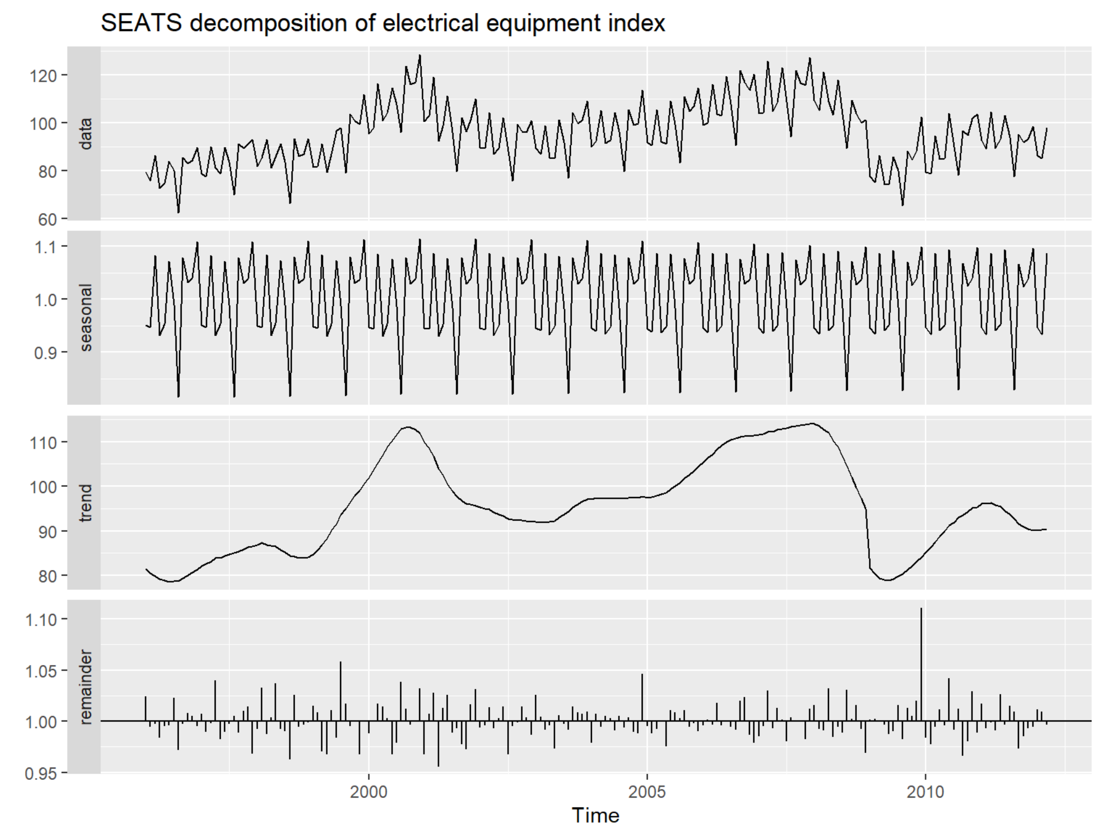
Note the very small changes in the seasonal component over time.

SEATS decomposition

“SEATS” is the acronym for “Seasonal Extraction in ARIMA Time Series” was developed at the Bank of Spain. It works only with quarterly and monthly data so it cannot be used for seasonality involving daily data, or hourly data, or weekly data.

The details are beyond the scope of this course. However, a complete discussion of the method is available in Dagum and Bianconcini (2016). The authors demostrate its use using the seasonal package. Other information are also available in the CRAN website.

```
library(seasonal)
elecequip %>% seas() %>%
autoplot() +
  ggtitle("SEATS decomposition of electrical equipment index")
```



The result is quite similar to the X11 decomposition shown above.

As with the X11 method, we can use the seasonal(), trendcycle() and remainder() functions to extract the individual components, and seasadj() to compute the seasonally adjusted time series.

The seasonal package has many options for handling variations of X11 and SEATS. See the package website for a detailed introduction to the options and features available (http://www.seasonal.website/seasonal.html) or in pdf (https://cran.r-project.org/web/packages/seasonal/vignettes/seas.pdf)

R-documentation is in (https://www.rdocumentation.org/packages/seasonal/versions/1.7.0/topics/seas)

STL decomposition

STL is an acronym for “Seasonal and Trend decomposition using Loess”, while Loess (Locally Weighted Regression and Scatterplot Smoothing) is a method for estimating nonlinear relationships.

STL has several advantages over the classical, SEATS and X-11 decomposition methods:

- Unlike the SEATS and X-11, STL will handle any type of seasonality, not only monthly and quarterly data.
- The seasonal component is allowed to change over time, and the rate of change can be controlled by the user.
- The smoothness of the trend-cycle can also be controlled by the user.
- It can be robust to outliers (i.e., the user can specify a robust decomposition).

On the other hand, STL has some disadvantages. In particular, it does not automatically handle trading day or calendar variation, and it only provides facilities for additive decompositions.

It is possible to obtain a multiplicative decomposition by first taking logs of the data, and then back-transforming the components. Additive and multiplicative decompositions can be obtained using a Box-Cox transformation of the data: a value of $\lambda = 0$ corresponds to the multiplicative decomposition while $\lambda = 1$ is equivalent to an additive decomposition.

Note that we used the STL command in earlier slides.

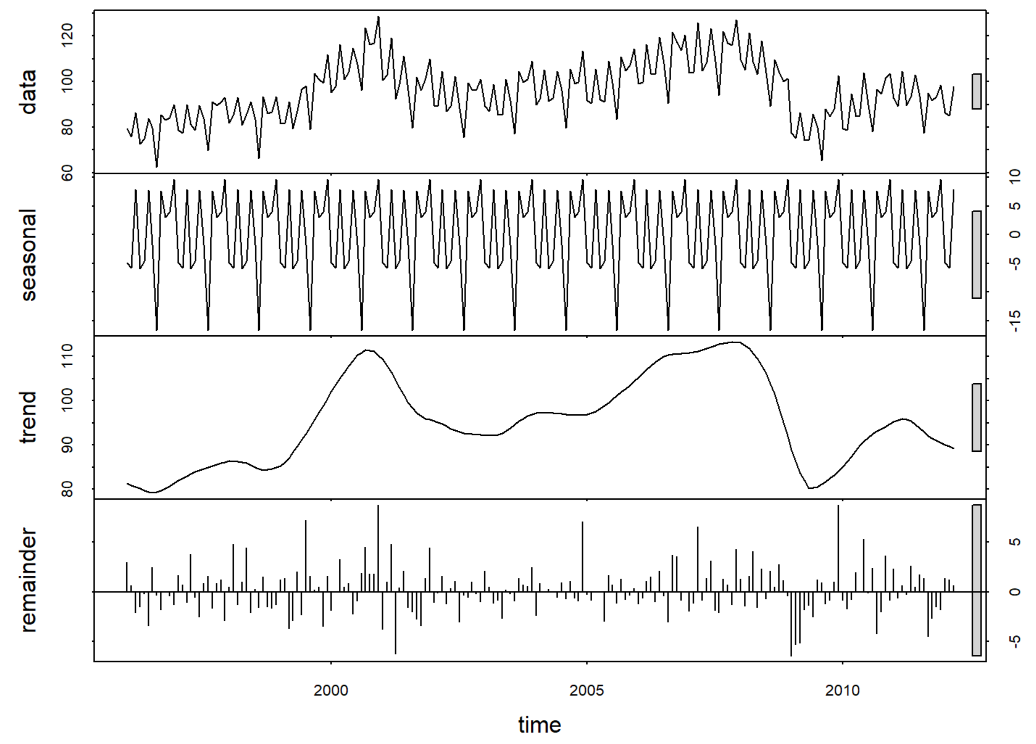
In the command below, the electrical equipment orders and its three additive components obtained from a robust STL decomposition with flexible trend-cycle and fixed seasonality.

We may use the pipe (%>%) operator or the usual command in R

```
elecequip %>%
  stl(t.window=13, s.window="periodic", robust=TRUE) %>%
  autoplot
```

Or

```r
fit <- stl(elecequip, t.window=13, s.window="periodic", robust=TRUE)
plot(fit)
```



The two main parameters to be chosen when using STL are the trend-cycle window (t.window) and the seasonal window (s.window). These control how rapidly the trend-cycle and seasonal components can change. Smaller values allow for more rapid changes. Both t.window and s.window should be odd numbers. t.windows refer to the number of consecutive observations to be used when estimating the trend-cycle and s.window refer to the number of consecutive years to be used in estimating each of the seasonal components.

The user must specify s.window as there is no default. Setting it to “periodic” forces the seasonal component to be identical across years. Specifying t.window is optional, and a default value will be used if it is omitted.

The mstl() function provides a convenient automated STL decomposition using s.window=13, and t.window also chosen automatically. This usually gives a good balance between overfitting the seasonality and allowing it to slowly change over time. But, as with any automated procedure, the default settings will need adjusting for some time series.

As with the other decomposition methods discussed in the text, to obtain the separate components in plots,

- use the seasonal() function for the seasonal component,
- the trendcycle() function for trend-cycle component, and
- the remainder() function for the remainder component.
- The seasadj() function can be used to compute the seasonally adjusted series.

For more information on the stl package, please refer to (https://www.rdocumentation.org/packages/stats/versions/3.4.3/topics/stl)

Forecasting with decomposition

While decomposition is primarily useful for studying time series data, and exploring the historical changes over time, it can also be used in forecasting.

Assuming an additive decomposition, the decomposed time series can be written as

$$y_t = \hat{S}_t + \hat{A}_t$$

where $\hat{A}_t = \hat{T}_t + \hat{R}_t$ is the seasonally adjusted component. Or if a multiplicative decomposition has been used, we can write

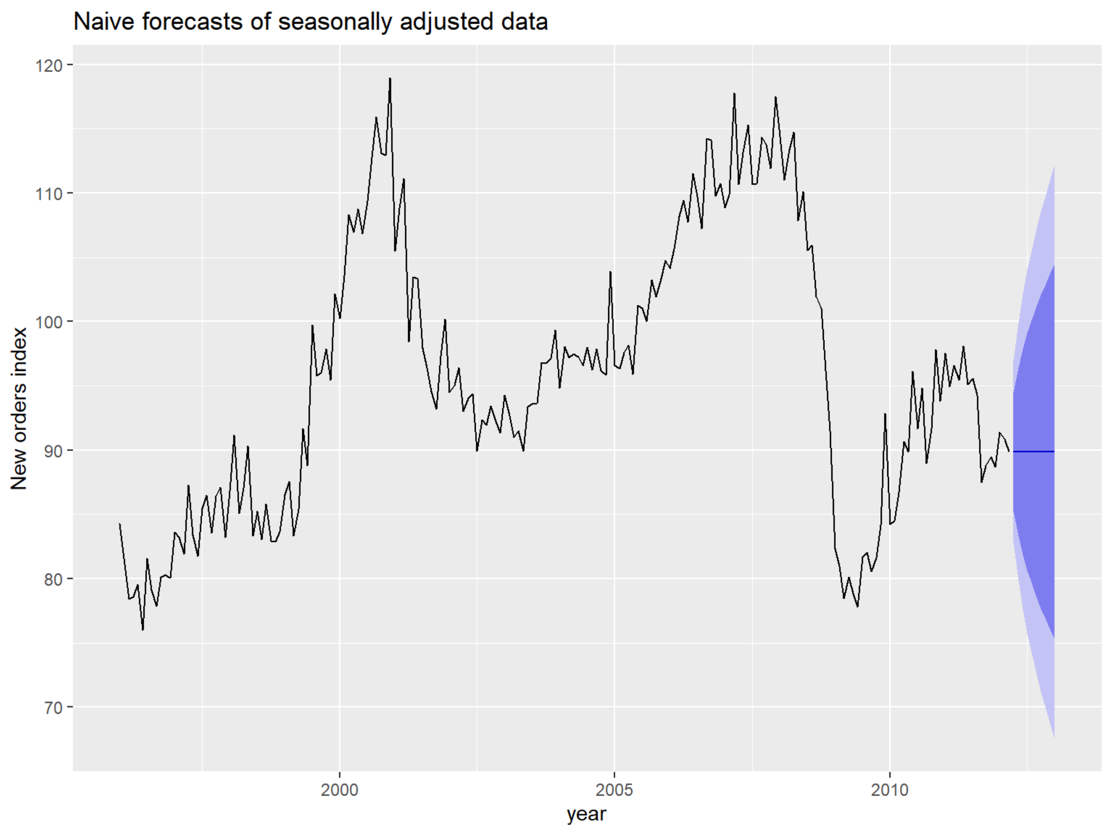
$$y_t = \hat{S}_t \hat{A}_t,$$

where $\hat{A}_t = \hat{T}_t \hat{R}_t$.

To forecast a decomposed time series, we separately forecast the seasonal component, $\hat{S}_t$, and the seasonally adjusted component $\hat{A}_t$. It is usually assumed that the seasonal component is unchanging, or changing extremely slowly, and so it is forecast by simply taking the last year of the estimated component. In other words, a seasonal naive method is used for the seasonal component.

To forecast the seasonally adjusted component, any non-seasonal forecasting method may be used. For example, a random walk with drift model, or Holt’s method, or a non-seasonal ARIMA model, may be used (discussed in Chapter 8 - 2^{nd} edition).

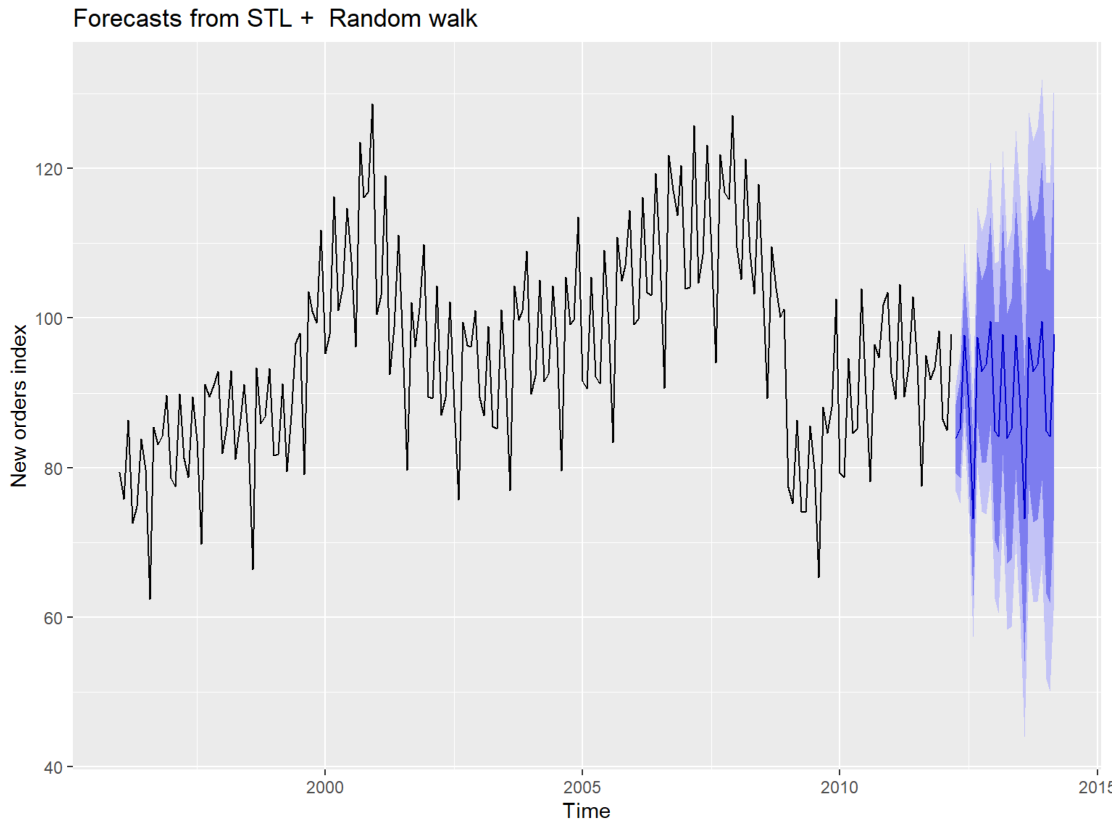
```
fit <- stl(elecequip, t.window=13, s.window="periodic", robust=TRUE)
fit %>% seasadj() %>% naive() %>% autoplot() + ylab("New orders index") + xlab("year") +
ggtitle("Naive forecasts of seasonally adjusted data")
```



In the next slide, the naive forecasts of seasonally adjusted electrical equipment data are “re-seasonalized” by adding in the seasonal naive forecasts of the seasonal component.

This is made easy with the forecast function applied to the ‘stl’ object. You need to specify the method being used on the seasonally adjusted data, and the function will do the re-seasonalizing for you.

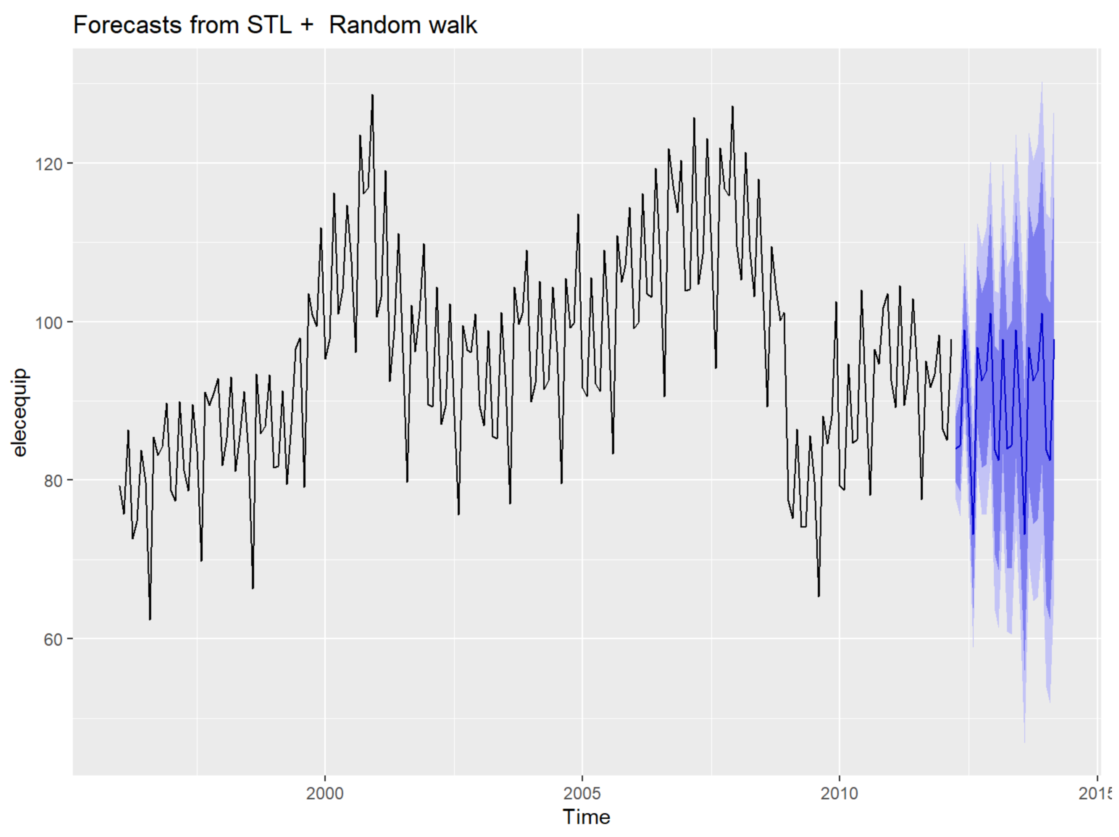
```r
fit %>% forecast(method="naive") %>% autoplot() + ylab("New orders index")
```



The prediction intervals shown in this graph are constructed in the same way as the point forecasts. That is, the upper and lower limits of the prediction intervals on the seasonally adjusted data are “reseasonalized” by adding in the forecasts of the seasonal component. In this calculation, the uncertainty in the forecasts of the seasonal component has been ignored. The rationale for this choice is that the uncertainty in the seasonal component is much smaller than that for the seasonally adjusted data, and so it is a reasonable approximation to ignore it.

A short-cut approach is to use the ‘stlf’ function. The following code will decompose the time series using STL, forecast the seasonally adjusted series, and return reseasonalize the forecasts.

```
fcast <- stlf(elecequip, method='naive')
autoplot(fcast)
```



The ‘stlf’ function uses default values for s.window and t.window .

As well as the naive method, several other possible forecasting methods are available with stlf, as described in the corresponding help file(https://www.rdocumentation.org/packages/forecast/versions/8.1/topics/forecast.stl).

If the method is not specified, it will use the ETS approach (discussed in the next chapter) applied to the seasonally adjusted series. This usually produces quite good forecasts for seasonal time series, and some companies use it routinely for all their operational forecasts.

Intro to ETS models

Exponential smoothing

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 spectrum of time series which is a great advantage and of major importance to applications in industry.

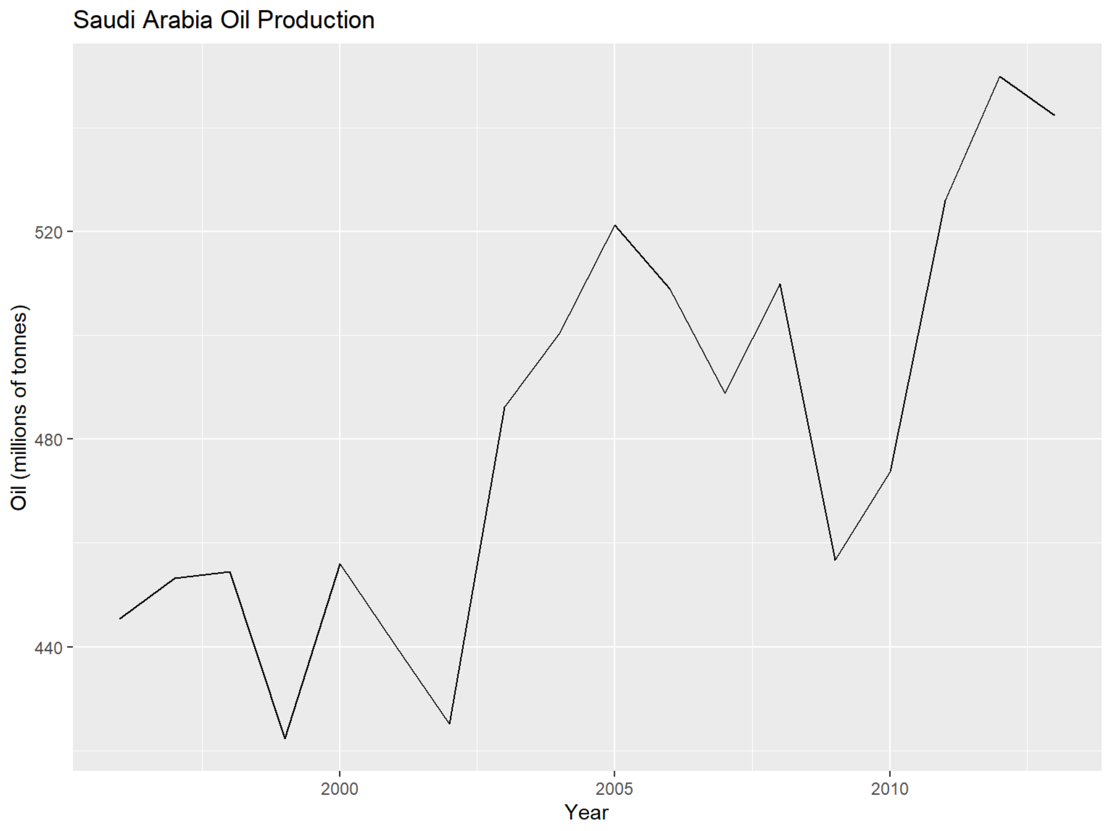
Simple exponential smoothing

The simple exponential smoothing method is suitable for forecasting data with no trend or seasonal pattern, although the mean of the data may be changing slowly over time.

The naive method assumes that the most current observation is the only important one and all previous observations provide no information for the future ($\hat{y}_{T+h|T} = y_T$, for $h = 1, 2, \ldots$). This can be thought of as a weighted average where all the weight is given to the last observation.

The average method assumes that all observations are of equal importance and they are given equal weight when generating forecasts ($\hat{y}_{T+h|T} = \frac{1}{T}\sum_{t=1}^{T} y_t$, for $h = 1, 2, \ldots$).

```r
oildata <- window(oil, start=1996)
autoplot(oildata) +
  ylab("Oil (millions of tonnes)") + xlab("Year")+
  ggtitle("Saudi Arabia Oil Production")
```



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:

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

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 the observations in the series $y_1, \ldots, y_T$. The rate at which the weights decrease is controlled by the parameter α.

Note that if we take the lag of the above equation, then multiply it by $1-\alpha$, we get:

$$\hat{y}_{T|T-1}(1-\alpha) = (1-\alpha)\alpha y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \alpha(1-\alpha)^3 y_{T-3} + \cdots,$$

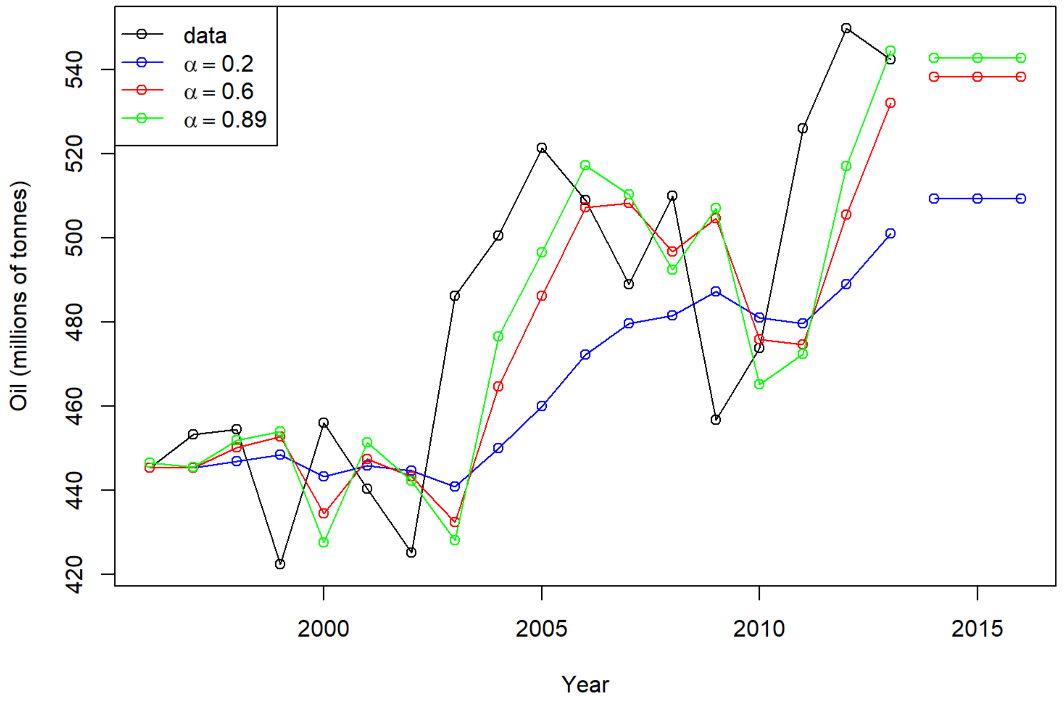
Subtracting this to the above equation gives the weighted average form.

Note that the sum of the weights even for a small α will be approximately equal to one.

For any α between 0 and 1, the weights attached to lagged observations decrease exponentially. If α is small (i.e., close to 0), more weight is given to observations from the more distant past. If α is large (i.e., close to 1), more weight is given to the more recent observations.

At the extreme case where $\alpha = 1$, $\hat{y}_{T+1|T} = y_T$ and forecasts are equal to the naive forecasts.

Example


data
α = 0.2
α = 0.6
α = 0.89
Oil (millions of tonnes)
420
440
460
480
500
520
540
2000
2005
2010
2015
Year

Weighted average form

The forecast at time $t+1$ is equal to a weighted average between the most recent observation y_t and the most recent forecast $\hat{y}_{t|t-1}$,

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

for $t = 1, \dots, T$, where $0 \le \alpha \le 1$ is the smoothing parameter.

The process has to start somewhere, so we let the first forecast of y_1 be denoted by ℓ_0. Then

$$\begin{aligned}
\hat{y}_{2|1} &= \alpha y_1 + (1-\alpha)\ell_0 \\
\hat{y}_{3|2} &= \alpha y_2 + (1-\alpha)\hat{y}_{2|1} \\
&. \\
&. \\
\hat{y}_{T+1|T} &= \alpha y_T + (1-\alpha)\hat{y}_{T|T-1}
\end{aligned}$$

Then substituting each equation into the following equation, we obtain

$$\begin{aligned}
\hat{y}_{3|2} &= \alpha y_2 + (1-\alpha)\left[\alpha y_1 + (1-\alpha)\ell_0\right] \\
&= \alpha y_2 + \alpha(1-\alpha) y_1 + (1-\alpha)^2 \ell_0 \\
\hat{y}_{4|3} &= \alpha y_3 + (1-\alpha)[\alpha y_2 + \alpha(1-\alpha) y_1 + (1-\alpha)^2 \ell_0] \\
&= \alpha y_3 + \alpha(1-\alpha) y_2 + \alpha(1-\alpha)^2 y_1 + (1-\alpha)^3 \ell_0 \\
&\vdots
\end{aligned}$$

$$\hat{y}_{T+1|T} = \sum_{j=0}^{T-1} \alpha(1-\alpha)^j y_{T-j} + (1-\alpha)^T \ell_0.$$

Component form

For simple exponential smoothing the only component included is the level, ℓ_t. (Other methods considered later in this lecture may also include a trend (slope) b_t and 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:

$$\begin{aligned}\text{Forecast equation} \quad & \hat{y}_{t+1|t} = \ell_t \\ \text{Smoothing equation} \quad & \ell_t = \alpha y_t + (1-\alpha)\ell_{t-1},\end{aligned}$$

where ℓ_t is the level (or the smoothed value) of the series at time t.

The forecast equation shows that the forecasted 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.

Applying the forecast equation for time T gives, $\hat{y}_{T+1|T} = \ell_T$, the most recent estimated level.

If we replace ℓ_t by $\hat{y}_{t+1|t}$ and ℓ_{t-1} by $\hat{y}_{t|t-1}$ in the smoothing equation, we will recover the weighted average form of simple exponential smoothing.

Multi-horizon Forecasts

Simple exponential smoothing has a flat forecast function, and therefore for longer forecast horizons,

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

Remember these forecasts will only be suitable if the time series has no trend or seasonal component.

Initialisation

For simple exponential smoothing we need to specify an initial value for the level, ℓ_0. Hence ℓ_0 plays a role in all forecasts generated by the process. In general, the weight attached to ℓ_0 is small.

However, in the case that α is small and/or the time series is relatively short, the weight may be large enough to have a noticeable effect on the resulting forecasts. Therefore, selecting suitable initial values can be quite important.

A common approach is to set $\ell_0 = y_1$ (recall that $\ell_0 = \hat{y}_{1|0}$).

An alternative approach is to use optimization to estimate the value of ℓ_0 rather than set it to some value. Even if optimization is used, selecting appropriate initial values can assist the speed and precision of the optimization process.

Optimization

For every exponential smoothing method we also need to choose the value for the smoothing parameters. For simple exponential smoothing, there is only one smoothing parameter (α).

There are cases where 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 robust and objective way to obtain values for the unknown parameters included in any exponential smoothing method is to estimate them from the observed data.

Similar to regression model, the unknown parameters and the initial values for any exponential smoothing method can be estimated by minimizing the SSE.

The errors are specified as $e_t = y_t - \hat{y}_{t|t-1}$ for $t = 1, \ldots, T$ (the one-step-ahead within-sample forecast errors).

$$\text{SSE} = \sum_{t=1}^{T} (y_t - \hat{y}_{t|t-1})^2 = \sum_{t=1}^{T} e_t^2.$$

This involves a non-linear minimization problem and we need to use an optimization tool to perform this.

In the forecast package, this is done with the function ses function in the forecast package (https://www.rdocumentation.org/packages/forecast/versions/8.1/topics/ses) or page 97 of the forecast package documentation (https://cran.r-project.org/web/packages/forecast/forecast.pdf)

Example: Oil Production

```
library(fpp2)
oildata <- window(oil, start=1996)
# Estimate parameters
fc <- ses(oildata, h=5)
# Accuracy of one-step-ahead training errors over period 1--12
round(accuracy(fc),2)
```

```
##                ME  RMSE   MAE MPE MAPE MASE  ACF1
## Training set 6.4 28.12 22.26 1.1 4.61 0.93 -0.03
```

```
#>              ME RMSE  MAE MPE MAPE MASE  ACF1
#> Training set 6.4 28.1 22.3 1.1 4.61 0.93 -0.03
```

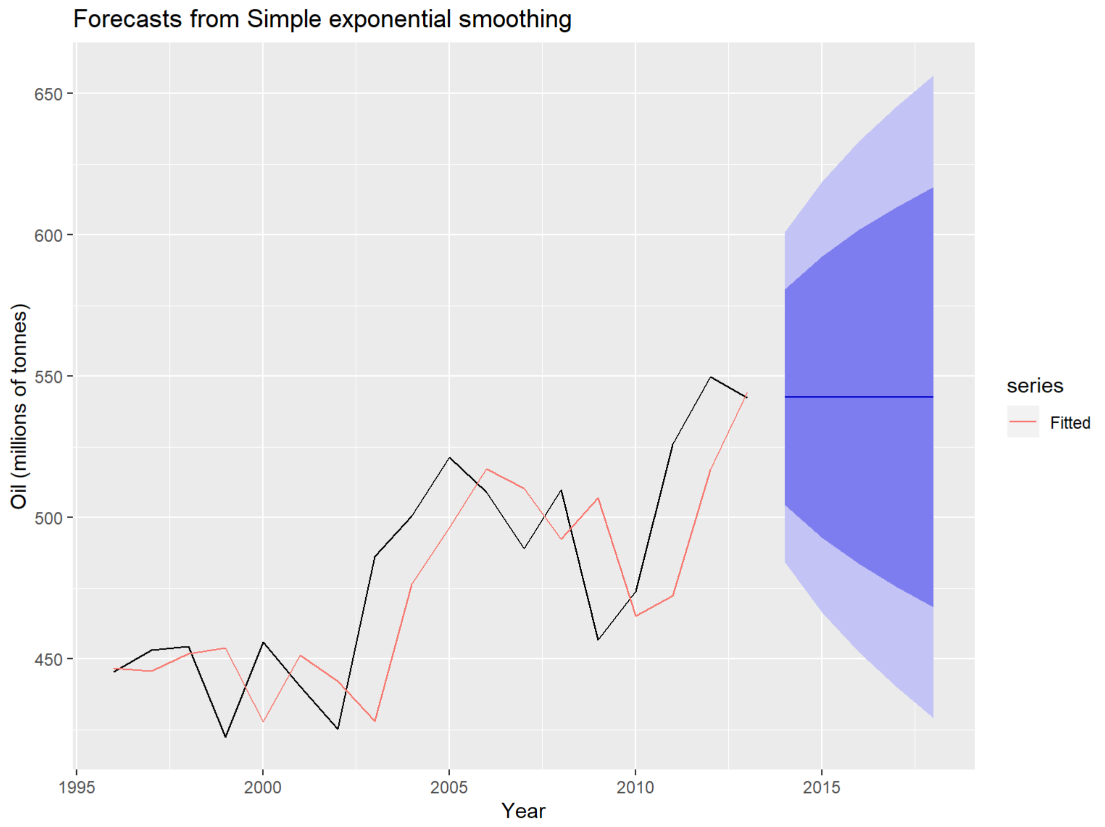
This gives parameters $\alpha = 0.83$ and $l_0 = 446.58$, obtained by minimizing SSE over periods $t = 1, 2, \ldots, 12$, subject to the restriction that $0 \leqslant \alpha \leqslant 1$.

Note that these values could be shown by running on R-prompt: >summary(fc) i.e.:

```
summary(fc)
```

This is plotted in the next slide

```
autoplot(fc) +
  forecast::autolayer(fitted(fc), series="Fitted") +
  ylab("Oil (millions of tonnes)") + xlab("Year")
```



The forecasts for the period 2014-2018 are plotted above. The fitted values (red line) are plotted alongside the data (black line) over the period 1996-2013. The large value of α is shown in the large adjustment in the estimated level l_t at each time. A smaller value of α would have shown smaller changes over time so that the series of fitted values would be smoother.

The prediction intervals at 80% and 95% confidence intervals are also shown over the forecast period.

Thank you for your attention