Week 3 & 4 Topics

1. Supporting e-Textbook:

Use the following link to a free e-book on Timeseries Analytics. It has more in depth explanation for the materials presented in our textbook. <https://otexts.com/fpp3/>

1. Chapter 4 – Forecasting Methods: Overview

# Model-Based vs. Data-Driven Methods

1. Definition
2. Advantages and disadvantages of each type
3. Differences

# Extrapolation Methods, Econometric Models, and External Information

# Manual vs. Automated Forecasting

1. Chapter 5 – Smoothing Methods

# Introduction to smoothing Methods

In statistics and image processing, to smooth a dataset is to create an approximating function that attempts to capture important patterns in the data, while leaving out noise or other fine-scale structures. In smoothing, the data points of a signal are modified so individual points (presumably because of noise) are reduced, and points that are lower than the adjacent points are increased leading to a smoother signal. Smoothing may be used in two important ways that can aid in data analysis:

1. by being able to extract more information from the data as long as the assumption of smoothing is reasonable and
2. by being able to provide analyses that are both flexible and robust.

Smoothing may be distinguished from the related and partially overlapping concept of curve fitting in the following ways:

* curve fitting often involves the use of an explicit function form for the result, whereas the immediate results from smoothing are the "smoothed" values with no later use made of a functional form if there is one;
* the aim of smoothing is to give a general idea of relatively slow changes of value with little attention paid to the close matching of data values, while curve fitting concentrates on achieving as close a match as possible.
* smoothing methods often have an associated tuning parameter which is used to control the extent of smoothing. Curve fitting will adjust any number of parameters of the function to obtain the 'best' fit.

In Summary. Data smoothing can be defined as a statistical approach of eliminating outliers from datasets to make the patterns more noticeable. The random method, simple moving average, random walk, simple exponential, and exponential moving average are some of the methods used for data smoothing.

However, the terminology used across applications is mixed. But in Data analytics for time series forecasting, smoothing and curve fitting process are well defined and are not the same.

# Moving Average (a smoothing method)

The two of the simplest models for predicting a model from its own history—the **mean** model and the **random walk** model. These models represent two extremes as far as time series forecasting is concerned. The **mean** model assumes that the best predictor of what will happen tomorrow is the average of everything that has happened up until now. The **random walk** model assumes that the best predictor of what will happen tomorrow is what happened today, and all previous history can be ignored. Intuitively there is a spectrum of possibilities in between these two extremes. Why not take an average of what has happened in some window of the recent past? That’s the concept of a “moving” average. You will often encounter time series that appear to be “locally stationary” in the sense- that they exhibit random variations around a local mean value that changes gradually over time in a nonsystematic way. Regardless the size, the window can be implement in two ways.

1. Centered window (Centered Moving Average - Visualization)
2. Trailing window (Trailing Moving Average - Forecasting)

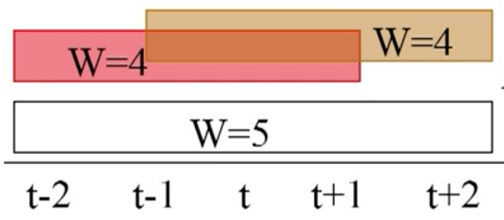
### Centered Window

Assume we want to predict the value of MA at the time t, then we compute average of values in window (of width W), centered at time t. The followings are reflected in figure 1 and 2.

For **Odd widow width**, the value of MA at the time ***t*** is calculated as:

Assuming the window width is equal to 5

For Even window width, we take the two “almost centered” widows and average the values in them. Let’s assume the window width is equal to 4, the value of MA at the time ***t*** is calculated as:



Even Width

Odd Width

Figure 1: Odd and Even window

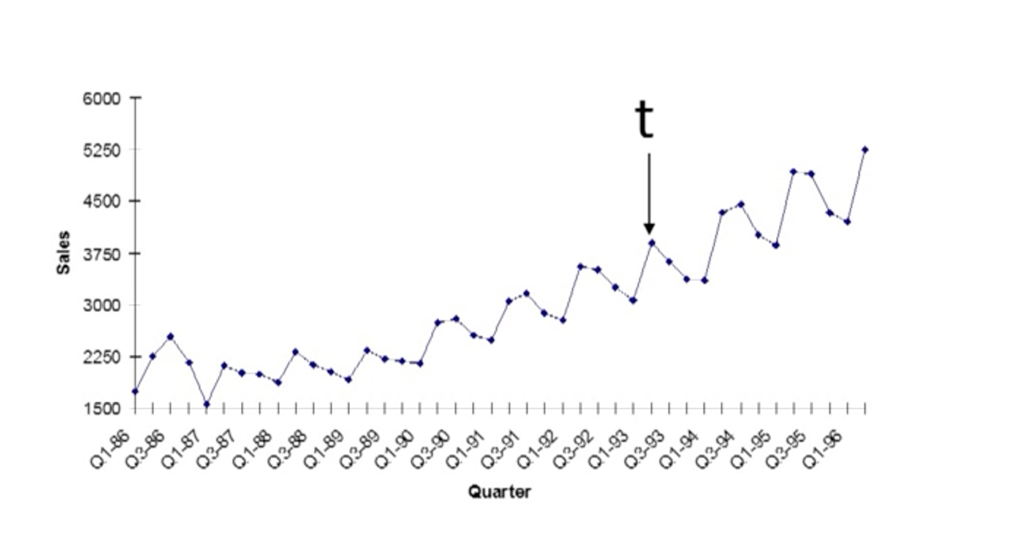
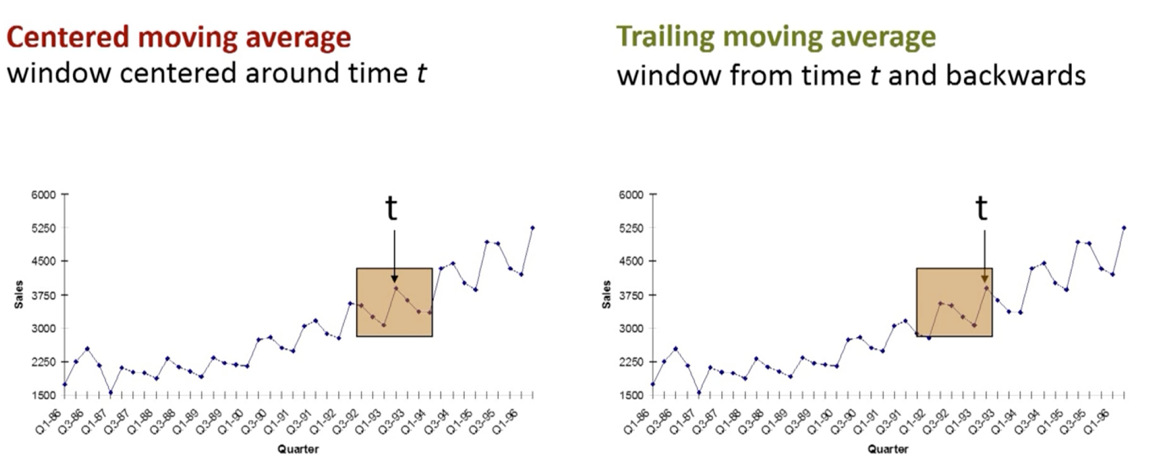


Figure 2: Adjacent observations to observation at the time t

Figure 3 and 4 show a centered and trailing moving average. The centered moving average is used for analyzing the data. The trailing moving average is for forecasting.

****

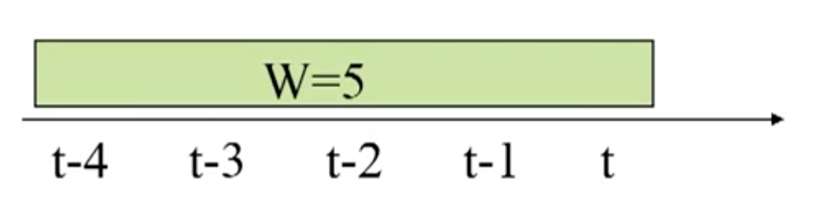
Figures 3 and 4: Windows of the centered and trailing moving average

Computing Trailing Moving Average for Forecasting

Following steps show how to implement the trailing moving average algorithm to forecast a time-series data

Choose a window width. The width depends on the pattern in data and how much we want to smooth.

For MA at the time t, (a future value), place window on the points t-W+1, …, t

****

(Assuming W = 5)

Compute average of values in the window. To forecast a series at time *t+k*, given data until time *t*:

Note: Moving average forecasting does not capture trend and seasonality

Example: the following figure 5 shows how moving average is used to forecast for each quarter in the training set and the in validation. The training set is 4 years (16 quarters). The widow size is 4 quarters

A screenshot of a spreadsheet

Description automatically generated

Figure 5: Moving average example (Training = 16 Quarters, Validation = 1 Quarter)

### The following R Codes are used in figure 5.3 on page 82.

*library(forecast)*

*library(zoo)*

*Amtrak.data <- read.csv("Amtrak data.csv")*

*ridership.ts <- ts(Amtrak.data$Ridership, start = c(1991, 1), end = c(2004, 3), freq = 12)*

*nValid <- 36*

*nTrain <- length(ridership.ts) - nValid*

*train.ts <- window(ridership.ts, start = c(1991, 1), end = c(1991, nTrain))*

*valid.ts <- window(ridership.ts, start = c(1991, nTrain + 1), end = c(1991, nTrain + nValid))*

*##apply the moving average on training dataset*

*ma.trailing <- rollmean(train.ts, k = 12, align = "right")*

*last.ma <- tail(ma.trailing, 1)*

*## create a time series dataset with the replicated last value of moving average value calculated in the training dataset*

*ma.trailing.pred <- ts(rep(last.ma, nValid), start = c(1991, nTrain + 1), end = c(1991, nTrain + nValid), freq = 12)*

*# Figure 5-3*

*plot(train.ts, ylim = c(1300, 2600), ylab = "Ridership", xlab = "Time", bty = "l", xaxt = "n", xlim = c(1991,2006.25), main = "")*

*axis(1, at = seq(1991, 2006, 1), labels = format(seq(1991, 2006, 1)))*

*lines(ma.trailing, lwd = 2)*

*lines(ma.trailing.pred, lwd = 2, col = "blue", lty = 2)*

*lines(valid.ts)lines*

*lines(c(2004.25 - 3, 2004.25 - 3), c(0, 3500))*

*lines(c(2004.25, 2004.25), c(0, 3500))*

*text(1996.25, 2500, "Training")*

*text(2002.75, 2500, "Validation")*

*text(2005.25, 2500, "Future")*

*arrows(2004 - 3, 2450, 1991.25, 2450, code = 3, length = 0.1, lwd = 1,angle = 30)*

*arrows(2004.5 - 3, 2450, 2004, 2450, code = 3, length = 0.1, lwd = 1,angle = 30)*

*arrows(2004.5, 2450, 2006, 2450, code = 3, length = 0.1, lwd = 1, angle = 30)*

*accuracy(ma.trailing.pred, valid.ts*

# Differencing

A simple and popular method for removing a trend and/or a seasonal pattern from a series is by the operation of differencing. Differencing means taking the difference between two values. A lag-1 difference (also called first difference) means taking the difference between every two consecutive values in the series (yt – y(t-1)). Differencing at lag-k means subtracting the value from k periods back (yt – y(t-k)). For example, for a daily series, lag-7 differencing means subtracting from each value the value on the same day in the previous week. In R, a lag-12 difference of the Amtrak ridership is created by running:

diff(ridership.ts, lag = 12)

where the ridership is a field in the data table showing number of people riding the train on a specific day.

To remove trends and seasonal patterns we can difference the original time series and obtain a differenced series that lacks trend and seasonality. Lag-1 differencing results in a differenced series that measures the changes from one period to the next.

### Removing Trend (De-Trending)

Lag-1 differencing is useful for removing a trend. The example of the Amtrak lag-1 differenced series is shown in figure 5. Compared to the original series (top left panel), which exhibits a U-shaped trend, the lag-1 differenced series contains no visible trend. One advantage of differencing over other methods (which we will see later) is that differencing does not assume that the trend is global: in other words, in differencing the trend shape is not assumed to be fixed throughout the entire period. For quadratic and exponential trends, often another round of lag-1 differencing must be applied in order to remove the trend. This means taking lag-1 differences of the differenced series.

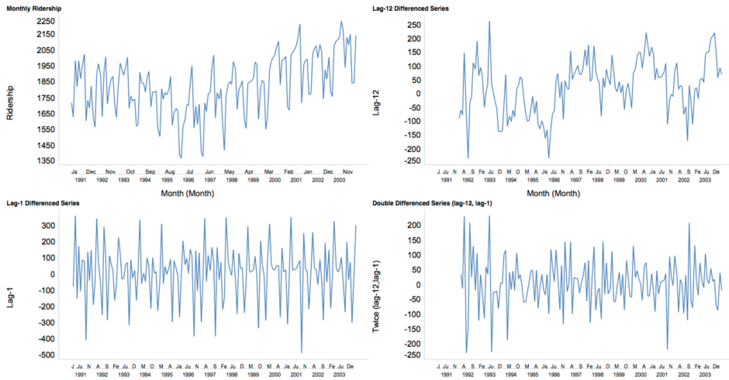


Figure 5: Differenced series for Amtrak ridership

### Removing Seasonality (Seasonal Adjustment, De-seasonalizing)

For removing a seasonal pattern with M seasons, we difference at lag M. For example, to remove a day-of-week pattern in daily data, we can take lag- 7 differences. In the figure 1 (top right panel) illustrates a lag-12 differenced series of the Amtrak monthly data. The monthly pattern no longer appears in this differenced series.

### Removing Trend and Seasonality

When both trend and seasonality exist, we can apply differencing twice to the series in order to de-trend and de-seasonalize it. We performed double differencing on the Amtrak data, which contain both trend and seasonality. The bottom right panel of the figure 1 shows the result after first differencing at lag-12, and then applying a lag-1 difference to the differenced series. The result is a series with no trend and no monthly seasonality. In R, this twice differencing is created by running

*diff(diff(ridership.ts, lag = 12), lag = 1)*

Note: Differencing is often used as a pre-processing step before applying a forecasting model to a series.

Simple Exponential Smoothing (SES)

Simple exponential smoothing (SES) is a popular forecasting method in business. Its popularity

derives from its flexibility, ease of automation, cheap computation, and good performance. SES

is similar to forecasting with a moving average, except that instead of taking a simple average

over the w (window size) most recent values, we take a weighted average of all past values, so

that the weights decrease exponentially into the past. The idea is to give more weight to recent

information, yet not to completely ignore older information. Like the moving average, SES

should only be used for forecasting series that have no clear trending behavior or any

seasonality. As we saw earlier, such series can be obtained by removing trend and/or

seasonality from raw series, and then applying exponential smoothing to the series of residuals

(which are assumed to contain no trend or seasonality). The SES generates a forecast at the

time t+1 (Ft+1) using the following formula:

(since we assume the timeseries has only level)

where L*t* is the level (or the smoothed value) of the series at time t. Setting h=1 (h is the forecasting horizon) gives the fitted values, while setting t = T (such as t+1, gives the true forecasts beyond the training data).

Expanding the above equality:

And Expanding for

Now we can see the sequence of component which expands all the way from components to

Considering our time series data set has ***t*** observations and the value of each observation, say i, is equal to then, the following determines the value of the level of observation ***t***.

And eventually we can write this series as:

As we learned from the naïve and moving average, the last observation’s value of the training

dataset is the on-step-ahead, two-step-ahead, and k-step-ahead forecast. Therefore, we can

write:

where  is a constant between 0 and 1 called the smoothing constant. The above formulation displays the exponential smoother as a weighted average of all past observations, with exponentially decaying weights. We can also write the exponential forecaster in another way, which is very useful in practice:

Using the first equation

where  is the forecast error at time t. This formulation presents the exponential forecaster as an “active learner”. It looks at the previous forecast () and at its distance from the actual value (), and then corrects the next forecast based on that information. If the forecast was too high in the last period, the next period is adjusted down. The amount of correction depends on the value of the smoothing constant α.

The second formula is also advantageous in terms of data storage and computation time: we need to store and use only the forecast and forecast error from the previous period, rather than the entire series. In applications where real-time forecasting is done, or many series are being forecasted in parallel and continuously, such savings are critical.

Note: forecasting further into the future yields the same forecasts as a one-step-ahead forecast () because the series is assumed to lack trend and seasonality.

# Choosing Smoothing Constant α

The smoothing constant *α*, which is set by the user, determines the rate of learning. A value close to 1 indicates fast learning (that is, only the most recent values influence the forecasts), whereas a value close to 0 indicates slow learning (past observations have a large influence on forecasts). This can be seen by plugging 0 or 1 into the two equations above. Hence, the choice of *α* depends on the required amount of smoothing, and on how relevant the history is for generating forecasts. Default values that have been shown to work well are in the range 0.1-0.2.

Trial and error can also help in the choice of *α*. Examine the time plot of the actual and predicted series, as well as the predictive accuracy (e.g., MAPE or RMSE of the validation period).

In R, forecasting using SES can be done via the *ets()* function in the forecast package. The three letters in *ets* stand for *error, trend, and seasonality*. Applying this function to a time series will yield forecasts and forecast errors (residuals) for both the training and validation periods. You can use a default value of *α*=0.2, set it to another value, or choose to find the optimal *α* in terms of minimizing RMSE over the training period. To choose an SES using the *ets* function, we set model = "ANN" (additive error (A), no trend (N), and no seasonality (N)). [For an illustration of using the *ets()* function and comparing default and optimized values, see Section 5.4 in the textbook Practical Time Series Forecasting with R, 2nd edition, by Shmueli & Lichtendahl ]

### The following R Codes are used in figure 5.5 on page 91.

*# Loading Data, Partitioning, and Forcasting the validation dataset.*

*nValid <- 36*

*nTrain <- length(diff.twice.ts) - nValid*

*train.ts <- window(diff.twice.ts, start = c(1992, 2), end = c(1992, nTrain + 1))*

*valid.ts <- window(diff.twice.ts, start = c(1992, nTrain + 2), end = c(1992, nTrain + 1 + nValid))*

*ses <- ets(train.ts, model = "ANN", alpha = 0.2)*

*ses.pred <- forecast(ses, h = nValid, level = 0)*

*# Generating Visualization. Figure 5-5 Textbook*

*plot(ses.pred, ylim = c(-250, 300), ylab = "Ridership (Twice-Differenced)", xlab = "Time", bty = "l", xaxt = "n", xlim = c(1991,2006.25), main = "", flty = 2)*

*axis(1, at = seq(1991, 2006, 1), labels = format(seq(1991, 2006, 1)))*

*lines(ses.pred$fitted, lwd = 2, col = "blue")*

*lines(valid.ts)*

*lines(c(2004.25 - 3, 2004.25 - 3), c(-250, 350))*

*lines(c(2004.25, 2004.25), c(-250, 350))*

*text(1996.25, 275, "Training")*

*text(2002.75, 275, "Validation")*

*text(2005.25, 275, "Future")*

*arrows(2004 - 3, 245, 1991.5, 245, code = 3, length = 0.1, lwd = 1,angle = 30)*

*arrows(2004.5 - 3, 245, 2004, 245, code = 3, length = 0.1, lwd = 1,angle = 30)*

*arrows(2004.5, 245, 2006, 245, code = 3, length = 0.1, lwd = 1, angle = 30)*

*# Generating Evaluations. Table 5.1 Textbook*

*ses.opt <- ets(train.ts, model = "ANN")*

*ses.opt.pred <- forecast(ses.opt, h = nValid, level = 0)*

*ses.opt*

*accuracy(ses.pred, valid.ts)*

*accuracy(ses.opt.pred, valid.ts)*

### Link Between Moving Average and Simple Exponential Smoothing

In both the moving average and simple exponential smoothing methods, the user must specify a single parameter: in moving averages, the window width (w); and in SES, the smoothing constant (*α*). In both cases, the parameter determines the importance of fresh information over older information. In fact, the two smoothers are approximately equal if the window width of the moving average is equal to w = 2/α − 1.

Holt’s (Double) Exponential Smoothing

Both moving average and SES should only be used for forecasting series with no trend or seasonality; series that have only a level and noise. One solution for forecasting series with trend and/or seasonality is first to remove those components (e.g., via differencing). Another solution is to use a more sophisticated version of exponential smoothing, which can capture trend and/or seasonality. Holt’s exponential smoothing is used for series that have a trend.

### Series with an Additive Trend

For series that contain a trend, we can use double exponential smoothing, also called Holt’s linear trend model. The trend in this model is not assumed to be global, but rather it can change over time. In double exponential smoothing, the local trend is estimated from the data and is updated as more data becomes available. Similar to SES, the level of the series is also estimated from the data, and is updated as more data become available.

The k-step-ahead forecast is given by combining the level estimate at time t ()  and the trend estimate (which is assumed additive) at time t ()

Note: in the presence of a trend, one-, two-, three-step-ahead (etc.) forecasts are no longer identical.

The level and trend are updated through a pair of updating equations:

(1)

(2)

The first equation (1) means that the level at time t is a weighted average of the actual value at time t and the level in the previous period, adjusted for trend (in the presence of a trend, moving from one period to the next requires factoring in the trend). The second equation (2) means that the trend at time t is a weighted average of the trend in the previous period and the more recent information on the change in level. (There are various ways to estimate the initial values  and , but the difference between estimates usually disappears after a few periods.) Two smoothing constants, and β, determine the rate of learning. As in SES, they are both constants between 0 and 1, set by the user, with higher values giving faster learning (more weight to most recent information).

### Series with a Multiplicative (Exponential) Trend

The additive trend model assumes that the level changes from one period to the next by a fixed amount. Hence, the forecasting equation adds k trend estimates. In contrast, a multiplicative trend model assumes that the level changes from one period to the next by a factor. Exponential smoothing with a multiplicative trend therefore produces k-step-ahead forecasts using the formula

and the updating equations for the level and trend are:

There are two types of errors an exponential smoothing method can capture. An additive error

And a multiplicative error

In R, to specify a Holt’s exponential smoothing model with an additive or multiplicative trend, use the ets() function and set the second parameter in model= to A (=Additive) or M (=Multiplicative). As you remember from previous sections, the first character is the model parameter is for error. That is Additive error (A \_ \_ ) and Multiplicative error (M \_ \_ ). Keep the third parameter equal to N (to specify no seasonality). For example, model = “AAN” gives Holt’s exponential smoothing with additive error and additive trend.

Holt-Winter’s (Triple) Exponential Smoothing

For series that contain both trend and seasonality, the Holt-Winter’s exponential smoothing method can be used. This is a further extension of double exponential smoothing, where the k-step-ahead forecast also takes into account the season at period ***t+k***. As for trend, there exist formulations for additive and multiplicative seasonality. In multiplicative seasonality, values on different seasons differ by percentage amounts, whereas in additive seasonality, they differ by a fixed amount.

### Additive Seasonality

The additive seasonality version of Holt-Winter’s exponential smoothing, where seasons differ by a constant amount, can be constructed from the multiplicative version by replacing multiplication/division signs by addition/subtraction signs for the seasonal component:

In R, to specify a Holt’s exponential smoothing model with additive or multiplicative seasonality, use the ***ets()*** function and set the third parameter in model= to A (=Additive) or M (=Multiplicative). For example, model = “MAA” gives Holt-Winter’s exponential smoothing with multiplicative error, additive trend, and additive seasonality.

### Multiplicative Seasonality

Assuming seasonality with M seasons (e.g., for weekly seasonality M = 7), the forecast for an additive trend and multiplicative seasonality is given by:

Note that in order to produce forecasts using this formula the series must have included at least one full cycle of seasons by the forecasting time **t, i.e., t > M**.

Being an adaptive method, Holt-Winter’s exponential smoothing allows the level, trend, and seasonality patterns to change over time. The three components are estimated and updated as new information arrives. The updating equations for this additive trend and multiplicative seasonality version are given by

The first equation is similar to that in double exponential smoothing, except that it uses the seasonally adjusted value at time ***t*** rather than the raw value. This is done by dividing  by its seasonal index, as estimated in the last cycle. The second equation is identical to double exponential smoothing. The third equation means that the seasonal index is updated by taking a weighted average of the seasonal index from the previous cycle and the current trend-adjusted value.

Summary of exponential smoothing in R

The following are some of most popular exponential smoothing functions in R. You can access and use them by installing

* Fpp3 (with which the forecast package will be automatically loaded)
* tidyverse

after preparing time series data and partitioning in training

wine.ses <-ses(train.ts, h = n.valid, alpha = 0.1, level = 0)

wine.holt <- holt(train.ts, h = n.valid, beta = 0.2, level = 0)

wine.train.HW<-HoltWinters(train.ts, alpha = 0.1, beta = 0.2, gamma = 0.1)

wine.HW<-forecast(wine.train.HW, h = n.valid, level = 0)

wine.train.ets<- ets(train.ts, model = "MAA")

wine.ets<-forecast(wine.train.ets, h = n.valid, level = 0)

A graph showing the growth of a number of smothering models

Description automatically generated

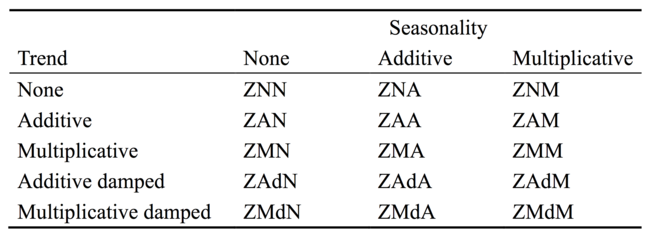
***ets()***

ETS() is based on Holtswinters and can be configure for many possibilities.For the additive seasonality and multiplicative seasonality versions of the exponential smoothing model introduced earlier, we can include either an additive or multiplicative error. We can also specify an additive trend, a multiplicative trend, or no trend at all. This flexibility of the ***ets()***function in R gives us 18 models from which to choose: 2 error types × 3 trend types × 3 seasonality types.

There are two other trend types that the ***ets*** function allows: the additive damped trend (Ad) and the multiplicative damped trend (Md). These advanced trend types apply to time series whose trends will eventually dampen to a flat line in the distant future. With these two other trend types, the number of possible models the ***ets()***function can fit increases to 30.

All these models are summarized in the Table 1 below, where the error (Z) can be either set to additive (A) or multiplicative (M).

Table 1: Possible exponential smoothing models in R



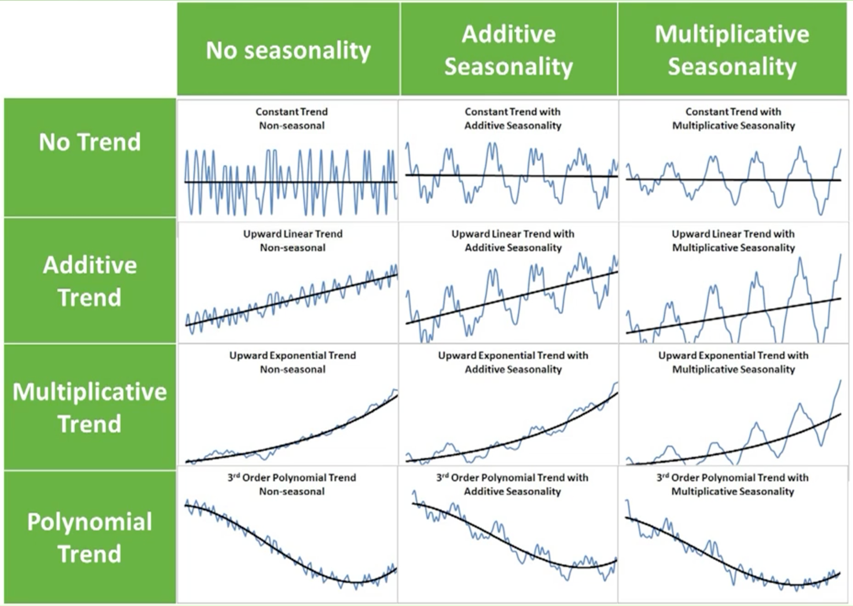
We note that the ***ets()***function by default restricts the use of some models in the Table 1. It does so because some combinations of error, trend, and seasonality can cause numerical difficulties when fit to some time series. For instance, ***ets*** function will not fit an AAM model unless we turn the restriction off by including the option **restrict = FALSE** in the ***ets()*** function. When you fit a restricted model, check that the results seem plausible. See figure 6.

Figure 6 All possible time series formats

### The following R Codes are used in figure 5.5 on page 91.

*library(forecast)*

*library(zoo)*

*Amtrak.data <- read.csv("Amtrak data.csv")*

*ridership.ts <- ts(Amtrak.data$Ridership, start = c(1991, 1), end = c(2004, 3), freq = 12)*

*nValid <- 36*

*nTrain <- length(ridership.ts) - nValid*

*train.ts <- window(ridership.ts, start = c(1991, 1), end = c(1991, nTrain))*

*valid.ts <- window(ridership.ts, start = c(1991, nTrain + 1), end = c(1991, nTrain + nValid))*

*hwin <- ets(train.ts, model = "MAA")*

*hwin.pred <- forecast(hwin, h = nValid, level = 0)*

*# Figure 5.6*

*plot(hwin.pred, ylim = c(1300, 2600), ylab = "Ridership", xlab = "Time", bty = "l", xaxt = "n", xlim = c(1991,2006.25), main = "", flty = 2)*

*axis(1, at = seq(1991, 2006, 1), labels = format(seq(1991, 2006, 1)))*

*lines(hwin.pred$fitted, lwd = 2, col = "blue")*

*lines(valid.ts)*

*lines(c(2004.25 - 3, 2004.25 - 3), c(0, 3500))*

*lines(c(2004.25, 2004.25), c(0, 3500))*

*text(1996.25, 2500, "Training")*

*text(2002.75, 2500, "Validation")*

*text(2005.25, 2500, "Future")*

*arrows(2004 - 3, 2450, 1991.25, 2450, code = 3, length = 0.1, lwd = 1,angle = 30)*

*arrows(2004.5 - 3, 2450, 2004, 2450, code = 3, length = 0.1, lwd = 1,angle = 30)*

*arrows(2004.5, 2450, 2006, 2450, code = 3, length = 0.1, lwd = 1, angle = 30)*

*# Table 5.2*

*hwin*

*hwin$states[1, ] # Initial states*

*hwin$states[nrow(hwin$states), ] # Final states*

*hwin$states*

*# Table 5.4*

*ets.opt <- ets(train.ts, restrict = FALSE, allow.multiplicative.trend = TRUE)*

*# Table 5.5*

*ets.opt.pred <- forecast(ets.opt, h = nValid, level = 0)*

*plot(ets.opt.pred)*

*accuracy(hwin.pred, valid.ts)*

*accuracy(ets.opt.pred, valid.ts)*