Smoothing Methods

Example Dataset

- We will use a time series dataset of Monthly Milk Production (in pounds) of cows from January 1962 to December 1975
- Source: http://data.is/1qY3LDd
- Website: www.datamarket.com

```r
mlk <- read.csv("monthly-milk-production-pounds-p.csv")
milk_ts <- ts(data = mlk$Milk, start = c(1962,1), frequency = 12 )
```

Data Partition

- We have divided the data into
  - Training Data : Data from Jan 1962 to November 1972
  - Validation Data : Data from December 1972 to December 1975

```
# Number of Observations in Validation data
nValid <- 38
# Number of Observations in Training data
nTrain <- nrow(mlk) - nValid
# Training data and Validation data partitioned using window()
train.ts <- window(milk_ts, start = c(1962,1), end = c(1962, nTrain + 1))
valid.ts <- window(milk_ts, start = c(1962, nTrain + 2))
```

Smoothing Methods

- Smoothing Methods are a kind of forecasting methods that are data driven
- These methods directly estimate time series components from the data
- We will be learning:
  - Moving Average
  - Simple Smoothing
  - Holt's Method
  - Holt-Winter's Method

Moving Average

- The consecutive values of the time series are averaged with a specific width maintained.
- A moving average with width ***w*** means average taken across each set of ***w*** consecutive time series values, where ***w*** is an integer input by the user.
- There are two types of moving averages:
  - Centered Moving Average
  - Trailing Moving Average

Centered Moving Average

- Centered Moving Average are powerful for visualization
- The value of the moving average at time ***t*** is computed by centering the time span around time ***t*** and averaging across ***w*** values within the time span
- The goal is to suppress the seasonality to better visualize the trend. Hence choosing width as length of seasonal cycle is more desirable

Centered MA Calculations

- With a time span w=5, the moving average at time point t=3 would be average of 1^{st}, 2^{nd}, 3^{rd}, 4^{th}, 5^{th} time points.
- At time span w=4, moving average would be average of 2^{nd} , 3^{rd} , 4^{th} , 5^{th} , 6^{th} time points
- Function **ma**() in the *forecast* package creates a moving average

Syntax : ma(x, order, center=TRUE)

Where

x : Object of class ts

order : span of moving average

center : logical value, if true then moving average is centered for even orders

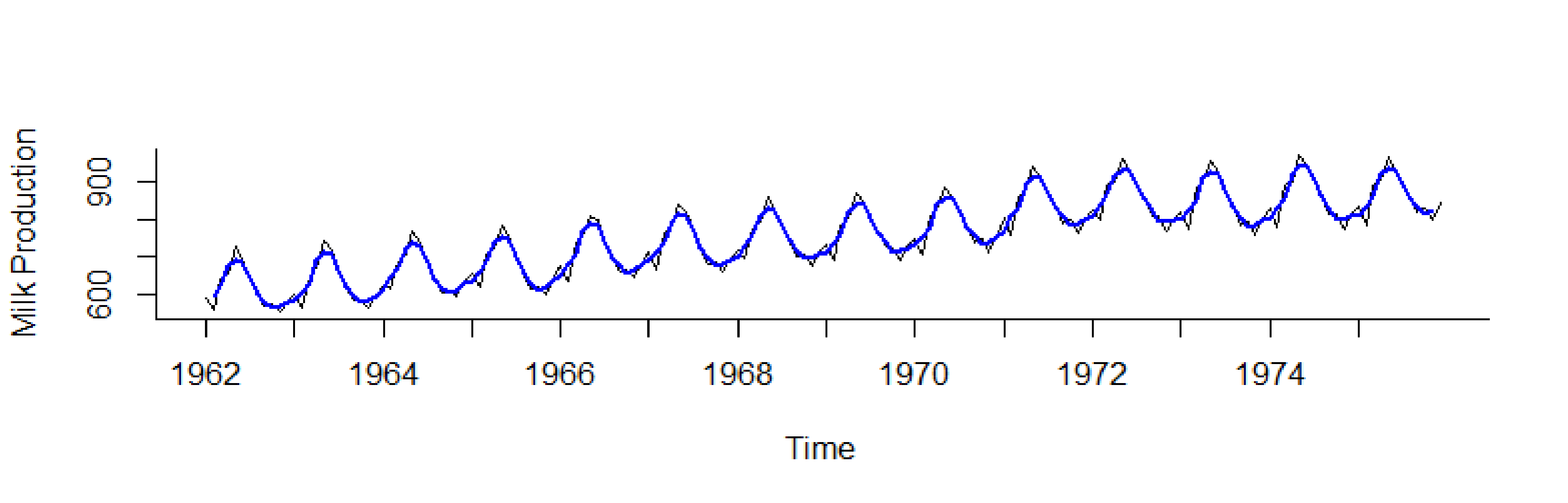
When w is even

- When order w is even then, centered MA is calculated as average of the two asymmetric moving averages
- When order w = 4,

$$MA_t = \frac{\left[\frac{(y_{t-2} + y_{t-1} + y_t + y_{t+1})}{4} + \frac{(y_{t-1} + y_t + y_{t+1} + y_{t+2})}{4}\right]}{2}$$

Centered MA Example

```
library(forecast)
maMilk1Cent  <- ma(milk_ts, order = 3)
plot(milk_ts,   ylab = "Milk Production", xlab = "Time", bty = "l", xaxt = "n", main = "")
axis(1, at = seq(1962, 1975, 1), labels = format(seq(1962, 1975, 1)))
lines(maMilk1Cent , lwd = 2, col = "blue")
```



Trailing Moving Average

- Centered MAs use both past and future time points, so they cannot be used for forecasting
- For forecasting trailing moving average can be used because here average is calculated in a time span for past and most recent time point
- The k-step ahead forecast F_{t+k} is computed with the formula:

$$F_{t+k} = (y_t + y_{t-1} + \ldots + y_{t-w+1}) / w$$

Trailing MA Calculation

- Trailing MA can be calculated with the function rollmean() in package zoo.

Syntax : rollmean(ts , w , align, …)

Where

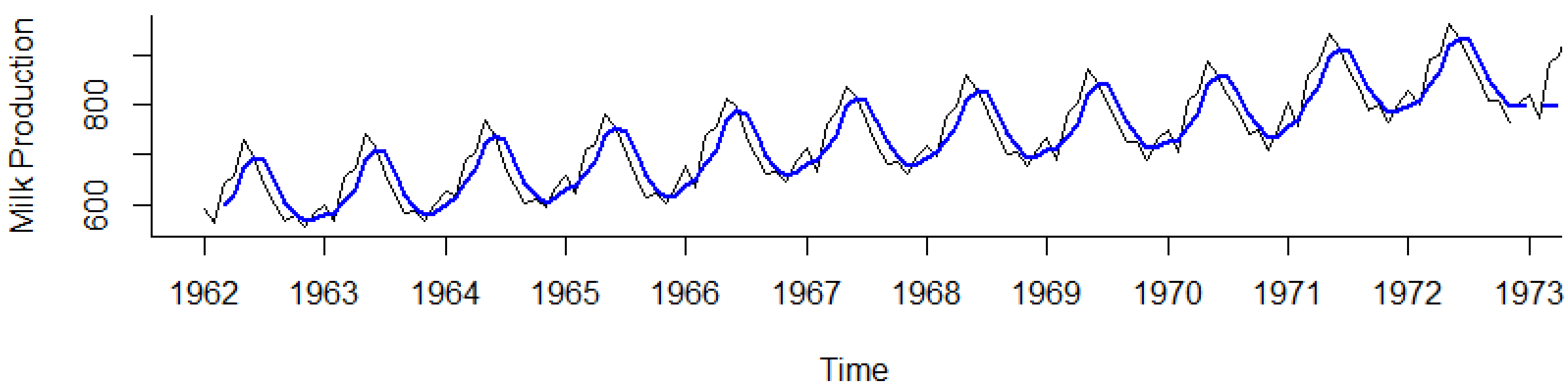
ts : Time series Object

w : Window span for moving average

align : character specifying whether the index of the result should be left- or right-aligned or centered

Trailing MA Example

```r
library(zoo)
maMilk3Trail <- rollmean(train.ts, k = 3, align = "right")
lastMA <- tail(maMilk3Trail, 1)
predMA <- ts(rep(lastMA, nValid), start = c(1962, nTrain + 1),
             end = c(1962, nTrain + nValid), freq = 12)
plot(train.ts,   ylab = "Milk Production", xlab = "Time", bty = "l",
     xaxt = "n",  main = "")
axis(1, at = seq(1962, 1975, 1), labels = format(seq(1962, 1975, 1)))
lines(maMilk3Trail, lwd = 2, col = "blue")
lines(predMA, lwd = 2, col = "blue", lty = 2)
lines(valid.ts)
```



Accuracy

- The function accuracy() calculates various summary measures of forecast accuracy

Syntax : accuracy(f , a , …)

where

f : an object of class forecast or a numerical vector containing the forecasted values

a : an object of class forecast or a numerical vector containing the actual values

Measures of Accuracy

The measures calculated are:

- ME: Mean Error
- RMSE: Root Mean Squared Error
- MAE: Mean Absolute Error
- MPE: Mean Percentage Error
- **MAPE: Mean Absolute Percentage Error**
- **MASE: Mean Absolute Scaled Error**
- ACF1: Autocorrelation of errors at lag 1.
- Theil's U: Uncertainty Coefficient

```
> accuracy(predMA,valid.ts)
               ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
Test set 58.21622 83.45836 64.91892 6.371956 7.239719 0.6635945  1.582979
```

Differencing

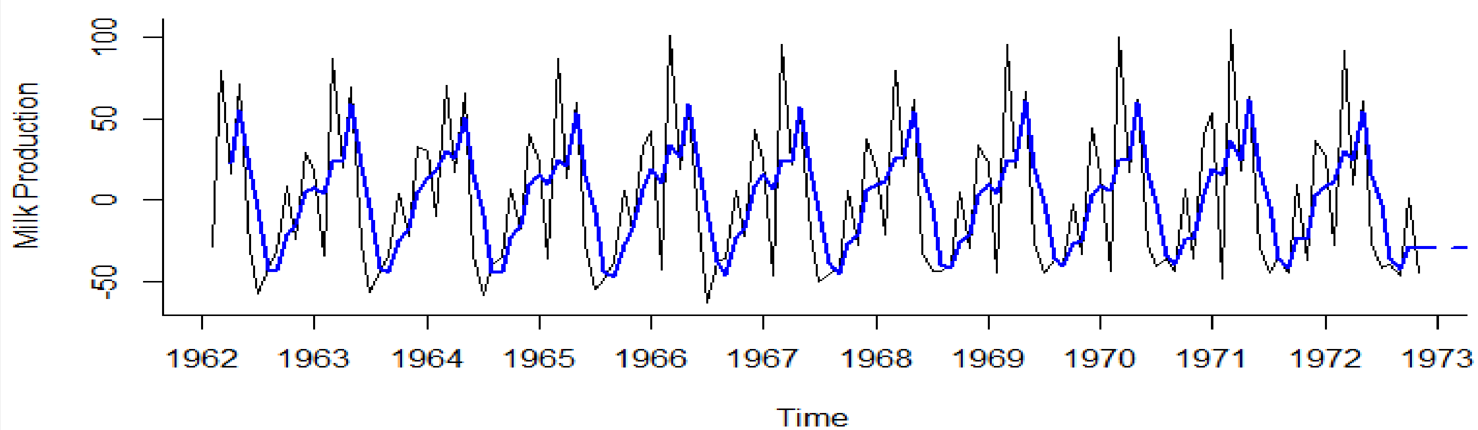
- Differencing means taking the difference between values of time points
- Differencing may be between consecutive values or between values with lag more than one
- A lag-1 difference means taking the difference between two consecutive time points $(y_t - y_{t-1})$
- A lag-k difference means taking difference between value and its value k-periods back $(y_t - y_{t-k})$

Removing Trend and De-seasonalizing

- Lag-1 differencing is useful for removing trend
- To remove seasonal pattern with M seasons, we can difference at lag M
- e.g. To remove a day-of-week pattern in daily data, we can take lag-7 differences

Differencing Example

```
df.train.1 <- diff(train.ts,lag = 1)
library(zoo)
maMilk3Trail <- rollmean(df.train.1, k = 3, align = "right")
lastMA <- tail(maMilk3Trail, 1)
predMA <- ts(rep(lastMA, nValid), start = c(1962, nTrain + 1),
             end = c(1962, nTrain + nValid), freq = 12)
plot(df.train.1,   ylab = "Milk Production", xlab = "Time", bty = "l",
     xaxt = "n",  main = "")
axis(1, at = seq(1962, 1975, 1), labels = format(seq(1962, 1975, 1)))
lines(maMilk3Trail, lwd = 2, col = "blue")
lines(predMA, lwd = 2, col = "blue", lty = 2)
```



Simple Exponential Smoothing

- In Simple Exponential Smoothing, weighted average of all past values is taken in such a way that the weights decrease exponentially into past
- Like Moving Average, this method is used for forecasting series that have no trend and no seasonality

Calculation

- The exponential smoother calculates a forecast at time t+1, F_{t+1}:

$$F_{t+1} = \propto y_t + \propto (1 - \propto) y_{t-1} + \propto (1 - \propto)^2 y_{t-2} + \cdots$$

  - Where $\propto$ is a constant between 0 and 1 called smoothing constant

- The above equation can also be written as:

$$F_{t+1} = F_t + \propto e_t$$

  - Where F_t is forecast error at time t and e_t is forecast error at time t

Choice of $\propto$

- The smoothing constant $\propto$ determines the rate of learning
- A value close to 1 implies fast learning, i.e. the most recent values influence the forecast most
- A value close to 0 implies slow learning, i.e. the past observations influence the forecast most
- The default values of $\propto$ that have been mostly observed to work well are between 0.1 and 0.2.

Simple Smoothing in R

- We can calculate time series estimates of simple exponential smoothing with the help of **ses()** in package ***forecast***.

Syntax : ses(ts , h , alpha)

where

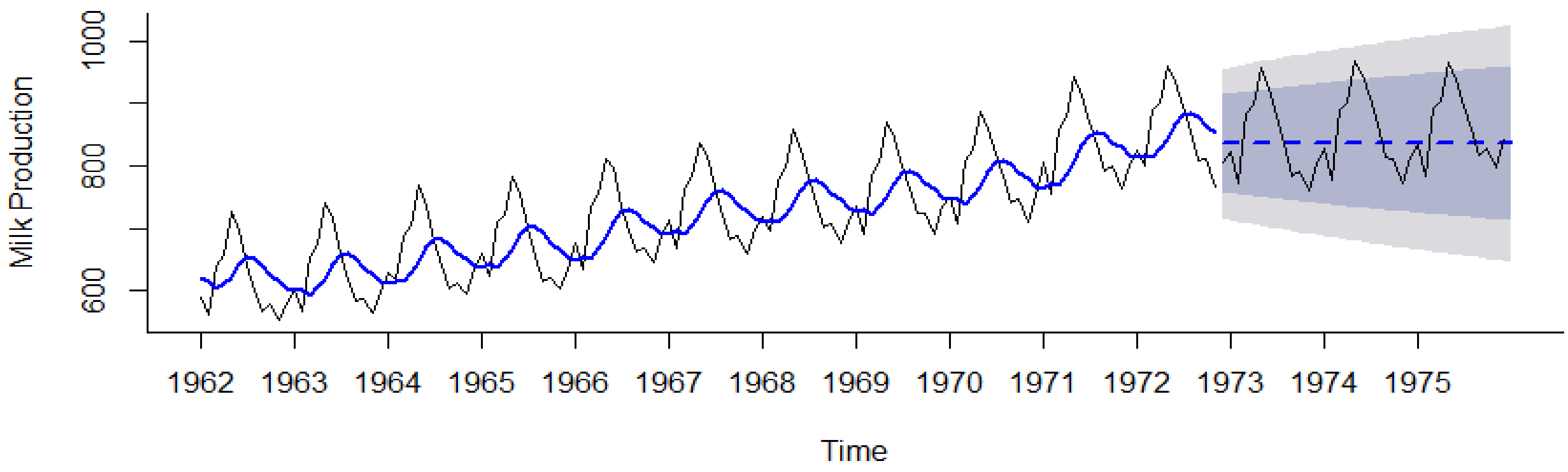
ts : a numeric vector or time series

h : Number of periods for forecasting

alpha : Smoothing constant for level, if not specified it will be estimated by the function

Simple Exponential Example

```
sesMilk <- ses(train.ts, h = nValid, alpha = 0.2)
plot(sesMilk,   ylab = "Milk Production", xlab = "Time", bty = "l",
     xaxt = "n",  main = "", flty = 2)
axis(1, at = seq(1962, 1975, 1), labels = format(seq(1962, 1975, 1)))
lines(sesMilk$fitted, lwd = 2, col = "blue")
lines(valid.ts)
```



Model & Accuracy

```
> sesMilk$model
Simple exponential smoothing

Call:
 ses(x = train.ts, h = nValid, alpha = 0.2)

  Smoothing parameters:
    alpha = 0.2

  Initial states:
    l = 620.6362

  sigma:  61.2608

     AIC     AICc      BIC
1718.818 1718.849 1721.693
```

```
accuracy(sesMilk , valid.ts)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	8.198829	61.26084	51.90699	0.5052553	7.075074	2.153742	0.6857870	NA
Test set	17.770688	62.38563	51.60958	1.6086402	5.907381	2.141402	0.6635945	1.194783

ses() without alpha

```
> sesMilk_opt <- ses(train.ts, h = nValid)
> plot(sesMilk_opt,   ylab = "Milk Production", xlab = "Time", bty = "l", xaxt = "n",  main = "", flty = 2)
> axis(1, at = seq(1962, 1975, 1), labels = format(seq(1962, 1975, 1)))
> lines(sesMilk_opt$fitted, lwd = 2, col = "blue")
> lines(valid.ts)
> sesMilk_opt$model
Simple exponential smoothing

Call:
 ses(x = train.ts, h = nValid)

  Smoothing parameters:
    alpha = 0.9999

  Initial states:
    l = 589.0255

  sigma:  44.1128

     AIC     AICc      BIC
1634.779 1634.873 1640.530
> accuracy(sesMilk_opt , valid.ts)
                    ME      RMSE      MAE        MPE     MAPE     MASE       ACF1 Theil's U
Training set  1.351119  44.11282 38.00845 0.01697799 5.209431 1.577059 0.06249857        NA
Test set     87.211813 105.74531 87.53638 9.78679961 9.829505 3.632088 0.66359455  2.017068
```

ses() without alpha