Microsoft

Time Series Forecasting with Applications in R

Microsoft Codess Event

Kostas Mammas – Senior Data & Applied Scientist Manager

Today's Agenda

Theory

45 minutes

- Quick introduction to time series
- Statistics background for forecasting
- Stationarity in time series
- Time series decomposition
- Introduction to stochastic models

Application

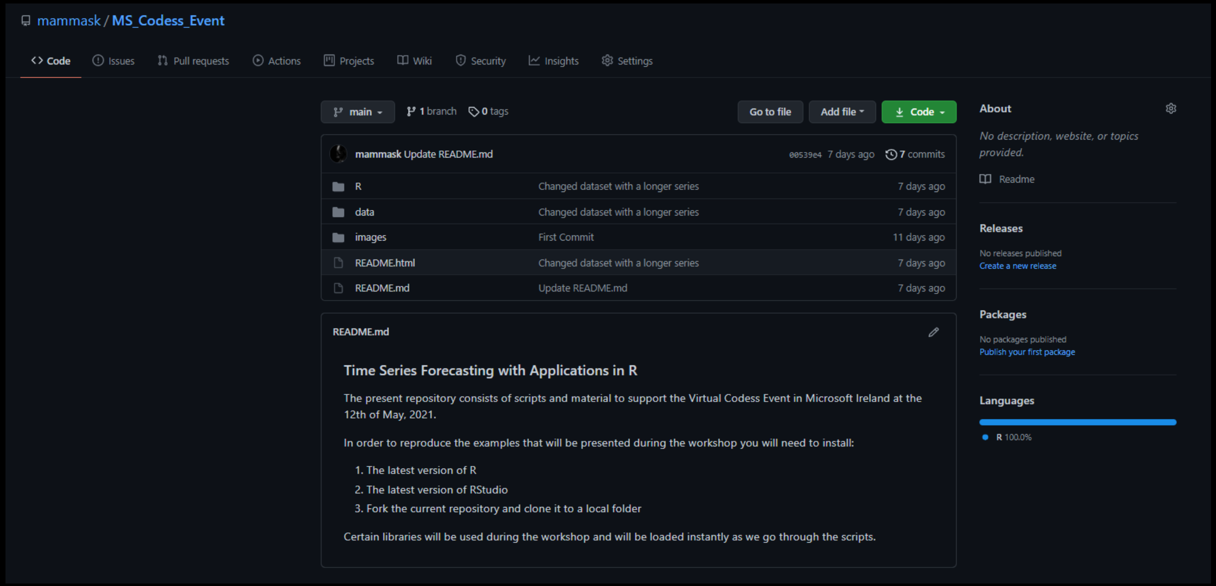
45 minutes

- Visualization and exploratory analysis of time series
- Stationarity checks
- Time series decomposition
- Forecasting time series using stochastic models

Requirements

In order to reproduce the examples that will be presented in today's session you will need to:

- Install the latest version of R
- Install the latest version of RStudio
- Fork → Clone the following repository: https://github.com/mammask/MS_Codess_Event.git
    - Available scripts: https://github.com/mammask/MS_Codess_Event/blob/main/R/TsForecastingMS.md



Microsoft

"A forecast is a prediction of some future event or events…"

Famous bad forecasts over the years

“The population is constant in size and will remain so right up till the end of mankind”,
L’ Encyclopedie,1756

“Computers are multiplying at a rapid rate. By the return of the century there will be 220,000 in the US”, *Wall Street Journal, 1966*

“Rising seas could obliterate nations by 2000”, *Associated Press, 1989*

Microsoft

Introduction to time series

Definition:

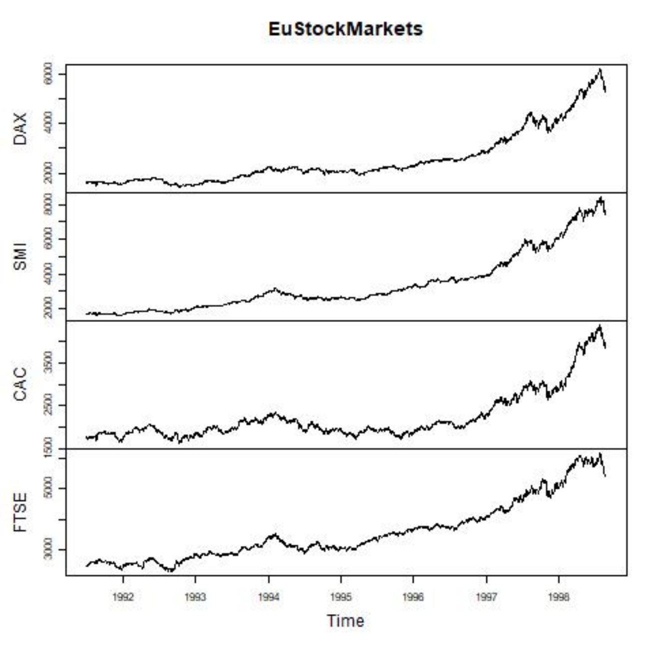
Time series is a set of sequence data points that occur in successive order over some period. It is mathematically defined as a set of vectors:

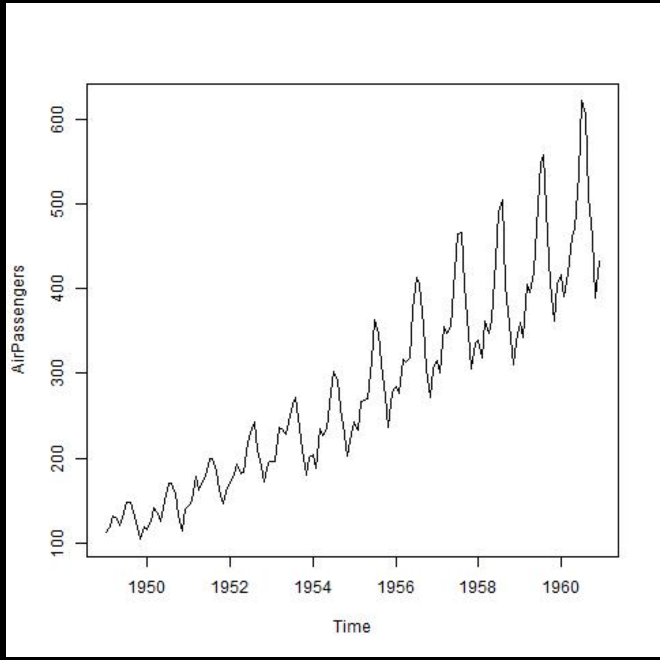
$x(t), t = 0,1,2, \dots n$

where t represents the time elapsed. The variable, $x(t)$, is treated as a random variable.

```
data("EuStockMarkets")
plot(EuStockMarkets)
```

```
data("AirPassengers")
plot(AirPassengers)
```



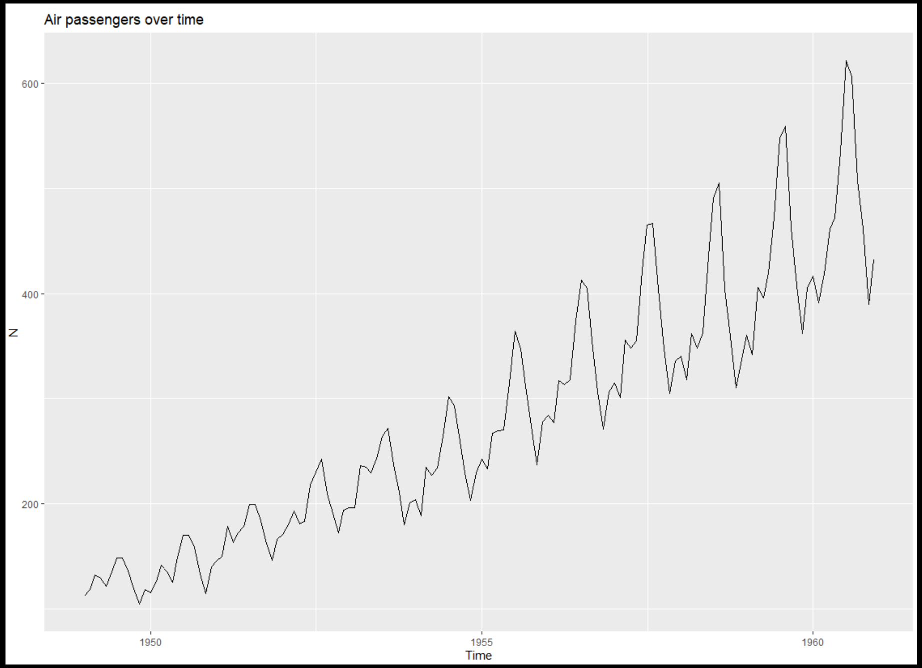


Microsoft

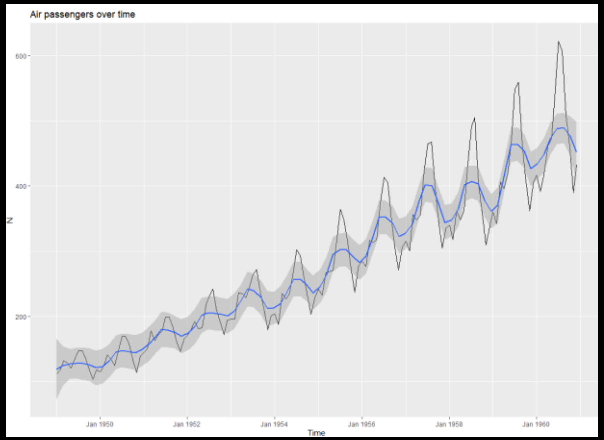
Statistics background for forecasting

Time series plots – Graphical description of time series

```r
library(ggplot2)
data("AirPassengers")
p = autoplot(AirPassengers, facets = FALSE) +
  ggtitle("Air passengers over time") +
  xlab("Time") +
  ylab("N")
```




```r
library(ggplot2)
library(data.table)
library(zoo)

data("AirPassengers")
mdf = data.table(Date = as.yearmon(time(AirPassengers)),
                 AirPassengers = as.numeric(AirPassengers)
                 )

p = ggplot(data = mdf) + geom_line(aes(x = Date, y = AirPassengers)) +
  geom_smooth(method = 'loess', aes(x = Date, y = AirPassengers), span = 0.15) +
  ggtitle("Air passengers over time") +
  xlab("Time") +
  ylab("N")
```



Microsoft

Numerical description of time series data

Stationary time series

Definition – Strictly stationary:

A time series is said to be strictly stationary if its properties is not affected by a change in the time origin. That is if the joint probability distribution of the observations $x_t, x_{t+1}, \ldots, x_{t+n}$ is exactly the same as the joint probability of the observations $x_{t+k}, x_{t+k+1}, \ldots, x_{t+k+n}$. This is also called shift invariance of its finite-dimensional distribution

Numerical description of time series data

Stationary time series

Definition – Weakly stationary:

A time series is said to be weakly stationary if:

1. The first moment of $x(t)$, the mean, μ_t, is constant and does not depend on time, t, and
2. The second moment of $x(t)$, if finite for all t, $E[x^2] < \infty$
3. The cross moment, the autocovariance, depends only on the difference $u - v$. For all u, v, k, $cov(x_u, x_v) = cov(x_{u+k}, x_{v+k})$

The 3rd condition implies that every lag, $t \in N$ has a constant variance associated with it: $cov(x_{t1}, x_{t2})$

This also implies that the variance of this process is also constant since we get that for all $t \in N$:

$$var(x_t) = cov(x_t, x_t) = d$$

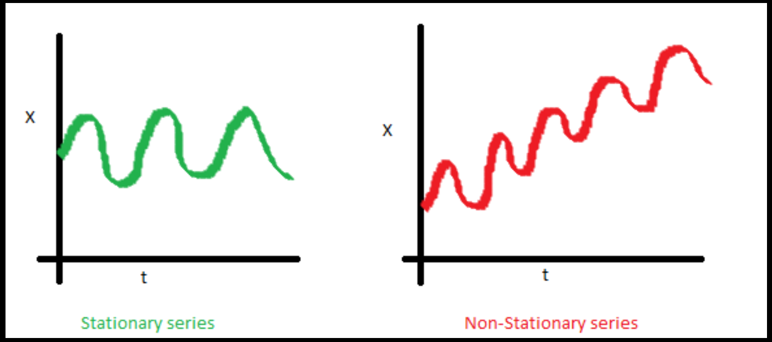
Definition – White noise:

1. First moment is zero
2. Second moment is finite
3. Cross-moment is zero

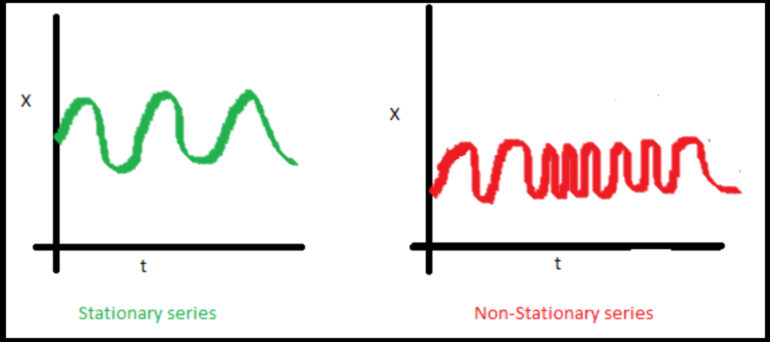
Numerical description of time series data

Stationary/ non-stationary time series?

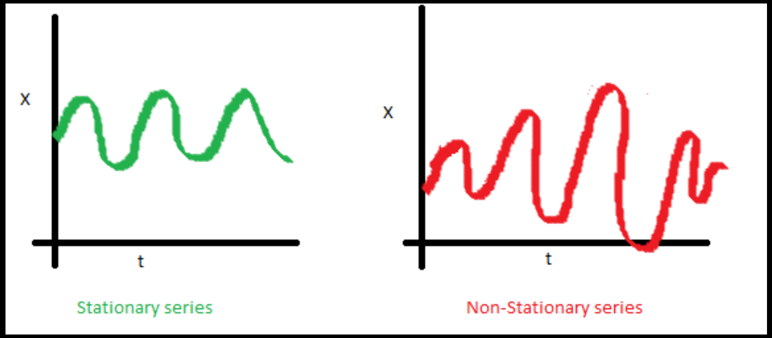
The mean is a function of time! It is time-dependent



The covariance is a function of time! It is time-dependent



The variance is a function of time! It is time-dependent



Microsoft

Numerical description of time series data

Stationary time series

Why stationarity is important?

1. Stationarity is very important for modelling/ forecasting non-independent data

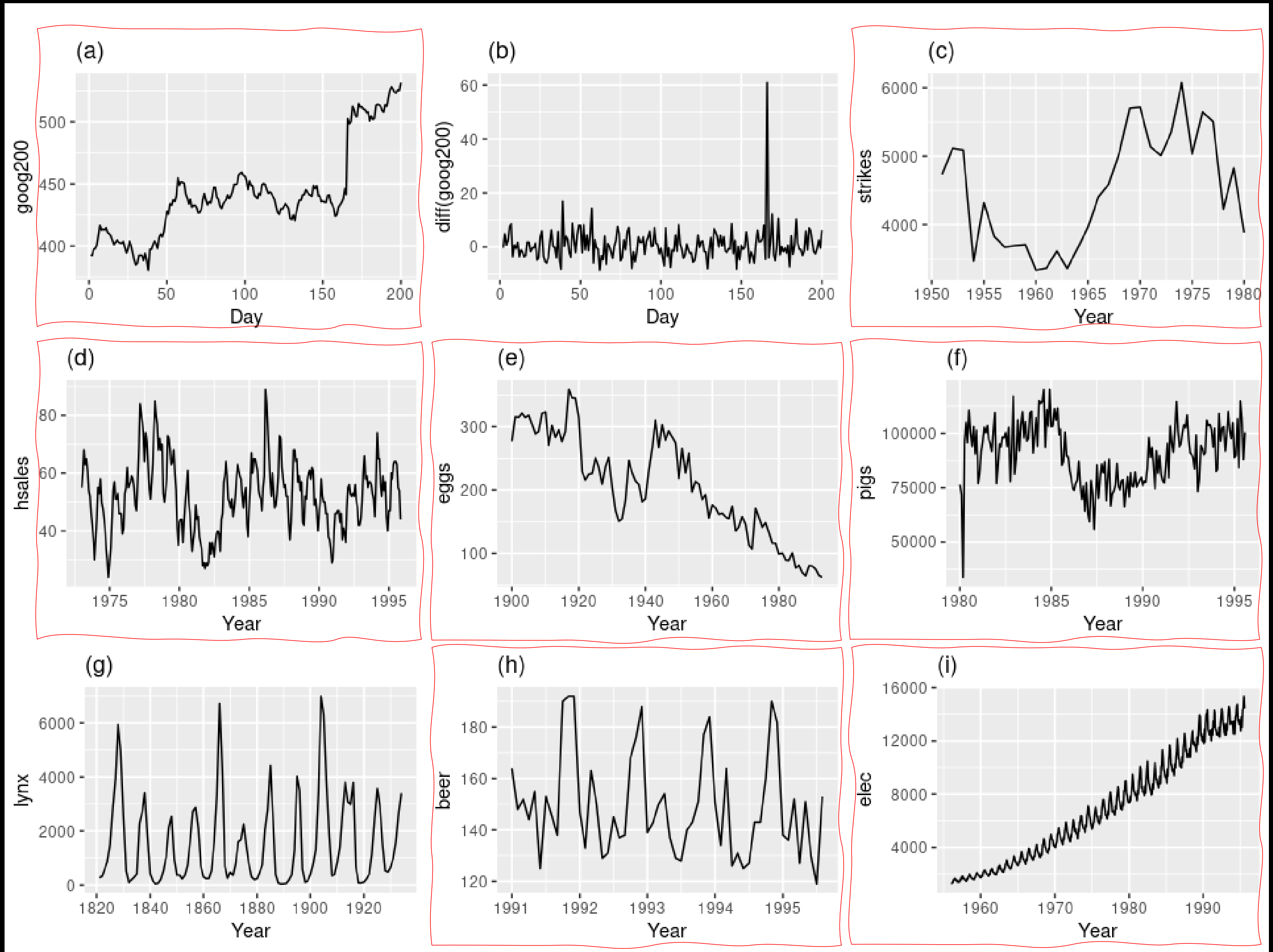
2. A stationary process is easier to predict – We simply predict that the statistical properties of the series will be the same in the future as they have been in the past

3. The derivation of simple statistics, such as the mean, variance, and correlation with other variables are useful as descriptors of the future only if the series is stationary

Microsoft

Numerical description of time series data

Stationary time series – Quiz – How many stationary time series appear in the graph?



Statistical investigation of time series stationarity

Stationary time series

- KPSS *(Kwiatkowski–Phillips–Schmidt–Shin, 1992)* is one of the most well-known stationarity tests and can be used to objectively identify whether a time-series is stationary or not.

- The null hypothesis is that the data are stationary, and we look for evidence to reject it.

- Consider the following AR(1) process:

$$y_t = \rho \times y_{t-1} + x_t' \times \delta + \varepsilon_t$$

- x_t' are optional exogenous variables
- ρ and δ are parameters to be estimated
- ε_t are assumed to be white noise
- If $|\rho| > 1$, the y is a non-stationary process

The KPSS Lagrange Multiplier test evaluates $H_0: \rho < 1$ against the alternative $H_1: \rho \geq 1$

How do we achieve stationarity?

Differencing time series

Differencing is a well-known technique that is used to remove the trend of a non-stationary time series and sometimes even the seasonality!

Backward shift operator:

$$\mathrm{B}y_t = y_{t-1}$$

as its name implies, the Backward shift operator simply outputs the previous observation

Lag-1 difference operator:

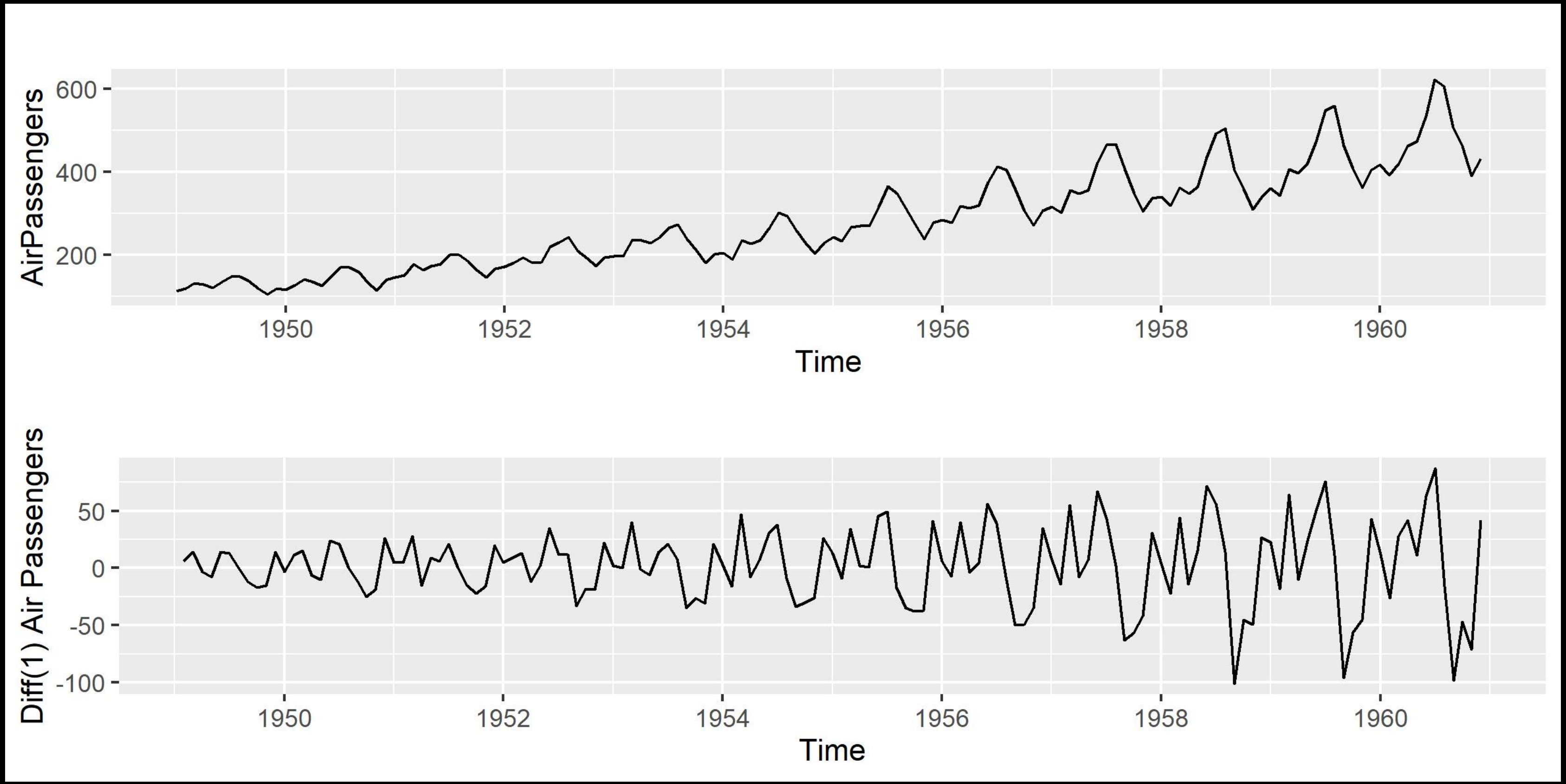
$$\nabla y_t = y_t - y_{t-1} = (1 - \mathrm{B})y_t$$

which is the difference between the current and the previous value

How do we achieve stationarity?

Example: Air passengers

```
library(ggplot2)
library(ggfortify)
library(forecast)
library(tseries)
library(gridExtra)

data("AirPassengers")

p = autoplot(AirPassengers)
pdiff = autoplot(diff(AirPassengers)) +
 ylab('Diff(1) Air Passengers')
ptotal = grid.arrange(p, pdiff, ncol = 1)

kpss.test(AirPassengers)

KPSS Test for Level Stationarity

data:  AirPassengers
KPSS Level = 2.7395, Truncation lag parameter
= 4, p-value = 0.01

kpss.test(diff(AirPassengers))

KPSS Test for Level Stationarity

data:  diff(AirPassengers)
KPSS Level = 0.014626, Truncation lag paramete
r = 4, p-value = 0.1
```



Time series Patterns

Trend, seasonal, cyclic components

Trend: Represents long/ short term increase/ decrease in the data. It does not have to be monotonic or linear

Seasonal: Occurs when time series is affected by a seasonal factors as the time of the year or day of the week. It should be a fixed value

Cyclic: Occurs when the data exhibits rises and falls that are not of a fixed frequency

Time series decomposition

How do we perform time series decomposition?

Additive decomposition

$$y_t = S_t + T_t + R_t$$

where y_t is the time series, S_t is the seasonal component, T_t, is the trend component and R_t is the remainder component under the assumption that additive decomposition can describe best the time series

Multiplicative decomposition

$$y_t = S_t \times T_t \times R_t$$

Which one is the most appropriate?

Additive decomposition works best when the magnitude of the seasonal variations or the variation around the trend – cycle does not vary with the level of the time series

Microsoft

Time series decomposition

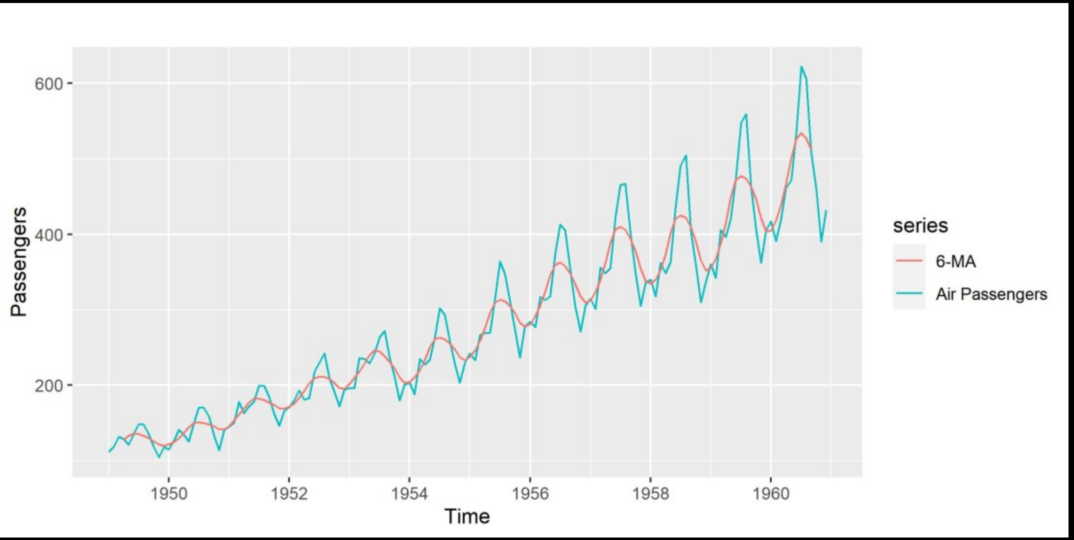
Moving average smoothing

Moving average smoothing

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

where $m = 2k + 1$, that is the estimate of the trend cycle at time t

```r
library(ggplot2)
library(ggfortify)
library(forecast)

data("AirPassengers")

autoplot(object = AirPassengers, series = 'Air Passengers') +
  autolayer(ma(AirPassengers,6), series="6-MA") +
  xlab('Time') + ylab('Passengers')
```



Microsoft

Time series decomposition

Additive decomposition

$$y_t = S_t + T_t + R_t$$

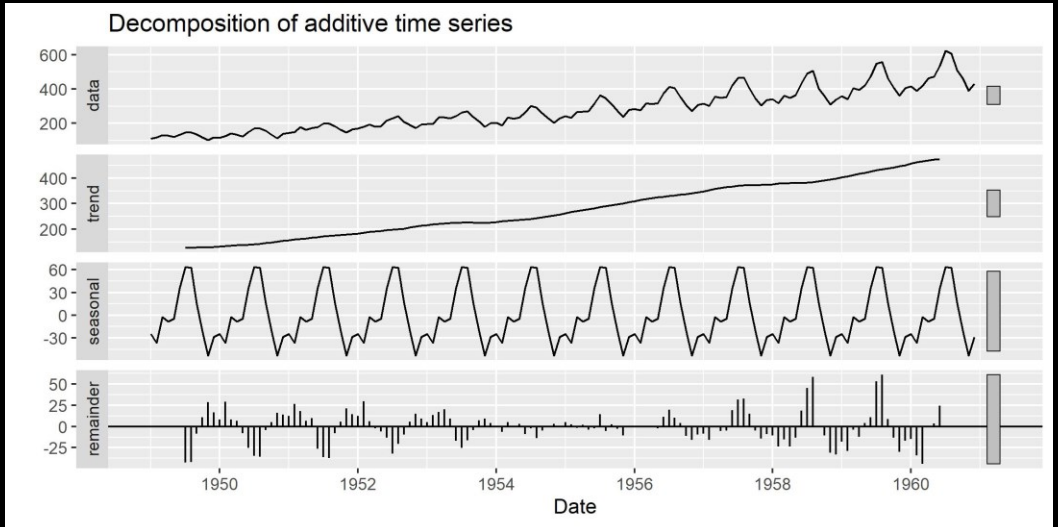
Step 1: Compute the trend cycle component T_t

Step 2: Calculate the detrended series $y_t - T_t$

Step 3: To estimate the seasonal component for each season, simply average the detrended values for that season

Step 4: Calculate the remainder component simply by: $R_t = y_t - S_t - T_t$

```
library(ggplot2)
library(ggfortify)
library(forecast)

data("AirPassengers")
p = autoplot(decompose(AirPassengers, type = "additive")) +
  xlab("Date")
```



Microsoft

Time series decomposition

Multiplicative decomposition

$$y_t = S_t \times T_t \times R_t$$

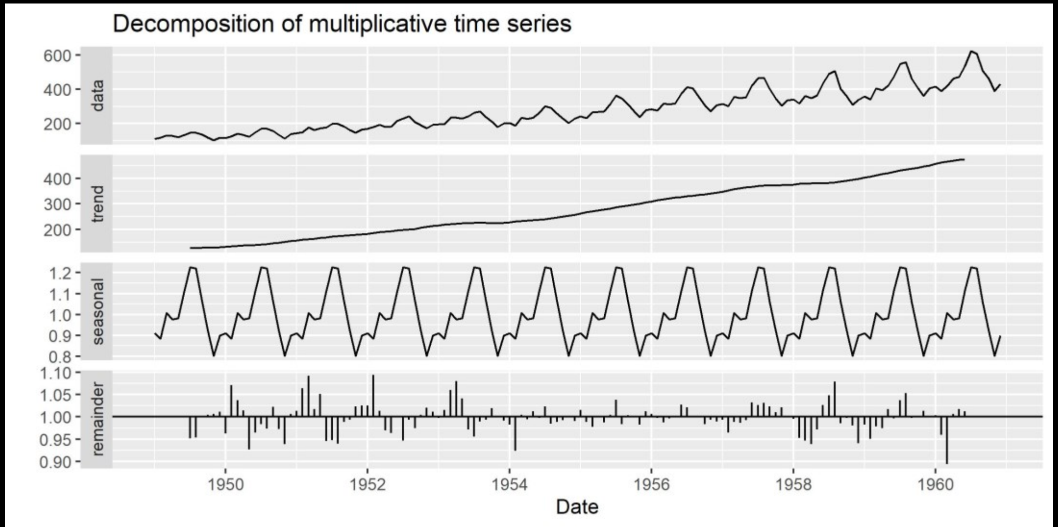
Step 1: Compute the trend cycle component T_t

Step 2: Calculate the detrended series $\frac{y_t}{T_t}$

Step 3: To estimate the seasonal component for each season, simply average the detrended values for that season

Step 4: Calculate the remainder component simply by: $R_t = \frac{y_t}{S_t \times T_t}$

```
library(ggplot2)
library(ggfortify)
library(forecast)

data("AirPassengers")
p = autoplot(decompose(AirPassengers, type = "multiplicative")) +
  xlab("Date")
```



Introduction to stochastic linear models

Autoregressive model (AR)

- Models the response y_t at time t as a linear function of its p previous values and some independent random noise:

$$y_t = \varphi_0 + \varphi_1 \times y_{t-1} + \varphi_2 \times y_{t-2} + \cdots + \varphi_p \times y_{t-p} + \varepsilon_t$$

this is defined as $\mathrm{AR}(p)$ model.

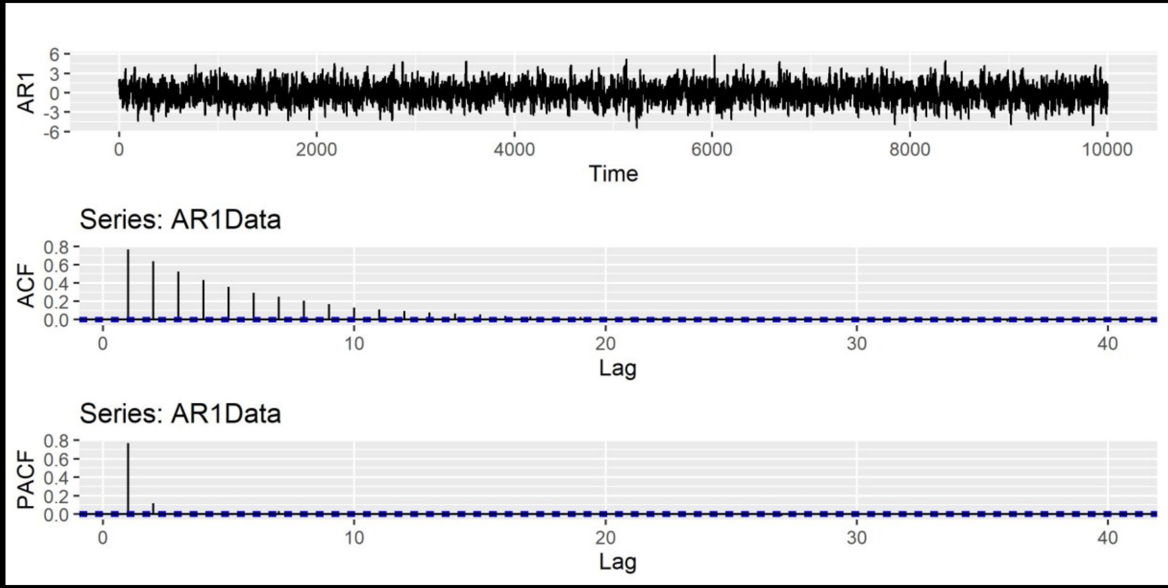
- For an $\mathrm{AR}(1)$ model, when:
  - When $\varphi_1 = 0$, y_t is equivalent to white noise
  - When $\varphi_1 = 1$ and $\varphi_0 = 0$, y_t is equivalent to a random walk
  - When $\varphi_1 = 1$ and $\varphi_0 \neq 0$, y_t is equivalent to a random walk with drift

Selecting the AR order

Autoregressive model (AR)

- Assuming that the series is stationary we can create ACF (auto-correlation) and PACF (partial auto-correlation) plots
- For an AR process we expect:
  - ACF plot will gradually decrease
  - PACF plot should have a sharp drop after p significant lags
- This leads to an $\mathrm{AR}(p)$ process

```
library(ggplot2)
library(ggfortify)
library(forecast)
library(gridExtra)

set.seed(885)

# Simulate an AR process
AR1 = list(order = c(1, 0, 0), ar = c(0.7))
AR1 = arima.sim(n = 10000, model = AR1)

p0 = autoplot(AR1)
p1 = ggAcf(AR1)
p2 = ggPacf(AR1)
p = grid.arrange(p0,p1,p2, ncol = 1)
```



Microsoft

Introduction to stochastic linear models

Moving average model (MA)

- Models the response y_t at time t as a linear function of its p previous values and some independent random noise:

$$y_t = \theta_0 + \theta_1 \times \varepsilon_{t-1} + \theta_2 \times \varepsilon_{t-2} + \cdots + \theta_p \times \varepsilon_{t-p} + \varepsilon_t$$

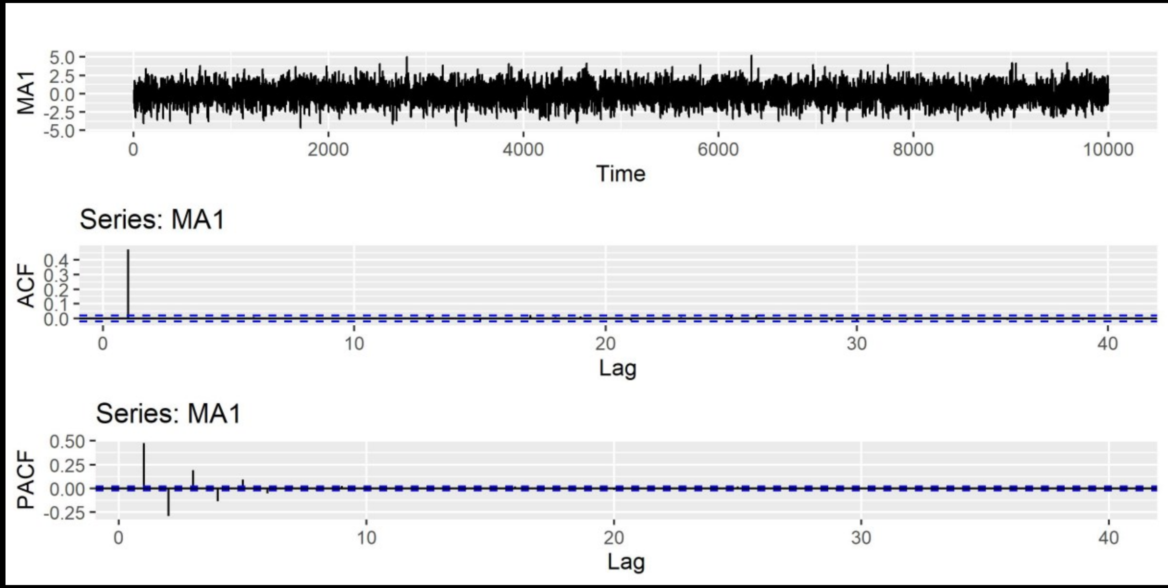
where ε_t is white noise. This is defined as $\mathrm{MA}(p)$ model.

Selecting the MA order

Moving average model (MA)

- Assuming that the series is stationary we can create ACF (auto-correlation) and PACF (partial auto-correlation) plots
- For an MA process we expect:
    - ACF plot should have a sharp drop after p significant lags
    - PACF plot would gradually decrease
- This leads to an $\mathrm{AR}(p)$ process

```
library(ggplot2)
library(ggfortify)
library(forecast)
library(gridExtra)

set.seed(885)

# Simulate an AR process
MA1 = list(order = c(0, 0, 1), ma = c(0.7))
MA1 = arima.sim(n = 10000, model = MA1)

p0 = autoplot(MA1)
p1 = ggAcf(MA1)
p2 = ggPacf(MA1)
p = grid.arrange(p0,p1,p2, ncol = 1)
```



Microsoft

Introduction to stochastic linear models

Autoregressive Integrated Moving Average (ARIMA)

- If we combine differencing with an autoregression and moving average model, then we have a non-seasonal ARIMA model:

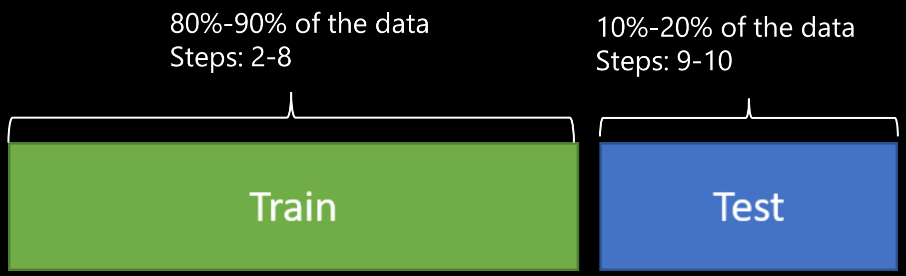
$$y'_t = c + \varphi_1 \times y'_{t-1} + \varphi_2 \times y'_{t-2} + \cdots + \varphi_p \times y'_{t-p} + \theta_1 \times \varepsilon_{t-1} + \theta_2 \times \varepsilon_{t-2} + \cdots + \theta_q \times \varepsilon_{t-q} + \varepsilon_t$$

where y'_t is the differenced time series. This model is called $\mathrm{ARIMA}(p, d, q)$ where:

p is the order of the autoregressive part
d is the degree of differencing
q is the order of the moving average part

Building a forecasting model

Out of sample validation (OOS)

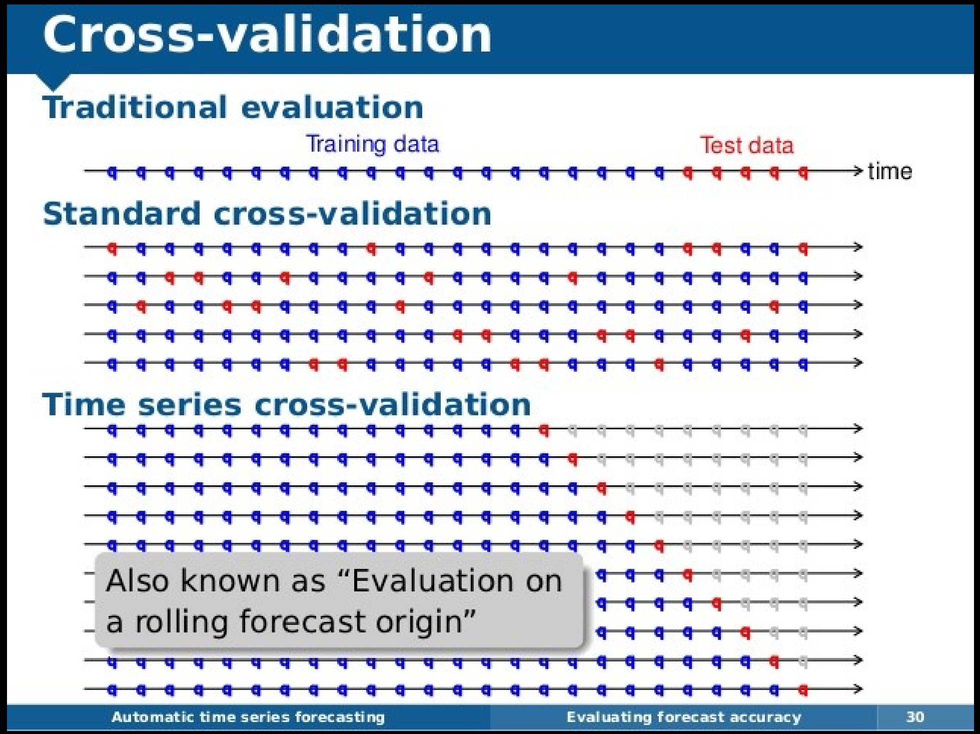


1. Split the dataset into training and test
2. Plot the time series in **the training set** to identify trends or seasonal characteristics
3. Investigate whether the time series is stationary using stationarity tests
4. Perform transformation of the time series to achieve stationarity, if necessary
5. Create ACF and PACF plots to define the number of AR and MA orders
6. Define the seasonal orders of the model if necessary
7. Fit different models using different parameterizations to make sure that the best model is developed
8. Obtain statistical measures such as the AIC/ BIC in the training set and select the model with the best performance
9. Use the best model and obtain the model performance in the test set
10. Walk forward one step at a time, update the model and predict the next step(s)

Building a forecasting model

Time series cross-validation

1. Split the dataset into k-folds by preserving the temporal order of the series and for each fold
2. Fit a model using a set of model hyperparameters
3. Obtain the forecast of the model in the next step(s)
4. Combine the model forecasts from the different folds
5. Calculate the test error and select the model with the optimal performance



Microsoft

Evaluating model performance

Scale dependent errors:

$$\text{Mean Absolute Error (MAE)} = \text{mean}(|e_t|)$$

$$\text{Root Mean Squared Error (RMSE)} = \sqrt{\text{mean}(e_t^2)}$$

Minimizing over MAE will lead to forecasts of the median, while minimizing RMSE will lead forecasts of the mean

Percentage errors:

$$\text{Mean Absolute Percentage Error (MAPE)} = \text{mean}(|p_t|)$$

main disadvantage when $y_t = 0$, leads to an undefined error and has extreme values when y_t is close to zero.

Thank you!