Module IN3031 / INM378 Digital Signal Processing and Audio Programming

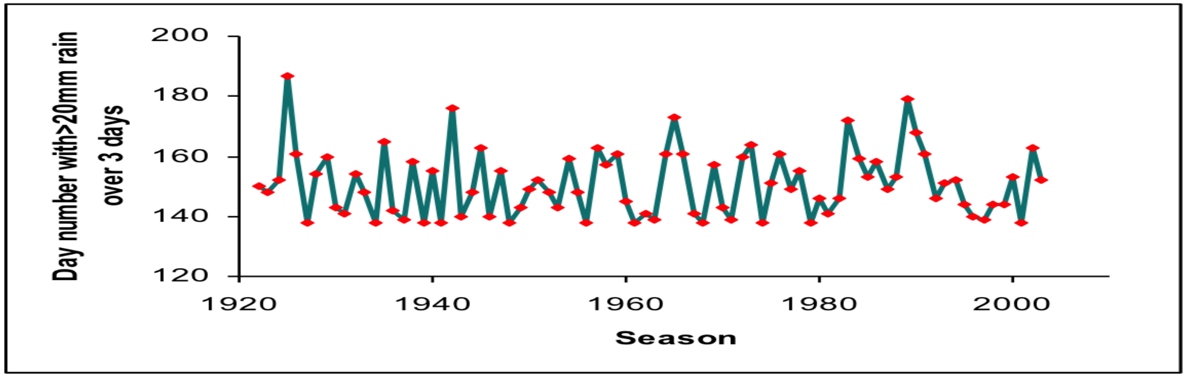
Johan Pauwels
johan.pauwels@city.ac.uk

based on slides by Tillman Weyde

Time Series Analysis and Prediction

Time Series Analysis

- Time Series: collection of **observations** $y_{t,}$ each one being recorded at **time *t***. (discrete, t = 1,2,3,... or continuous t > 0.)
- So it's (more or less) a **signal**, **but**
  - Might have **missing values**
  - Might be **sampled** at **unequal** time **intervals**
  - Typically at **longer time scale** than signals



Time Series Examples

- **Measurements**:
  - Meteorology: sun activity, tides, rainfall ...
- **Surveys**:
  - Moods, preferences, ...
- **Prices**
  - Stock markets, crop, livestock …
- **Etc** ...

Objectives of Time Series Analysis

Data **compression**

provide compact description of the data.

Explanation

seasonal factors

relationships with other variables (temperature, humidity, pollution, etc)

Signal **processing**

extracting a signal in the presence of noise

Prediction

use the model to predict future values of the time series.

Simple Signal / Time Series Descriptors

- Descriptive statistics
  - **Mean** $\bar{y} = 1/n \sum_{t=1}^{n} y_t$
  - **Variance** $\sigma^2 = 1/n \sum_{t=1}^{n} (y_t - \bar{y})^2$
  - **Skewness** $\sum_{t=1}^{n} (y_t - \bar{y})^3 / \sum_{t=1}^{n} [(y_t - \bar{y})^2]^{3/2}$
  - **Mode**: most frequent value
  - **Median**: half below, half above
    S=sort(y), median := $S[(n+1)/2]$ if n odd

    $(S[n/2]+S[(n/2)+1])$ if n even

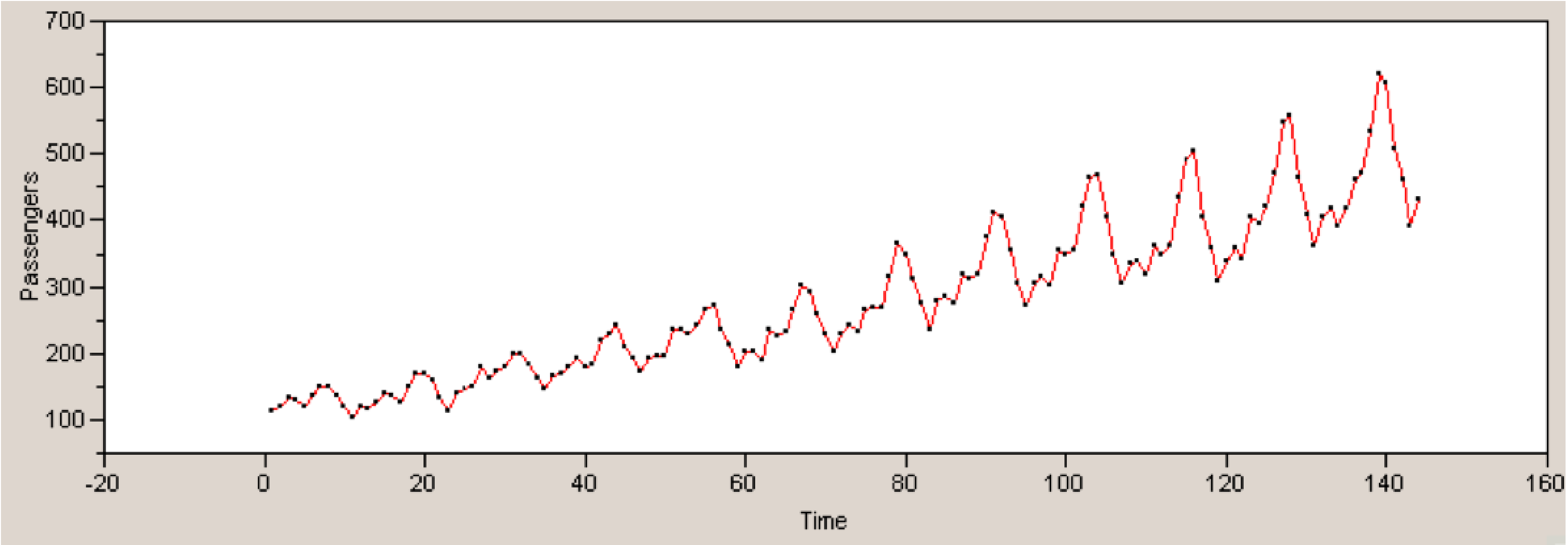
    using 1-based indices

General Modelling Approach

- **Deterministic + noise**: $y_t = f(t) + \varepsilon_t, E[\varepsilon_t] = 0$
- Modelling with different **types of *f***
  - *Autoregressive, harmonic, …*
- **Assumptions** about noise
- **Estimation** of **parameters** from **data**

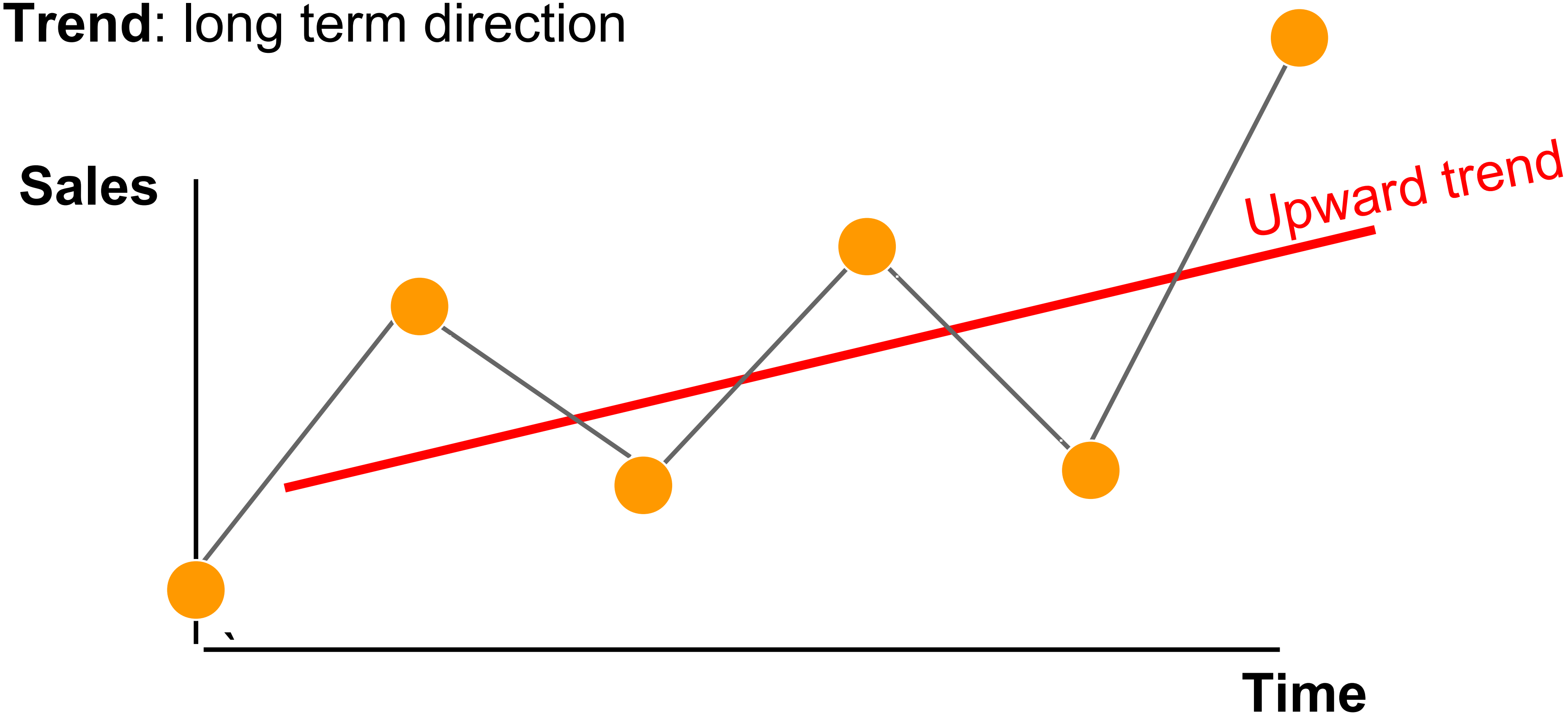
Time series model components

- **Trend + Seasonal + Cyclical + Irregular (noise)**



Trend component

- **Trend**: long term direction



Smoothing with Moving Average

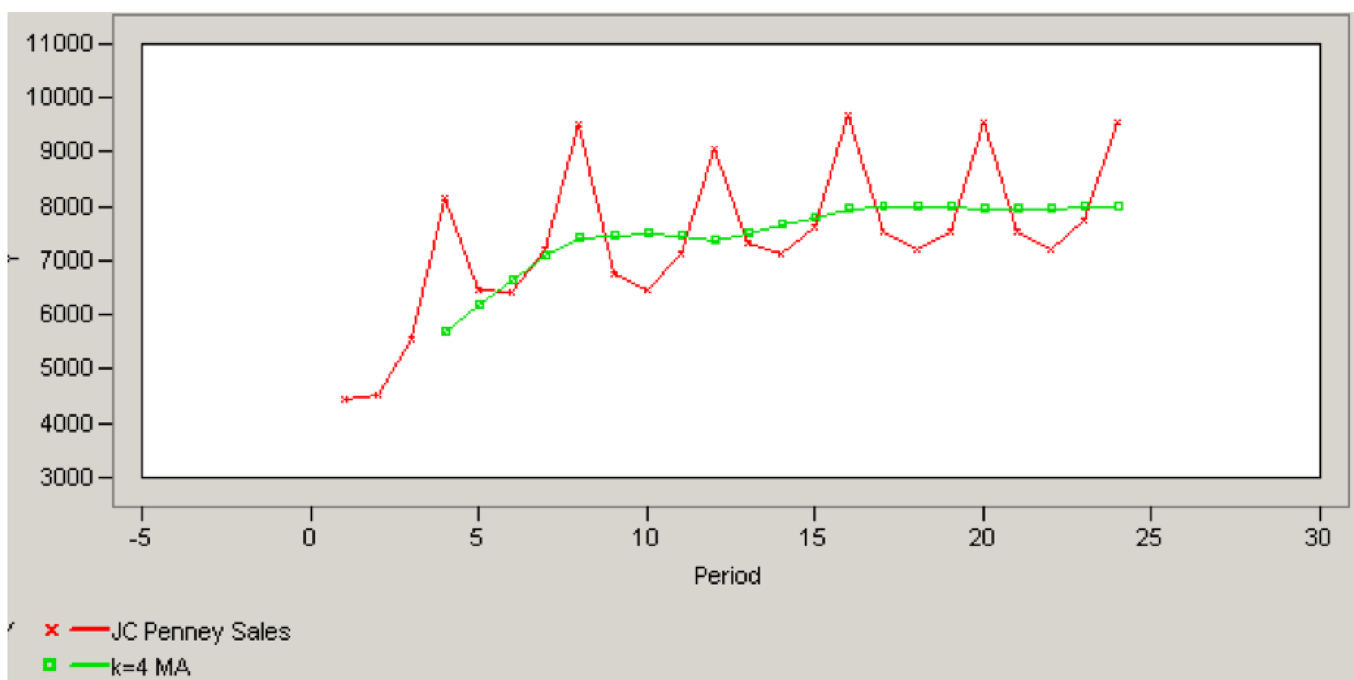
- **Moving average** of **span *k*** smoothes the data

$$\tilde{y}_t = (y_t + y_{t-1} + \ldots + y_{t-k-1})/k$$

- A **low pass FIR** filter with coefficients 1/k,1/k, … , 1/k
- In **Python**: `signal.lfilter([.25,.25,.25,.25], 1, x)`

Smoothing with Moving Average

- **Assumption**:
  - **high frequencies** are just **noise**, the long-term trend (low frequencies) matters
  - When **seasonal** effects with **cycle** = n are expected, use k = n



Exponentially Weighted Moving Average

- **Recursive average** of the data

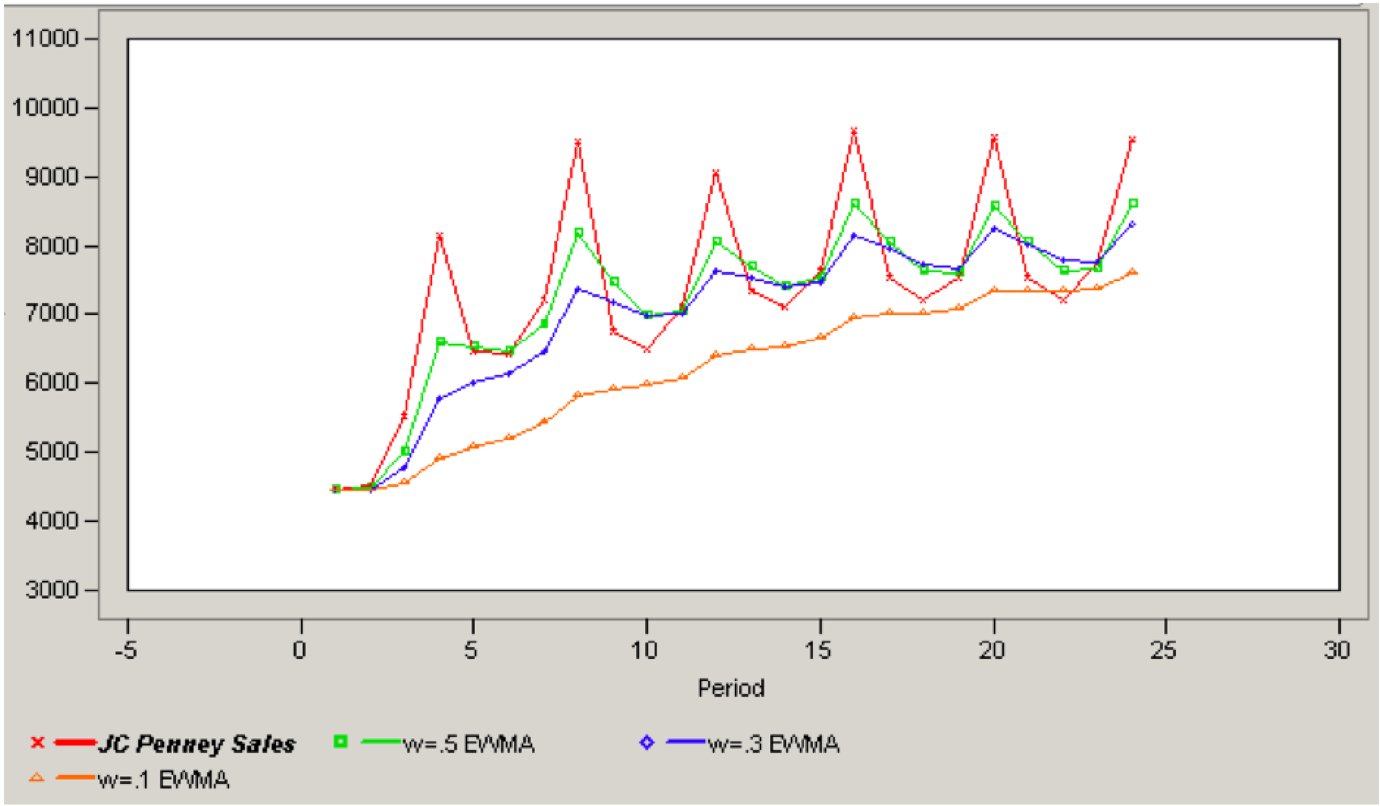
$$\tilde{y}_t = w\, y_t + (1-w)\, \tilde{y}_{t-1}$$

- A **low-pass IIR** filter with coefficients *w* and (1-*w*)
  In Python: `signal.lfilter([.25], [1, -(1-.25)], x)`

- **Assumption**:
  - **Recent values** are **more important** than older ones

Exponentially Weighted Moving Average

- Greater ***w*** means less filtering:



Linear Trend Estimation

- **Linear regression**: find a **straight line** to fit the **data**

$$\hat{y}_t = a_0 + a_1 t$$

- Determine a_0 and a_1 to **minimise the sum or squares error**

$$sse = \sum_t (\hat{y}_t - y_t)^2$$

- Solve the system of equations

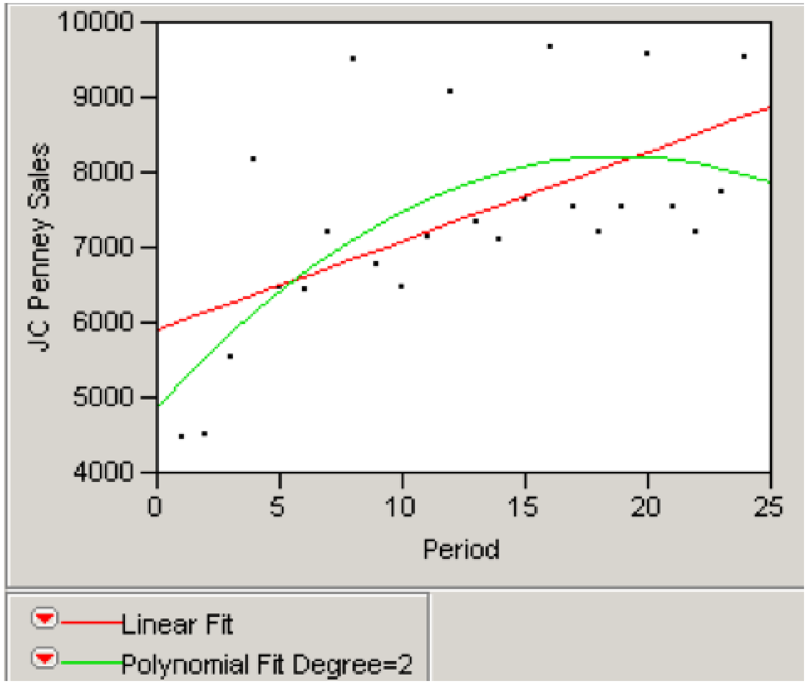
In Python:

```
from numpy.polynomial.polynomial import polyfit
coeff = polyfit(t, y, 1)
```

Quadratic Trend Estimation

- If the underlying trend is not linear, a **quadratic polynomial** may give a better fit.

  In Python: `coeff = polyfit(t, x, 2)`



Seasonal Structure

- **Natural pattern** of known period in **many types** of data (e.g. rainfall, heating, travel, employment, …)
- Modelling **seasonal regularity** can **reveal trends** and unusual **developments**
- Modelling **per month** or **quarter** per **year**, **hour** per day, **day** per week

Seasonal Average Method

- **Seasonal averages** = seasonal values total / # of years
- **General average** = seasonal averages total / # of seasons
- **Multiplicative modelling**:
  Seasonal index = seasonal average / general average
- **Additive modelling**:
  Seasonal offset = seasonal average – general average

Seasonal Average Example

Period, t	y_t	$\hat{y}_t$	$y_t - \hat{y}_t$	$\frac{y_t}{\hat{y}_t}$
1	4452	6022	−1570	.7393
5	6481	6497	−16	.9975
9	6755	6972	−217	.9685
13	7339	7447	−108	.9855
17	7528	7922	−394	.9503
21	7522	8397	−875	.8958
			−3180	5.5369

Then note that the average $y_t - \hat{y}_t$ is

$$\frac{-3180}{6} = -530$$

and the average $y_t/\hat{y}_t$ is

$$\frac{5.5369}{6} = .9228$$

Seasonal Average Example (2)

- **Linear model** (prediction for quarter 25)

$$\begin{aligned}\hat{y}_{25} &= 5903.2174 + 118.75261(25) \\ &= 8872\end{aligned}$$

- **Additive seasonal adjustment**

$$\hat{y}_{25} = 8872 + (-530) = 8342$$

- **Multiplicative adjustment**

$$\hat{y}_{25} = 8872(.9228) = 8187$$

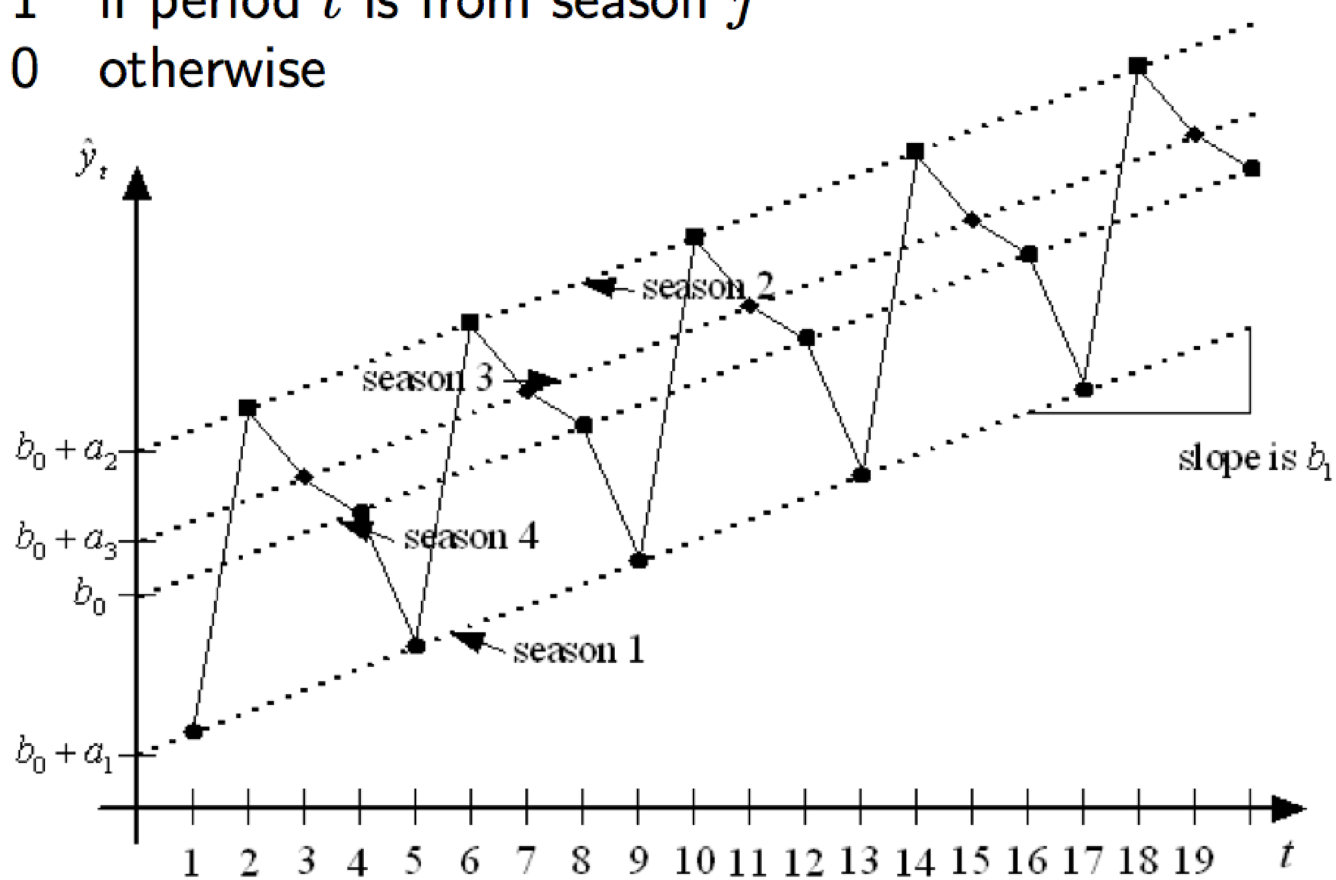
Season Variables

- **Seasonality** adjustment can be re-formulated with **dummy variables**

$$y_t \approx b_0 + b_1 t + a_1 x_{1,t} + a_2 x_{2,t} + \cdots + a_{k-1} x_{k-1,t}$$

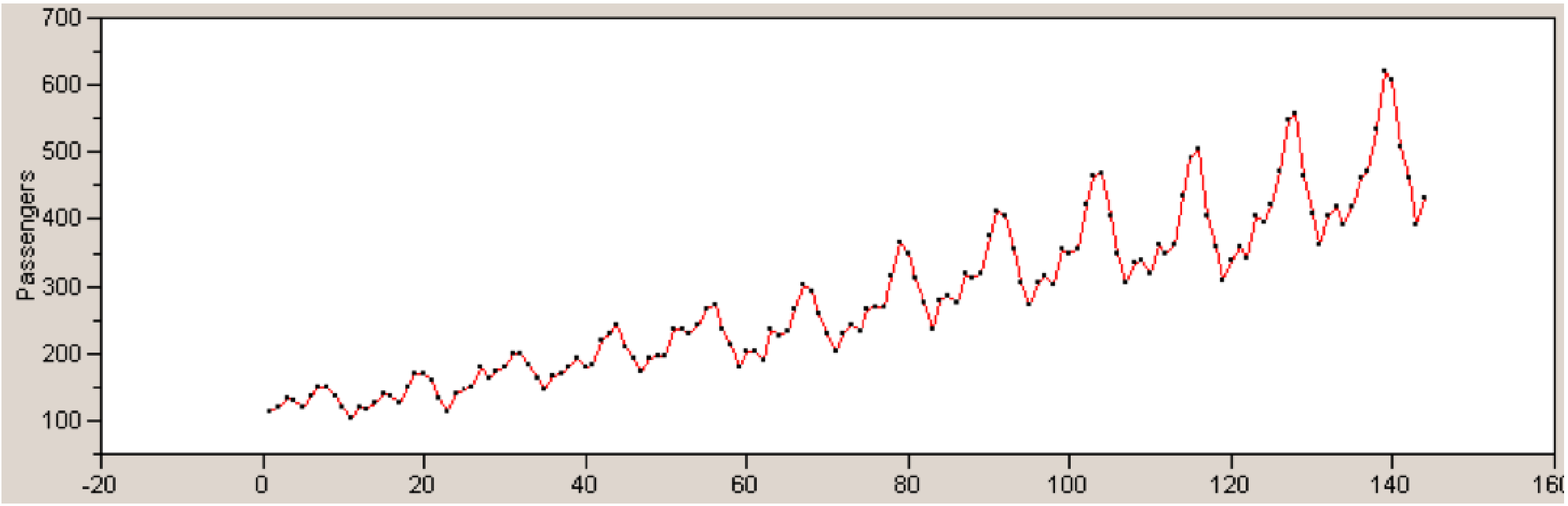
$$x_{j,t} = \begin{cases} 1 & \text{if period } t \text{ is from season } j \\ 0 & \text{otherwise} \end{cases}$$

Effectively changes the intercept per season



Rescaling

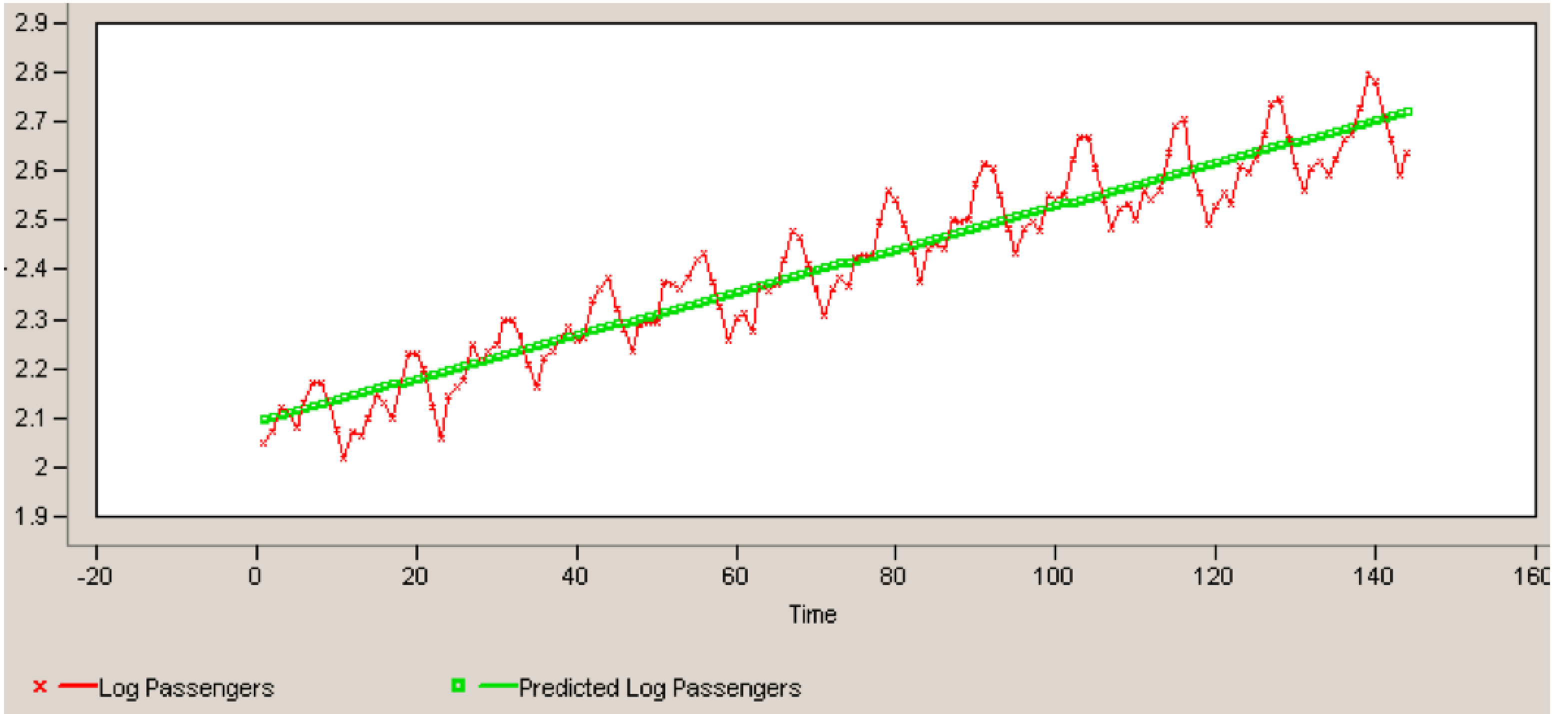
- Data can have an **exponential** structure (unrestricted growth, natural decay)



Rescaling (2)

- **Log scaling** can help reveal structure

$$y_t = \log_{10}(\text{passenger count at period } t)$$



Residual Autocorrelation

- After linear modelling and seasonal adjustment we can study the **autocorrelation** of the residuals $e_t = y_t - \hat{y}_t$

Correlations

	Residual Log Passengers	Lag 1 Residuals	Lag 2 Residuals
Residual Log Passengers	1.0000	0.7896	0.6722
Lag 1 Residuals	0.7896	1.0000	0.7832
Lag 2 Residuals	0.6722	0.7832	1.0000

- With linear regression we can improve the prediction based on residuals

$$\hat{e}_t = -0.000153 + 0.7918985 e_{t-1}$$

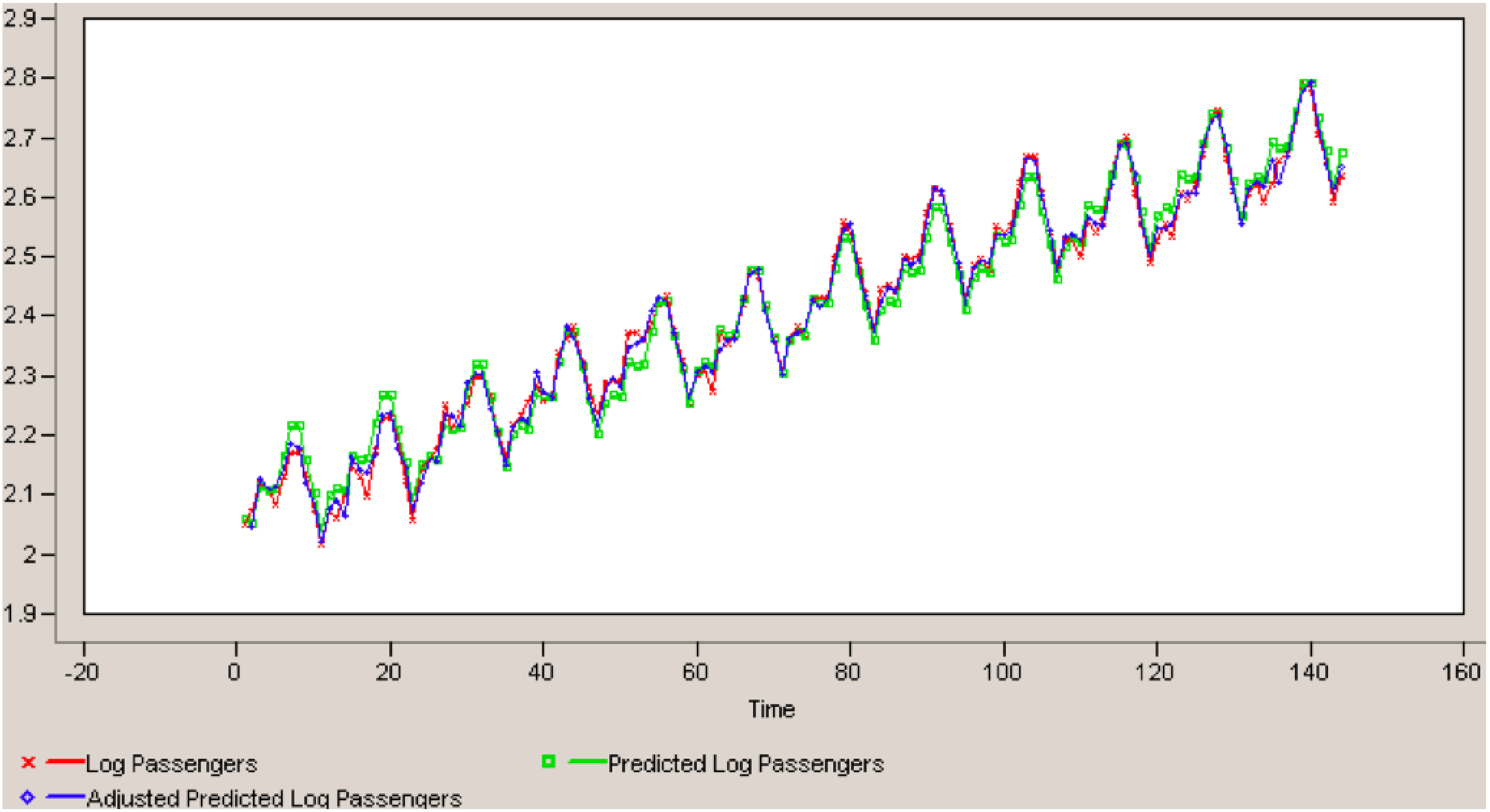
- This is a form of a **generalised** (weighted) **moving average**

$$y_t = \hat{y} + e_t + \sum_{i=1}^{q} \theta_i e_{t-i}$$

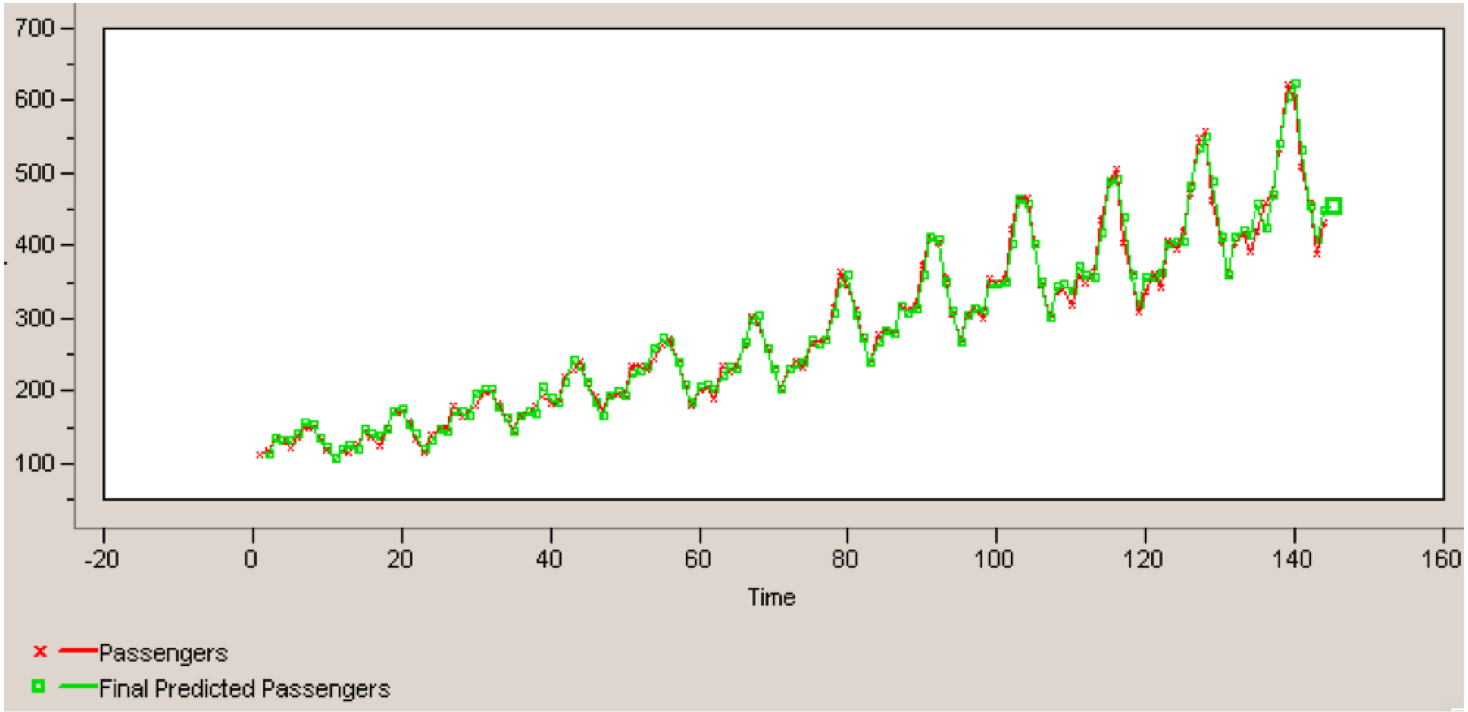
Adjusted Model

- We can **adjust** the prediction based on **residuals**

$$(adjusted\ fit)_t = \hat{y}_t + \hat{e}_t$$



Transformed Back to Linear



ARMA Model

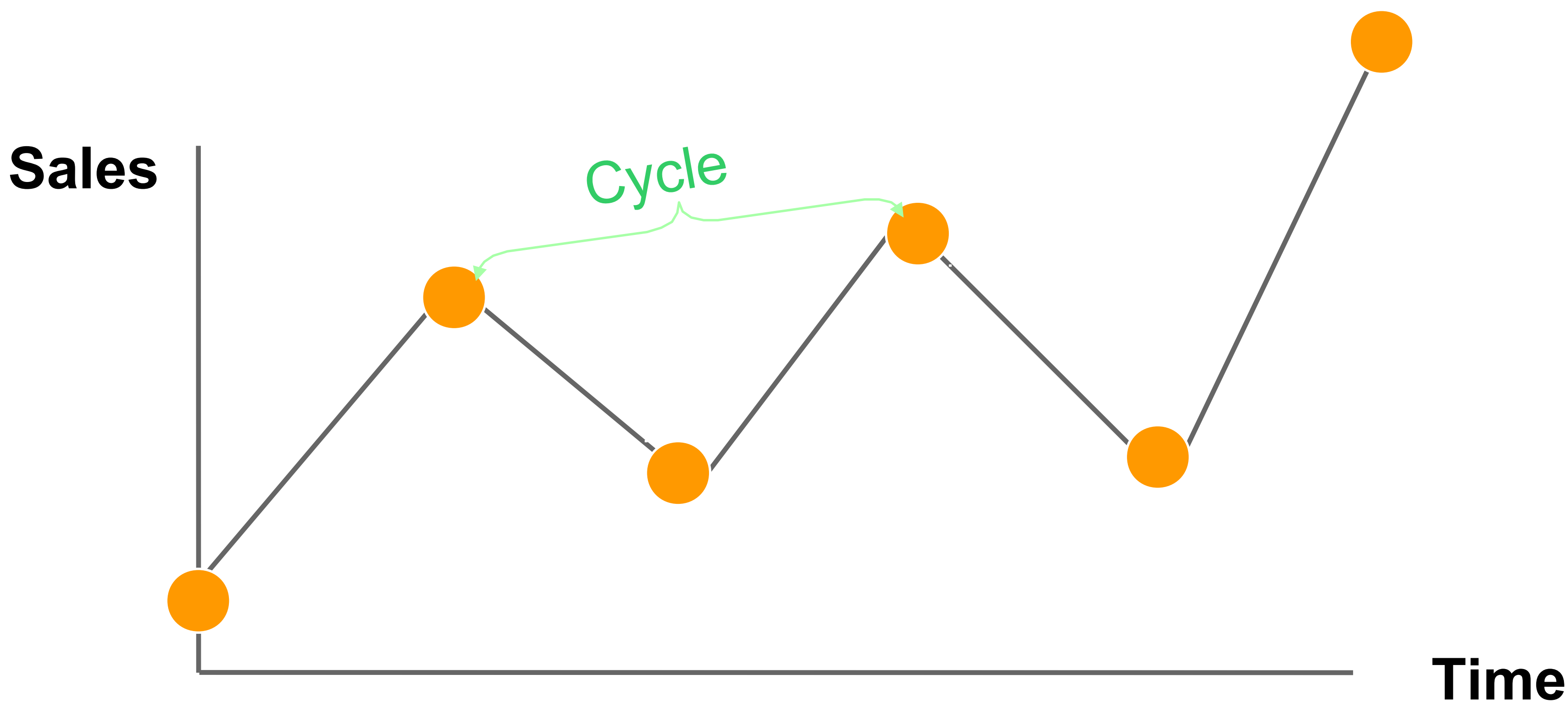
- Auto-Regressive Moving Average
  - **recursive model** (generalisation of exponential weighted moving average)
  - combined with **generalised moving** average

$$\hat{y}_t = c + e_t + \sum_{i=1}^{p} \phi_i \hat{y}_{t-i} + \sum_{i=1}^{q} \theta_i e_{t-i}$$

  - … which has the **structure of an IIR filter**
  - **optimise** the **parameters** using linear regression
  - available in Python library `statsmodels` (not used in class)

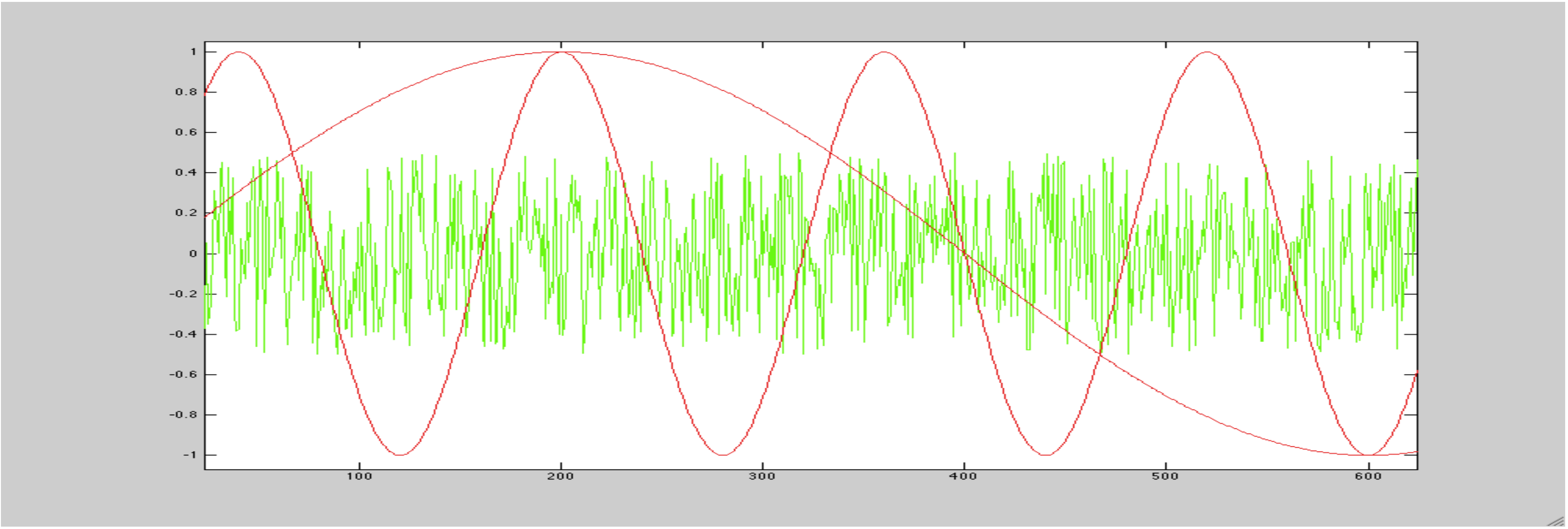
Cyclic Component

- There can be components of unknown **period** (e.g. economic cycles of 5-10 years)



Fourier Analysis

- Fourier (aka harmonic) analysis identifies **periodic components** (sinusoids)



Fourier Model

- Fourier coefficients are a **linear model** (remember lecture 3)

$$
\begin{aligned}
x(t) = b_0 \\
&+ a_1 \sin(2\pi 1 f t) + b_1 \cos(2\pi 1 f t) \\
&+ a_2 \sin(2\pi 2 f t) + b_1 \cos(2\pi 2 f t) \\
&+ a_3 \sin(2\pi 3 f t) + b_3 \cos(2\pi 3 f t) \\
&+ \ldots. \qquad = \sum_{k=0}^{\infty} a_k \sin(2\pi k t) + b_k \cos(2\pi k t)
\end{aligned}
$$

Predictive Modelling

- **ARMA, Fourier** and other approaches can provide **predictions**
- Approaches can be **combined** in generalised linear models
- Coefficients **minimising** sum of squared errors (SSE) can be calculated directly

Linear Model in Python

- Model with **arbitrary** component functions

$$y = a_0 + a_1 e^{-t} + a_2 t e^{-t}$$

- Vector with **input** values

```
t = np.array([0, 0.3, 0.8, 1.1, 1.6, 2.3])
```

- Vector with **output** values

```
y = np.array([0.6, 0.67, 1.01, 1.35, 1.47, 1.25])
```

- Create the design matrix (one column per component)
- ```
  X = np.stack([np.ones_like(t), np.exp(-t),
                      t * np.exp(-t)], axis=1)
  ```
- Calculate the model coefficients by solving the *Xa* = *y* for *a*, finding the least-squares solution
  ```

Linear Model in Python

- Calculate the **model coefficients** as least-squares solution

```
a = np.linalg.lstsq(X, y, rcond=None)[0]
```

- Calculate the model outputs (for the given values)

```
y_hat = np.matmul(X, a)
```

- And the (maximal) error

```
MaxErr = np.amax(np.abs(y_hat - y))
```

- And the squared error

```
SSE = np.sum(pow(y_hat - y, 2))
```

- And on new data

```
t2 = np.arange(0, 2.6, 0.1)
y2 = np.matmul(np.stack([np.ones_like(t2),
    np.exp(-t2), t2 * np.exp(-t2)], axis=1), a)
plt.plot(t2, y2), plt.plot(t, y, 'o')
```

Modelling Caveats

- Overfitting:
  - too many parameters → model learns noise in the data but not the trend
  - need to test on data not used in building the model
  - cross-validation can when data is scarce
- Predictions get less reliable further away from sample data

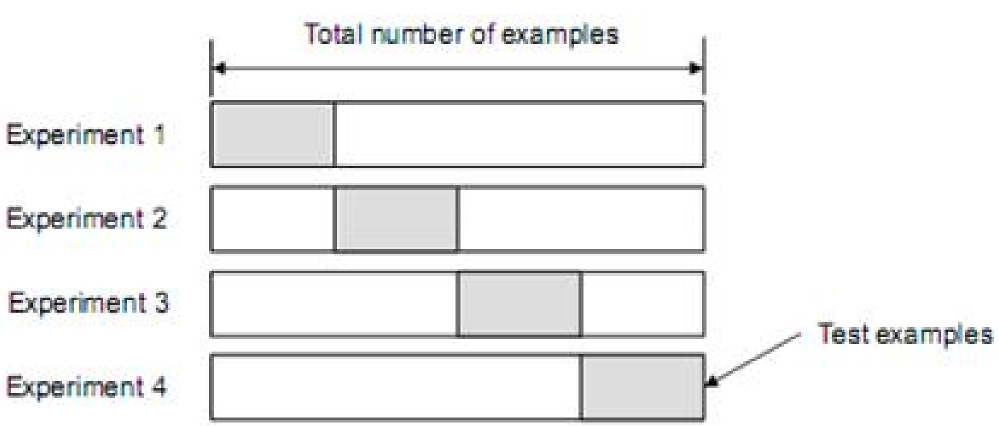
Maximum Likelihood and Regularisation

- Common approach: Linear models with least squares optimisation
  - **Maximise** the **likelihood** of the data given the prediction (assuming normal distribution)

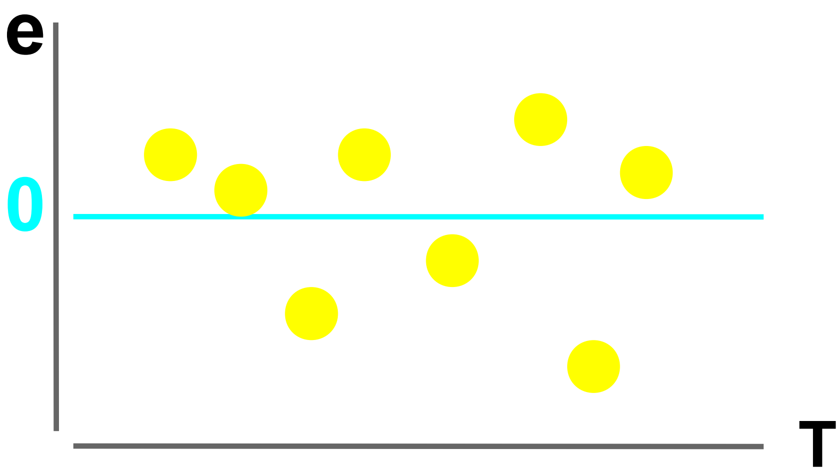
- **Regularisation** helps avoid overfitting (especially with small datasets)
  - Most popular: keep size of parameters low using a '**penalty term**': sum of squares or absolutes of the parameters (*ridge* or *lasso*)
  - Add penalty term to errors and calculate gradient to optimise (or use packaged solutions)

Cross-Validation for Regularisation

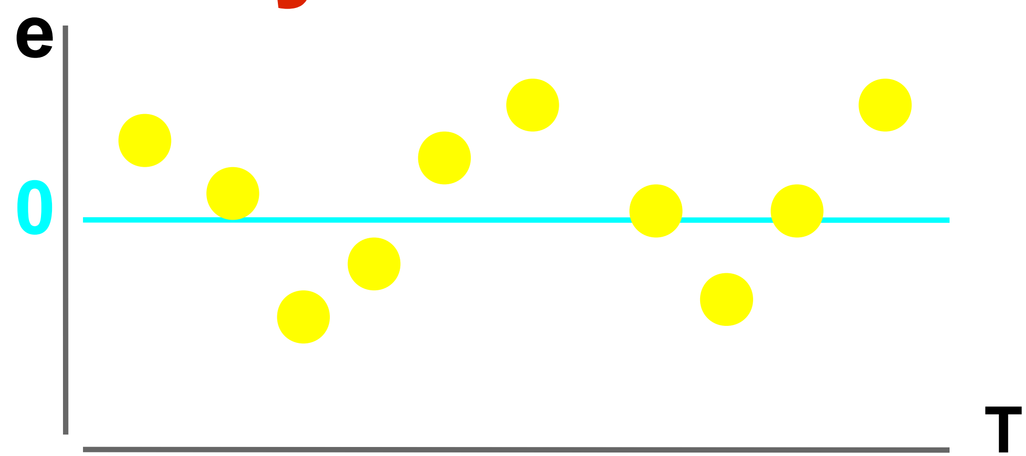
- **Optimise regularisation and other parameters:**
  - Divide the data into *k* equally sized subsets ('folds')
  - Adapt the model to k-1 joint subsets, test on the remaining subset, and iterate through all folds
  - Test a grid of regularisation values (or other parameters) and choose the one with best results on test sets



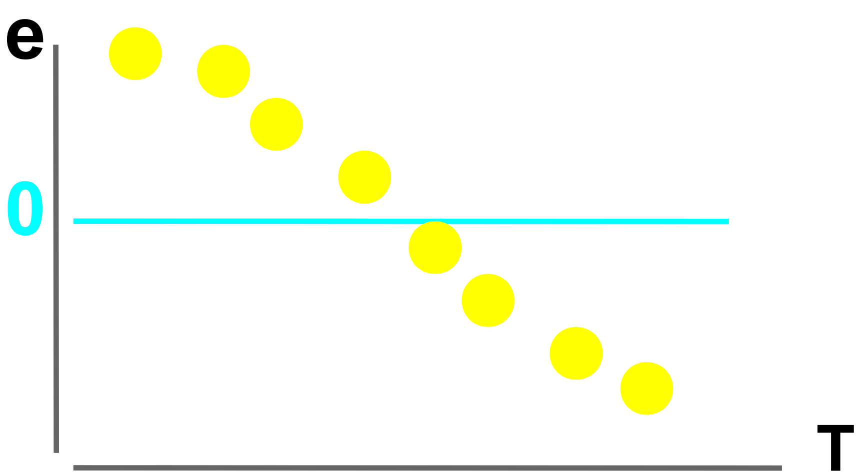
Residual Analysis



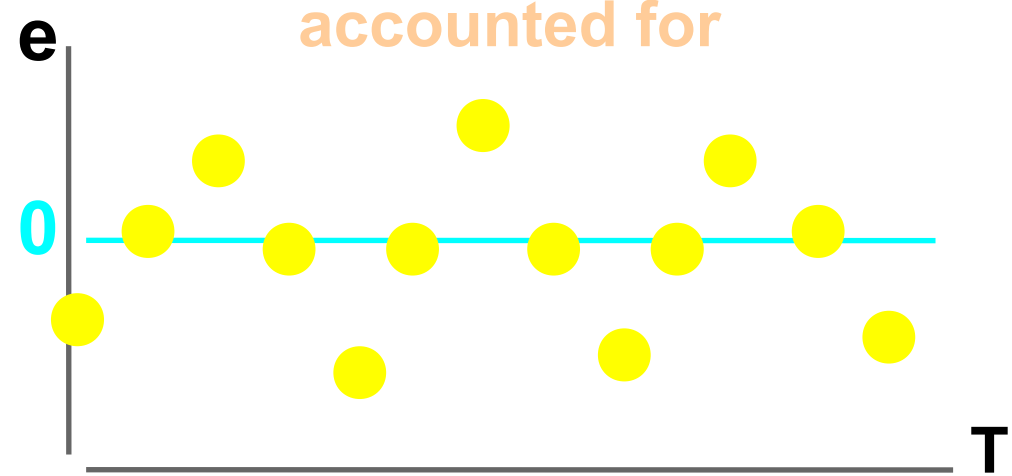
Random errors



Cyclical effects not accounted for



Trend not accounted for



Seasonal effects not accoun
for

Datasets

Several repositories:
UCI – well know, often used
https://archive.ics.uci.edu/ml/datasets.html?type=ts
UCR – another good source
http://www.cs.ucr.edu/~eamonn/time_series_data/
KDNuggest – lost of advertising, but many links
http://www.kdnuggets.com/datasets/index.html

Reading:

Brockwell & Davis: Introduction to Time Series and Forecasting, Springer 2002

Montgomery et al: Introduction to Time Series and Forecasting, Wiley 2008