Lecture 15: Autoregressive integrated Moving Average model (2)

Wengang Mao (Marine Technology)
Department of Mechanics and Maritime Sciences,
Chalmers University of Technology,
Goteborg, Sweden

Outline of ARIMA models

- Basis of autoregressive moving average models
- Difference equations
- Autocorrelation (ACF) and partial autocorrelation (PACF)
- Forecasting
- **Estimation**
- **Integrated Models for Nonstationary Data**
- **Steps to build ARIMA Models**

ARIMA: estimation overview (1)

For an **ARIMA**(*p*, *d*, *q*) written as $\phi(B)X_t = \theta(B)W_t$, where $W_t \sim wn(0, \sigma_W^2)$

- The parameters within the model are:
  - ✓$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p$
  - ✓$\theta(B) = 1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q$
  - ✓Assume that we have *n* observations $x_1, x_2, \ldots, x_n$

The objective of such an estimation is to get the parameters through n observations

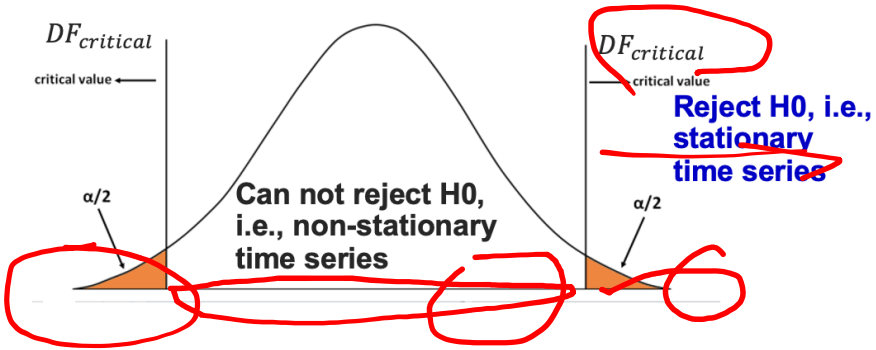
ARIMA: estimation overview (2)

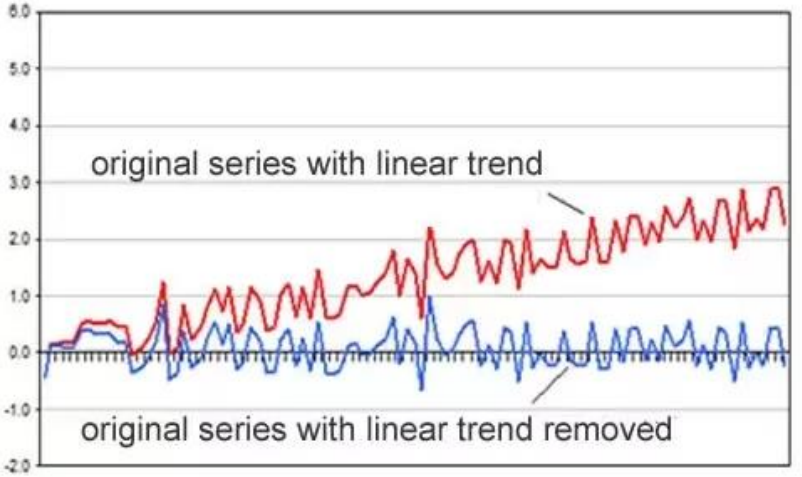
How to get the parameters through n observations?

1) Check the stationary of the (detrend/difference/transformed) time series by **Augmented Dickey–Fuller (ADF)** test. **Find the difference order *d*.**

Hypothesis H0: the time series present certain trend (non-stationary)

$$\mathrm{DF}_\tau = \frac{\hat{\gamma}}{\mathrm{SE}(\hat{\gamma})}$$

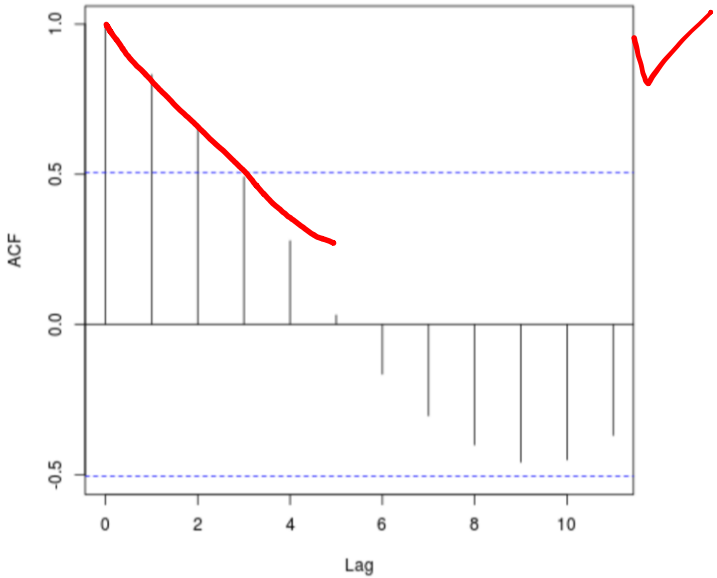


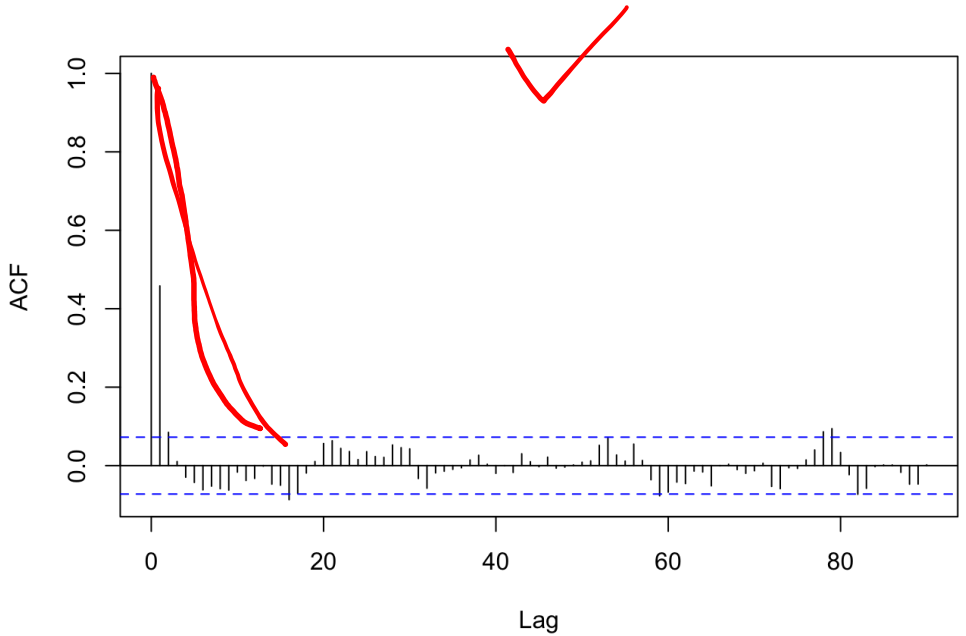


ARMA: estimation overview (2)

How to get the parameters through n observations?

2) We need to determine the orders of the ARMA model, i.e., *p*, *q*. *The value of p represent the time lag h of* ***AR*** *when PACF(h) = 0; while q represents order of MA when ACF(h) = 0.*





ARMA: estimation overview (2)

How to get the parameters through n observations?

3) For the estimation of ϕ_1, ϕ_2,…, ϕ_p, θ_1, θ_2,…, θ_q, there are three methods

- Estimation of AR(p) by MM
- Estimation of AR(p) by Least Square Method
- Estimation of MA(q) by MM
- Estimate of ARMA(p, q), by Maximum likelihood method

4) Most of the ARMA packages have already implemented method for the estimation. We will mainly talk about some basics for the estimation process.

ARMA: estimation of AR(p) by MM

- For a AR(p) model:

$$x_t = \phi_1 x_{t-1} + \cdots + \phi_p x_{t-p} + w_t,$$

- The Yule-Walker Equations:

$$\gamma(h) = \phi_1 \gamma(h-1) + \cdots + \phi_p \gamma(h-p), \quad h = 1, 2, \ldots, p,$$
$$\sigma_w^2 = \gamma(0) - \phi_1 \gamma(1) - \cdots - \phi_p \gamma(p).$$

In matrix notation, the Yule–Walker equations are

$$\Gamma_p \phi = \gamma_p, \quad \sigma_w^2 = \gamma(0) - \phi' \gamma_p,$$

- Then, using the method of moments (covariance is a second moments), we can get

$$\hat{\phi} = \hat{\Gamma}_p^{-1} \hat{\gamma}_p, \quad \hat{\sigma}_w^2 = \hat{\gamma}(0) - \hat{\gamma}_p' \hat{\Gamma}_p^{-1} \hat{\gamma}_p.$$

NB: **Durbin–Levinson algorithm** can be used to calculate $\hat{\phi}$ without above inversed matrix.

ARMA: AR(p) by Least Square Method

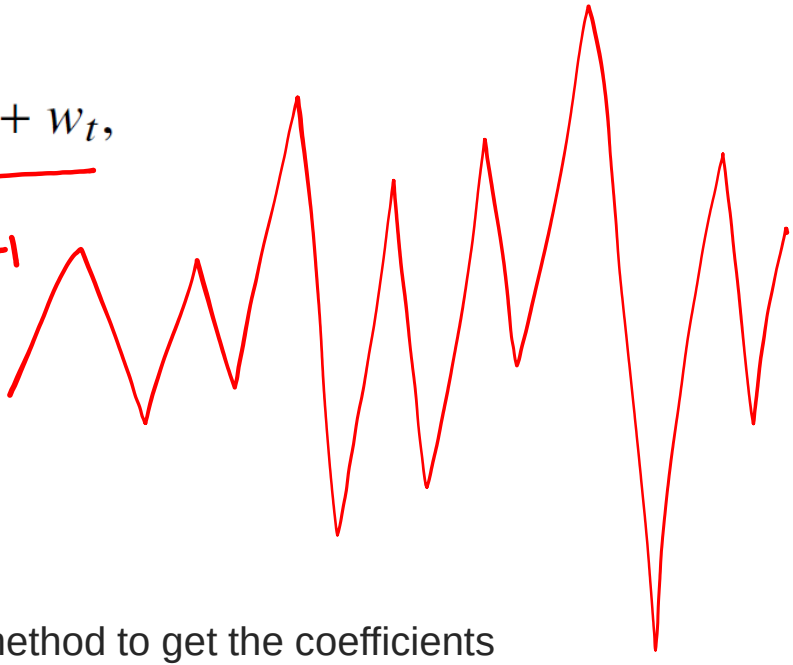
- For a **AR**(*p*) model:

$$x_t = \phi_1 x_{t-1} + \cdots + \phi_p x_{t-p} + w_t,$$

- We have the data observations $\{x_1, x_2, \ldots, x_n\}$
- We can organize the data into the following way:
-

$$\begin{bmatrix} x_n \\ x_{n-1} \\ \vdots \\ x_{p+1} \end{bmatrix} = \boldsymbol{\phi} \begin{bmatrix} x_{n-1}, x_{n-2}, \ldots & x_{n-p} \\ x_{n-2}, x_{n-3}, \ldots & x_{n-p-1} \\ \vdots \quad \vdots & \ddots \quad \vdots \\ x_p, \quad x_{p-1}, \cdots & x_1 \end{bmatrix}$$

- Then, it becomes a standard least square regression method to get the coefficients



ARMA: estimation of MA(q) by MM

For the **MA**(1) model: $X_t = W_t + \theta W_{t-1}$, where $|\theta| < 1$. Rewrite this model as

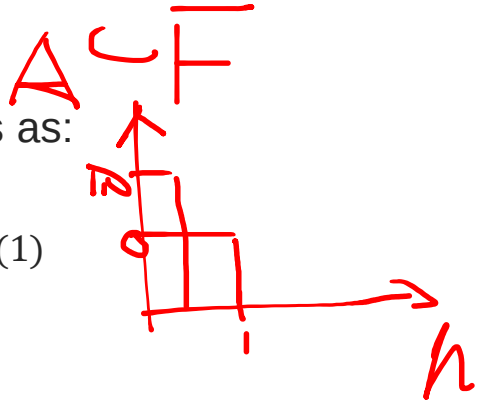
- $X_t = \sum_{j=1}^{\infty} (-\theta)^j X_{t-j} + W_t$
- By estimating the above process, we can get the first two moments as:
  - $\gamma(0) = \sigma_W^2(1+\theta^2)$ and $\gamma(1) = \sigma_W^2\theta$
  - We can estimate the values of $\gamma(0)$ and $\gamma(1)$ from the data as $\hat{\gamma}(0)$ and $\hat{\gamma}(1)$

$$\hat{\rho}(1) = \frac{\hat{\gamma}(1)}{\hat{\gamma}(0)} = \frac{\hat{\theta}}{1+\hat{\theta}^2}$$

  - It is a nonlinear equation, and we will have two solutions. Then the invertible estimate will be picked

$$\hat{\theta} = \frac{1 - \sqrt{1 - 4\hat{\rho}(1)^2}}{2\hat{\rho}(1)}$$

$$\hat{\theta} \sim \text{AN}\left(\theta, \frac{1+\theta^2+4\theta^4+\theta^6+\theta^8}{n(1-\theta^2)^2}\right)$$



ARMA: estimation by ML method

- To understand how the idea of maximum likelihood method can be used to estimate the order parameter of ***ARMA***(p, q), let first look at the ***AR***(1) case.

$$x_t = \mu + \phi(x_{t-1} - \mu) + w_t$$

where $|\phi| < 1$ and $w_t \sim \text{iid N}(0, \sigma_w^2)$. Given data $x_1, x_2, \ldots, x_n$, we seek the likelihood

$$L(\mu, \phi, \sigma_w^2) = f\left(x_1, x_2, \ldots, x_n \mid \mu, \phi, \sigma_w^2\right).$$

In the case of an AR(1), we may write the likelihood as

$$L(\mu, \phi, \sigma_w^2) = f(x_1) f(x_2 \mid x_1) \cdots f(x_n \mid x_{n-1}),$$

$$\frac{\partial L}{\partial \phi_i} = 0$$

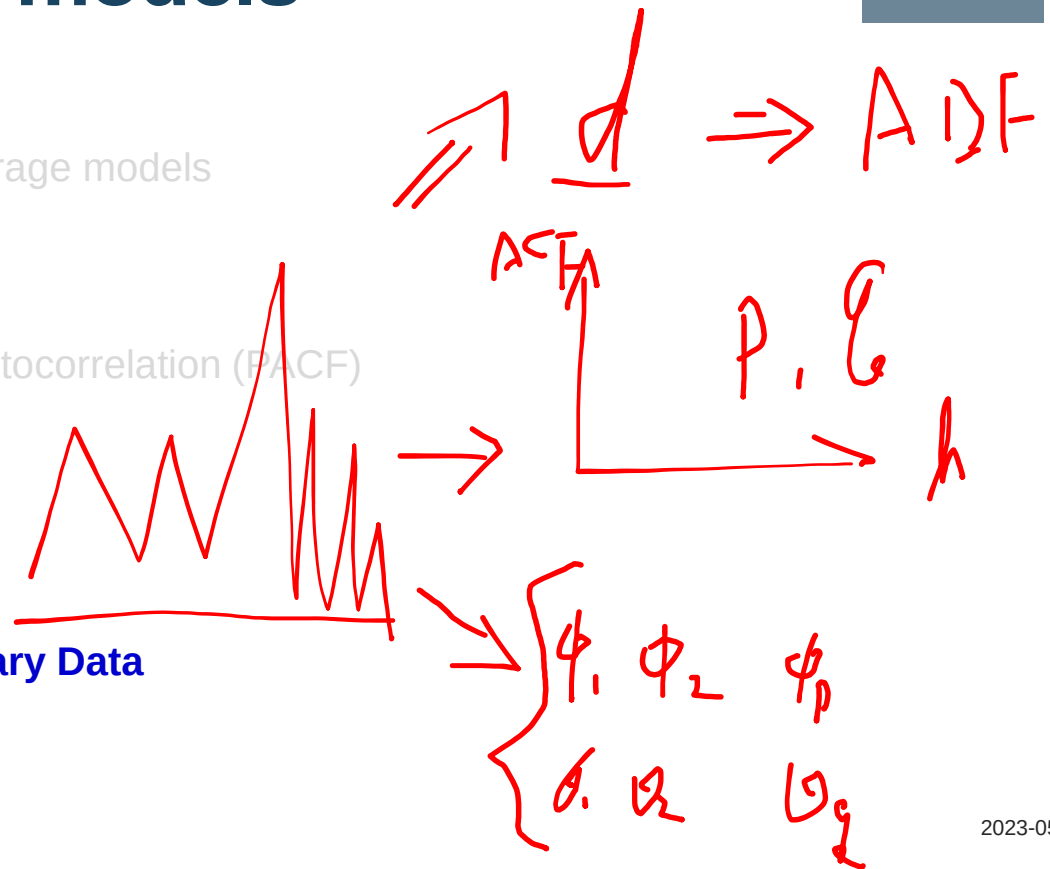
- This is because x_{k+1} only dependent on its previous value of x_k

Because $x_t \mid x_{t-1} \sim \text{N}\left(\mu + \phi(x_{t-1} - \mu), \sigma_w^2\right)$, we have

$$f(x_t \mid x_{t-1}) = f_w[(x_t - \mu) - \phi(x_{t-1} - \mu)],$$

Outline of ARIMA models

- Basis of autoregressive moving average models
- Difference equations
- Autocorrelation (ACF) and partial autocorrelation (PACF)
- Forecasting
- **Estimation**
- **Integrated Models for Nonstationary Data**
- **Building ARIMA Models**



ARMA: nonstationary data by ARIMA

Definition 3.11 *A process x_t is said to be* **ARIMA**(p, d, q) *if*

$$\nabla^d x_t = (1 - B)^d x_t$$

is ARMA(p, q). In general, we will write the model as

$$\phi(B)(1 - B)^d x_t = \theta(B) w_t .$$

If $\mathrm{E}(\nabla^d x_t) = \mu$, *we write the model as*

$$\phi(B)(1 - B)^d x_t = \delta + \theta(B) w_t ,$$

where $\delta = \mu(1 - \phi_1 - \cdots - \phi_p)$.

ARMA: nonstationary data by ARIMA

Definition: ***the integrated ARMA (ARIMA) model is a broadening of the class of ARMA models to include differencing.***

- ✓For a random walk $X_t = X_{t-1} + W_t$, it is not a stationary process, but by differencing the process, we can get $\nabla X_t = W_t$, which becomes a stationary process.
- ✓In addition, for a process with trend, $X_t = \mu_t + Y_t$, where $\mu_t = \beta_0 + \beta_1 t$ represents trend, by differencing the process $\nabla X_t = X_t - X_{t-1} = \beta_1 + \nabla Y_t$, which becomes a stationary process.
- ✓Furthermore, if the trend μ_t is a *k*-th order polynomial in t, $\mu_t = \sum_{j=0}^{k} \beta_j t^j$, by differencing the process *k* times $\nabla^k X_t$ can lead to a stationary process.

Example:

Building ARIMA Models

ARIMA: Building ARIMA Models

There are some basic steps to fit ARIMA models to time series data.

- plotting the data,
- possibly transforming the data,
- identifying the dependence orders of the model,
- parameter estimation,
- diagnostics, and
- model choice.

ARIMA: Building ARIMA model

How to determine the order of differencing:

- A time plot of the data will typically suggest whether any differencing is needed, such as trend, or very irregular process (that might indicate a transform is needed).
- ACF can help in indicating whether differencing is needed. The sample ACF will not decay to zero fast as *h* increases. Thus, a slow decay ACF(h) is an indication that differencing may be needed.

How to diagnose the ARIMA model:

- Inspection of the time plot of the standardized residuals should show no obvious patterns.
- Notice that there may be outliers, with a few values exceeding 3 standard deviations in magnitude.
- The normal Q-Q plot of the residuals shows that the assumption of normality is reasonable, with the exception of the possible outliers.

ARIMA: a simple example

Forecast the annual water usage in Baltimore

- The steps work through are as follows.
  1. Environment (get code libraries)
  2. Problem Description (mathematical model: inputs/outputs)
  3. Test Harness.
  4. Data Analysis.
  5. ARIMA Models.
  6. Model Validation.

ARIMA: a simple example

3. Test Harness

- We must develop a test harness to investigate the data and evaluate candidate models.

1. Defining a Validation Dataset.
2. Developing a Method for Model Evaluation.
   - The dataset is not current. This means that we cannot easily collect updated data to validate the model.
   - Therefore, we will pretend that it is 1953 and withhold the last 10 years of data from analysis and model selection.
   - This final decade of data will be used to validate the final model.

ARIMA: a simple example

4. Data analysis

Summary statistics and data plots to quickly learn more about the structure of the prediction problem.

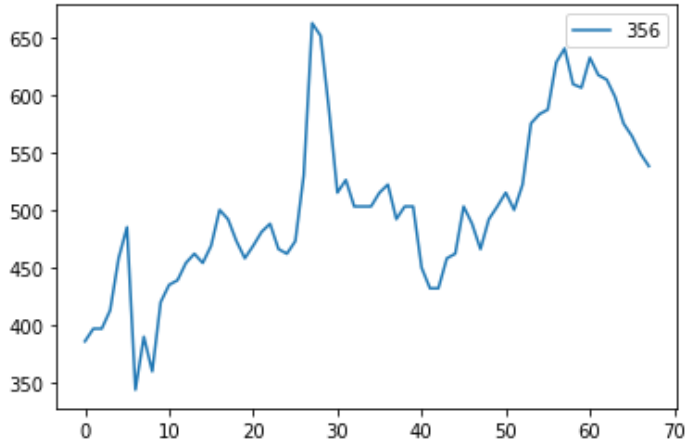
1. Summary Statistics.
2. Line Plot.
3. Density Plots.
4. Box and Whisker Plot.

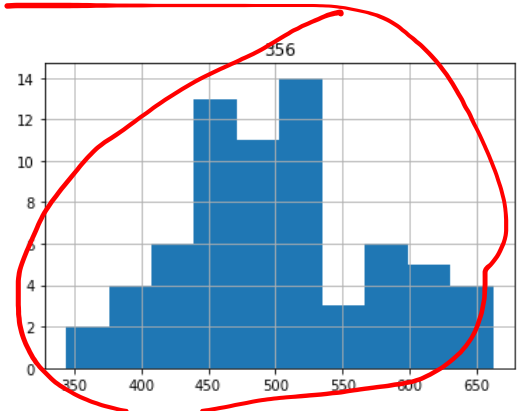
ARIMA: a simple example

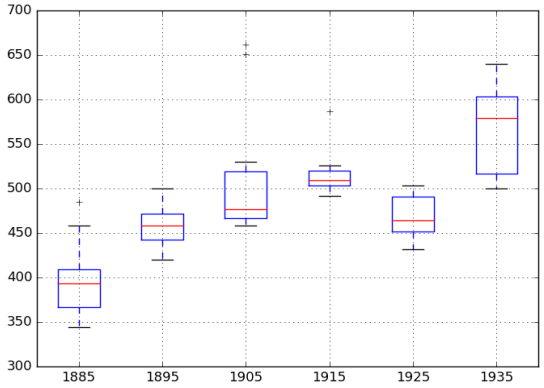
4. Data analysis

1. Summary Statistics.
2. Line Plot.
3. Density Plots.
4. Box and Whisker Plot.

```
count    68.000000
mean    502.602941
std      72.296888
min     344.000000
25%     458.000000
50%     496.000000
75%     540.750000
max     662.000000
```







ARIMA: a simple example

5. ARIMA models

Three steps to make and test the ARIMA model for forecasting and prediction

- Manually Configure the ARIMA.
- Automatically Configure the ARIMA.
- Review Residual Errors.

- The ARIMA(*p*,*d*,*q*) model requires three parameters and is traditionally configured manually.
- Analysis of the time series data assumes that we are working with a stationary time series.
- NB: the time series is likely non-stationary. We can make it stationary by first differencing the series and using a statistical test to confirm that the result is stationary.

ARIMA: a simple example

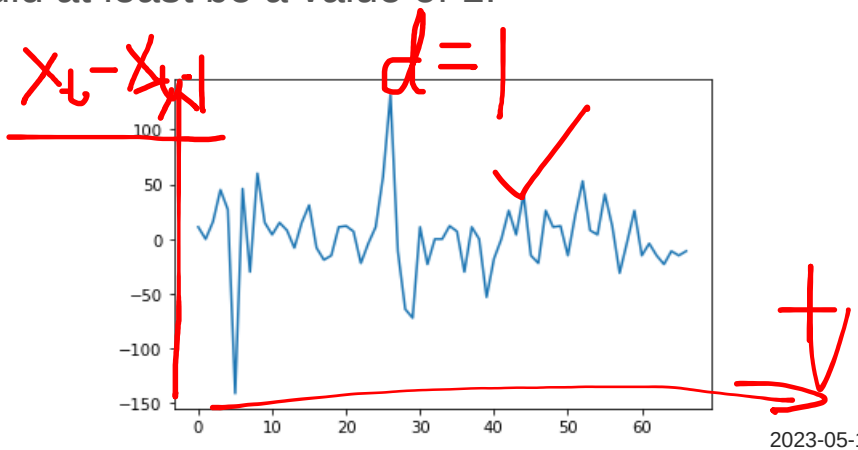
```
ADF Statistic: -2.309209
p-value: 0.169015
Critical Values:
    1%: -3.535
    5%: -2.907
    10%: -2.591
```

5. ARIMA models: Manually Configure the ARIMA:

5.1, choose the differential order *d*

- First check the ADF statistics of original data
- Then, after first difference and check the following items:
- The ADF statistics suggests that at least one level of differencing is required. The d parameter in our ARIMA model should at least be a value of 1.

```
ADF Statistic: -6.055566
p-value: 0.000000
Critical Values:
    1%: -3.535
    5%: -2.907
    10%: -2.591
```



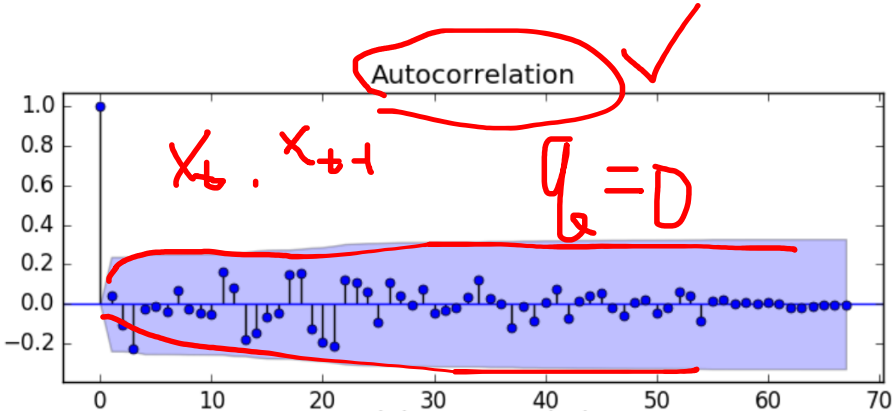
ARIMA: a simple example

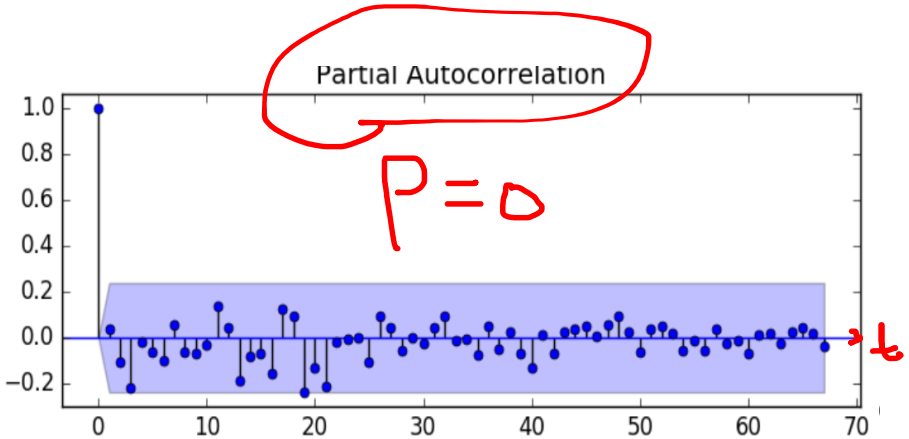
5. ARIMA models: Manually Configure the ARIMA:

5.2, select the lag values for the Autoregression (AR) and Moving Average (MA) parameters, p and q respectively.

- The ACF shows no significant lags.
- The PACF also shows no significant lags.

This quick manual analysis suggests an ARIMA(0,1,0) on the raw data may be a good starting point.



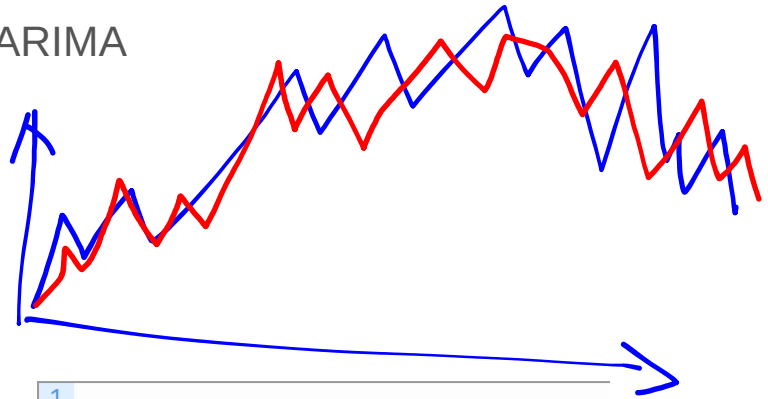


ARIMA: a simple example

5. ARIMA models: Manually Configure the ARIMA

5.3, estimate the model

```python
from pandas import read_csv
from sklearn.metrics import mean_squared_error
from statsmodels.tsa.arima_model import ARIMA
from math import sqrt
# load data
series = read_csv('dataset.csv')
# prepare data
X = series.values
X = X.astype('float32')
train_size = int(len(X) * 0.50)
train, test = X[0:train_size], X[train_size:]
# walk-forward validation
history = [x for x in train]
predictions = list()
for i in range(len(test)):
    # predict
    model = ARIMA(history, order=(0,1,0))
    model_fit = model.fit(disp=0)
    yhat = model_fit.forecast()[0]
    predictions.append(yhat)
    # observation
    obs = test[i]
    history.append(obs)
    print('>Predicted=%.3f, Expected=%3.f' % (yhat, obs))
# report performance
mse = mean_squared_error(test, predictions)
rmse = sqrt(mse)
print('RMSE: %.3f' % rmse)
```



```
...
>Predicted=617.079, Expected=598
>Predicted=601.781, Expected=575
>Predicted=578.369, Expected=564
>Predicted=567.152, Expected=549
>Predicted=551.881, Expected=538
RMSE: 22.311
```

ARIMA: a simple example

5. ARIMA models: Automatically Configure ARIMA

5.4, Grid Search ARIMA Hyperparameters

- *p*: 0 to 4. = 2
- *d*: 0 to 2. = 1
- *q*: 0 to 4. = 0

```
...
ARIMA(4, 1, 0) RMSE=24.802
ARIMA(4, 1, 1) RMSE=25.103
ARIMA(4, 2, 0) RMSE=27.089
ARIMA(4, 2, 1) RMSE=25.932
ARIMA(4, 2, 2) RMSE=25.418
Best ARIMA(2, 1, 0) RMSE=21.733
```

We will select this ARIMA(2, 1, 0) model going forward.

```
import warnings
from pandas import read_csv
from statsmodels.tsa.arima_model import ARIMA
from sklearn.metrics import mean_squared_error
from math import sqrt

# evaluate an ARIMA model for a given order (p,d,q) and return RMSE
def evaluate_arima_model(X, arima_order):
    # prepare training dataset
    X = X.astype('float32')
    train_size = int(len(X) * 0.50)
    train, test = X[0:train_size], X[train_size:]
    history = [x for x in train]
    # make predictions
    predictions = list()
    for t in range(len(test)):
        model = ARIMA(history, order=arima_order)
        # model_fit = model.fit(disp=0)
        model_fit = model.fit(trend='nc', disp=0)
        yhat = model_fit.forecast()[0]
        predictions.append(yhat)
        history.append(test[t])
    # calculate out of sample error
    mse = mean_squared_error(test, predictions)
    rmse = sqrt(mse)
    return rmse

# evaluate combinations of p, d and q values for an ARIMA model
def evaluate_models(dataset, p_values, d_values, q_values):
    dataset = dataset.astype('float32')
    best_score, best_cfg = float("inf"), None
    for p in p_values:
        for d in d_values:
            for q in q_values:
                order = (p,d,q)
                try:
                    mse = evaluate_arima_model(dataset, order)
                    if mse < best_score:
                        best_score, best_cfg = mse, order
                    print('ARIMA%s RMSE=%.3f' % (order,mse))
                except:
                    continue
    print('Best ARIMA%s RMSE=%.3f' % (best_cfg, best_score))

# load dataset
series = read_csv('dataset.csv')
# evaluate parameters
p_values = range(0, 5)
d_values = range(0, 3)
q_values = range(0, 5)
warnings.filterwarnings("ignore")
evaluate_models(series.values, p_values, d_values, q_values)
```

ARIMA: a simple example

5. ARIMA models: Check the residual errors

- Ideally, the distribution of residual errors should be a Gaussian with a zero mean.

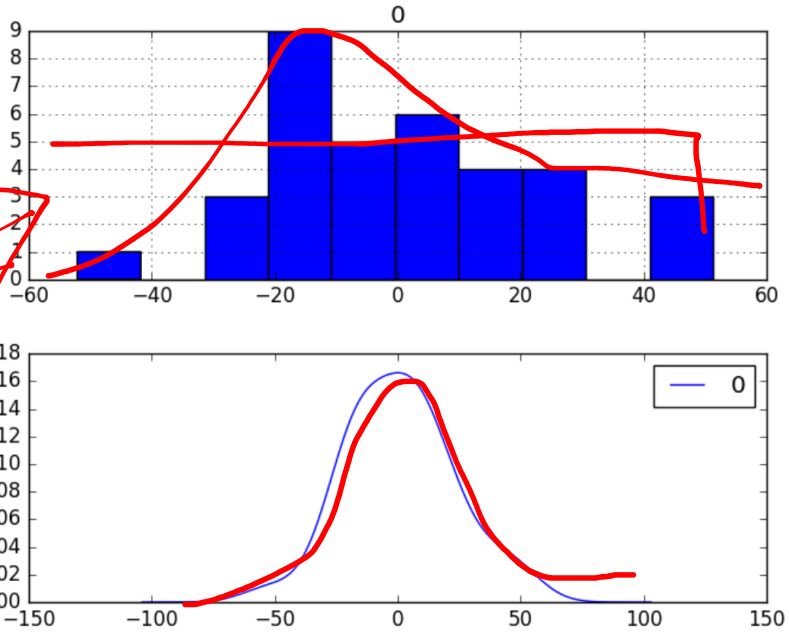
(1) Basic statistics of the residuals

1	count	35.000000
2	mean	1.081624
3	std	22.022566
4	min	-52.103811
5	25%	-16.202283
6	50%	-0.459801
7	75%	12.085091
8	max	51.284336

(2) Check distribution of residuals

(3) A longer right tail ➔

perhaps a power transform is worth exploring

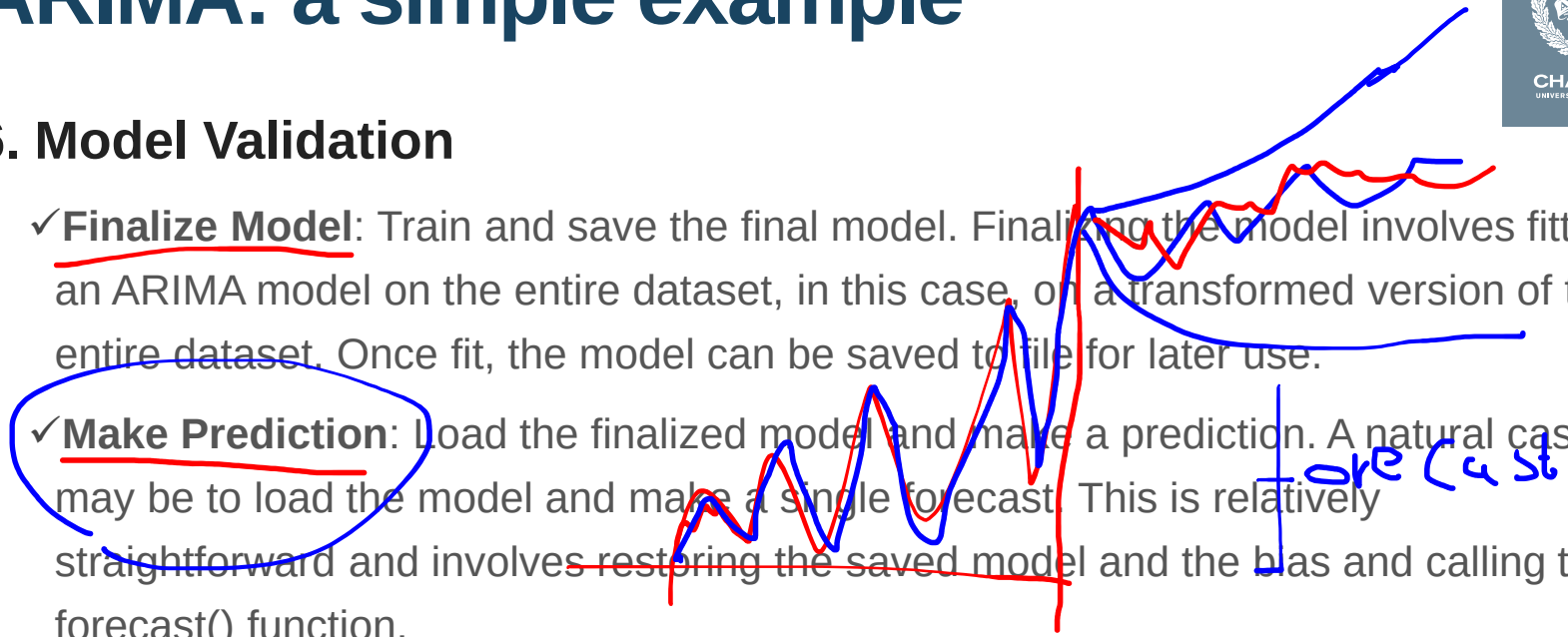


NB: Bias-correct predictions by adding the mean residual error of 1.081624 to each forecast made.

ARIMA: a simple example

6. Model Validation

- ✓**Finalize Model**: Train and save the final model. Finalizing the model involves fitting an ARIMA model on the entire dataset, in this case, on a transformed version of the entire dataset. Once fit, the model can be saved to file for later use.
- ✓**Make Prediction**: Load the finalized model and make a prediction. A natural case may be to load the model and make a single forecast. This is relatively straightforward and involves restoring the saved model and the bias and calling the forecast() function.
- ✓**Validate Model**: Load and validate the final model. In the test harness section, we saved the final 10 years of the original dataset in a separate file to validate the final model.



ARIMA: a simple example

6.1 Model Validation -- Finalize Model

- Develop code to load and fit data with the ARIMA ➔ creates two local files:
  - *model.pkl* This is the ARIMAResult object from the call to *ARIMA.fit()*. This includes the coefficients and all other internal data returned when fitting the model.
  - *model_bias.npy* This is the bias value stored as a one-row, one-column NumPy array.

```
# monkey patch around bug in ARIMA class
def __getnewargs__(self):
    return ((self.endog),(self.k_lags, self.k_diff, self.k_ma))

ARIMA.__getnewargs__ = __getnewargs__

# load data
series = read_csv('dataset.csv')
# prepare data
X = series.values
X = X.astype('float32')
# fit model
model = ARIMA(X, order=(2,1,0))
model_fit = model.fit(trend='nc', disp=0)
# bias constant, could be calculated from in-sample mean residual
bias = 1.081624
# save model
model_fit.save('model.pkl')
numpy.save('model_bias.npy', [bias])
```

ARIMA: a simple example

6.2 Model Validation -- Make Prediction

This is relatively straightforward and involves restoring the saved model and the bias and calling the forecast() function.

```
1 from statsmodels.tsa.arima_model import ARIMAResults
2 import numpy
3 model_fit = ARIMAResults.load('model.pkl')
4 bias = numpy.load('model_bias.npy')
5 yhat = bias + float(model_fit.forecast()[0])
6 print('Predicted: %.3f' % yhat)
```

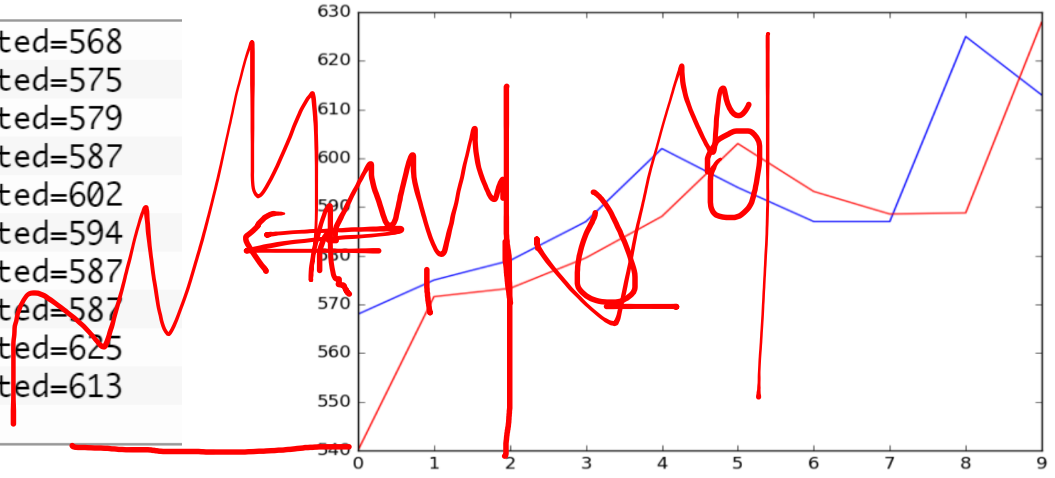
Running the example prints the prediction of about 540.

ARIMA: a simple example

6.3 Model Validation -- Validate Model with two ways to proceed

- Use the model to forecast next 10 years: forecast degrades quickly
- Use the model in a rolling-forecast manner, updating the transform and model for each step (preferred method and achieve best performance).

```
>Predicted=540.013, Expected=568
>Predicted=571.589, Expected=575
>Predicted=573.289, Expected=579
>Predicted=579.561, Expected=587
>Predicted=588.063, Expected=602
>Predicted=603.022, Expected=594
>Predicted=593.178, Expected=587
>Predicted=588.558, Expected=587
>Predicted=588.797, Expected=625
>Predicted=627.941, Expected=613
RMSE: 16.532
```



CHALMERS

UNIVERSITY OF TECHNOLOGY