Forecasting

BITS Pilani
Pilani Campus

Akanksha Bharadwaj
Asst. Professor, CS/IS

innovate achieve lead

BITS Pilani
Pilani Campus

SS ZG536, ADV STAT TECHNIQUES FOR ANALYTICS

Contact Session 11

Python example: Minimum Daily Temperatures Dataset

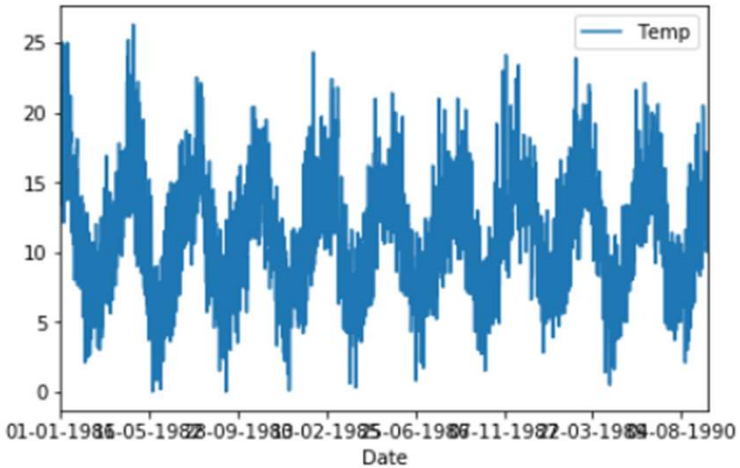
- The dataset describes the minimum daily temperatures over 10 years (1981-1990) in the city Melbourne, Australia.

- The units are in degrees Celsius and there are 3,650 observations. The source of the data is credited as the Australian Bureau of Meteorology.

Load the dataset as a Pandas Series.

- Running the example prints the first 5 rows from the loaded dataset.
- A line plot of the dataset is then created.

```
            Temp
Date
01-01-1981  20.7
02-01-1981  17.9
03-01-1981  18.8
04-01-1981  14.6
05-01-1981  15.8
```




```
from pandas import read_csv
from matplotlib import pyplot
series = read_csv('C:\Akanksha\Data\daily-minimum-
    temperatures.csv', header=0, index_col=0)
print(series.head())
series.plot()
pyplot.show()
```

Quick Check for Autocorrelation

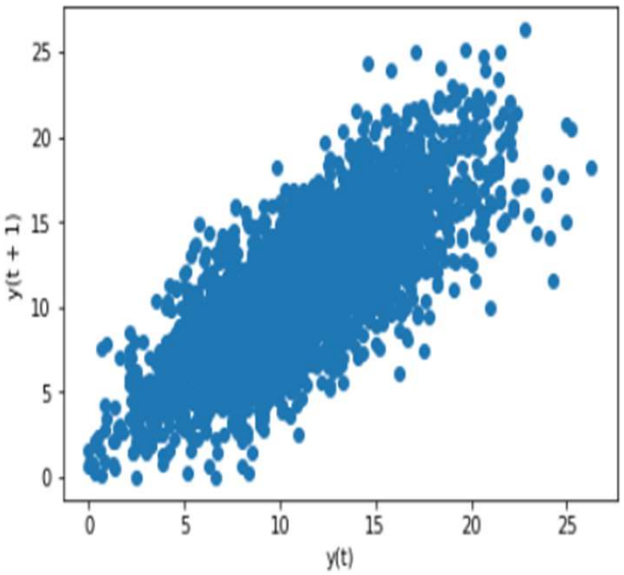
- We can plot the observation at the previous time step (t) with the observation at the next time step (t+1) as a scatter plot.

- This could be done manually by first creating a lag version of the time series dataset and using a built-in scatter plot function in the Pandas library.

- Easy method: Pandas provides a built-in plot to do exactly this, called the lag_plot() function.

- Running the example plots the temperature data (t) on the x-axis against the temperature on the previous day (t+1) on the y-axis.

```
from pandas import read_csv
from matplotlib import pyplot
from pandas.plotting import lag_plot
series = read_csv('C:\Akanksha\Data\daily-minimum-
    temperatures.csv', header=0, index_col=0)
lag_plot(series)
pyplot.show()
```



- We can see a large ball of observations along a diagonal line of the plot. It clearly shows a relationship or **some correlation**.

Pearson correlation coefficient

- We can use a statistical test like the Pearson correlation coefficient.

- This produces a number to summarize how correlated two variables are between -1 (negatively correlated) and +1 (positively correlated) with small values close to zero indicating low correlation and high values above 0.5 or below -0.5 showing high correlation.

- Correlation can be calculated easily using the corr() function on the DataFrame of the lagged dataset.

- It shows a strong positive correlation (0.77) between the observation and the lag=1 value.
- This is good but tedious if we want to check a large number of lag variables in our time series.

```
from pandas import read_csv
from pandas import DataFrame
from pandas import concat
from matplotlib import pyplot
series = read_csv('C:\Akanksha\Data\daily-minimum-temperatures.csv', header=0, index_col=0)
values = DataFrame(series.values)
dataframe = concat([values.shift(1), values], axis=1)
dataframe.columns = ['t-1', 't+1']
result = dataframe.corr()
print(result)
```

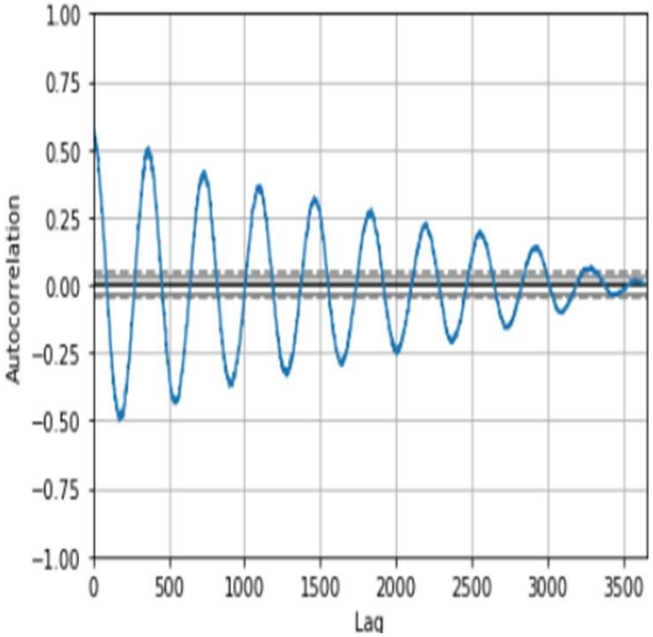
```
          t-1     t+1
t-1  1.00000  0.77487
t+1  0.77487  1.00000
```

Autocorrelation Plots

- We can plot the correlation coefficient for each lag variable.
- We could manually calculate the correlation values for each lag variable and plot the result.
- But, Pandas provides a built-in plot called the autocorrelation_plot() function.
- The plot provides the lag number along the x-axis and the correlation coefficient value between -1 and 1 on the y-axis.
- The plot also includes solid and dashed lines that indicate the 95% and 99% confidence interval for the correlation values.
- Correlation values above these lines are more significant than those below the line, providing a threshold or cutoff for selecting more relevant lag values.

- Figure shows the swing in positive and negative correlation as the temperature values change cross summer and winter seasons each previous year.

```
from pandas import read_csv
from matplotlib import pyplot
from pandas.plotting import autocorrelation_plot
series = read_csv('C:\Akanksha\Data\daily-minimum-
    temperatures.csv', header=0, index_col=0)
autocorrelation_plot(series)
pyplot.show()
```



Autoregression Model

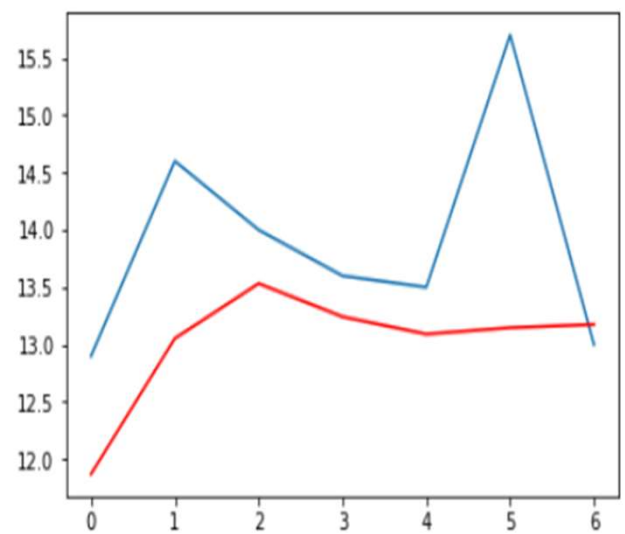
- An autoregression model is a linear regression model that uses lagged variables as input variables.
- We could calculate the linear regression model manually using the LinearRegession class in scikit-learn and manually specify the lag input variables to use.
- Alternately, the statsmodels library provides an autoregression model that automatically selects an appropriate lag value using statistical tests and trains a linear regression model. It is provided in the AR class.
- We can use this model by first creating the model AR() and then calling fit() to train it on our dataset. This returns an AR Result object.
- Once fit, we can use the model to make a prediction by calling the predict() function for a number of observations in the future.

- Running the example first prints the chosen optimal lag and the list of coefficients in the trained linear regression model.
- We can see that a 29-lag model was chosen and trained. This is interesting given how close this lag is to the average number of days in a month.
- The 7 day forecast is then printed and the mean squared error of the forecast is summarized

```
Lag: 29
Coefficients: [ 5.57543506e-01  5.88595221e-01 -9.08257090e-02  4.82615092e-02
  4.00650265e-02  3.93020055e-02  2.59463738e-02  4.46675960e-02
  1.27681498e-02  3.74362239e-02 -8.11700276e-04  4.79081949e-03
  1.84731397e-02  2.68908418e-02  5.75906178e-04  2.48096415e-02
  7.40316579e-03  9.91622149e-03  3.41599123e-02 -9.11961877e-03
  2.42127561e-02  1.87870751e-02  1.21841870e-02 -1.85534575e-02
 -1.77162867e-03  1.67319894e-02  1.97615668e-02  9.83245087e-03
  6.22710723e-03 -1.37732255e-03]
predicted=11.871275, expected=12.900000
predicted=13.053794, expected=14.600000
predicted=13.532591, expected=14.000000
predicted=13.243126, expected=13.600000
predicted=13.091438, expected=13.500000
predicted=13.146989, expected=15.700000
predicted=13.176153, expected=13.000000
Test MSE: 1.502
```

Predictions from fixed AR model

- A plot of the expected (blue) vs the predicted values (red) is made.
- The forecast does look pretty good (about 1 degree Celsius out each day), with big deviation on day 5.
- The statsmodels API does not make it easy to update the model as new observations become available.
- One way would be to re-train the AR model each day as new observations become available, and that may be a valid approach, if not computationally expensive.



Moving Averages model

- The moving average (MA) method models the next step in the sequence as a linear function of the residual errors from a mean process at prior time steps.
- A moving average model is different from calculating the moving average of the time series.
- The notation for the model involves specifying the order of the model q as a parameter to the MA function, e.g. MA(q). For example, MA(1) is a first-order moving average model.
- The method is suitable for univariate time series without trend and seasonal components.

Moving Average (MA) Models

The notation MA(q) refers to the moving average model of order q:

$$X_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q}$$

- where μ is the mean of the series, the θ_1, ..., θ_q are the parameters of the model and the ε_t, ε_{t-1},..., ε_{t-q} are white noise error terms. The value of q is called the order of the MA model.
- Thus, a moving-average model is conceptually a linear regression of the current value of the series against current and previous (observed) white noise error terms or random shocks.
- The random shocks at each point are assumed to be mutually independent and to come from the same distribution, typically a normal distribution, with location at zero and constant scale.
- The distinction in this model is that these random shocks are propagated to future values of the time series. Fitting the MA estimates is more complicated than with AR models because the error terms are not observable.

Moving Average Models

- The MA model should not be confused with Moving Average Smoothing as they are not the same thing.
- A **moving average** term in a time series model is a past error (multiplied by a coefficient).

Let r.v. $w_t \sim iid(0,\sigma^2)$, meaning that the w_t are identically, independently distributed, having mean 0 and the same variance.

The **1st order moving average** model, denoted by MA(1) is:
$x_t=\mu+w_t+\theta_1 w_{t-1}$
The **2nd order moving average** model, denoted by MA(2) is:
$x_t=\mu+w_t+\theta_1 w_{t-1}+\theta_2 w_{t-2}$
The **qth order moving average** model, denoted by MA(q) is:
$x_t=\mu+w_t+\theta_1 w_{t-1}+\theta_2 w_{t-2}+\cdots+\theta_q w_{t-q}$

Theoretical Properties of a Time Series with an MA(1) Model

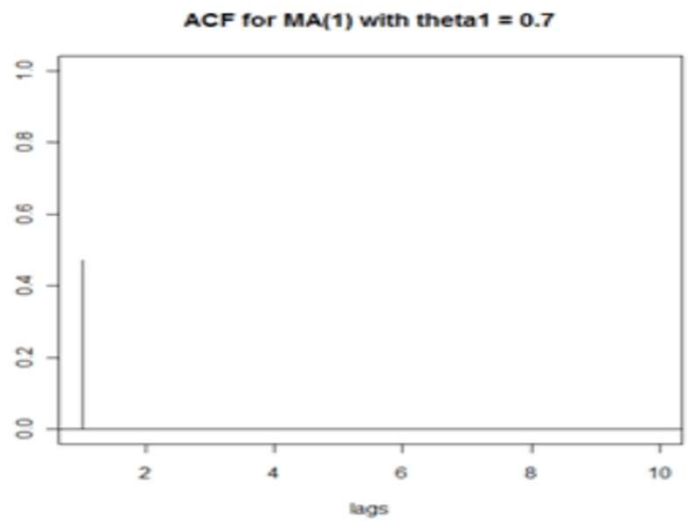
- Mean is $E(x_t) = \mu$
- Variance is $Var(x_t) = \sigma_w^2(1 + \theta_1^2)$
- Autocorrelation function (ACF) is:

$$\rho_1 = \frac{\theta_1}{1 + \theta_1^2}, \text{ and } \rho_h = 0 \text{ for } h \geq 2$$

Example

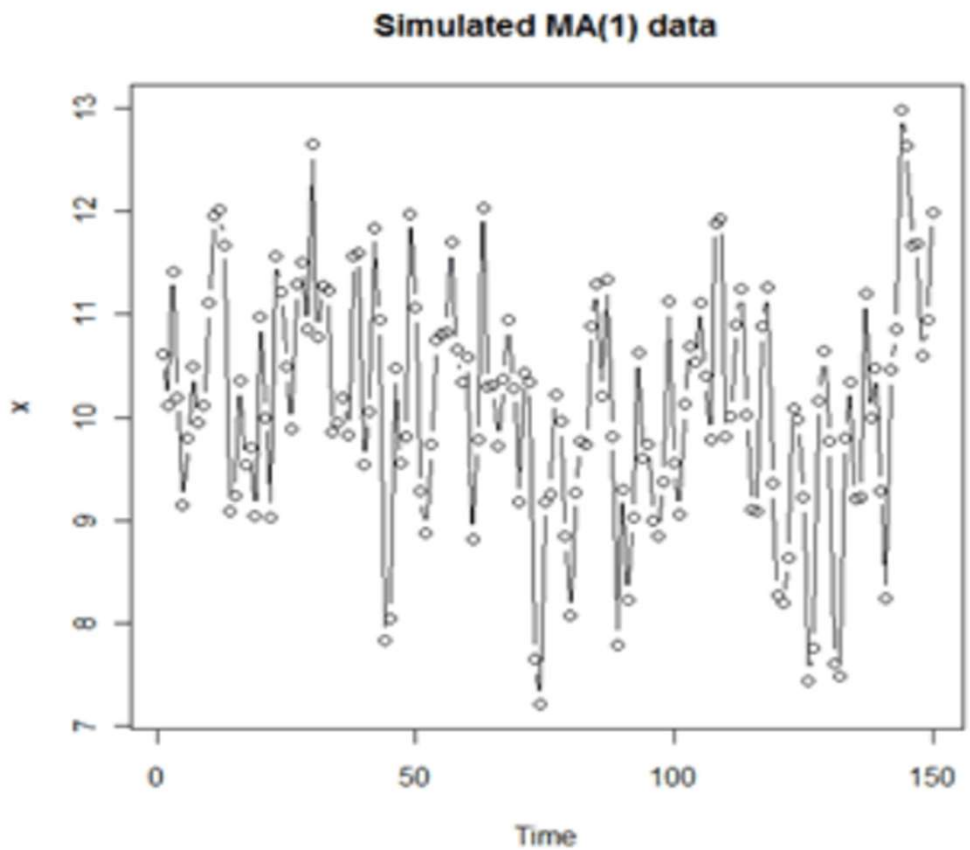
Suppose that an MA(1) model is $x_t = 10 + w_t + .7w_{t-1}$, where $w_t \overset{iid}{\sim} N(0,1)$. Thus the coefficient $\theta_1 = 0.7$. The theoretical ACF is given by:

$$\rho_1 = \frac{0.7}{1+0.7^2} = 0.4698, \text{ and } \rho_h = 0 \text{ for all lags } h \geq 2$$



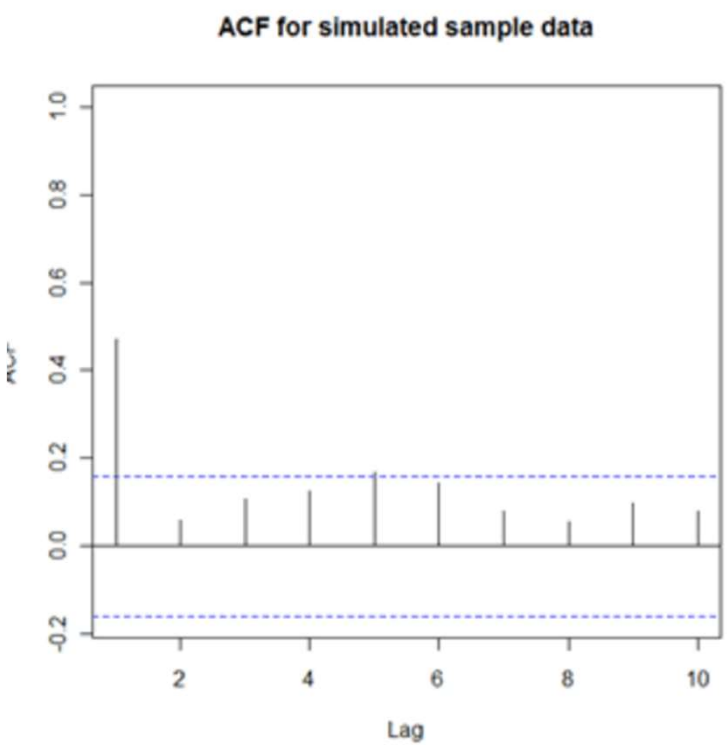
- That the *only nonzero value in the theoretical ACF is for lag 1*. All other autocorrelations are 0. Thus a sample ACF with a significant autocorrelation only at lag 1 is an indicator of a possible MA(1) model.

Using R, we simulated n = 100 sample values using the model $x_t = 10 + w_t + .7w_{t-1}$ where $w_t \overset{iid}{\sim} N(0, 1)$. For this simulation, a time series plot of the sample data follows. We can't tell much from this plot.



- The sample ACF does not match the theoretical pattern of the underlying MA(1), which is that all autocorrelations for lags past 1 will be 0.

- A different sample would have a slightly different sample ACF shown below, but would likely have the same broad features.



Theoretical Properties of a Time Series with an MA(2) Model

Theoretical Properties of a Time Series with an MA(2) Model

For the MA(2) model, theoretical properties are the following:

- Mean is $E(x_t) = \mu$
- Variance is $Var(x_t) = \sigma_w^2(1 + \theta_1^2 + \theta_2^2)$
- Autocorrelation function (ACF) is:

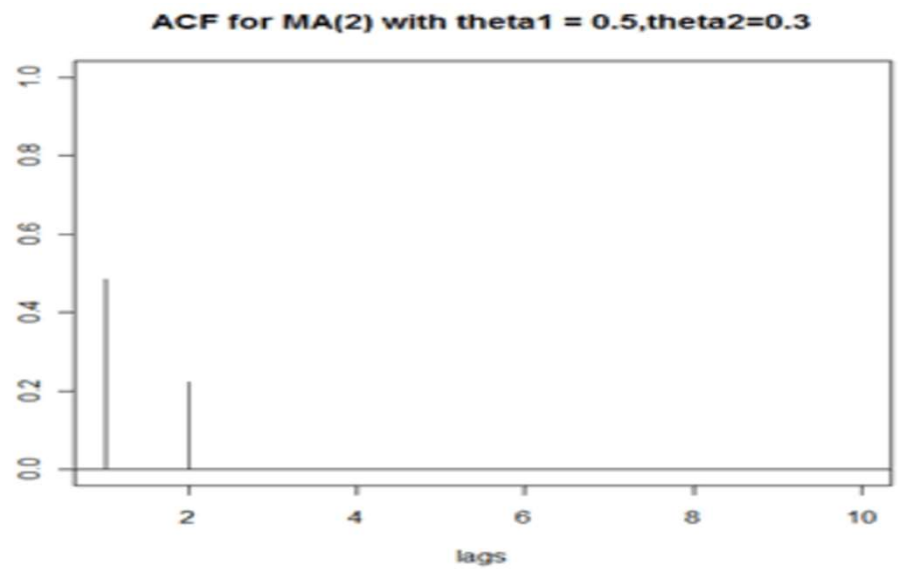
$$\rho_1 = \frac{\theta_1 + \theta_1\theta_2}{1 + \theta_1^2 + \theta_2^2}, \ \rho_2 = \frac{\theta_2}{1 + \theta_1^2 + \theta_2^2}, \text{ and } \rho_h = 0 \text{ for } h \geq 3$$

Example

Consider the MA(2) model $x_t = 10 + w_t + .5w_{t-1} + .3w_{t-2}$, where $w_t \overset{iid}{\sim} N(0, 1)$. The coefficients are $\theta_1 = 0.5$ and $\theta_2 = 0.3$. Because this is an MA(2), the theoretical ACF will have nonzero values only at lags 1 and 2.

Values of the two nonzero autocorrelations are:

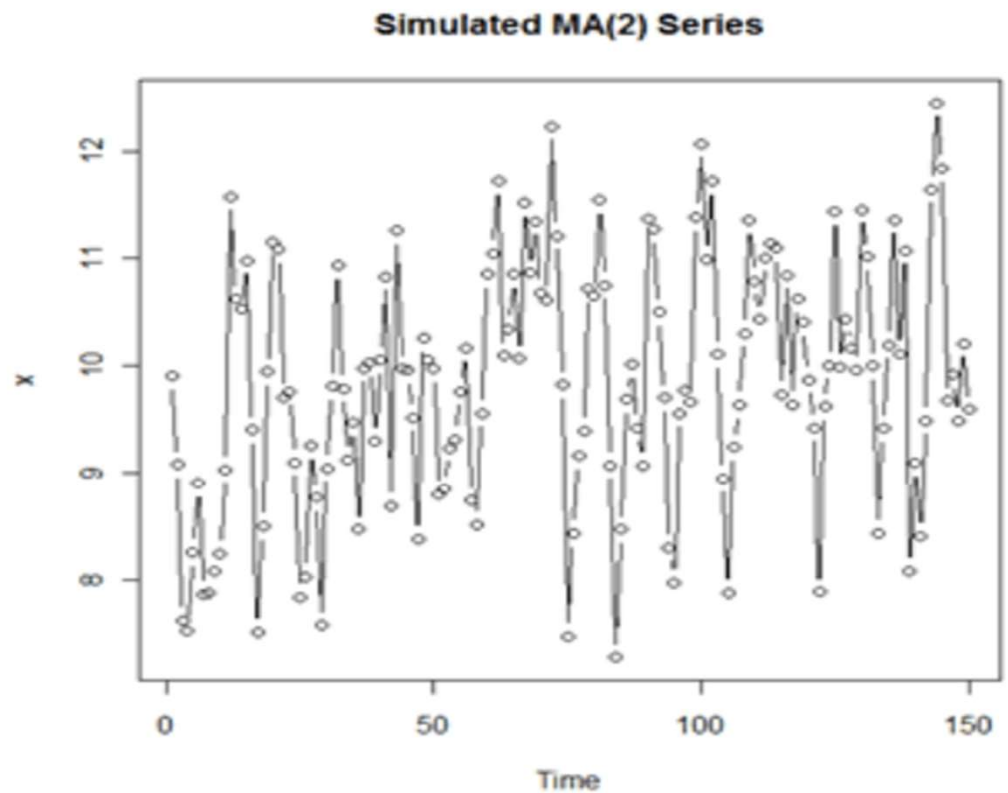
$$\rho_1 = \frac{0.5 + 0.5 \times 0.3}{1 + 0.5^2 + 0.3^2} = 0.4851 \text{ and } \rho_2 = \frac{0.3}{1 + 0.5^2 + 0.3^2} = 0.2239$$



The only nonzero values in the theoretical ACF are for lags 1 and 2. Autocorrelations for higher lags are 0. So, a sample ACF with significant autocorrelations at lags 1 and 2, but non-significant autocorrelations for higher lags indicates a possible MA(2) model.

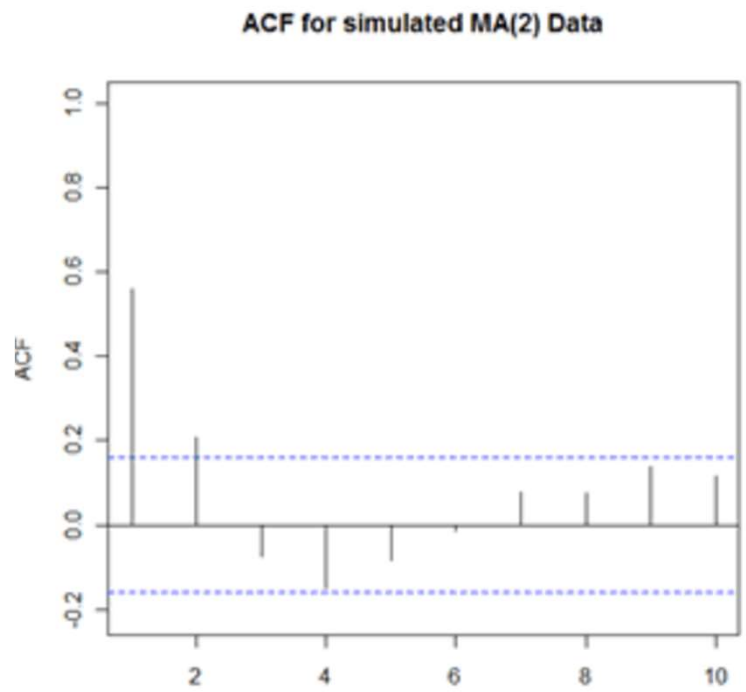
We simulated n = 150 sample values for the model

. $x_t = 10 + w_t + .5w_{t-1} + .3w_{t-2}$, where $w_t \overset{iid}{\sim} N(0, 1)$. The time series plot of the data follows. A
with the time series plot for the MA(1) sample data, you can't tell much from it.



- The pattern is typical for situations where an MA(2) model may be useful. There are two statistically significant “spikes” at lags 1 and 2 followed by non-significant values for other lags.

- Note that due to sampling error, the sample ACF did not match the theoretical pattern exactly.



ACF for General MA(q) Models

A property of MA(q) models in general is that there are nonzero autocorrelations for the first q lags and autocorrelations = 0 for all lags > q.

Non-uniqueness of connection between values of θ_1 and ρ_1 in MA(1) Model.

In the MA(1) model, for any value of θ_1, the reciprocal $1/\theta_1$ gives the same value for:

$$\rho_1 = \frac{\theta_1}{1+\theta_1^2}$$

As an example, use +0.5 for θ_1, and then use 1/(0.5) = 2 for θ_1. You'll get $\rho_1 = 0.4$ in both instances.

To satisfy a theoretical restriction called **invertibility**, we restrict MA(1) models to have values with absolute value less than 1. In the example just given, $\theta_1 = 0.5$ will be an allowable parameter value, whereas $\theta_1 = 1/0.5 = 2$ will not.

Python Code

- We can use the ARMA class to create an MA model and setting a zeroth-order AR model. We must specify the order of the MA model in the order argument.

```
# MA example
from statsmodels.tsa.arima_model import ARMA
from random import random
# contrived dataset
data = [x + random() for x in range(1, 100)]
# fit model
model = ARMA(data, order=(0, 1))
model_fit = model.fit(disp=False)
# make prediction
yhat = model_fit.predict(len(data), len(data))
print(yhat)
```

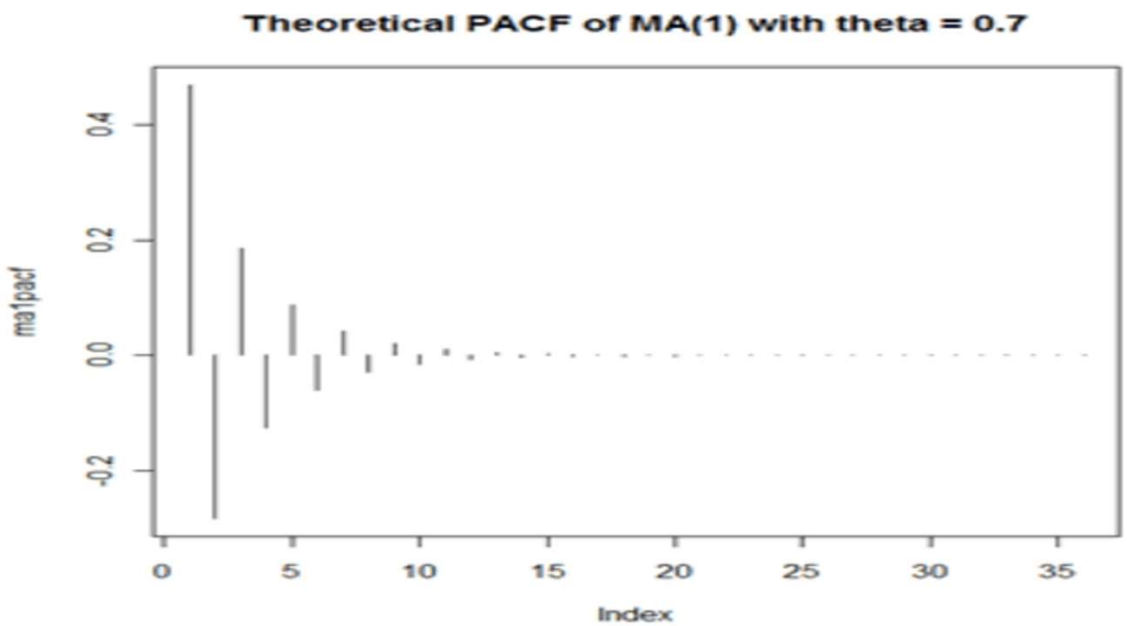
```
In [1]: # MA example
        from statsmodels.tsa.arima_model import ARMA
        from random import random
        # contrived dataset
        data = [x + random() for x in range(1, 100)]
        # fit model
        model = ARMA(data, order=(0, 1))
        model_fit = model.fit(disp=False)
        # make prediction
        yhat = model_fit.predict(len(data), len(data))
        print(yhat)

        [75.00274978]
```

Important conclusion for MA(q) Process

- For an MA model, the theoretical PACF does not shut off, but instead tapers toward 0 in some manner.

- A clearer pattern for an MA model is in the ACF. The ACF will have non-zero autocorrelations only at lags involved in the model.



Box-Jenkins models

ARIMA models, also called Box-Jenkins models, are models that may possibly include autoregressive terms, moving average terms, and differencing operations. Various abbreviations are used:

- When a model only involves autoregressive terms it may be referred to as an AR model. When a model only involves moving average terms, it may be referred to as an MA model.

- When no differencing is involved, the abbreviation ARMA may be used.

Autoregressive Moving Average (ARMA)

- The Autoregressive Moving Average (ARMA) method models the next step in the sequence as a linear function of the observations and residual errors at prior time steps.

- It combines both Autoregression (AR) and Moving Average (MA) models.

- The notation for the model involves specifying the order for the AR(p) and MA(q) models as parameters to an ARMA function, e.g. ARMA(p, q). An ARIMA model can be used to develop AR or MA models.

- The method is suitable for univariate time series without trend and seasonal components.

ARMA

ARMA model is simply the merger between AR(p) and MA(q) models:

- AR(p) models try to explain the momentum and mean reversion effects often observed in trading markets (market participant effects).

- MA(q) models try to capture the shock effects observed in the white noise terms. These shock effects could be thought of as unexpected events affecting the observation process e.g. Surprise earnings, wars, attacks, etc.

ARMA Models

- Autoregressive and Moving Average Models can be combined to form ARMA models

- ARMA(p,q) time series model has the following form:

$$X_t = w_t + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{j=1}^{q} \theta_j w_{t-j}$$

where ϕ_p , $\theta_q \neq 0$ and w_t ~white noise

ARMA(1,1)

ARMA(1,1) model is:

$x(t) = a*x(t-1) + b*e(t-1) + e(t)$

$e(t)$ is white noise with $E[e(t)] = 0$

Identifying the Model

- Three items should be considered to determine a first guess at an ARIMA model: a time series plot of the data, the ACF and the PACF.

- Time series plot will show if there is a trend, if so, detrend it by taking first difference of the consecutive time series values, transforming the series, smoothing or by subtracting a regression estimate of the trend

- For an ARMA model, ACF and PACF gives only a guess to select the p , q values.

	$AR(p)$	$MA(q)$	$ARMA(p,q)$
ACF	Tails off	Cuts off after lag q	Tails off
PACF	Cuts off after lag p	Tails off	Tails off

Estimating & diagnosing a possible model

- Estimate the model using software such as R, SAS,Minitab.etc.

- Once the model has been estimated, do the following for diagnosis:
  - Look at the significance of the coefficients. Compare p values to the significance level or calculate a t-statistic, t = estimated coeff./std. error of coeff & compare with $t_{\alpha,df}$. When n is large, coeff./std. error of coeff can be compared to 1.96.

  - Look at ACF of the residuals. For a good model, ACF of the residual series should be non-significant.

  - Look at time series plot of the residuals for randomness

  - Look at Box-Pierce tests for possible residual autocorrelation at various lags.

- If something looks wrong, revise your guess of the model and do the steps again.

What if more than one model looks OK !!

- Possibly choose the model with the fewest parameter
- Examine standard error of forecast values such as MSE, MAPE and pick the model with the lowest standard errors
- Compare models using criterion such as AIC and BIC.

Example: choosing from competing models

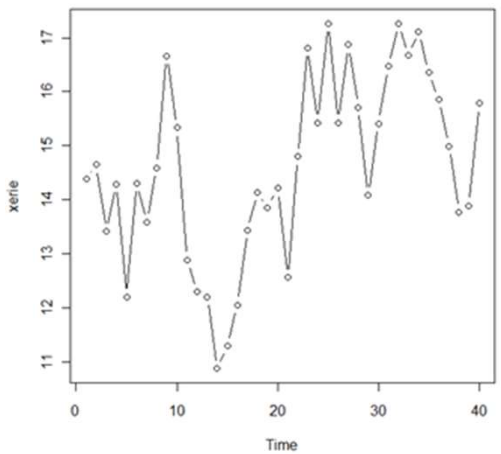
- Let an AR process gives the following data

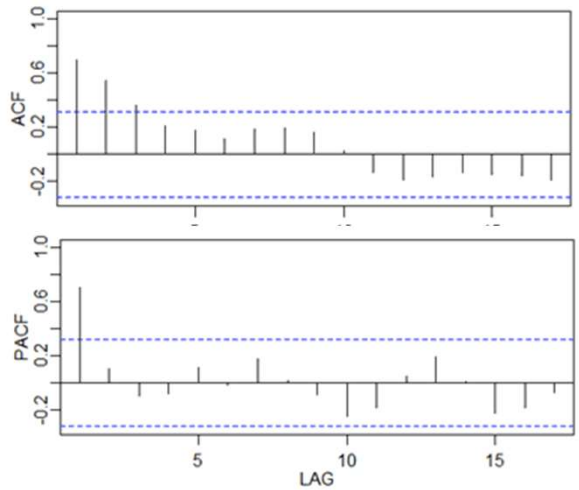
Order of AR process, p	1	2	3	4	5
coeff	0.5970	0.7111 -0.1912	0.7136 -0.2003 0.0128	0.7137 -0.2016 0.0176 -0.0066	0.7141 -0.2027 0.0302 -0.0515 0.629
AIC	5751.32	5679.274	5680.945	5682.857	5676.917
SSE	2072.832	1997.007	1996.678	1996.590	1988.660

- Which AR process order will you choose?
- AR(1) is the best as it is parsimonious given not much difference in AIC.

Example

- N=40 consecutive annual measurements of the lake Erie in a particular month are given
- Model identification: A time series plot of the data is obtained, the ACF and the PACF are drawn

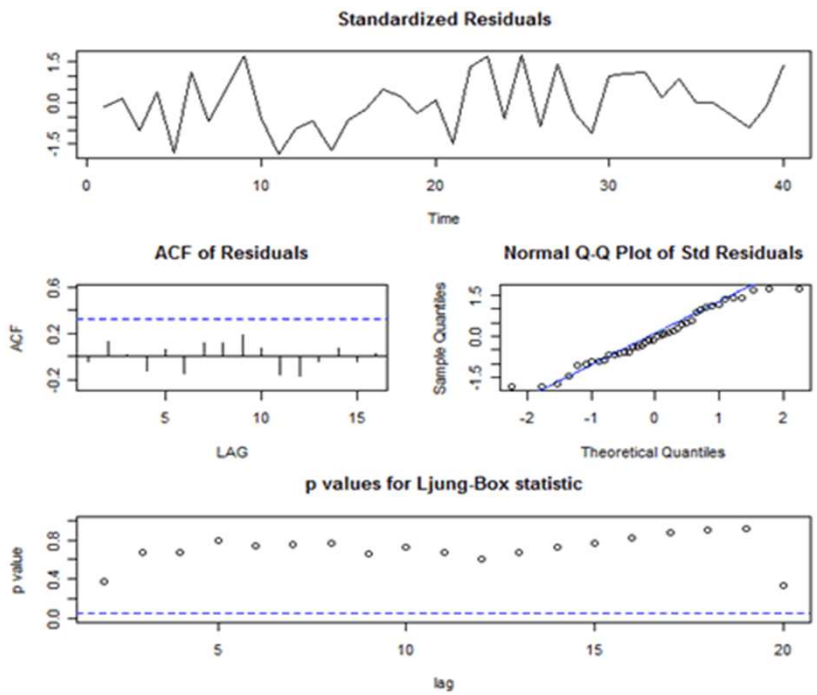




- The plot shows almost no trend. The ACF tapers off & the PACF shows a single spike. So a AR(1) _____ model is indicated.

Example

- N=40 consecutive annual measurements of the lake Erie in a particular month are given
- Model estimation: Using the software, the AR(1) model was estimated to be $x_t = 4.522 + 0.6909\, x_{t-1}$
- Model diagnosis: We check the z value = 6.315 for the AR coeff. which is statistically significant
  and do the residual diagnostics
- The time series plot of the residuals no trend, no change in variance – Good!
  The ACF shows no significant autocorrelations – Good!
  Q-Q plot shows residuals are normally distributed - Good!
- Thus the estimated model can be used for forecasting



So how do we decide the values of *p* and *q* ?

- To fit data to an ARMA model, we use the Akaike Information Criterion (AIC)across a subset of values for *p,q* to find the model with minimum AIC and then apply the Ljung-Box test to determine if a good fit has been achieved, for particular values of *p,q*.

- If the p-value of the test is greater the required significance, we can conclude that the residuals are independent and white noise.

Python code

```
# ARMA example
from statsmodels.tsa.arima_model import ARMA
from random import random
# contrived dataset
data = [random() for x in range(1, 100)]
# fit model
model = ARMA(data, order=(2, 1))
model_fit = model.fit(disp=False)
# make prediction
yhat = model_fit.predict(len(data), len(data))
print(yhat)
```

```
# ARMA example
from statsmodels.tsa.arima_model import ARMA
from random import random
# contrived dataset
data = [random() for x in range(1, 100)]
# fit model
model = ARMA(data, order=(2, 1))
model_fit = model.fit(disp=False)
# make prediction
yhat = model_fit.predict(len(data), len(data))
print(yhat)
```

```
[0.48230949]
```

Autoregressive Integrated Moving Average (ARIMA)

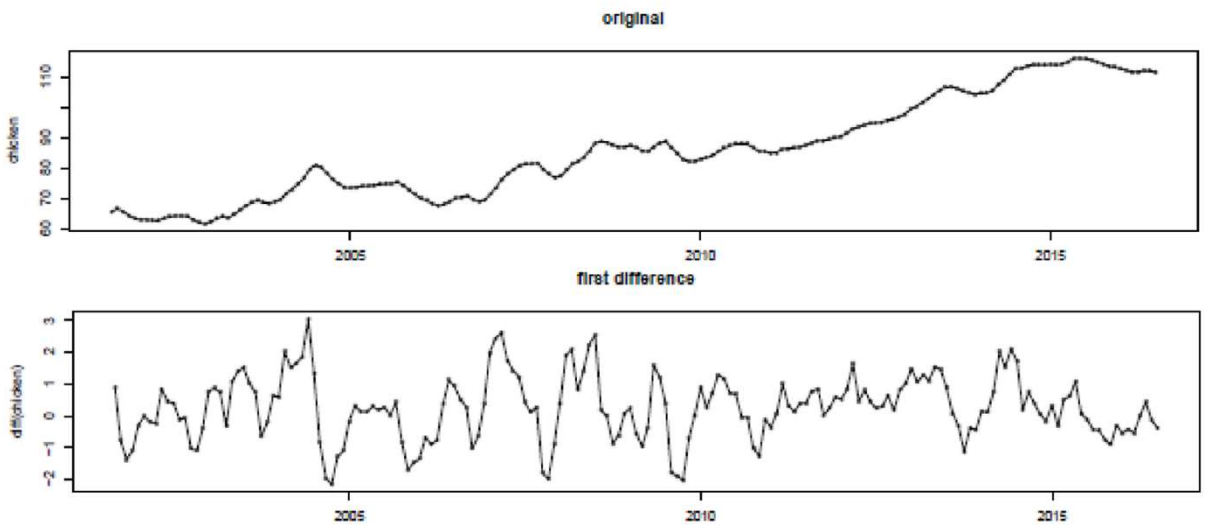
- The Autoregressive Integrated Moving Average (ARIMA) method models the next step in the sequence as a linear function of the differenced observations and residual errors at prior time steps.
- It combines both Autoregression (AR) and Moving Average (MA) models as well as a differencing pre-processing step of the sequence to make the sequence stationary, called integration (I).
- The notation for the model involves specifying the order for the AR(p), I(d), and MA(q) models as parameters to an ARIMA function, e.g. ARIMA(p, d, q). An ARIMA model can also be used to develop AR, MA, and ARMA models.
- The method is suitable for univariate time series with trend and without seasonal components

ARIMA Models

- ARIMA stands for Autoregressive, Integrated, Moving Average
- AR and MA Models work on stationary time series and I is a preprocessing procedure to “stationarize” the time series, if required, through differencing operation.
- ARIMA Model is specified by 3 parameters ARIMA(p,d,q)
- Eg. A MA(2) Model will be specified as ARIMA(0,0,2)
- ARIMA Model can be configured to perform the function of an ARMA Model, and even a simple AR, I or MA Model.

ARIMA Models

- Real life data sets are non-stationary.
- During the model identification stage, if the time series plot indicates non-stationarity, we stationarize the process
- Eg. If the process has a trend then we can stationarize by differencing and the ARMA process so obtained will be analysed using the tools already discussed.



Python code

```
# ARIMA example
from statsmodels.tsa.arima_model import ARIMA
from random import random
# contrived dataset
data = [x + random() for x in range(1, 100)]
# fit model
model = ARIMA(data, order=(1, 1, 1))
model_fit = model.fit(disp=False)
# make prediction
yhat = model_fit.predict(len(data), len(data), typ='levels')
print(yhat)
```

```
# ARIMA example
from statsmodels.tsa.arima_model import ARIMA
from random import random
# contrived dataset
data = [x + random() for x in range(1, 100)]
# fit model
model = ARIMA(data, order=(1, 1, 1))
model_fit = model.fit(disp=False)
# make prediction
yhat = model_fit.predict(len(data), len(data), typ='levels')
print(yhat)
```

```
[100.51557447]
```

References

- Introduction to Time Series and Forecasting by Peter J. Brockwell, Richard A. Davis
- Applied Business Statistics by Ken Black
- Introduction-to-time-series-forecasting by Jason Brownlee
- https://www.math-stat.unibe.ch/e237483/e237655/e243381/e281679/files281691/Chap12_ger.pdf
- https://faculty.chicagobooth.edu/jeffrey.russell/teaching/bstats/timeseries.pdf