INTRODUCTION TO TIME SERIES ANALYSIS IN PYTHON

Introducing an AR Model

Rob Reider
Adjunct Professor, NYU-Courant
Consultant, Quantopian

Mathematical Decription of AR(1) Model

today value = mean + phi * yesterday's value + noise

$$R_t = \mu + \phi \, R_{t-1} + \epsilon_t$$

- Since only one lagged value on right hand side, this is called:
  - AR model of order 1, or
  - AR(1) model
- AR parameter is ϕ =1 (random walk), =0(white noise)
- For stationarity, $-1 < \phi < 1$ (to be stable and stationary)

Interpretation of AR(1) Parameter

R_t is a time series of stock returns

$$R_t = \mu + \phi R_{t-1} + \epsilon_t$$

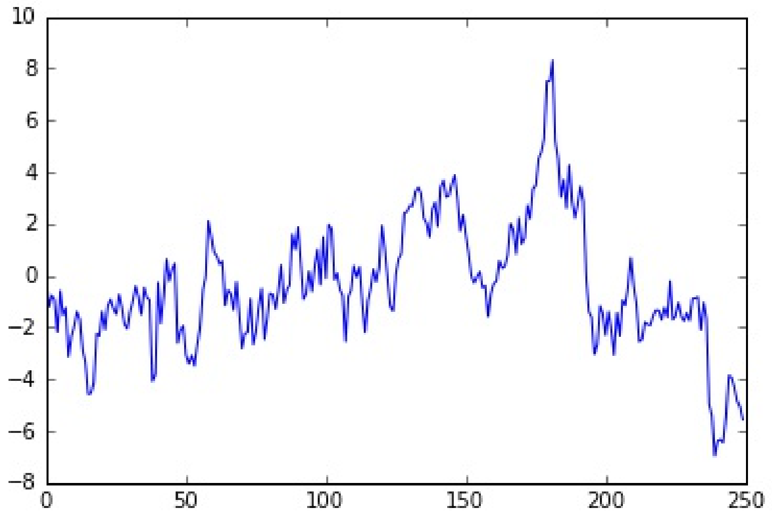
phi is negative, a positive return last period, at time t-1 implies that this period's return is more likely to be negative

- Negative ϕ: Mean Reversion
- Positive ϕ: Momentum

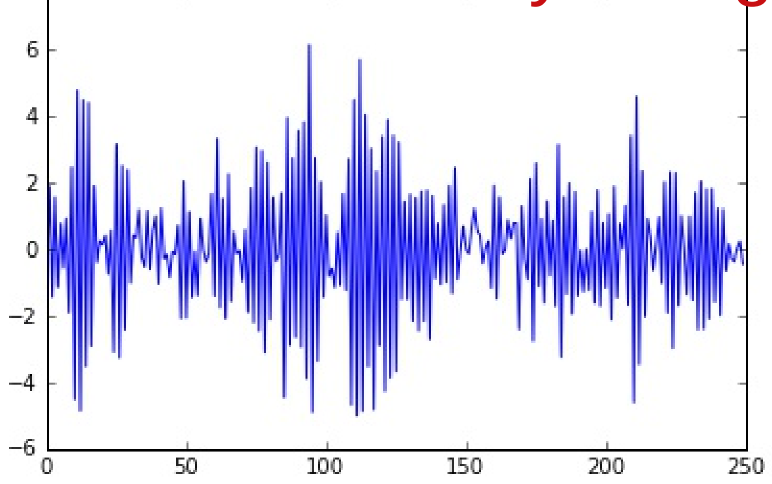
a positive return last period implies that this period's return is expected to be positive

Comparison of AR(1) Time Series

- $\phi = 0.9$ close to random walk

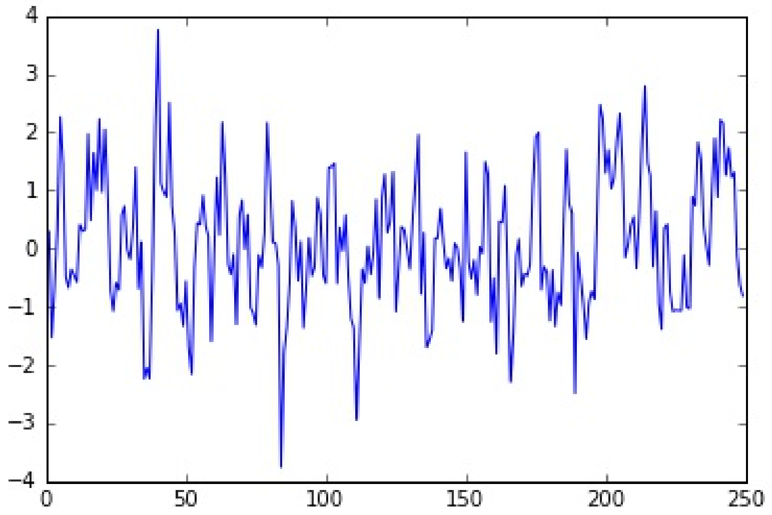


- $\phi = -0.9$ more erratic (a large + value is usually followed by a large - value)

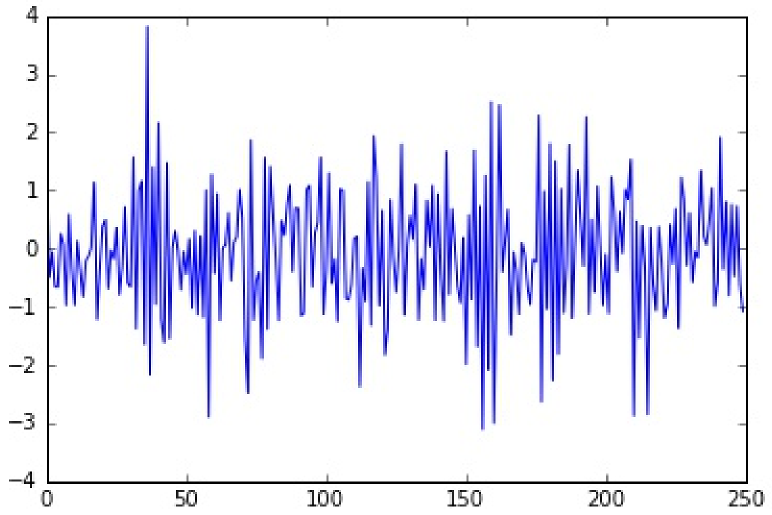


bottom two are similar, but are less exaggerated and closer to white noise

- $\phi = 0.5$



- $\phi = -0.5$

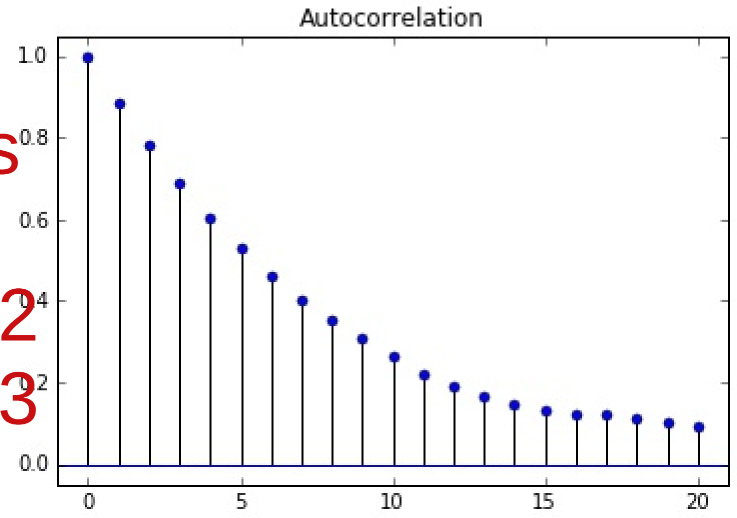


Comparison of AR(1) Autocorrelation Functions

autocorrelation decays exponentially at a rate of phi

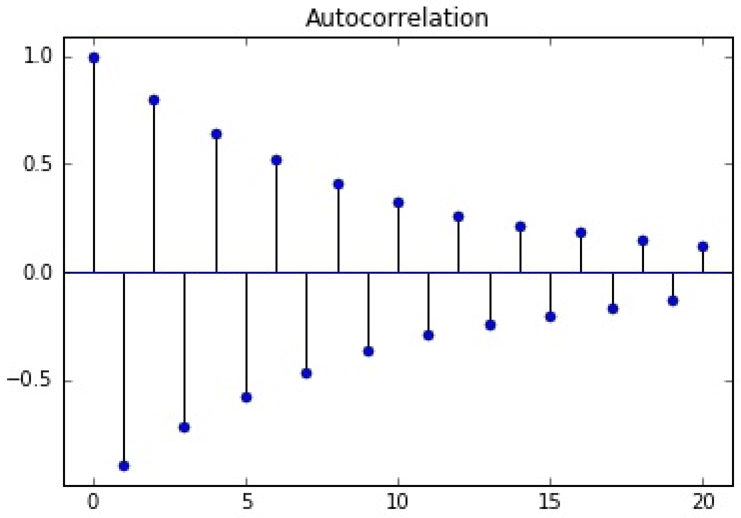
- $\phi = 0.9$

lag-1 auto correlation is 0.9
lag-2 is 0.9^2
lag-3 is 0.9^3

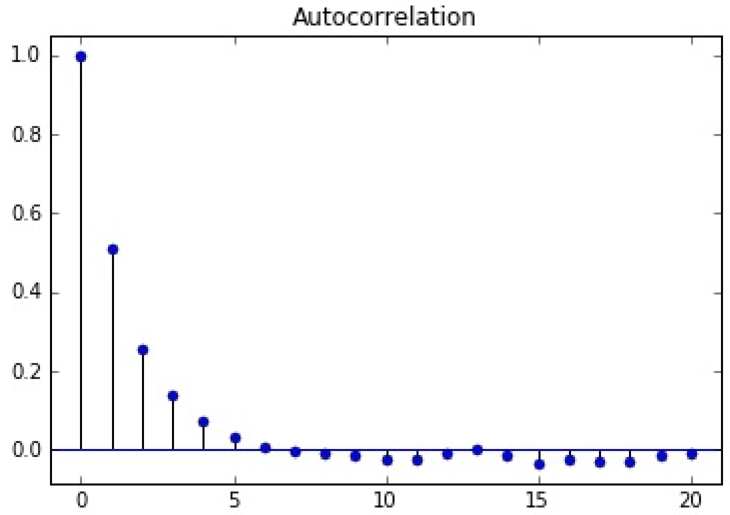


autocorrelation function is still decays exponentialy, but the signs of acf reverse at each lag

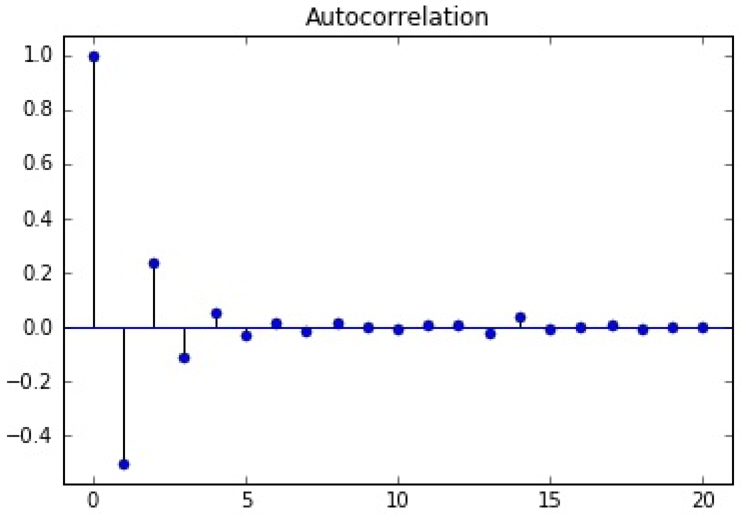
- $\phi = -0.9$



- $\phi = 0.5$



- $\phi = -0.5$



Higher Order AR Models

can be extended to include more lagged values and more phi parameters.

- AR(1)

$$R_t = \mu + \phi_1 R_{t-1} + \epsilon_t$$

- AR(2)

$$R_t = \mu + \phi_1 R_{t-1} + \phi_2 R_{t-2} + \epsilon_t$$

- AR(3)

$$R_t = \mu + \phi_1 R_{t-1} + \phi_2 R_{t-2} + \phi_3 R_{t-3} + \epsilon_t$$

- ...

Simulating an AR Process

want to study and understand a pure AR process, it is useful to work with simulated data

```
from statsmodels.tsa.arima_process import ArmaProcess
ar = np.array([1, -0.9])
ma = np.array([1])
AR_object = ArmaProcess(ar, ma)
simulated_data = AR_object.generate_sample(nsample=1000)
plt.plot(simulated_data)
```

define order and parameters

the convention is a little counterintuitive: must include the zero-lag coeff of 1, and sign of the other coeff is the opposite of what we have been using.

EX: AR(1) with phi=0.9 --> 2nd element of 'ar' array should be -0.9
this is consistent with the time series literature in the field of signal processing

also have to input MA parameters

INTRODUCTION TO TIME SERIES ANALYSIS IN PYTHON

Let's practice!

INTRODUCTION TO TIME SERIES ANALYSIS IN PYTHON

Estimating and Forecasting an AR Model

Rob Reider
Adjunct Professor, NYU-Courant
Consultant, Quantopian

Estimating an AR Model

- To estimate parameters from data (simulated)

```
from statsmodels.tsa.arima_model import ARMA
mod = ARMA(simulated_data, order=(1,0))
result = mod.fit()
```

create an instance of class called mod (data, order of the model)
(1,0) mean --> fit the data to AR(1)
(2,0) --> AR(2)
2nd part of order is MA part

.fit() to estimate model

Estimating an AR Model

- Full output (true $\mu = 0$ and $\phi = 0.9$)

estimated parameters are very close to true parameters

```
print(result.summary())
```

```
                              ARMA Model Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 5000
Model:                     ARMA(1, 0)   Log Likelihood               -7178.386
Method:                       css-mle   S.D. of innovations              1.017
Date:                Fri, 01 Dec 2017   AIC                          14362.772
Time:                        15:34:50   BIC                          14382.324
Sample:                             0   HQIC                         14369.625

==============================================================================
                 coef    std err          z      P>|z|      [95.0% Conf. Int.]
------------------------------------------------------------------------------
const         -0.0361      0.152     -0.238      0.812      -0.333       0.261
ar.L1.y        0.9054      0.006    151.020      0.000       0.894       0.917
                                    Roots
=============================================================================
                 Real           Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           1.1045           +0.0000j            1.1045            0.0000
-----------------------------------------------------------------------------
```

mean mu

AR(1) parameter

Estimating an AR Model

- Only the estimates of μ and ϕ (true $\mu = 0$ and $\phi = 0.9$)

```
print(result.params)
array([-0.03605989,  0.90535667])
```

see only coeff

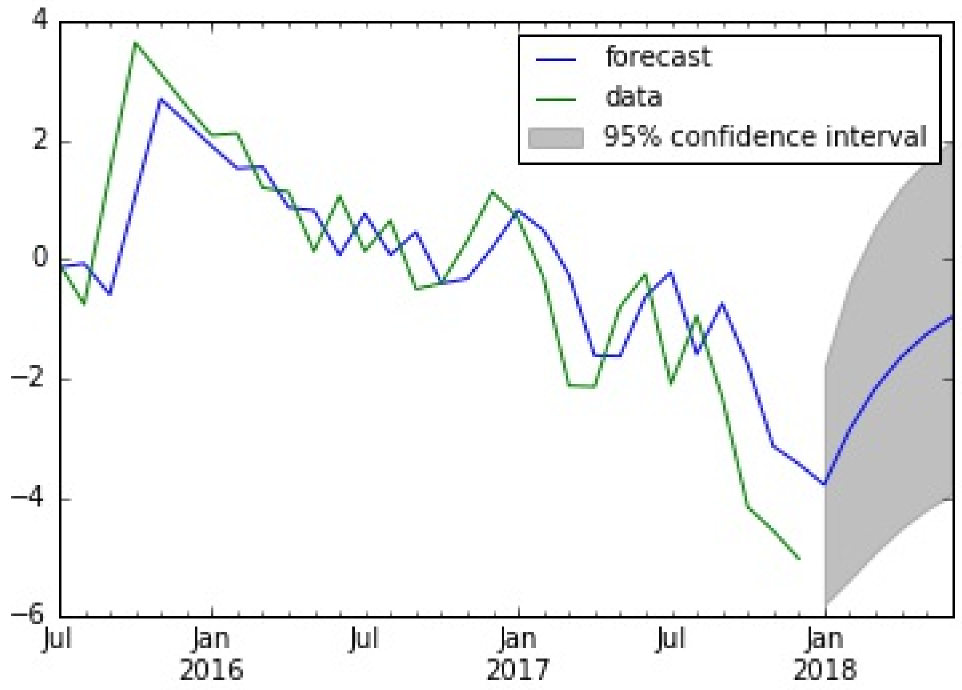
mu

phi

Forecasting an AR Model

```
from statsmodels.tsa.arima_model import ARMA
mod = ARMA(simulated_data, order=(1,0))
res = mod.fit()
res.plot_predict(start='2016-07-01', end='2017-06-01')
plt.show()
```

to do forecast



plot also gives confidence intervals around the out of sample forecasts.

notice how the confidence interval gets wider, the farther out the forecast is

INTRODUCTION TO TIME SERIES ANALYSIS IN PYTHON

Let's practice!

INTRODUCTION TO TIME SERIES ANALYSIS IN PYTHON

Choosing the Right Model

Rob Reider
Adjunct Professor, NYU-Courant
Consultant, Quantopian

in practice, you will ordinarily not be told the order of the model that you are trying to estimate

Identifying the Order of an AR Model

- The order of an AR(p) model will usually be unknown
- Two techniques to determine order
  - Partial Autocorrelation Function
  - Information criteria

Partial Autocorrelation Funcion (PACF)

measures the incremental benefit of adding another lag.

imagine running several regressions, where u regress returns on more and more lagged values.

$$R_t = \phi_{0,1} + \boxed{\phi_{1,1}} R_{t-1} + \epsilon_{1t}$$

$$R_t = \phi_{0,2} + \phi_{1,2} R_{t-1} + \boxed{\phi_{2,2}} R_{t-2} + \epsilon_{2t}$$

$$R_t = \phi_{0,3} + \phi_{1,3} R_{t-1} + \phi_{2,3} R_{t-2} + \boxed{\phi_{3,3}} R_{t-3} + \epsilon_{3t}$$

$$R_t = \phi_{0,4} + \phi_{1,4} R_{t-1} + \phi_{2,4} R_{t-2} + \phi_{3,4} R_{t-3} + \boxed{\phi_{4,4}} R_{t-4} + \epsilon_{4t}$$

$$\vdots$$

is the lag-4 value of the partial acf, it represent how significant adding a fourth lag is when u already have 3 lags

coefficients in red boxes represent values of the partial autocorrelation function for different lags.

Plot PACF in Python

- Same as ACF, but use plot_pacf instead of plt_acf

- Import module

```
from statsmodels.graphics.tsaplots import plot_pacf
```

- Plot the PACF

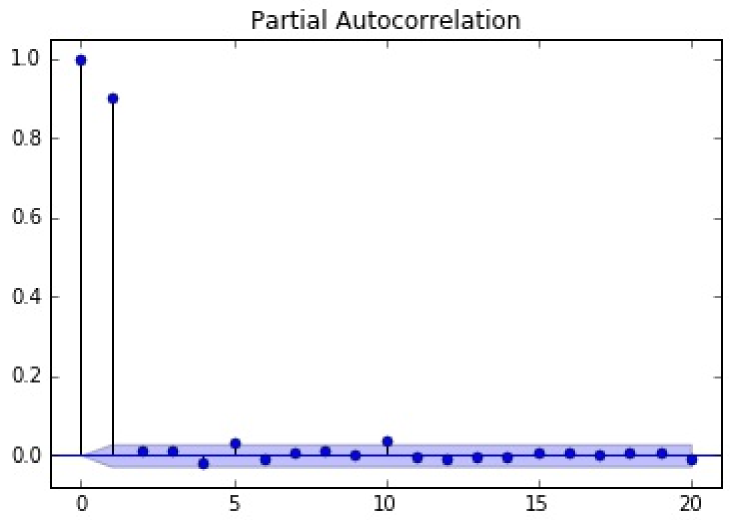
```
plot_pacf(x, lags= 20, alpha=0.05)
```

how many lags of pacf will be plotted
alpha: set the width of the confidence interval

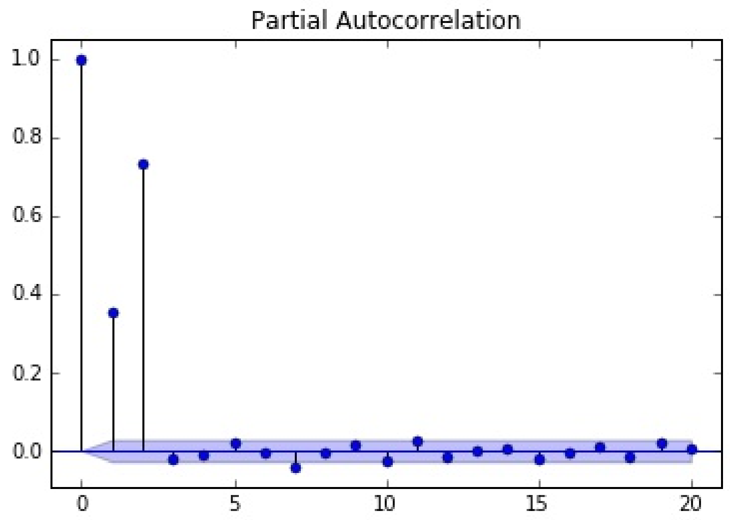
Comparison of PACF for Different AR Models

- AR(1)

only lag-1 pacf is significantly different from zero



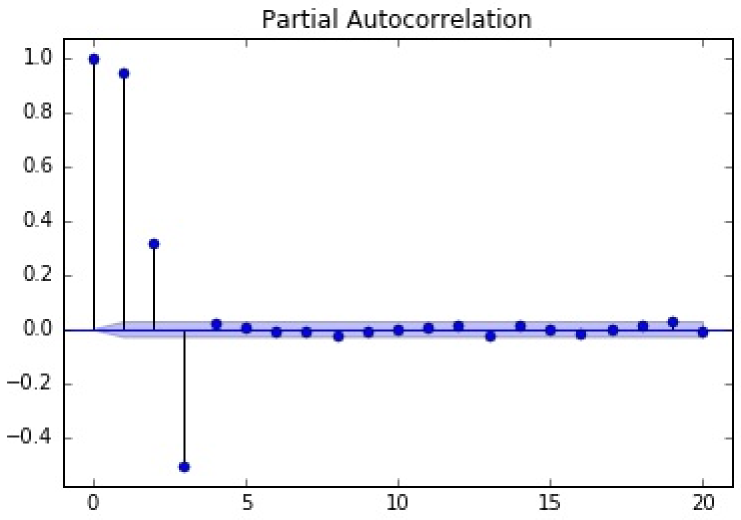
- AR(2)



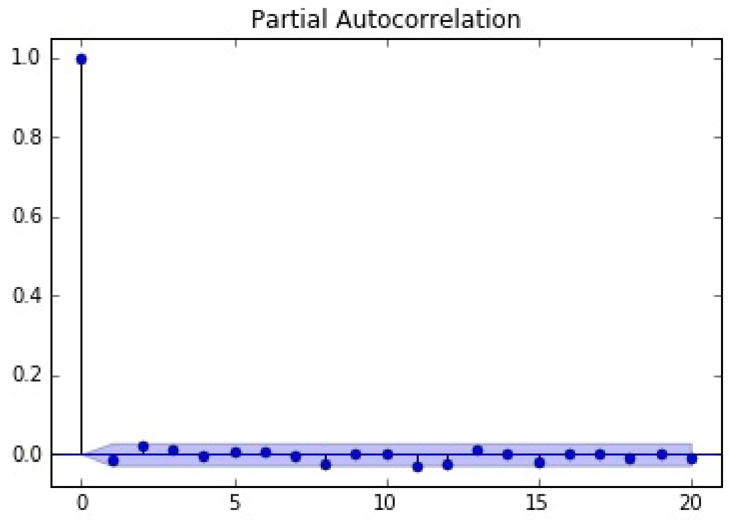
2 lags are different from zero

- AR(3)

3 lags are different from zero



- White Noise



no lags are significantly different from zero

Information Criteria

more parameters in a model, better the model will fit the data.
but this can lead to overfitting of the data

- Information criteria: adjusts goodness-of-fit for number of parameters
- Two popular adjusted goodness-of-fit meaures
  - AIC (Akaike Information Criterion)
  - BIC (Bayesian Information Criterion)

by imposing a penalty based on the no of parameters used.

Information Criteria

- Estimation output

```
                              ARMA Model Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 2500
Model:                     ARMA(2, 0)   Log Likelihood               -3536.481
Method:                       css-mle   S.D. of innovations              0.996
Date:                Fri, 29 Dec 2017   AIC                           7080.963
Time:                        22:53:24   BIC                           7104.259
Sample:                             0   HQIC                          7089.420

==============================================================================
                 coef    std err          z      P>|z|      [95.0% Conf. Int.]
------------------------------------------------------------------------------
const          0.0054      0.010      0.517      0.605        -0.015     0.026
ar.L1.y       -0.6130      0.019    -32.243      0.000        -0.650    -0.576
ar.L2.y       -0.3109      0.019    -16.351      0.000        -0.348    -0.274
                                    Roots
=============================================================================
                 Real           Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1          -0.9859             -1.4982j           1.7935           -0.3426
AR.2          -0.9859             +1.4982j           1.7935            0.3426
-----------------------------------------------------------------------------
```

Getting Information Criteria From statsmodels

- You learned earlier how to fit an AR model

```
from statsmodels.tsa.arima_model import ARMA
mod = ARMA(simulated_data, order=(1,0))
result = mod.fit()
```

- And to get full output

```
result.summary()
```

- Or just the parameters

```
result.params
```

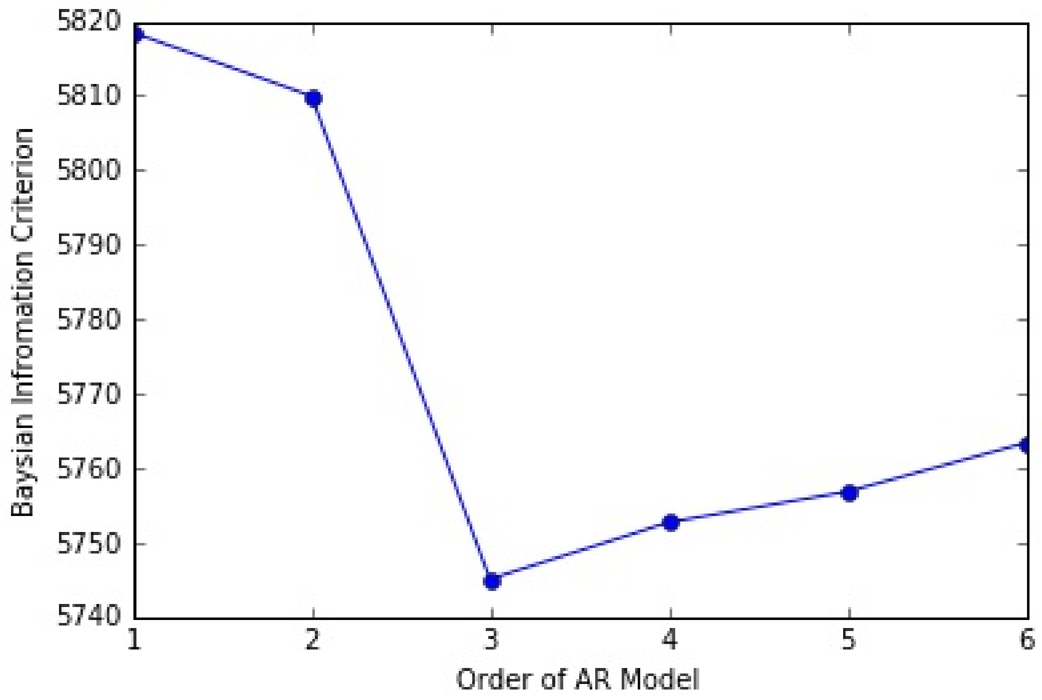
- To get the AIC and BIC

```
result.aic
result.bic
```

Information Criteria

in practice, the way to use the info crieteria is to fit several models, each with a different no of parameters, and choose the one with the lowest BIC

- Fit a simulated AR(3) to different AR(p) models
- Choose p with the lowest BIC



INTRODUCTION TO TIME SERIES ANALYSIS IN PYTHON

Let's practice!