Introducing an AR Model

TIME SERIES ANALYSIS IN PYTHON

Rob Reider

Adjunct Professor, NYU-Courant
Consultant, Quantopian

Mathematical Description of AR(1) Model

Today's value equals a mean, plus a fraction phi, of yesterday's value, plus noise.

AR = Auto Regressive Model

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

- 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 <— If Phi = 1 then the series is a random walk, if phi is 0 then the process is white noise.
- For stationarity, $-1 < \phi < 1$ <— In order for the process ot be stables, phi has to be between 1 and -1

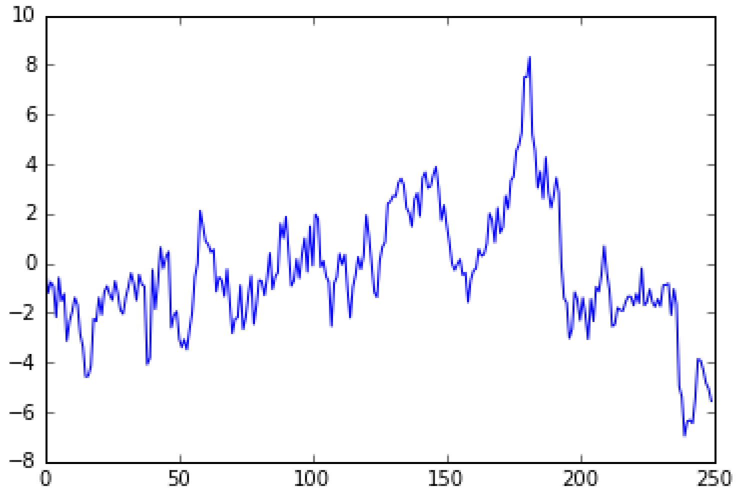
Interpretation of AR(1) Parameter

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

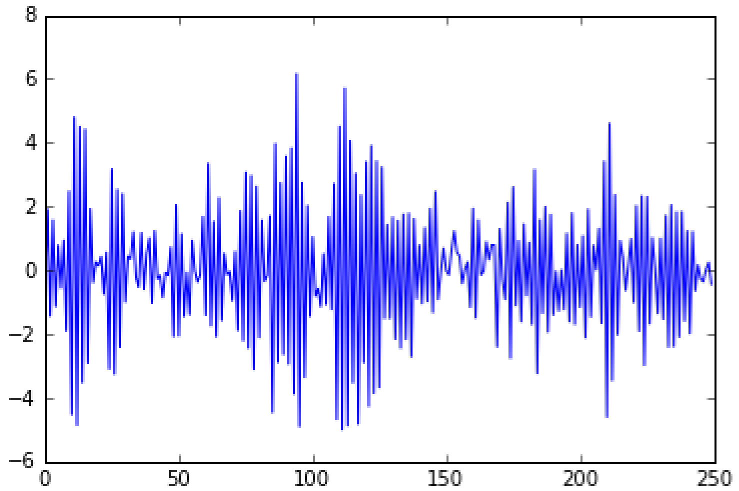
- Negative ϕ: Mean Reversion
- Positive ϕ: Momentum

Comparison of AR(1) Time Series

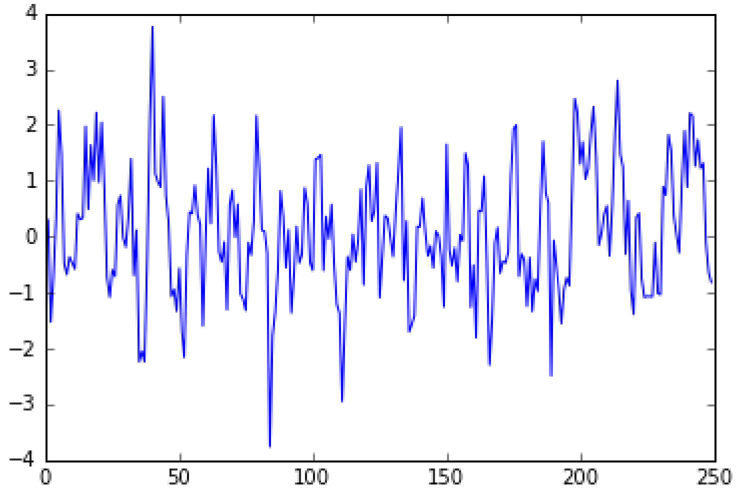
- $\phi = 0.9$



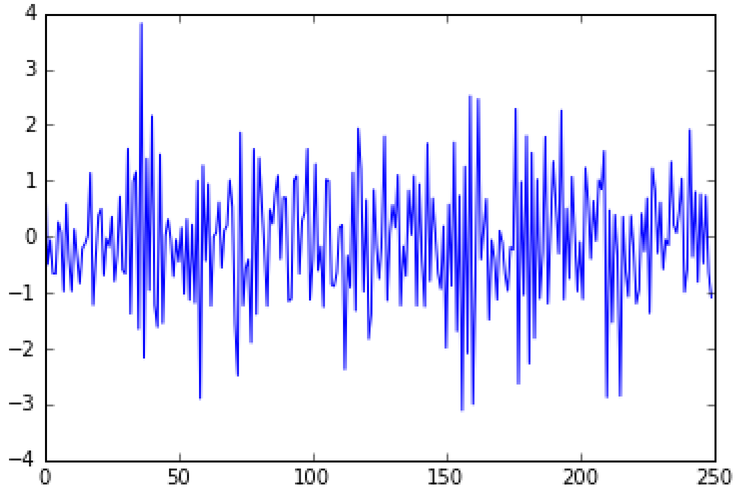
- $\phi = -0.9$



- $\phi = 0.5$

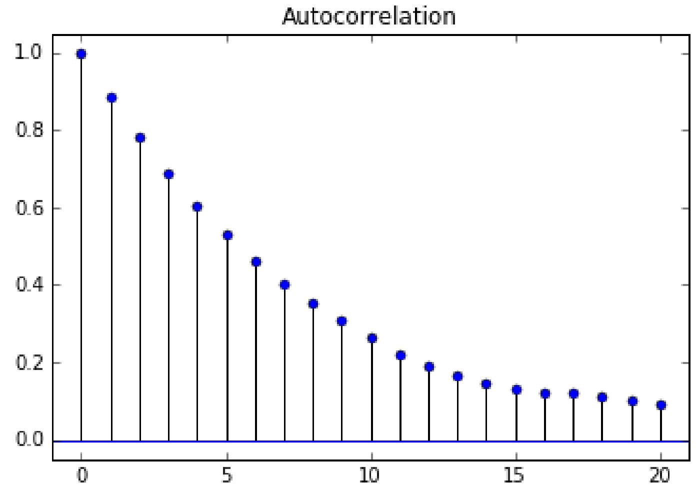


- $\phi = -0.5$

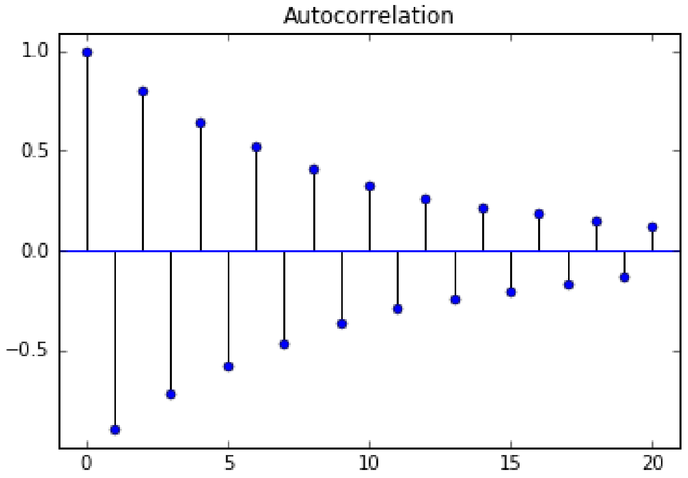


Comparison of AR(1) Autocorrelation Functions

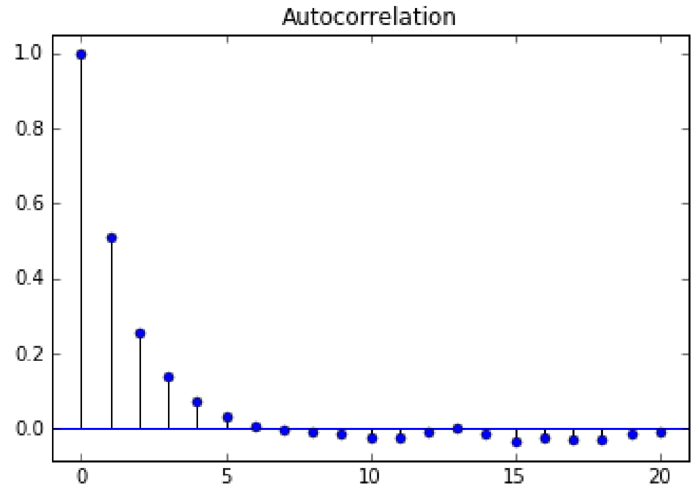
- $\phi = 0.9$



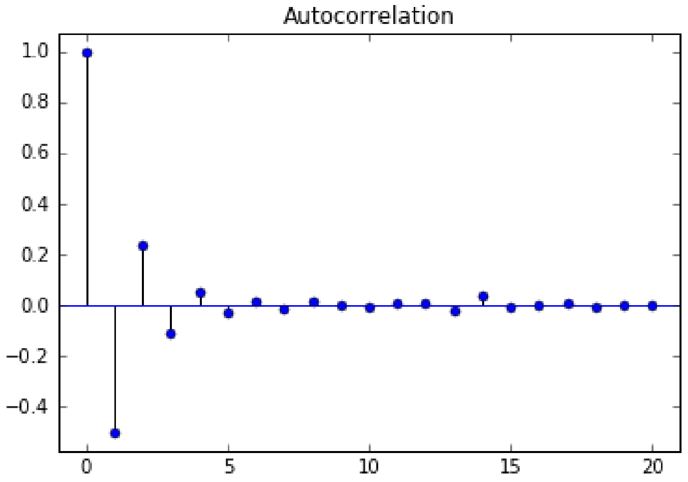
- $\phi = -0.9$



- $\phi = 0.5$



- $\phi = -0.5$



Higher Order AR Models

- 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

```
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)
```

<— You must include the 0 lag coefficient of 1, and the sign of the other coefficient is the opposite of what we've been using.

Ex: For a AR(1) process with phi equal to +0.9, the second element of the AR array should be the opposite sign: -0.9

```
# import the module for simulating data
from statsmodels.tsa.arima_process import ArmaProcess

# Plot 1: AR parameter = +0.9
plt.subplot(2,1,1)
ar1 = np.array([1, -0.9])
ma1 = np.array([1])
AR_object1 = ArmaProcess(ar1, ma1)
simulated_data_1 = AR_object1.generate_sample(nsample=1000)
plt.plot(simulated_data_1)

# Plot 2: AR parameter = -0.9
plt.subplot(2,1,2)
ar2 = np.array([1, 0.9])
ma2 = np.array([1])
AR_object2 = ArmaProcess(ar2, ma2)
simulated_data_2 = AR_object2.generate_sample(nsample=1000)
plt.plot(simulated_data_2)
plt.show()
```

The two AR parameters produce very different looking time series plots, but in the next exercise you'll really be able to distinguish the time series.

Let's practice!

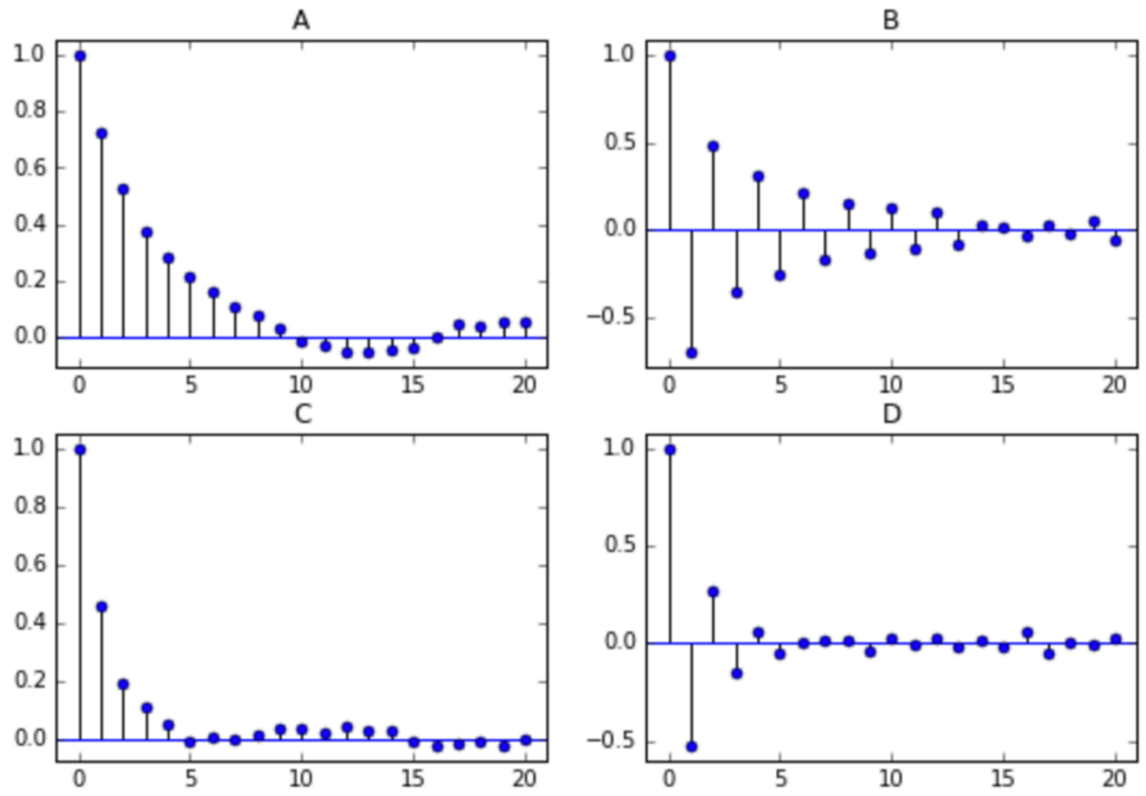
TIME SERIES ANALYSIS IN PYTHON

```
# Import the plot_acf module from statsmodels
from statsmodels.graphics.tsaplots import plot_acf

# Plot 1: AR parameter = +0.9
plot_acf(simulated_data_1, alpha=1, lags=20)
plt.show()

# Plot 2: AR parameter = -0.9
plot_acf(simulated_data_2, alpha=1, lags=20)
plt.show()

# Plot 3: AR parameter = +0.3
plot_acf(simulated_data_3, alpha=1, lags=20)
plt.show()
```

The autocorrelation function decays exponentially for an AR time series at a rate of the AR parameter. For example, if the AR parameter, $\phi = +0.9$, the first-lag autocorrelation will be 0.9, the second-lag will be (0.9) ^^ 2=0.81, the third-lag will be (0.9) ^^ 3 = 0.729, etc. A smaller AR parameter will have a steeper decay, and for a negative AR parameter, say -0.9, the decay will flip signs, so the first-lag autocorrelation will be -0.9, the second-lag will be (−0.9) ^^ 2 = 0.81 , the third-lag will be (−0.9) ^^ 3 = −0.729, etc.

Match AR Model with ACF

Here are four Autocorrelation plots:



Which figure corresponds to an AR(1) model with an AR parameter of -0.5?

Answer the question 50 XP

Possible Answers

- ○ A press 1
- ○ B press 2
- ○ C press 3
- ◉ D press 4

Estimating and Forecasting an AR Model

TIME SERIES ANALYSIS IN PYTHON

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()
```

Estimating an AR Model

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

```
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
-----------------------------------------------------------------------------
```

Estimating an AR Model

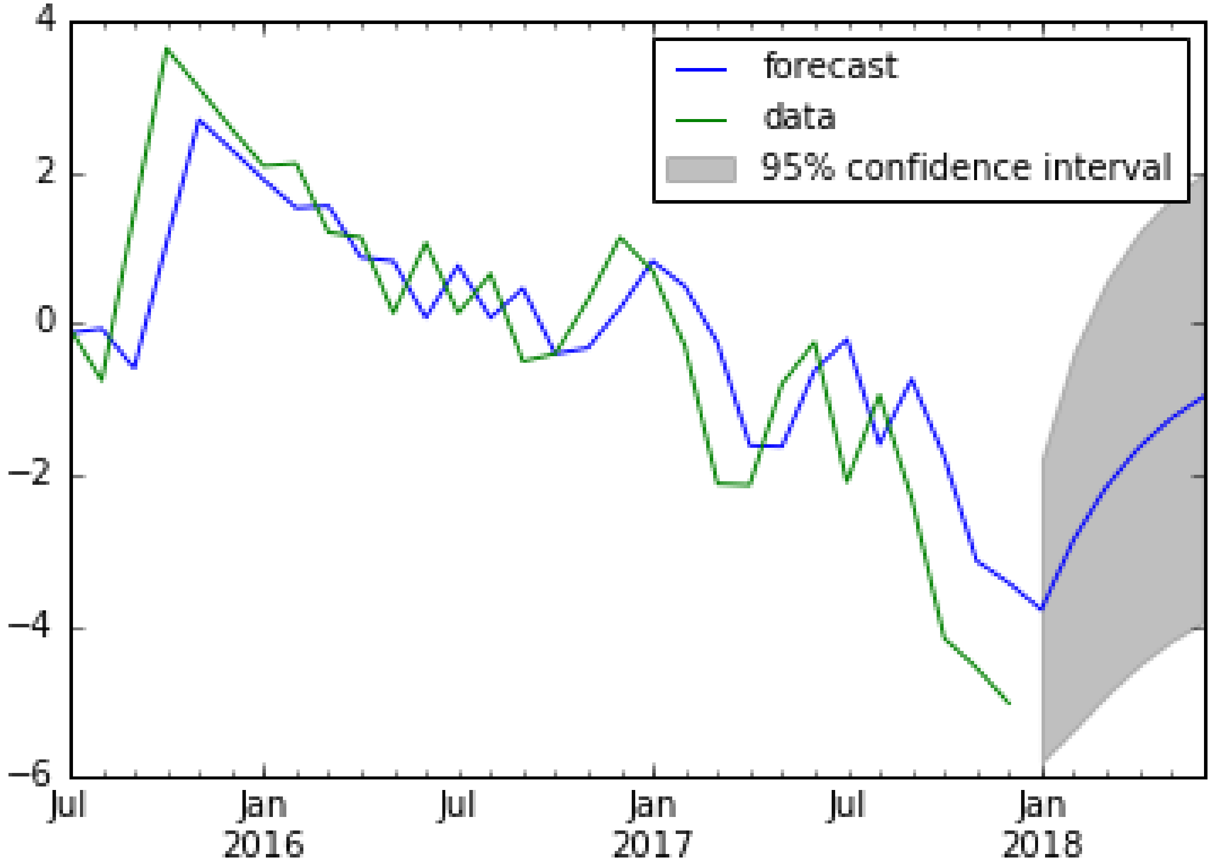
- Only the estimates of μ and ϕ (true $\mu = 0$ and $\phi = 0.9$)

```
print(result.params)
```

```
array([-0.03605989,  0.90535667])
```

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()
```



The confidence interval gets wider the further out the forecast is.

Let's practice!

TIME SERIES ANALYSIS IN PYTHON

Choosing the Right Model

TIME SERIES ANALYSIS IN PYTHON

Rob Reider

Adjunct Professor, NYU-Courant
Consultant, Quantopian

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 Function (PACF)

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

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

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

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

$$\vdots$$

Plot PACF in Python

- Same as ACF, but use `plot_pacf` instead of `plt_acf`
- Import module

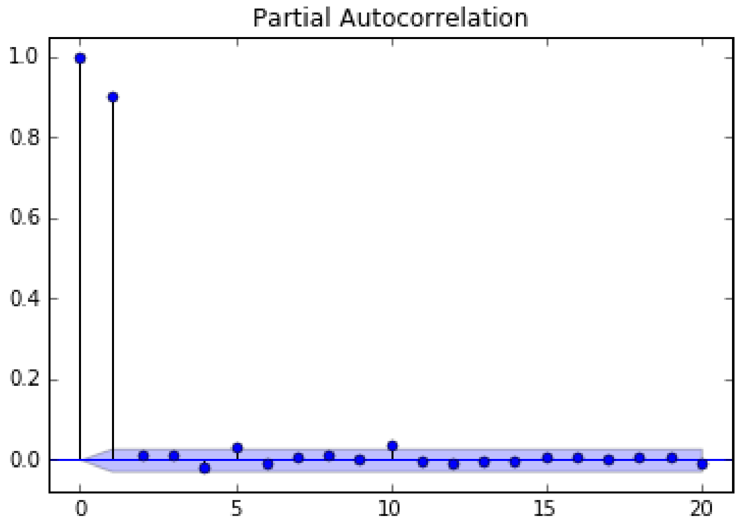
```
from statsmodels.graphics.tsaplots import plot_pacf
```

- Plot the PACF

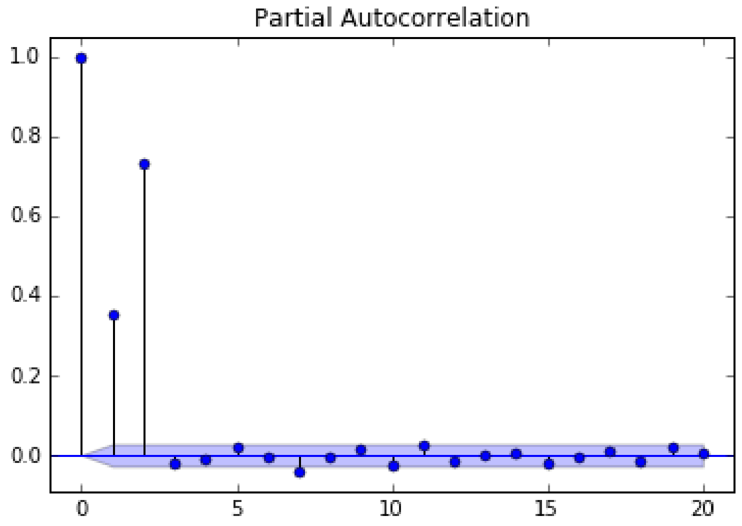
```
plot_pacf(x, lags= 20, alpha=0.05)
```

Comparison of PACF for Different AR Models

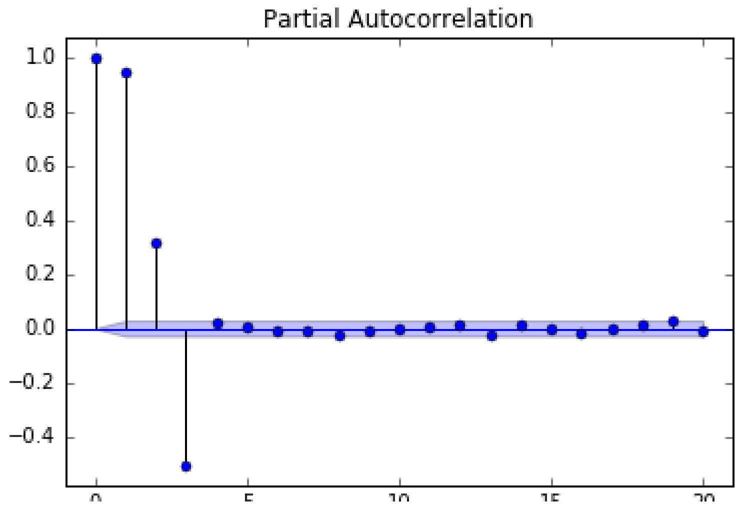
- AR(1)



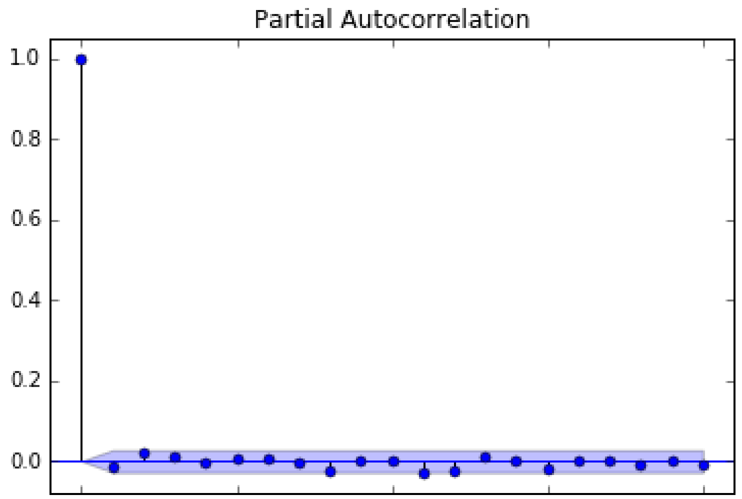
- AR(2)



- AR(3)



- White Noise



Information Criteria

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

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

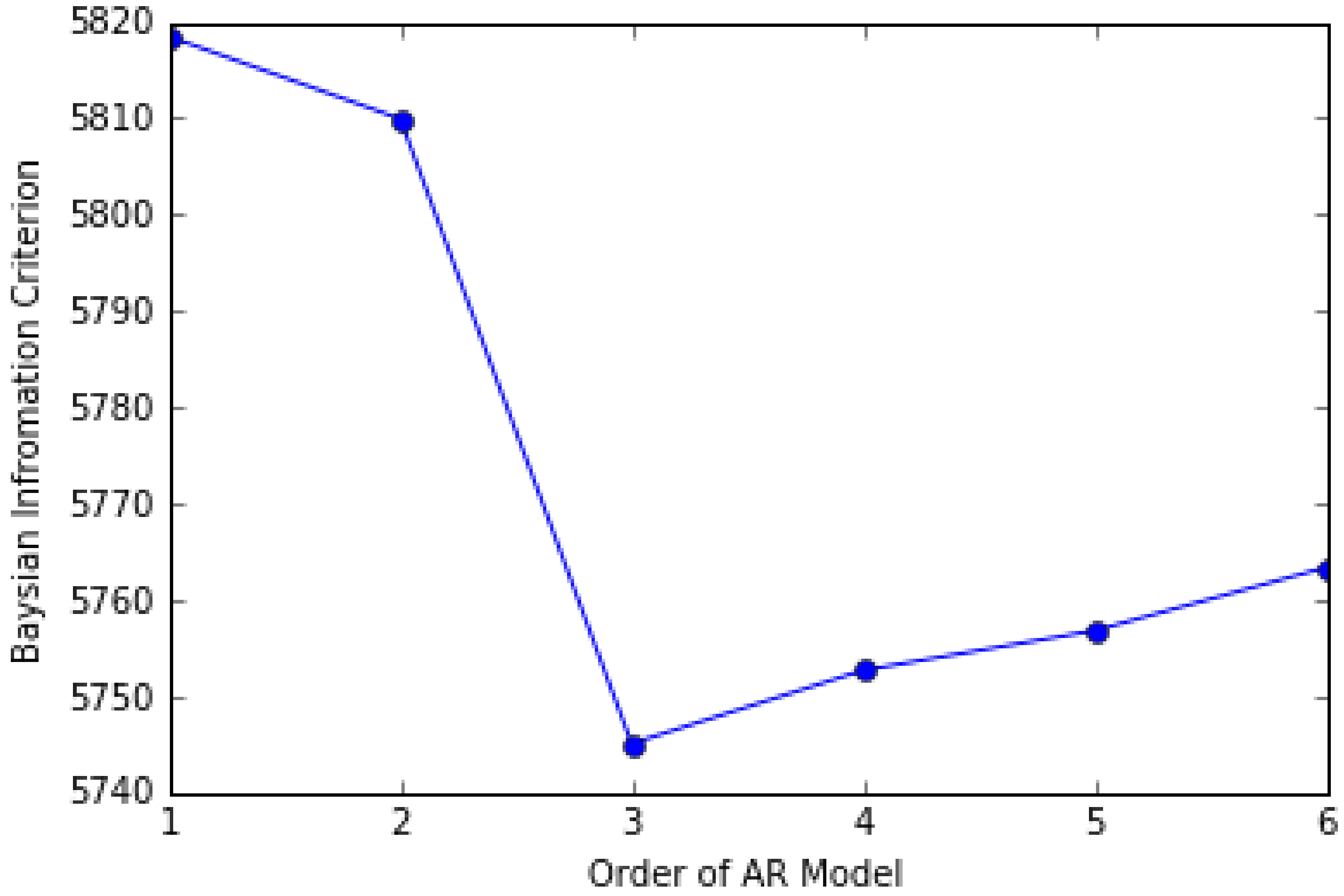
```
result.params
```

- To get the AIC and BIC

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

Information Criteria

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



Let's practice!

TIME SERIES ANALYSIS IN PYTHON