Autocorrelation Function

TIME SERIES ANALYSIS IN PYTHON

Rob Reider
Adjunct Professor, NYU-Courant
Consultant, Quantopian

Autocorrelation Function

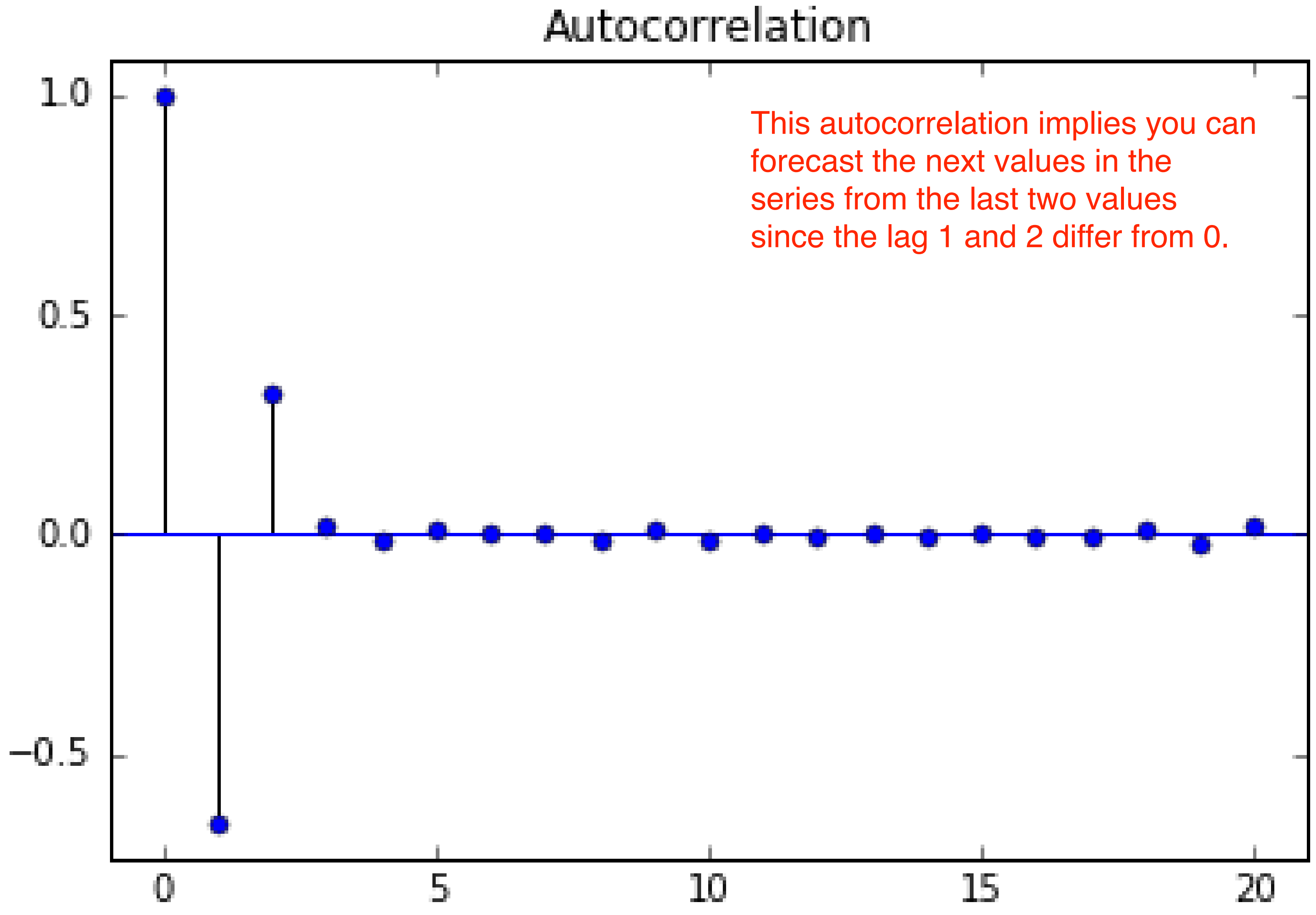
- Autocorrelation Function (ACF): The autocorrelation as a function of the lag

Any significant non-zero autocorrelation implies that the serirs can be forecast from the past.

- Equals one at lag-zero
- Interesting information beyond lag-one

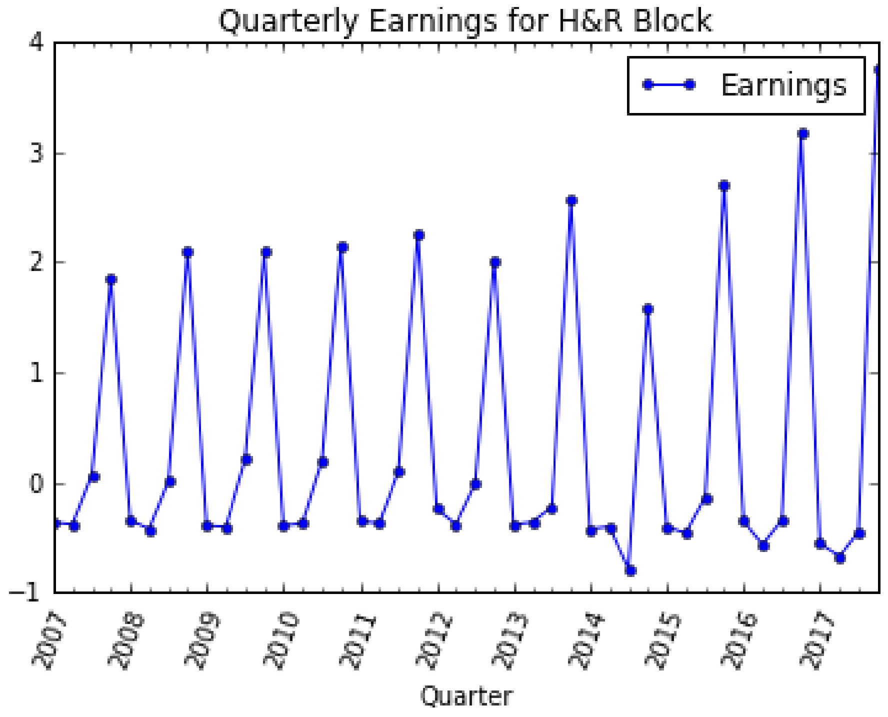
ACF Example 1: Simple Autocorrelation Function

- Can use last two values in series for forecasting

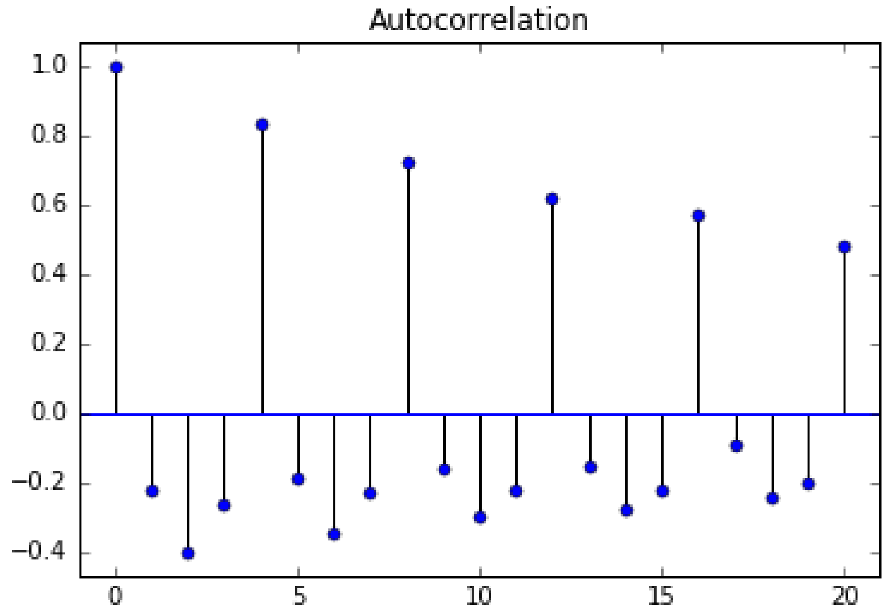


ACF Example 2: Seasonal Earnings

- Earnings for H&R Block

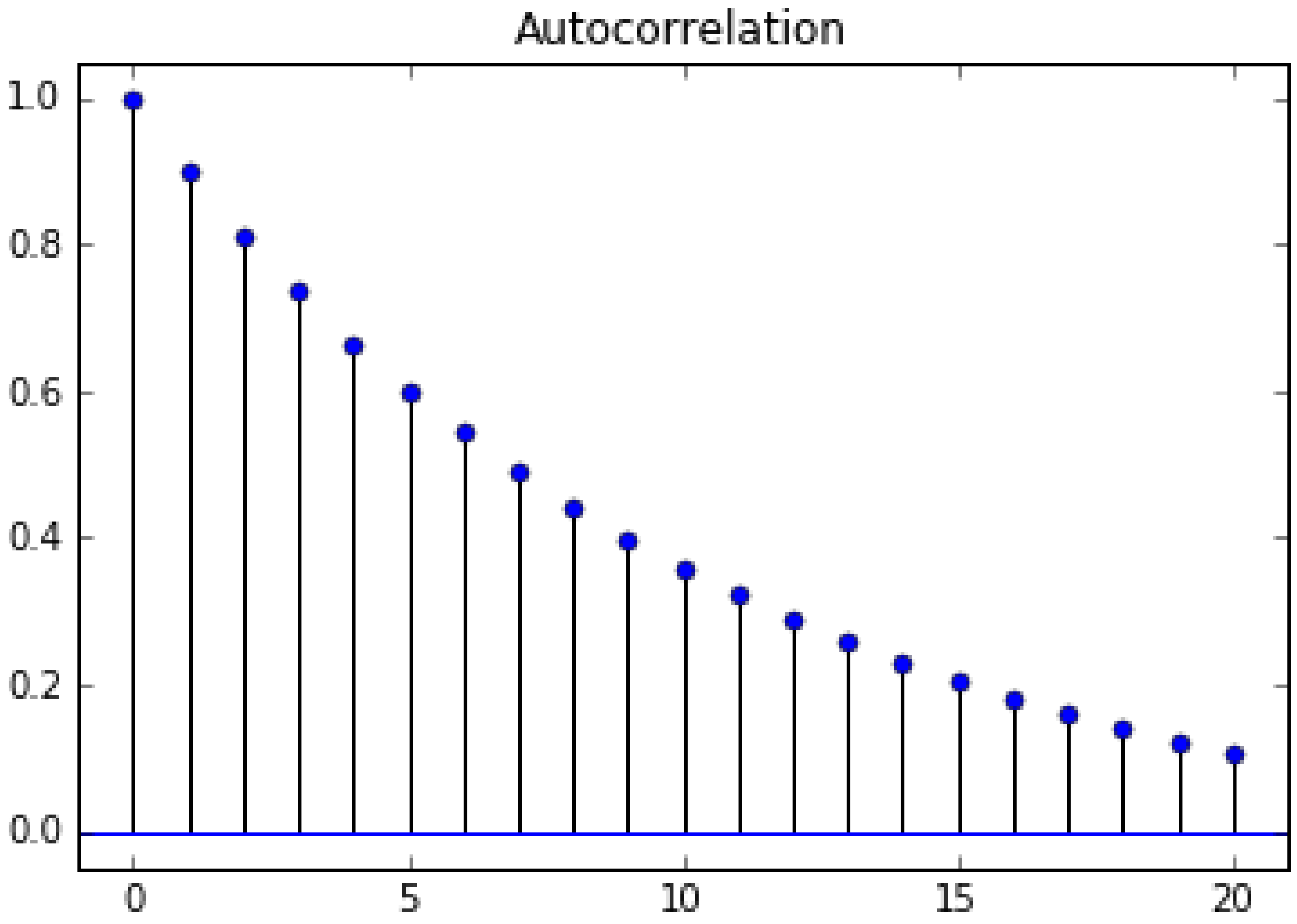


- ACF for H&R Block



This shows strong autocorrelation at lags 4, 8, 12, 16 & 20.

ACF Example 3: Useful for Model Selection



ACF can also be used for a parsimonious model for fitting the data.

- Model selection

Plot ACF in Python

- Import module:

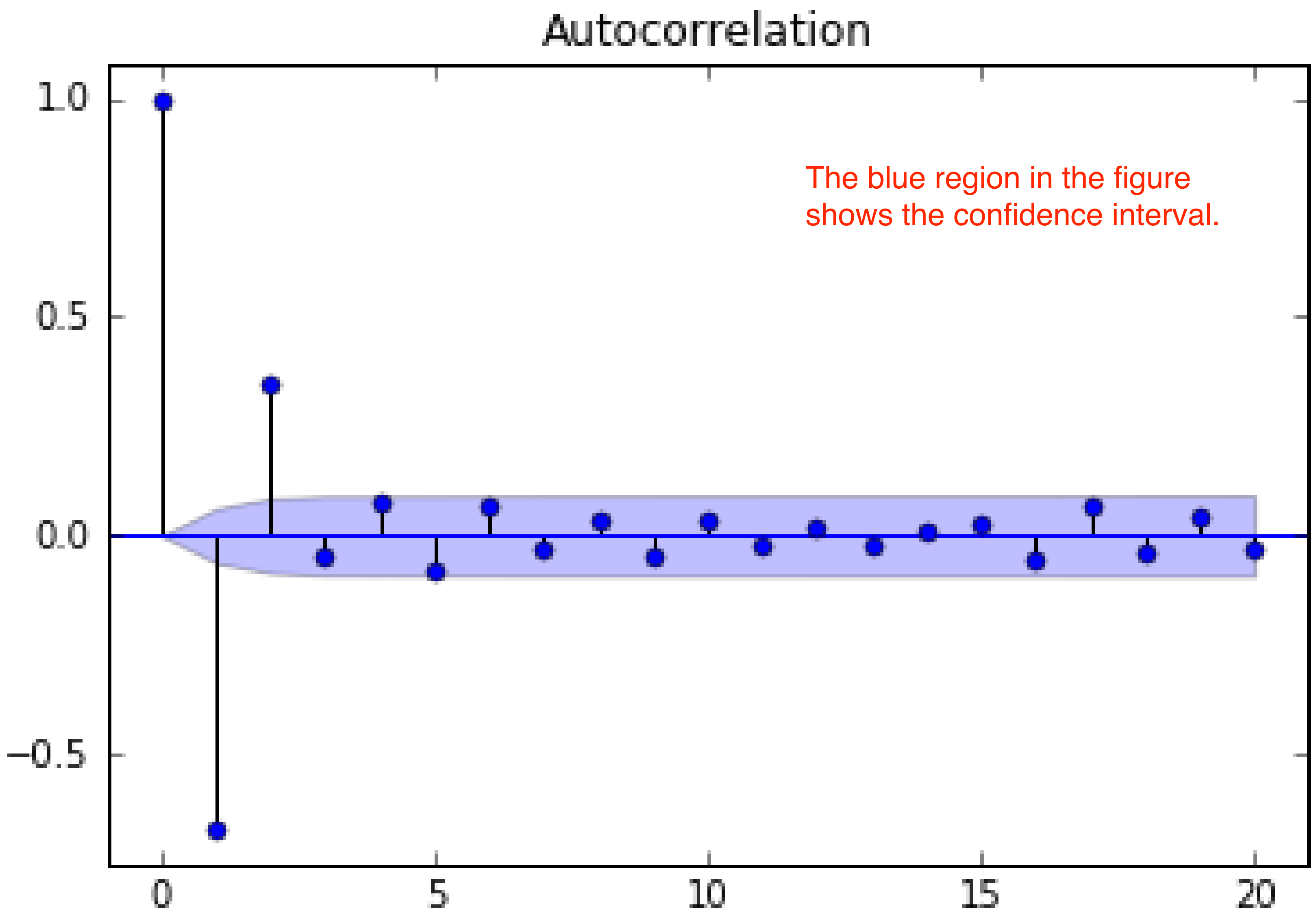
```
from statsmodels.graphics.tsaplots import plot_acf
```

- Plot the ACF:

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

x is the series or array
lags is how many of the lags of the ACF will be plotted
alpha sets the width of confidence interval.

Confidence Interval of ACF



Confidence Interval of ACF

- Argument `alpha` sets the width of confidence interval
- Example: `alpha=0.05`
  - 5% chance that if true autocorrelation is zero, it will fall outside blue band

In other words, can you say that there is less than a 5% chance that we would observe such an autocorrelation value if the true autocorrelation were really zero?

- Confidence bands are wider if:
  - Alpha lower
  - Fewer observations
- Under some simplifying assumptions, 95% confidence bands are $\pm 2/\sqrt{N}$
- If you want no bands on plot, set `alpha=1`

ACF Values Instead of Plot

```
from statsmodels.tsa.stattools import acf
print(acf(x))
```

```
[ 1.          -0.6765505   0.34989905 -0.01629415 -0.0250701
 -0.03186545  0.01399904 -0.03518128  0.02063168 -0.0262064
...
  0.07191516 -0.12211912  0.14514481 -0.09644228  0.0521588
```

```
# Import the acf module and the plot_acf module from statsmodels
from statsmodels.tsa.stattools import acf
from statsmodels.graphics.tsaplots import plot_acf

# Compute the acf array of HRB
acf_array = acf(HRB)
print(acf_array)

# Plot the acf function
plot_acf(HRB, alpha = 1)
plt.show()
```

Notice the strong positive autocorrelation at lags 4, 8, 12, 16,20, ...

Let's practice!

TIME SERIES ANALYSIS IN PYTHON

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

# Compute and print the autocorrelation of MSFT weekly returns
autocorrelation = returns['Adj Close'].autocorr()
print("The autocorrelation of weekly MSFT returns is %4.2f" %(autocorrelation))

# Find the number of observations by taking the length of the returns DataFrame
nobs = len(returns)

# Compute the approximate confidence interval
conf = 1.96/sqrt(nobs)
print("The approximate confidence interval is +/- %4.2f" %(conf))

# Plot the autocorrelation function with 95% confidence intervals and 20 lags using plot_acf
plot_acf(returns, alpha=0.05, lags = 20)
plt.show()
```

Notice that the autocorrelation with lag 1 is significantly negative, but none of the other lags are significantly different from zero

White Noise

TIME SERIES ANALYSIS IN PYTHON

Rob Reider
Adjunct Professor, NYU-Courant
Consultant, Quantopian

What is White Noise?

- White Noise is a series with:
  - Constant mean
  - Constant variance
  - Zero autocorrelations at all lags
- Special Case: if data has normal distribution, then *Gaussian White Noise*

Simulating White Noise

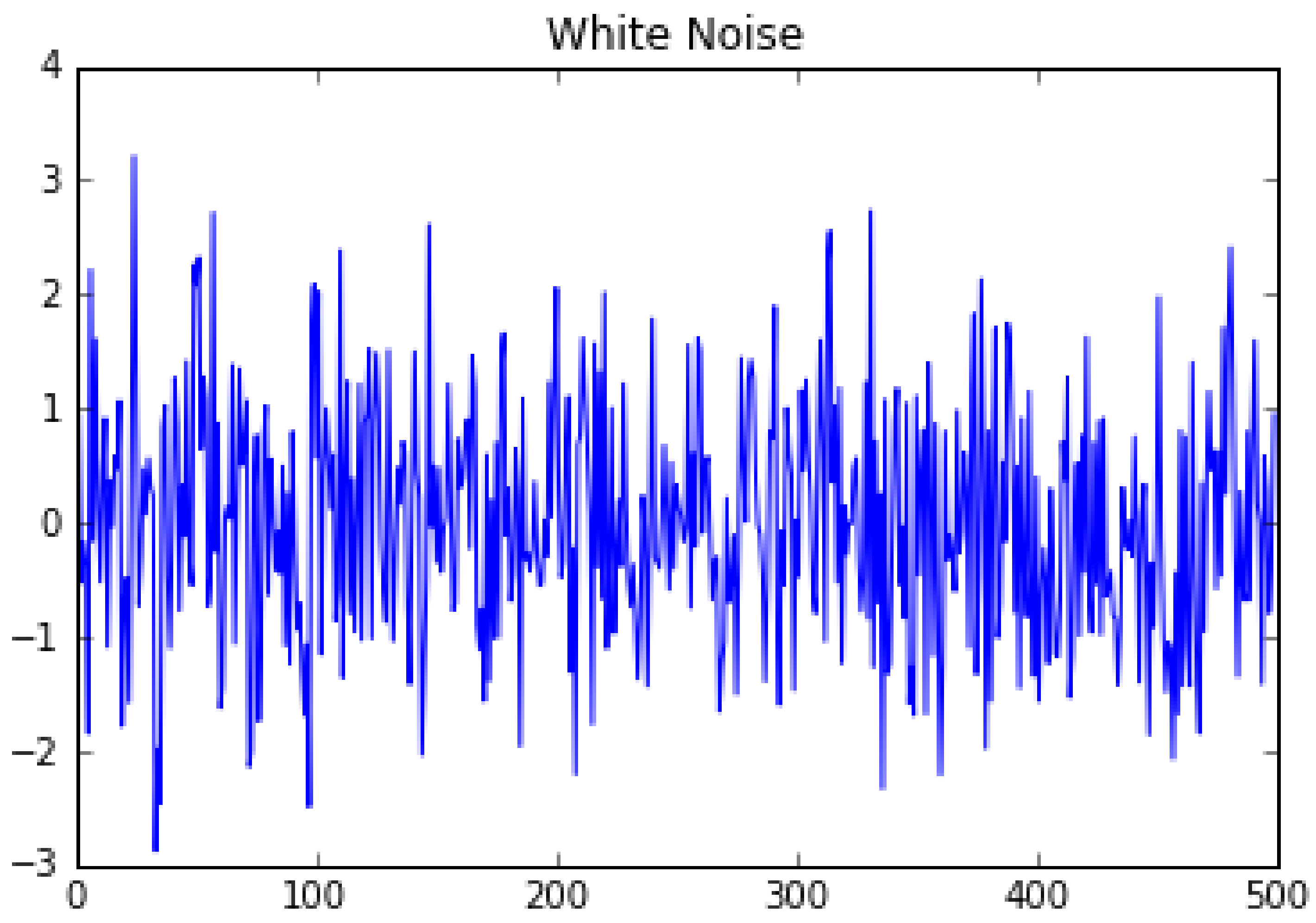
- It's very easy to generate white noise

```
import numpy as np
noise = np.random.normal(loc=0, scale=1, size=500)
```

loc is the mean and scale is the standard deviation

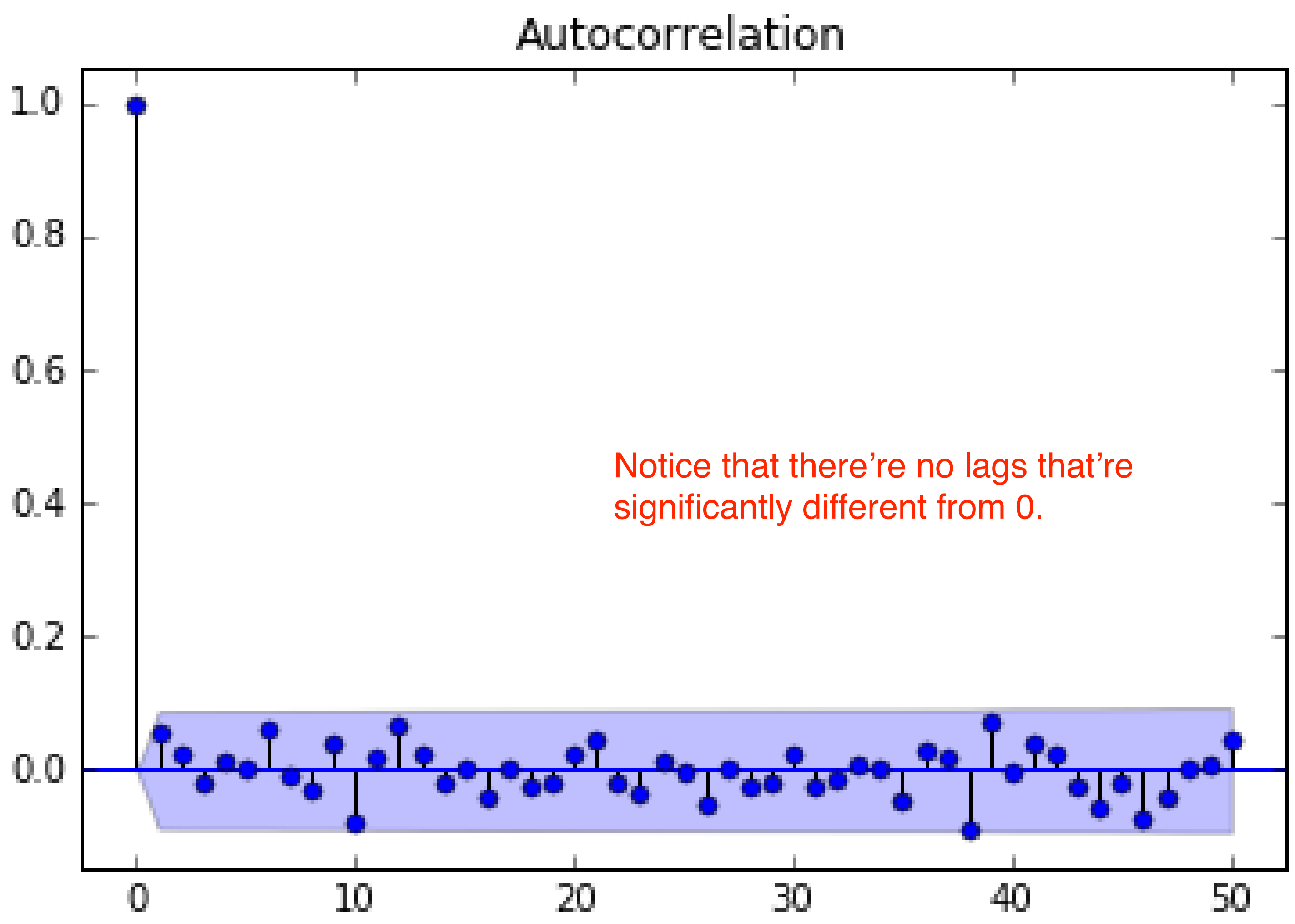
What Does White Noise Look Like?

```
plt.plot(noise)
```



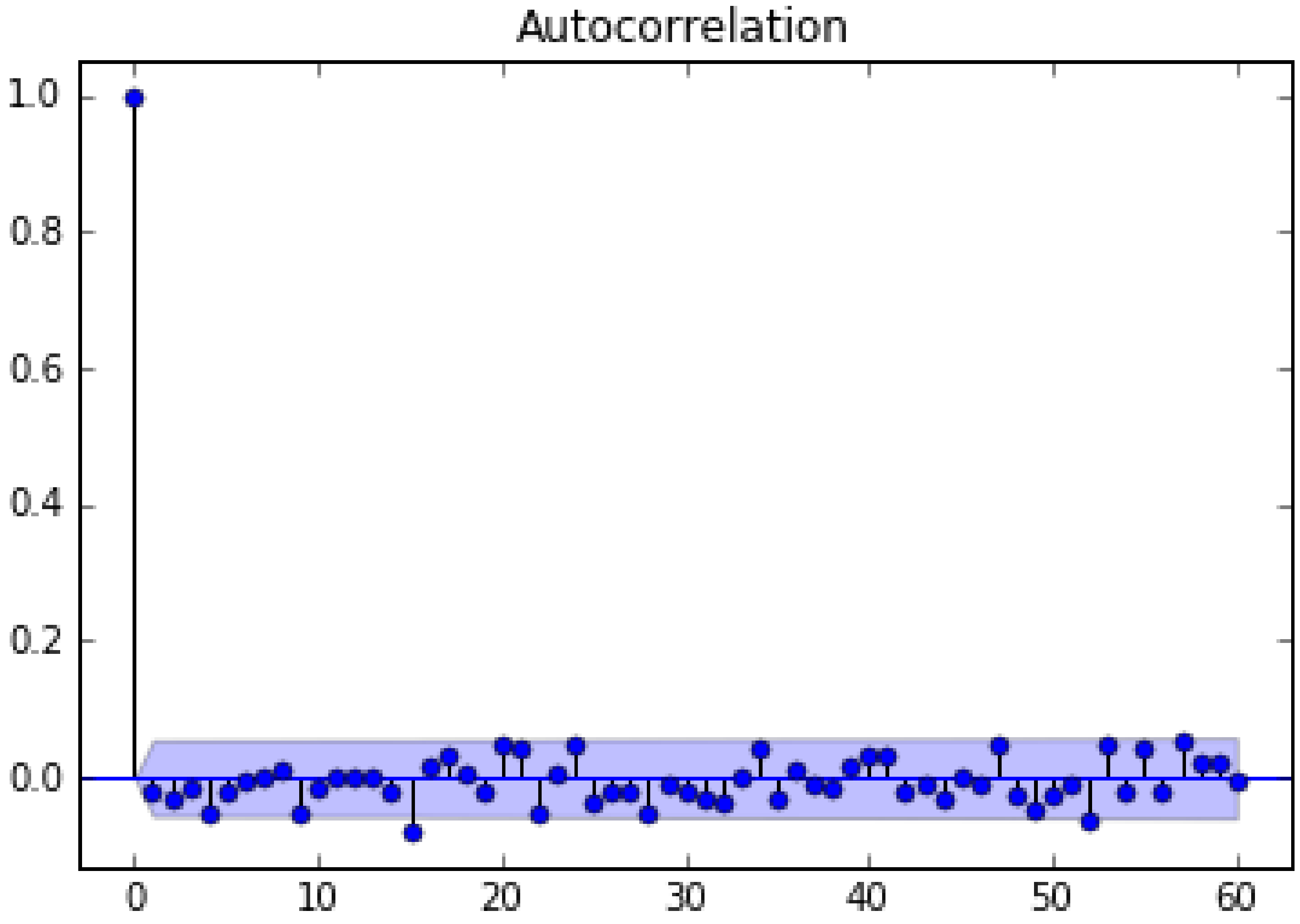
Autocorrelation of White Noise

```
plot_acf(noise, lags=50)
```



Stock Market Returns: Close to White Noise

- Autocorrelation Function for the S&P500



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

# Simulate white noise returns
returns = np.random.normal(loc=0.02, scale=0.05, size=1000)

# Print out the mean and standard deviation of returns
mean = np.mean(returns)
std = np.std(returns)
print("The mean is %5.3f and the standard deviation is %5.3f" %(mean,std))

# Plot returns series
plt.plot(returns)
plt.show()

# Plot autocorrelation function of white noise returns
plot_acf(returns, lags=20)
plt.show()
```

Notice that for a white noise time series, all the autocorrelations are close to zero, so the past will not help you forecast the future.

Let's practice!

TIME SERIES ANALYSIS IN PYTHON

Random Walk

TIME SERIES ANALYSIS IN PYTHON

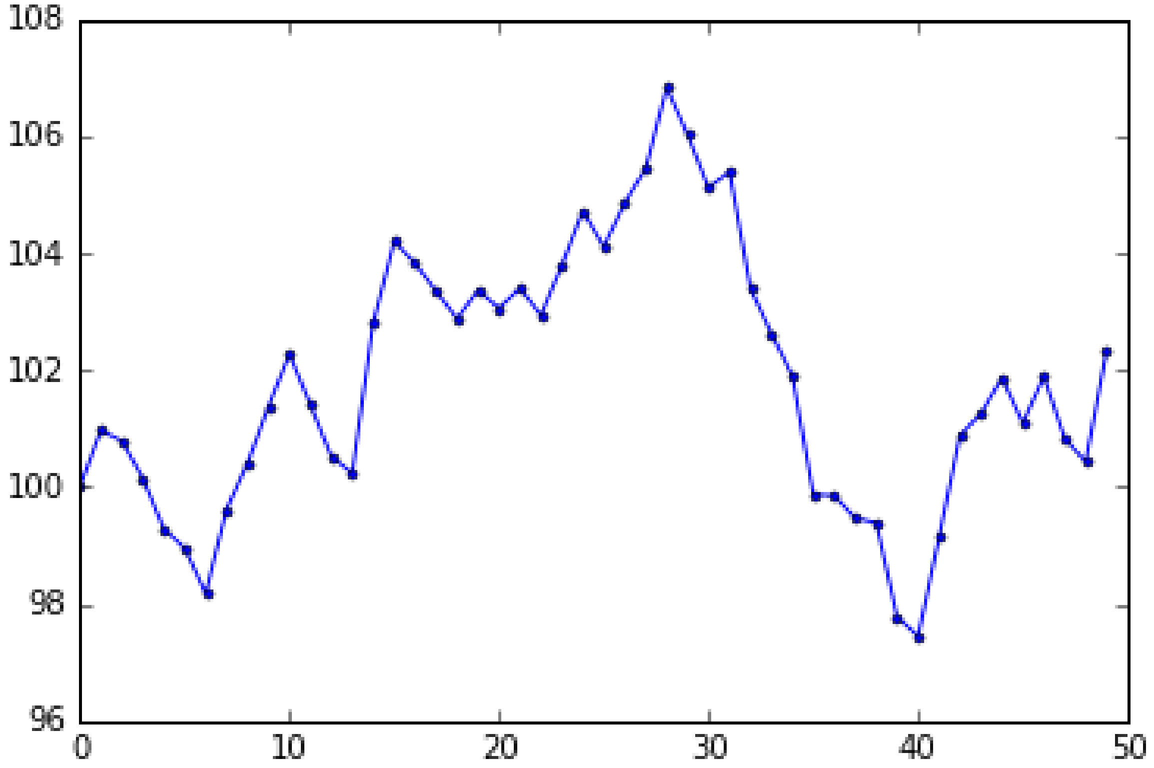
Rob Reider
Adjunct Professor, NYU-Courant
Consultant, Quantopian

What is a Random Walk?

- Today's Price = Yesterday's Price + Noise

$$P_t \quad = \quad P_{t-1} \quad + \quad \epsilon_t$$

- Plot of simulated data



What is a Random Walk?

- Today's Price = Yesterday's Price + Noise

$$P_t = P_{t-1} + \epsilon_t$$

- Change in price is white noise

$$P_t - P_{t-1} = \epsilon_t$$

- Can't forecast a random walk
- Best forecast for tomorrow's price is today's price

What is a Random Walk?

- Today's Price = Yesterday's Price + Noise

$$P_t = P_{t-1} + \epsilon_t$$

- Random walk with drift:

$$P_t = \mu + P_{t-1} + \epsilon_t$$

- Change in price is white noise with non-zero mean:

$$P_t - P_{t-1} = \mu + \epsilon_t$$

Statistical Test for Random Walk

- Random walk with drift

$$P_t = \mu + P_{t-1} + \epsilon_t$$

- Regression test for random walk

Regress current prices on lag prices

$$P_t = \alpha + \beta\, P_{t-1} + \epsilon_t$$

- Test: $H_0 : \beta = 1$ (random walk) $H_1 : \beta < 1$ (not random walk)

If the slope coefficient, Beta, is not significantly different from 1, then we cannot reject the null hypothesis that the series is a random walk, if the slope coefficient is significantly less than 1, then we can rejects the null hypothesis that the series is a random walk.

Statistical Test for Random Walk

- Regression test for random walk

Regress the difference in price on the lag price.

$$P_t = \alpha + \beta P_{t-1} + \epsilon_t$$

- Equivalent to

$$P_t - P_{t-1} = \alpha + \beta P_{t-1} + \epsilon_t$$

- Test: $H_0 : \beta = 0$ (random walk) $H_1 : \beta < 0$ (not random walk)

Instead of testing if the slope coefficient is less than 1 now we're testing if it's 0

Statistical Test for Random Walk

- Regression test for random walk

$$P_t - P_{t-1} = \alpha + \beta P_{t-1} + \epsilon_t$$

- Test: $H_0 : \beta = 0$ (random walk) $H_1 : \beta < 0$ (not random walk)
- This test is called the **Dickey-Fuller** test
- If you add more lagged changes on the right hand side, it's the **Augmented Dickey-Fuller** test

ADF Test in Python

- Import module from statsmodels

```
from statsmodels.tsa.stattools import adfuller
```

- Run Augmented Dickey-Test

```
adfuller(x)
```

Example: Is the S&P500 a Random Walk?

```
# Run Augmented Dickey-Fuller Test on SPX data
results = adfuller(df['SPX'])
```

```
# Print p-value
print(results[1])
```

```
0.782253808587
```

```
# Print full results
print(results)
```

```
(-0.917204903311227869,
 0.78225380858668414,
 0,
 1257,
 {'1%': -3.4355629707955395,
  '10%': -2.567995644141416,
  '5%': -2.8638420633876671},
```

If the p-value is less than 5% then we can reject the null hypotheses that the series is a random walk.with 95% confidence.

In this case the p-value is much higher than 0.05, it's 0.78 therefore we cannot reject the null hypothesuis that the S&P 500 is a random walk.

```
# Generate 500 random steps with mean=0 and standard deviation=1
steps = np.random.normal(loc=0, scale=1, size=500)

# Set first element to 0 so that the first price will be the starting stock price
steps[0]=0

# Simulate stock prices, P with a starting price of 100
P = 100 + np.cumsum(steps)

# Plot the simulated stock prices
plt.plot(P)
plt.title("Simulated Random Walk")
plt.show()
```

The simulated price series you plotted should closely resemble a random walk.

```
# Import the adfuller module from statsmodels
from statsmodels.tsa.stattools import adfuller

# Run the ADF test on the price series and print out the results
results = adfuller(AMZN['Adj Close'])
print(results)

# Just print out the p-value
print('The p-value of the test on prices is: ' + str(results[1]))
```

According to this test, we cannot reject the hypothesis that Amazon prices follow a random walk.

Let's practice!

TIME SERIES ANALYSIS IN PYTHON

```
# Generate 500 random steps
steps = np.random.normal(loc=0.001, scale=0.01, size=500) +
1

# Set first element to 1
steps[0]=1

# Simulate the stock price, P, by taking the cumulative product
P = 100 * np.cumprod(steps)

# Plot the simulated stock prices
plt.plot(P)
plt.title("Simulated Random Walk with Drift")
plt.show()
```

This simulated price series you plotted should closely resemble a random walk for a high flying stock

```
# Import the adfuller module from statsmodels
from statsmodels.tsa.stattools import adfuller

# Create a DataFrame of AMZN returns
AMZN_ret = AMZN.pct_change()

# Eliminate the NaN in the first row of returns
AMZN_ret = AMZN_ret.dropna()

# Run the ADF test on the return series and print out the p-
value
results = adfuller(AMZN_ret['Adj Close'])
print('The p-value of the test on returns is: ' + str(results[1]))
```

The p-value is extremely small, so we can easily reject the hypothesis that returns are a random walk at all levels of significance.

Stationarity

TIME SERIES ANALYSIS IN PYTHON

Rob Reider
Adjunct Professor, NYU-Courant
Consultant, Quantopian

What is Stationarity?

- **Strong stationarity**: entire distribution of data is time-invariant
- **Weak stationarity**: mean, variance and autocorrelation are time-invariant (i.e., for autocorrelation, corr$(X_t, X_{t-\tau})$ is only a function of τ) and not a function of time.

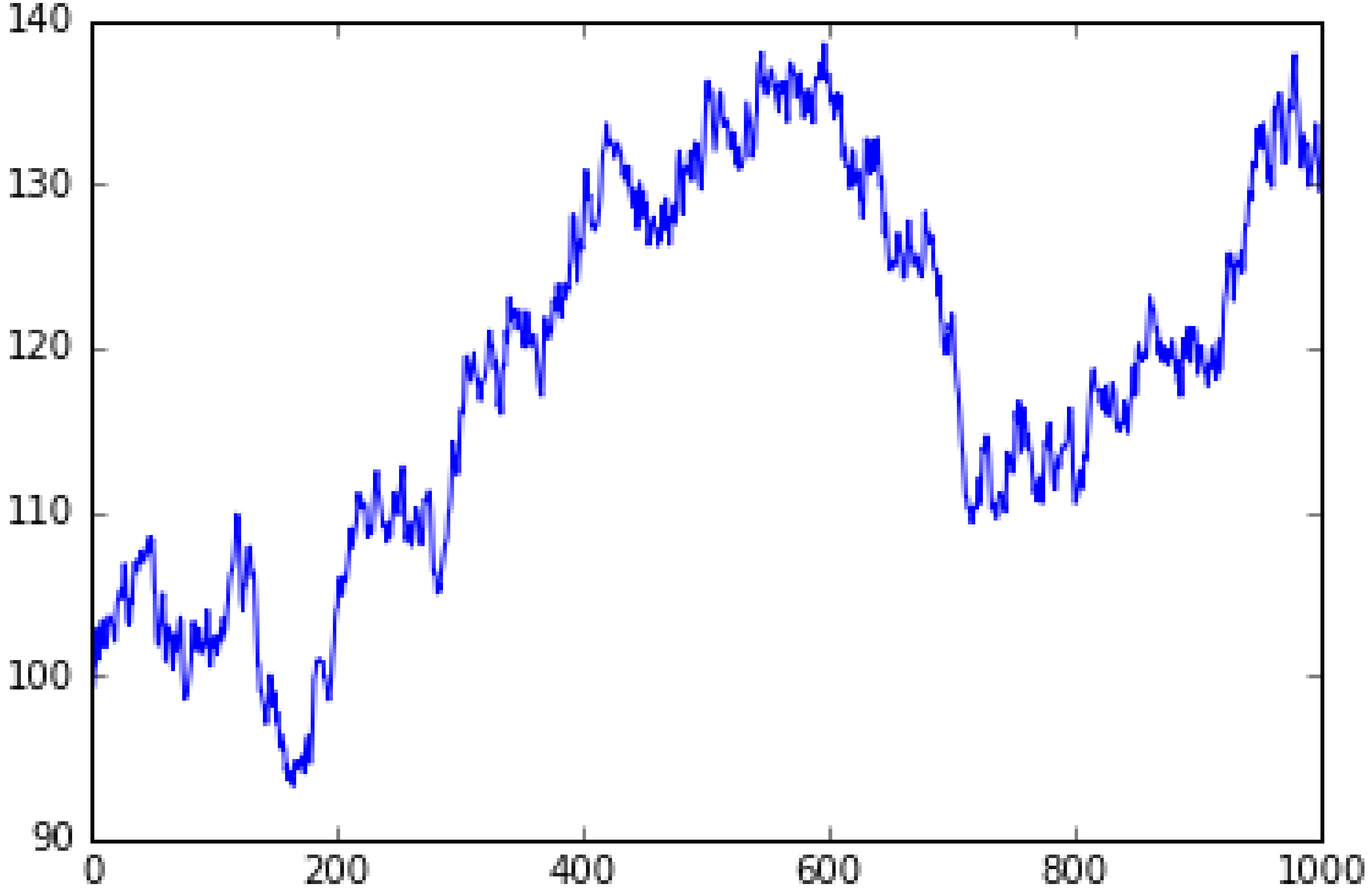
Why Do We Care?

If a process is not stationary then it becomes difficult to model.

- If parameters vary with time, too many parameters to estimate
- Can only estimate a parsimonious model with a few parameters

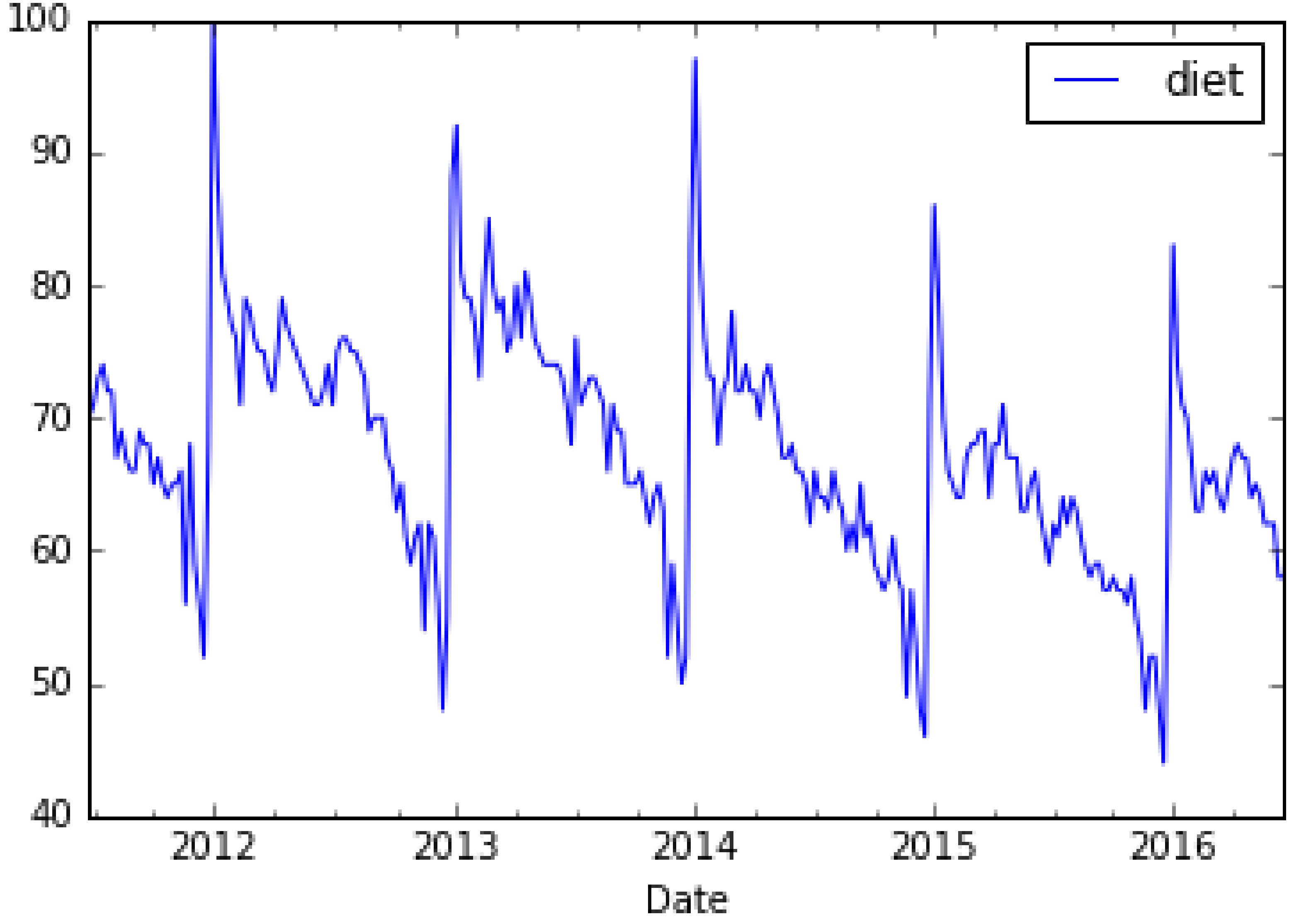
Examples of Nonstationary Series

- Random Walk



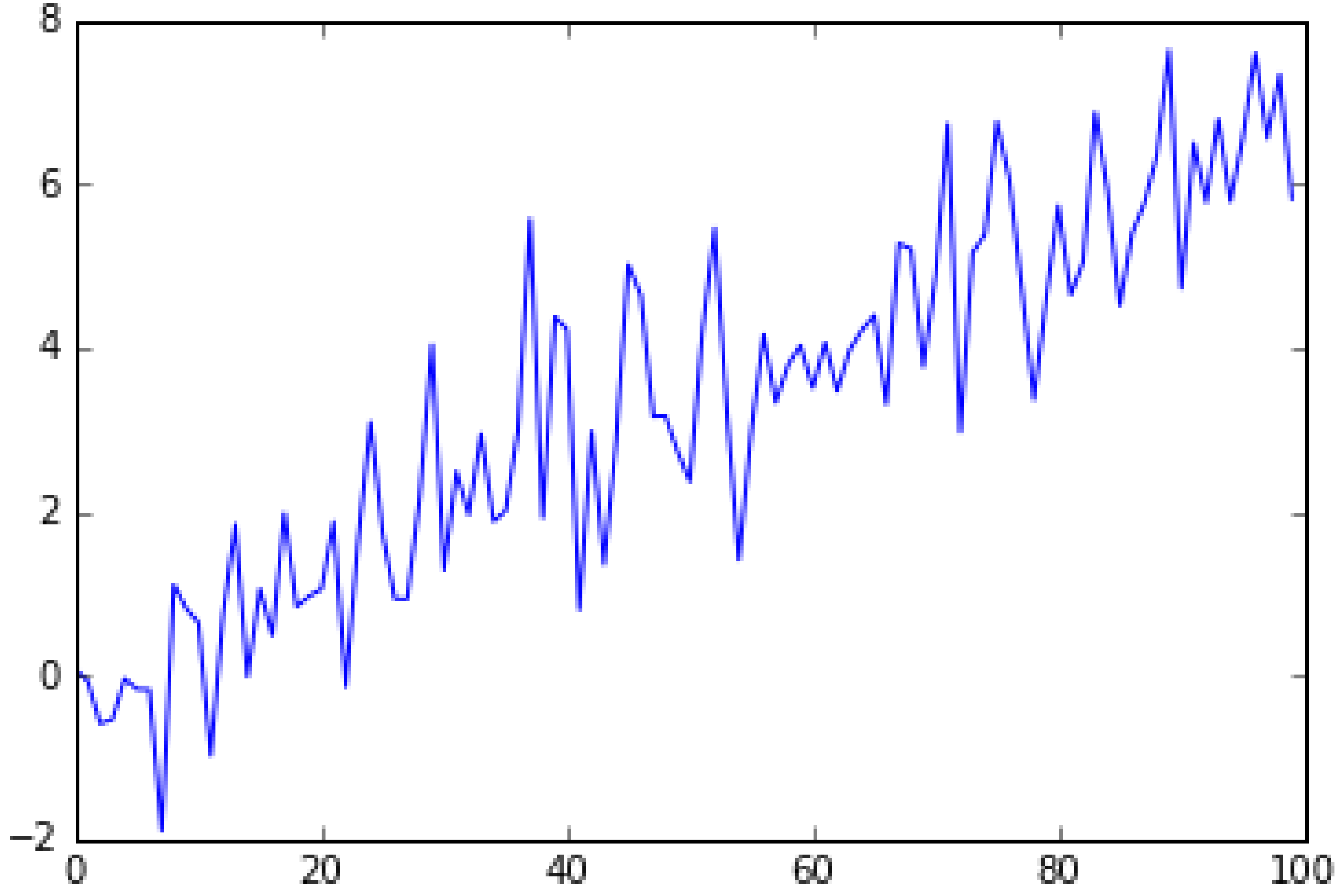
Examples of Nonstationary Series

- Seasonality in series



Examples of Nonstationary Series

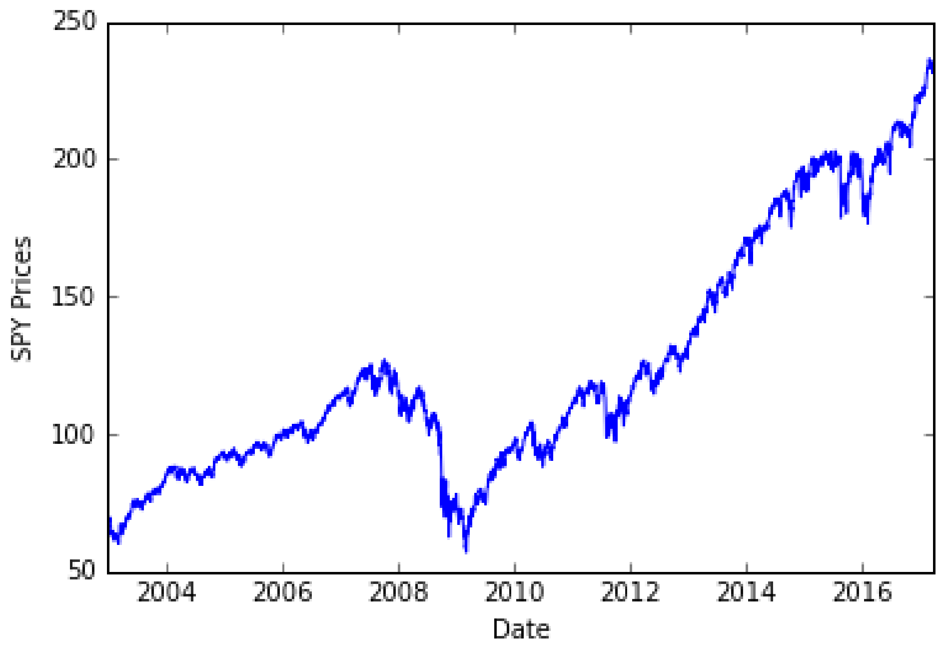
- Change in Mean or Standard Deviation over time



Transforming Nonstationary Series Into Stationary Series

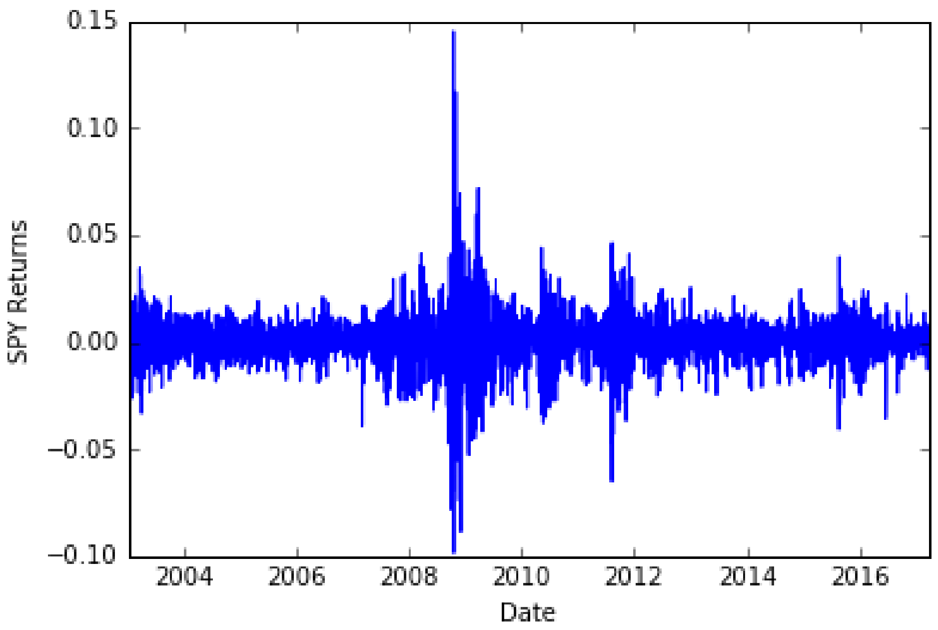
- Random Walk

```
plot.plot(SPY)
```



- First difference

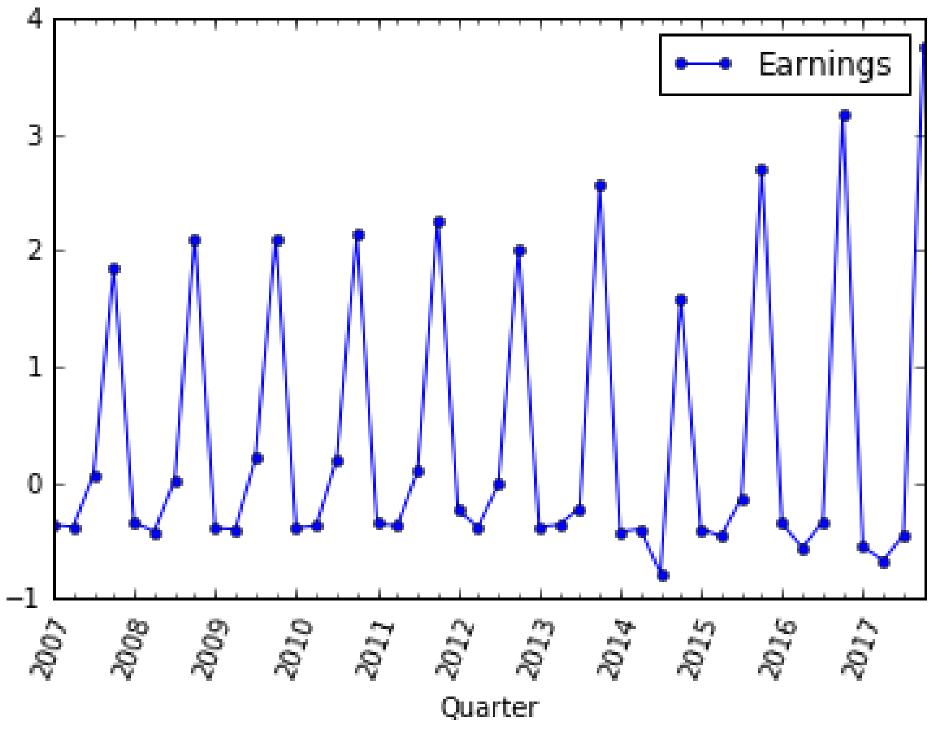
```
plot.plot(SPY.diff())
```



Transforming Nonstationary Series Into Stationary Series

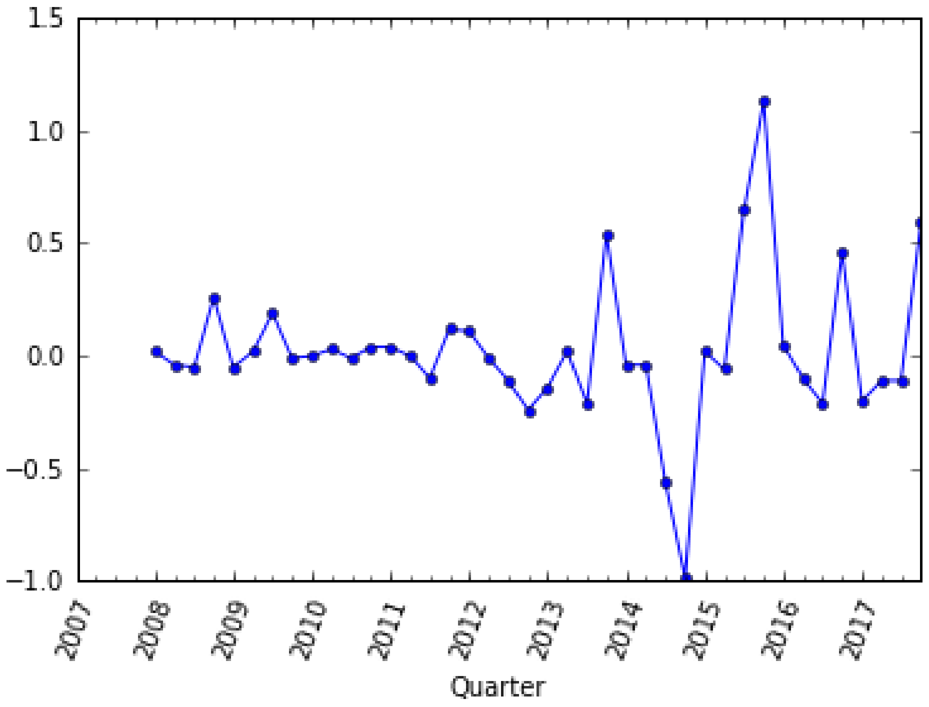
- Seasonality

```
plot.plot(HRB)
```



- Seasonal difference

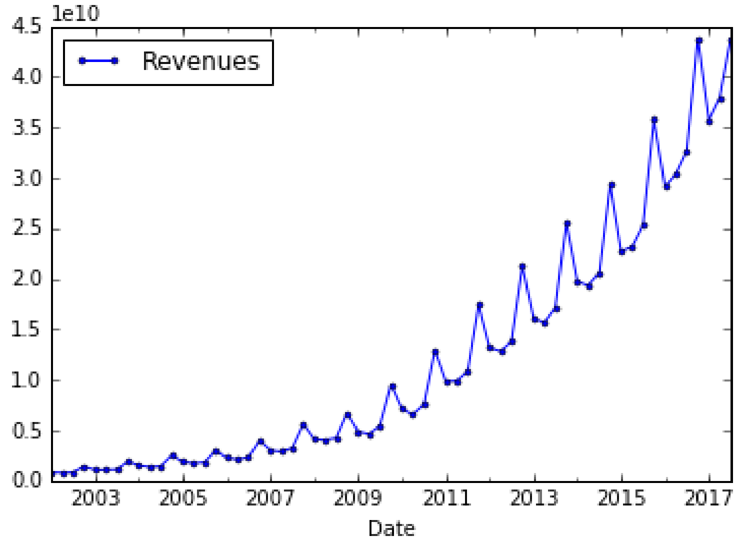
```
plot.plot(HRB.diff(4))
```



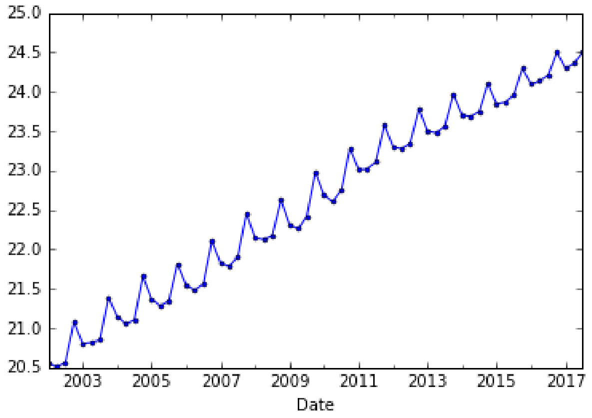
Transforming Nonstationary Series Into Stationary Series

- AMZN Quarterly Revenues

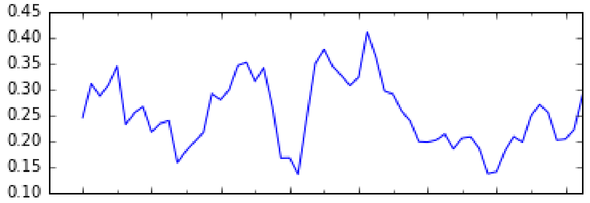
```
plt.plot(AMZN)
```




```
# Log of AMZN Revenues
plt.plot(np.log(AMZN))
```




```
# Log, then seasonal difference
plt.plot(np.log(AMZN).diff(4))
```



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

# Seasonally adjust quarterly earnings
HRBsa = HRB.diff(4)

# Print the first 10 rows of the seasonally adjusted series
print(HRBsa.head(10))

# Drop the NaN data in the first four rows
HRBsa = HRBsa.dropna()

# Plot the autocorrelation function of the seasonally adjusted series
plot_acf(HRBsa)
plt.show()
```

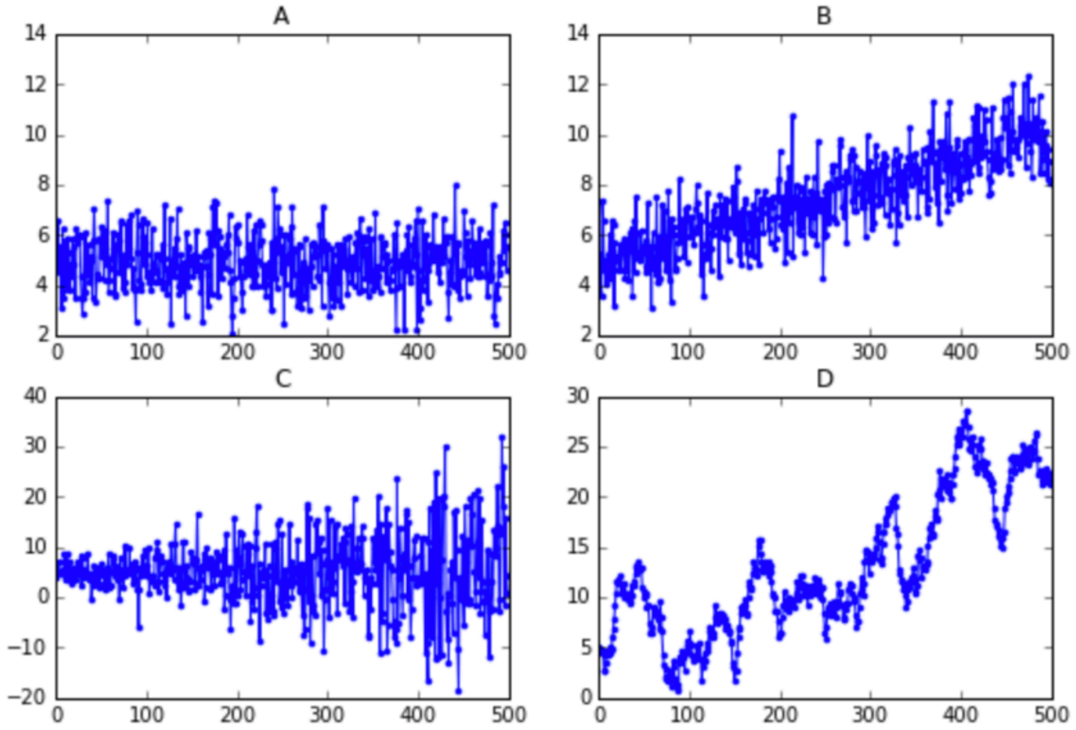
By seasonally adjusting the series, we eliminated the seasonal pattern in the autocorrelation function

Let's practice!

TIME SERIES ANALYSIS IN PYTHON

Is it Stationary?

Here are four time series plots:



Which one is stationary?

Answer the question 50 XP

Possible Answers

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

Take Hint (-15xp)

Submit Answer