Autocorrelation Function

TIME SERIES ANALYSIS IN PYTHON

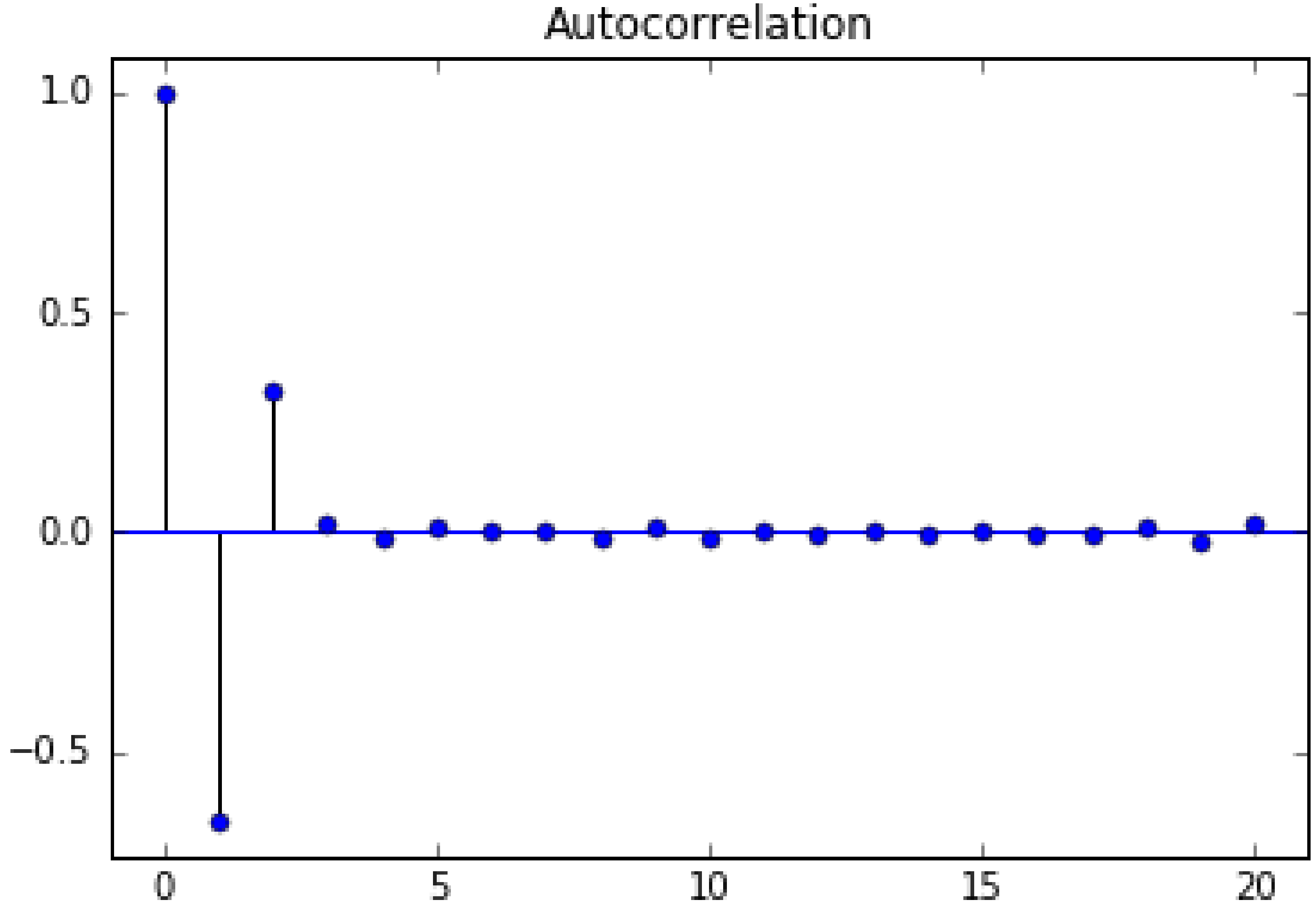
Rob Reider
Adjunct Professor, NYU-Courant
Consultant, Quantopian

Autocorrelation Function

- Autocorrelation Function (ACF): The autocorrelation as a function of the lag
- Equals one at lag-zero
- Interesting information beyond lag-one

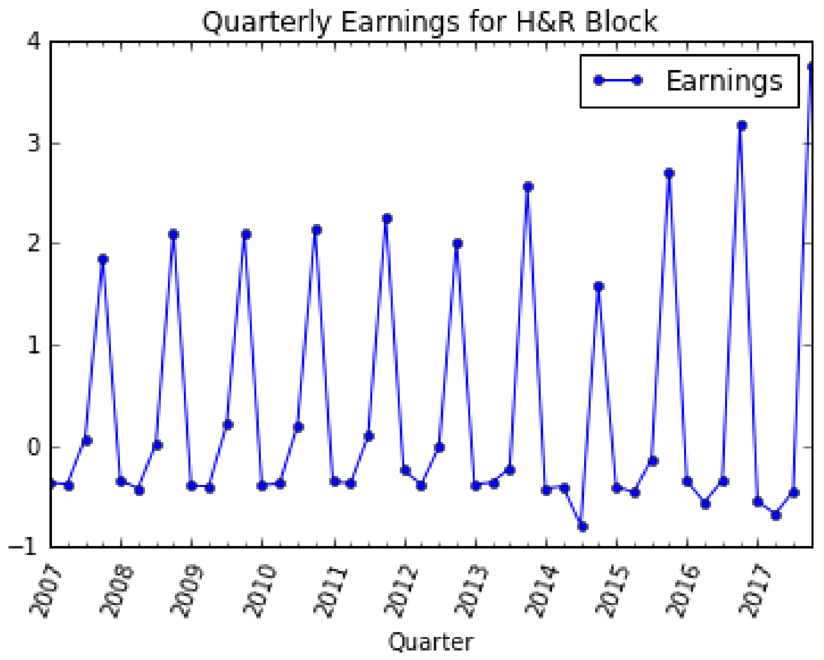
ACF Example 1: Simple Autocorrelation Function

- Can use last two values in series for forecasting

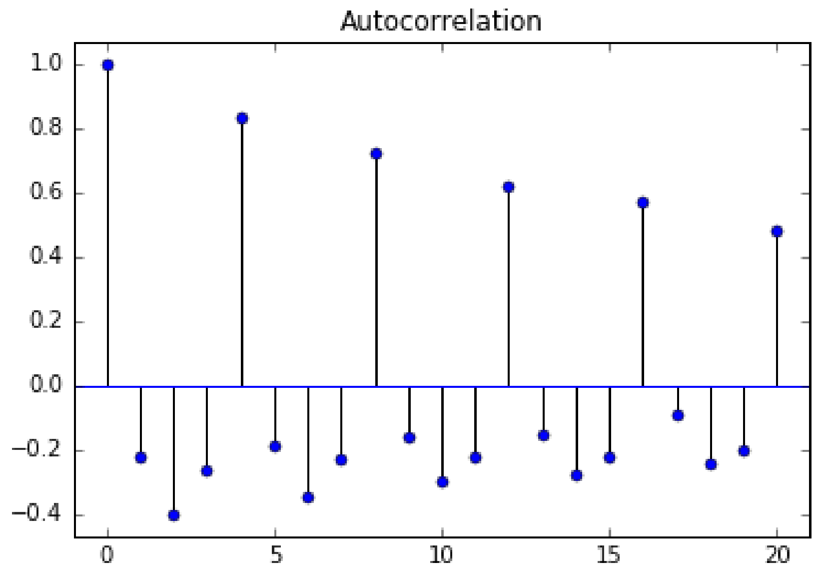


ACF Example 2: Seasonal Earnings

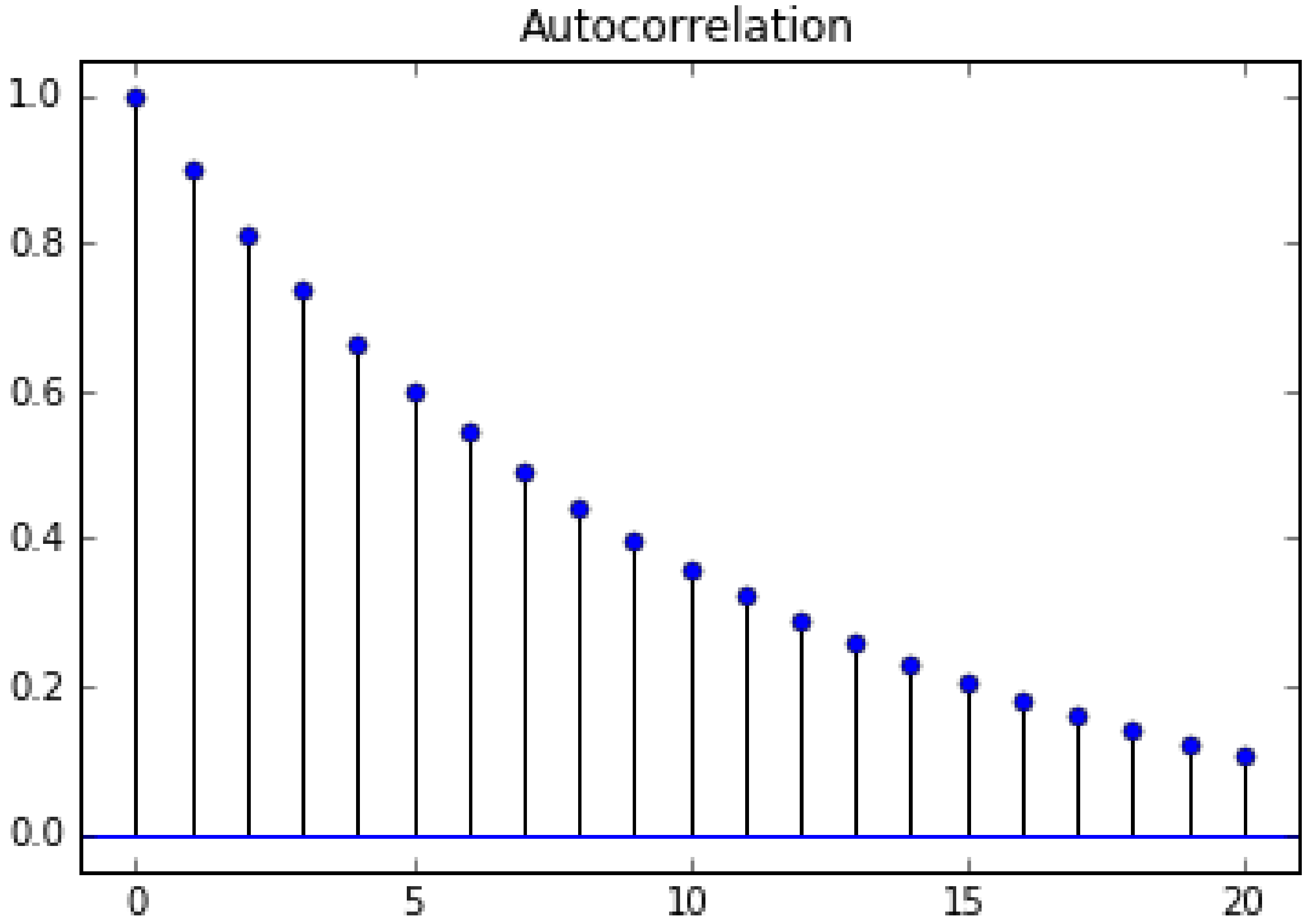
- Earnings for H&R Block



- ACF for H&R Block



ACF Example 3: Useful for Model Selection



- Model selection

Plot ACF in Python

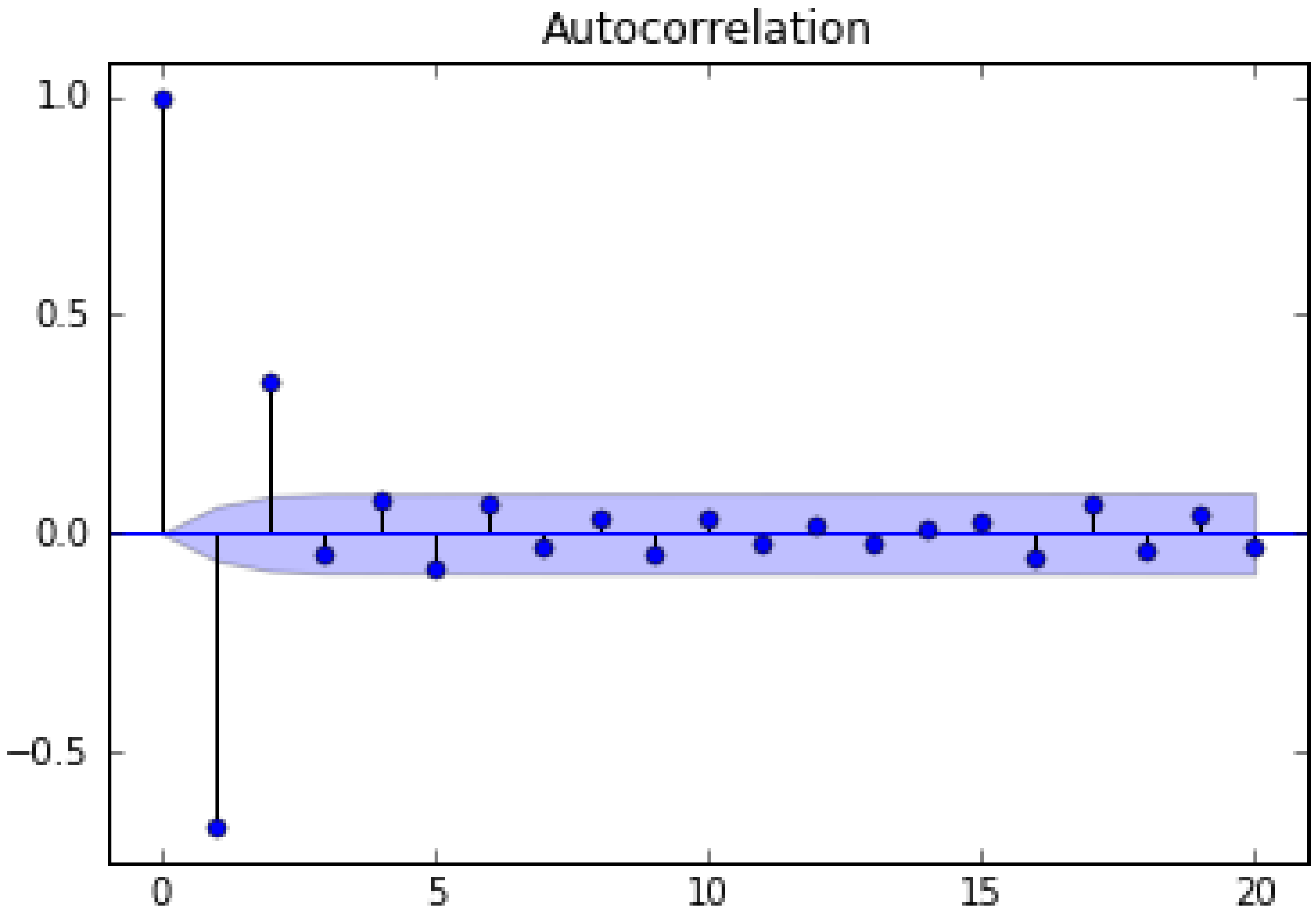
- Import module:

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

- Plot the ACF:

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

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
- 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
```

Let's practice!

TIME SERIES ANALYSIS IN PYTHON

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*

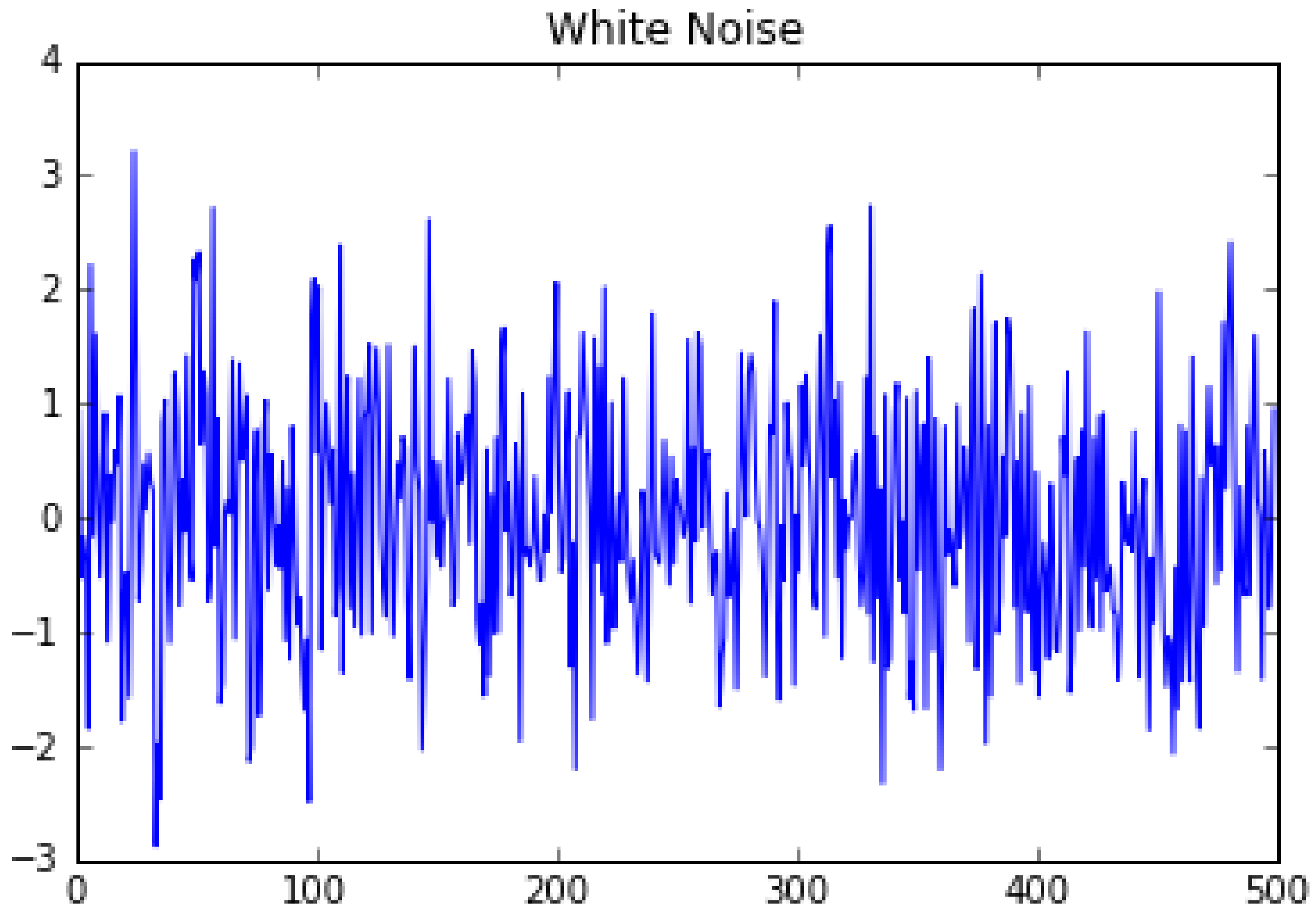
Simulating White Noise

- It's very easy to generate white noise

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

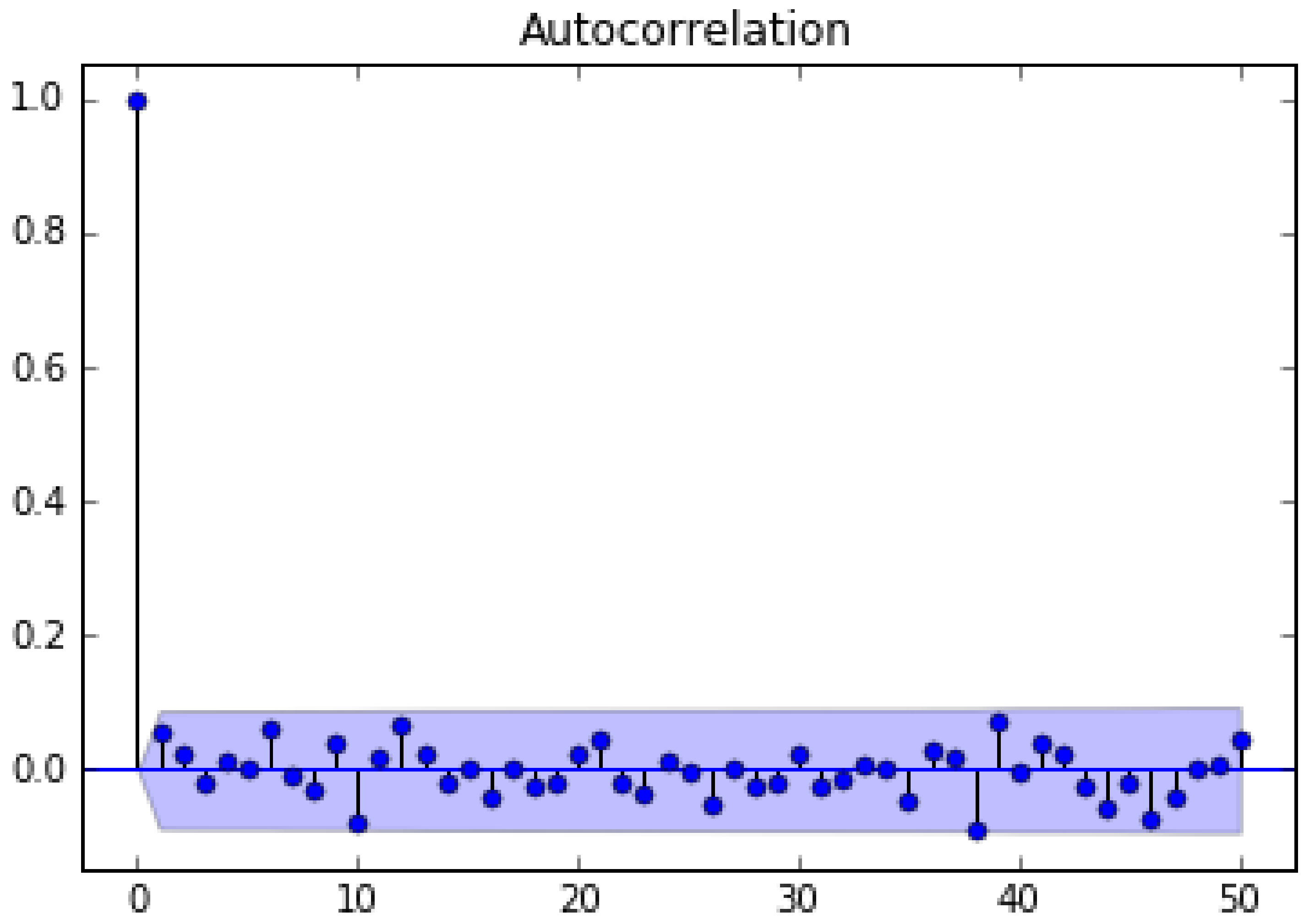
What Does White Noise Look Like?

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



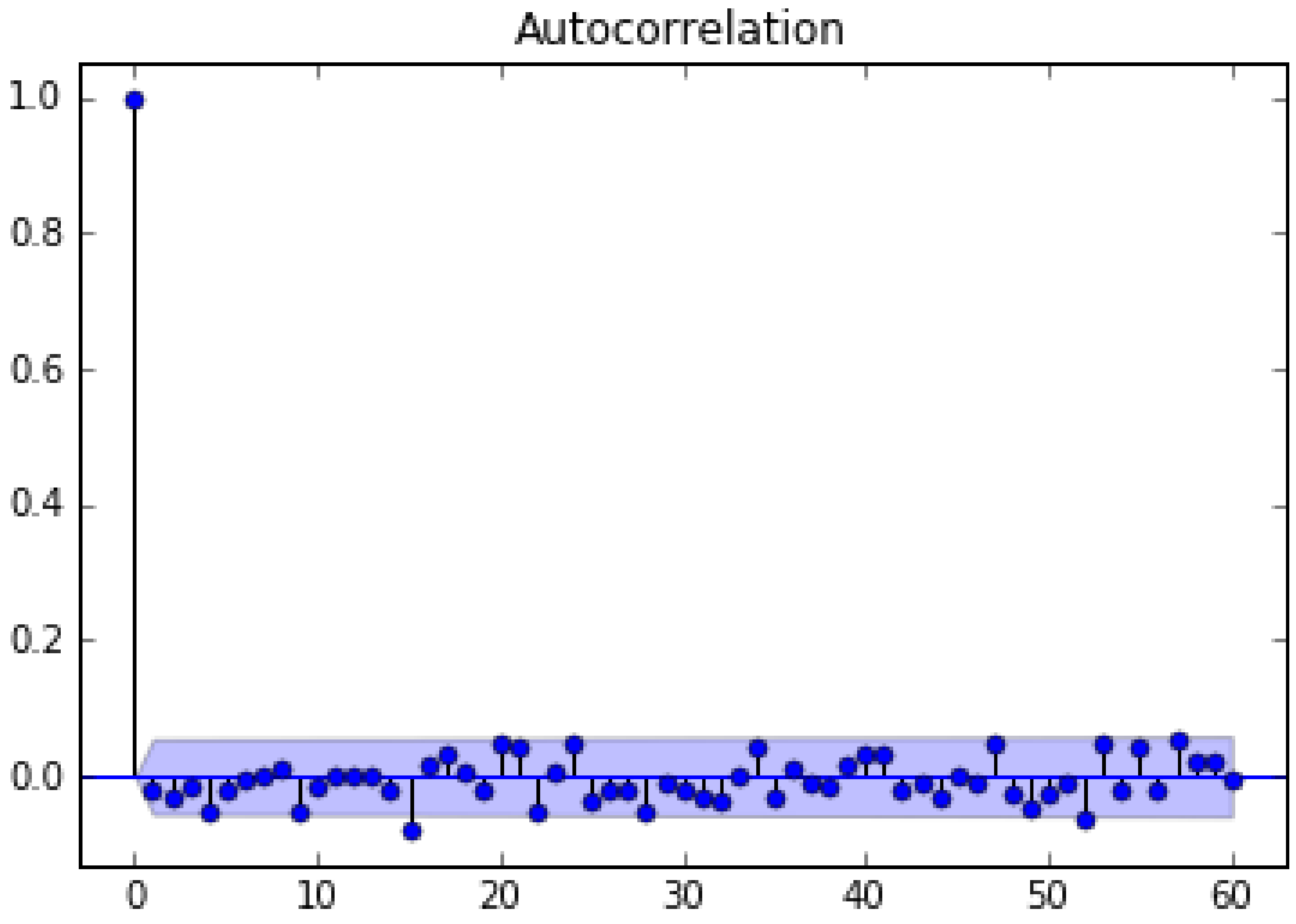
Autocorrelation of White Noise

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



Stock Market Returns: Close to White Noise

- Autocorrelation Function for the S&P500



Let's practice!

TIME SERIES ANALYSIS IN PYTHON

Random Walk

TIME SERIES ANALYSIS IN PYTHON

Rob Reider

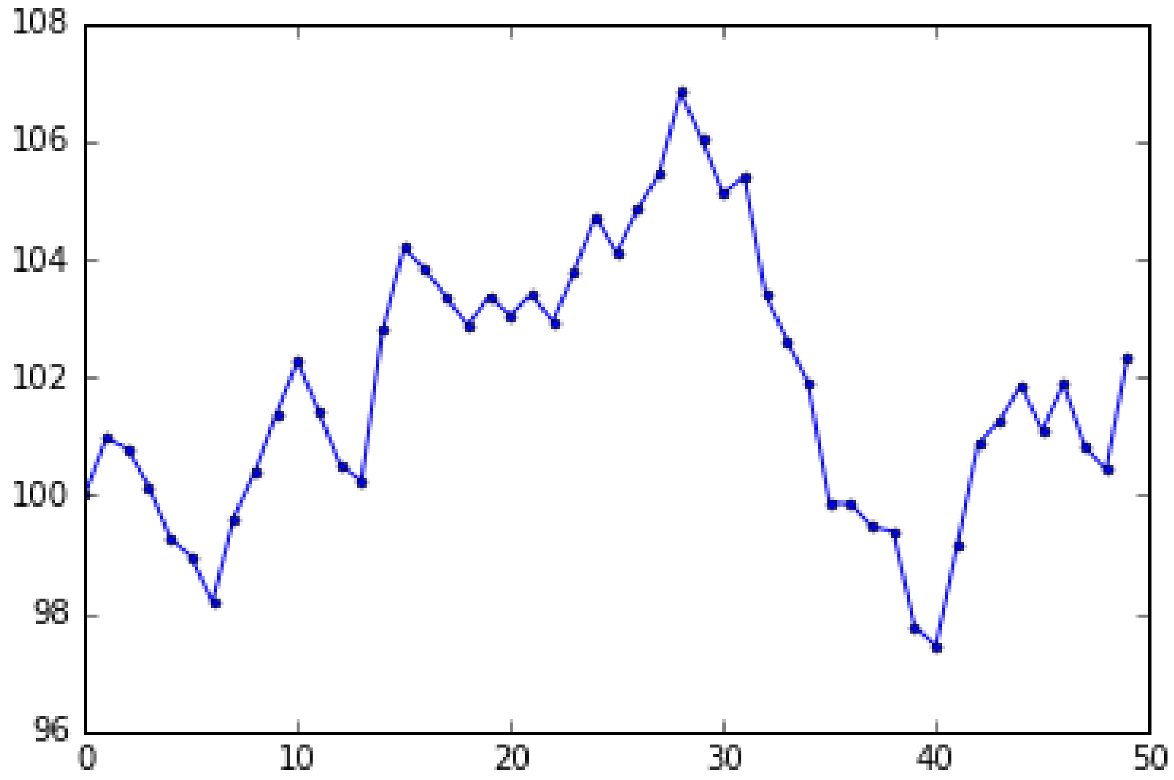
Adjunct Professor, NYU-Courant
Consultant, Quantopian

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

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

- Random walk with drift:

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

- 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

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

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

Statistical Test for Random Walk

- Regression test for random walk

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

- 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)

Statistical Test for Random Walk

- Regression test for random walk

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

- 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 adfulle
```

- 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)
```

```
(-0.91720490331127869,
 0.78225380858668414,
 0,
 1257,
 {'1%': -3.4355629707955395,
  '10%': -2.567995644141416,
  '5%': -2.8638420633876671},
 10161.888789598503)
```

Let's practice!

TIME SERIES ANALYSIS IN PYTHON

Stationarity

TIME SERIES ANALYSIS IN PYTHON

Rob Reider
Adjunct Professor, NYU-Courant
Consultant, Quantopian

What is Stationarity?

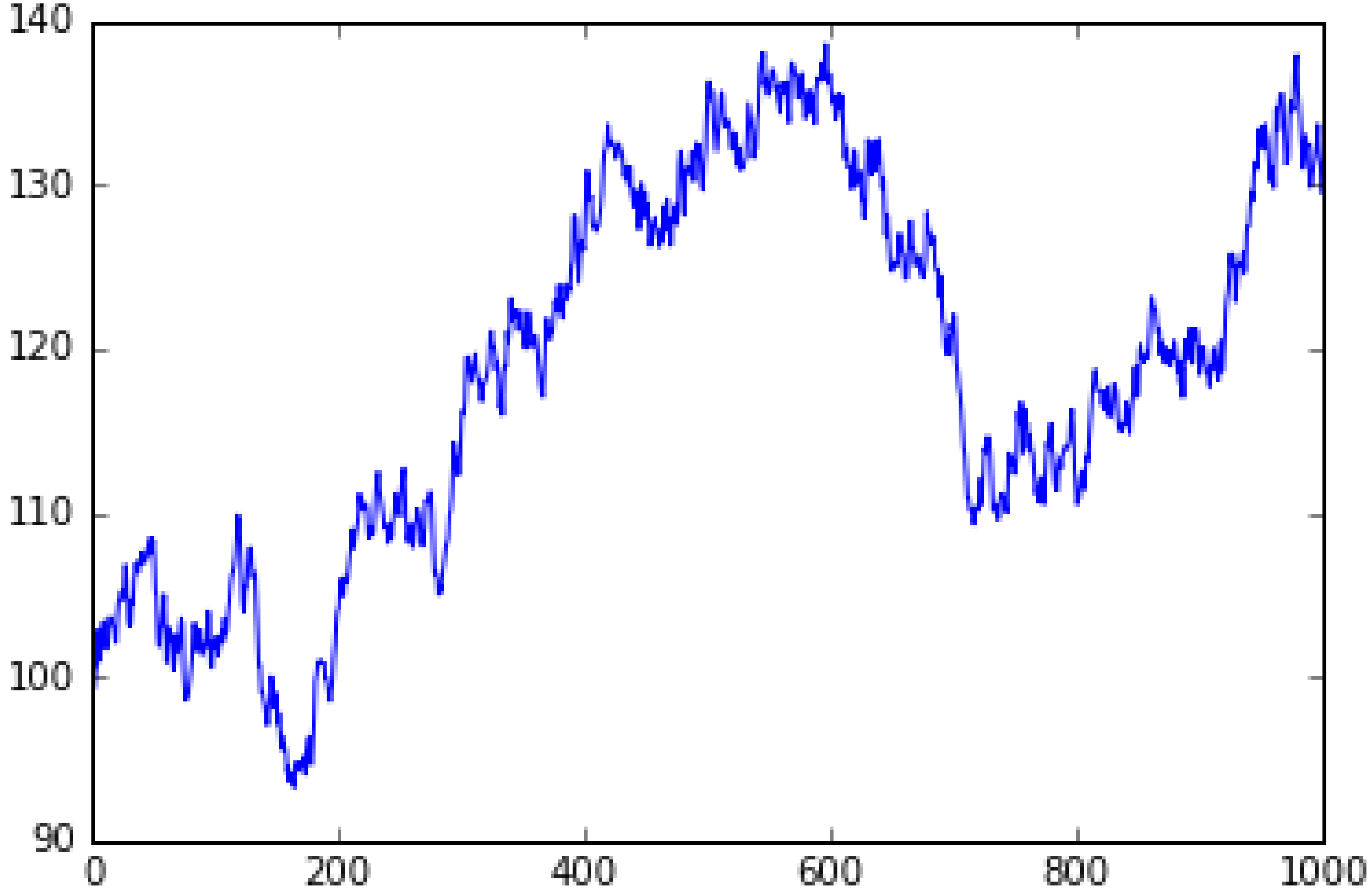
- **Strong stationarity**: entire distribution of data is time-invariant
- **Weak stationarity**: mean, variance and autocorrelation are time-invariant (i.e., for autocorrelation, corr$(X_t, X_{t-\tau})$ is only a function of τ)

Why Do We Care?

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

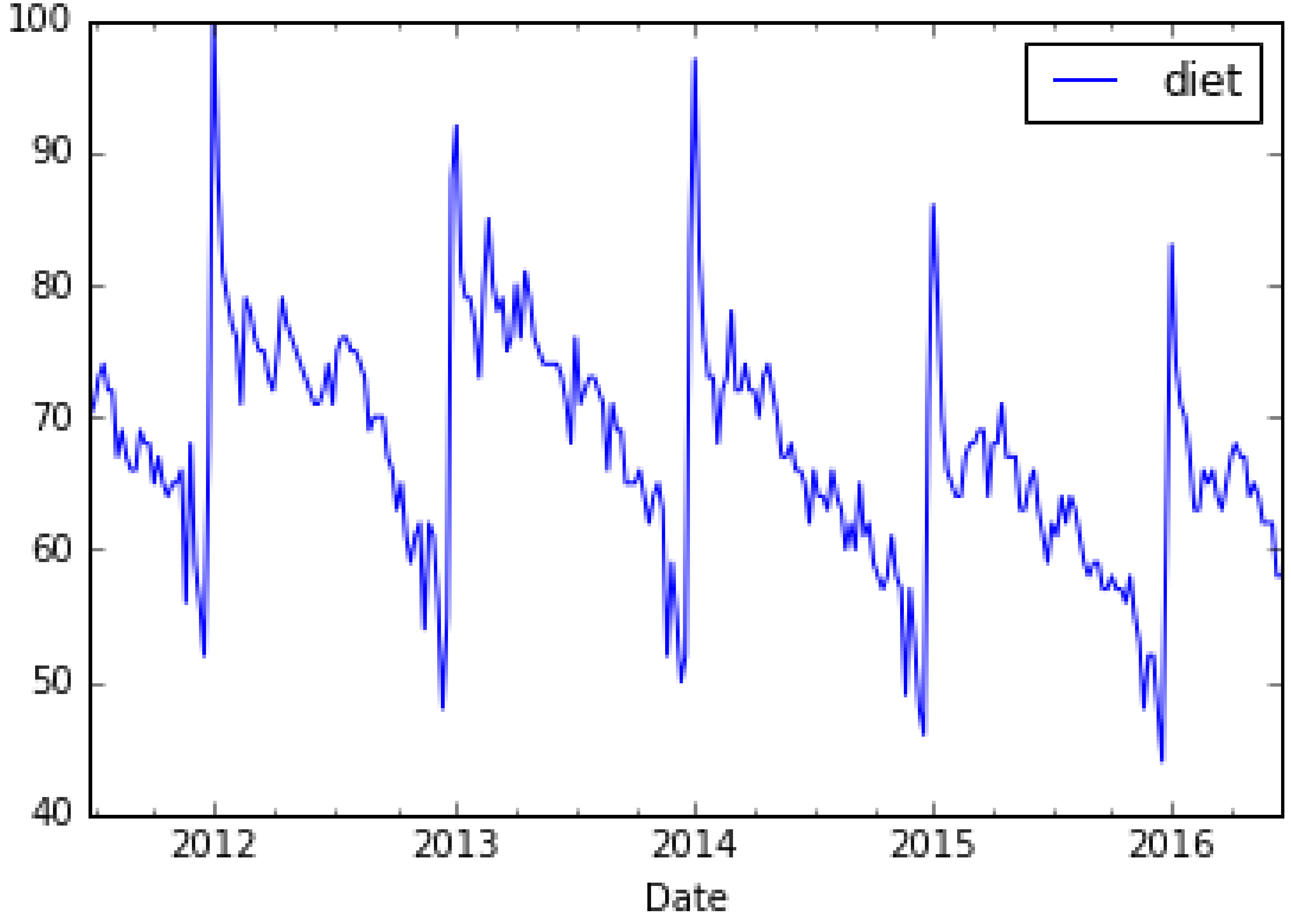
Examples of Nonstationary Series

- Random Walk



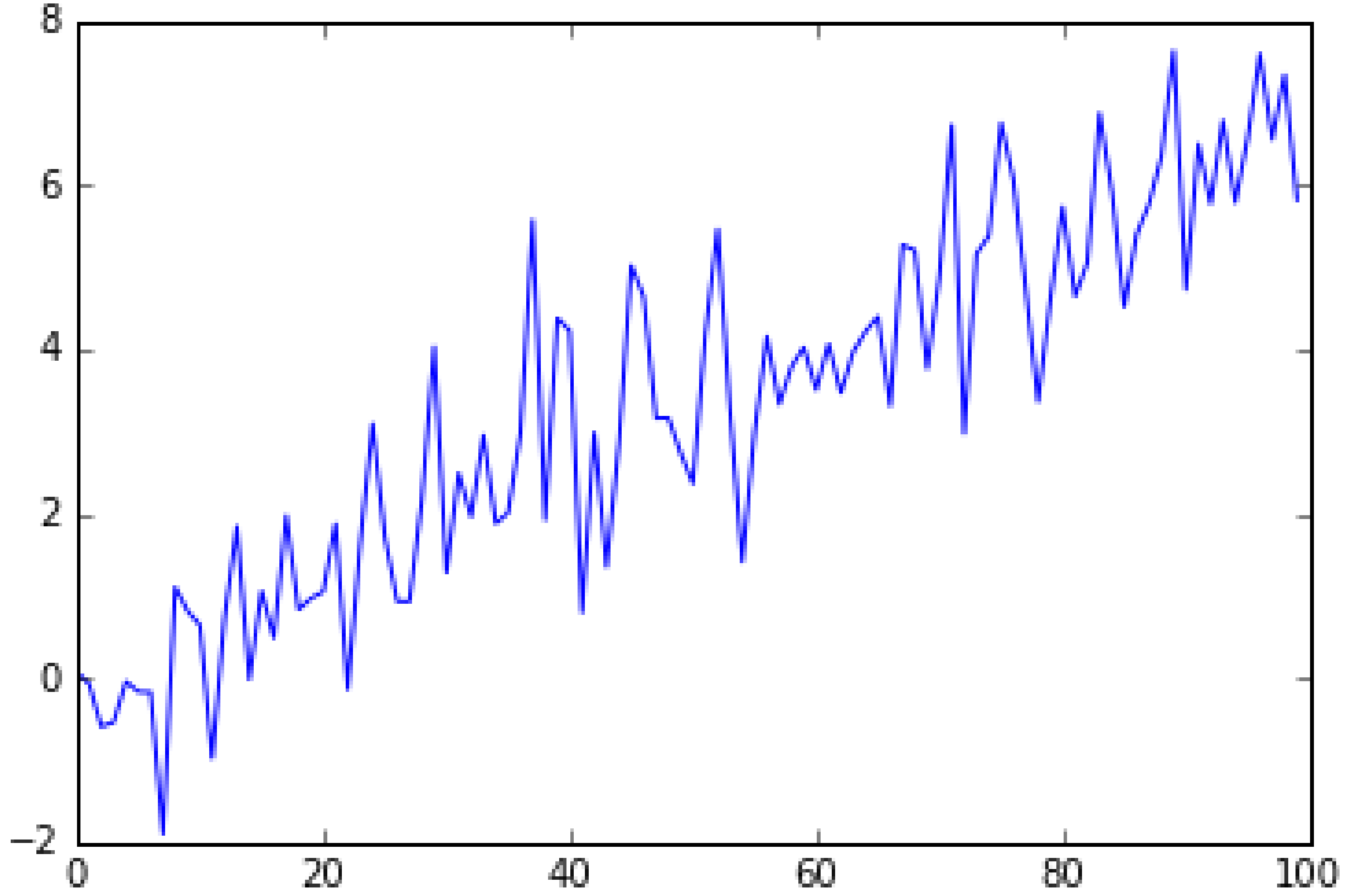
Examples of Nonstationary Series

- Seasonality in series



Examples of Nonstationary Series

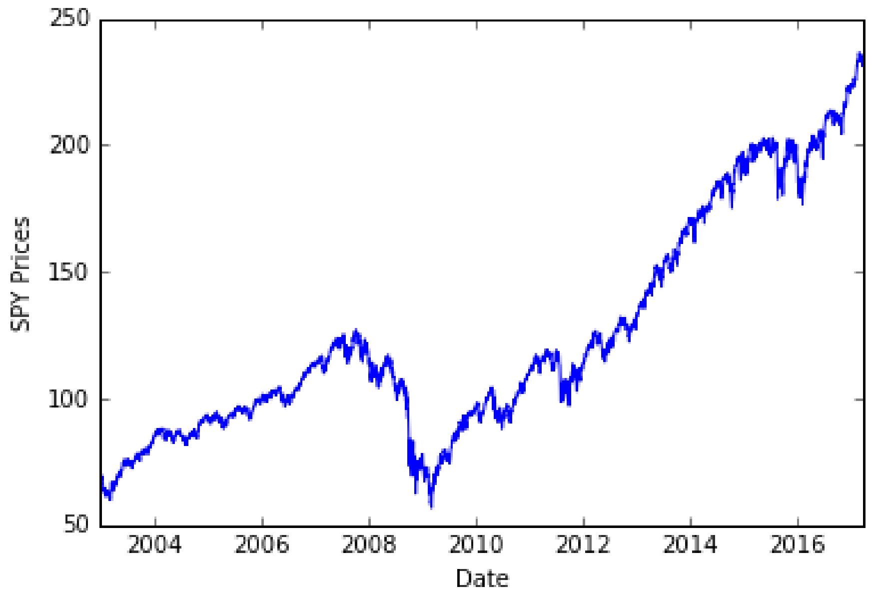
- Change in Mean or Standard Deviation over time



Transforming Nonstationary Series Into Stationary Series

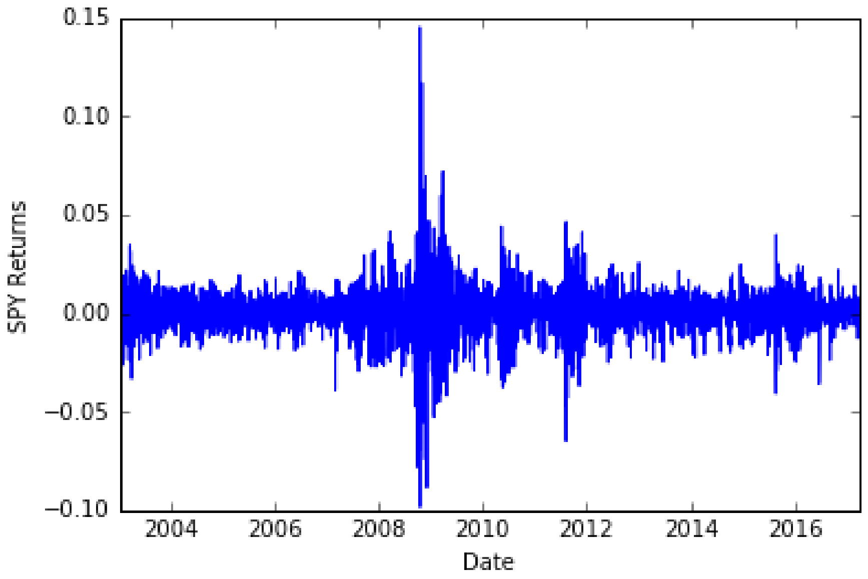
- Random Walk

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



- First difference

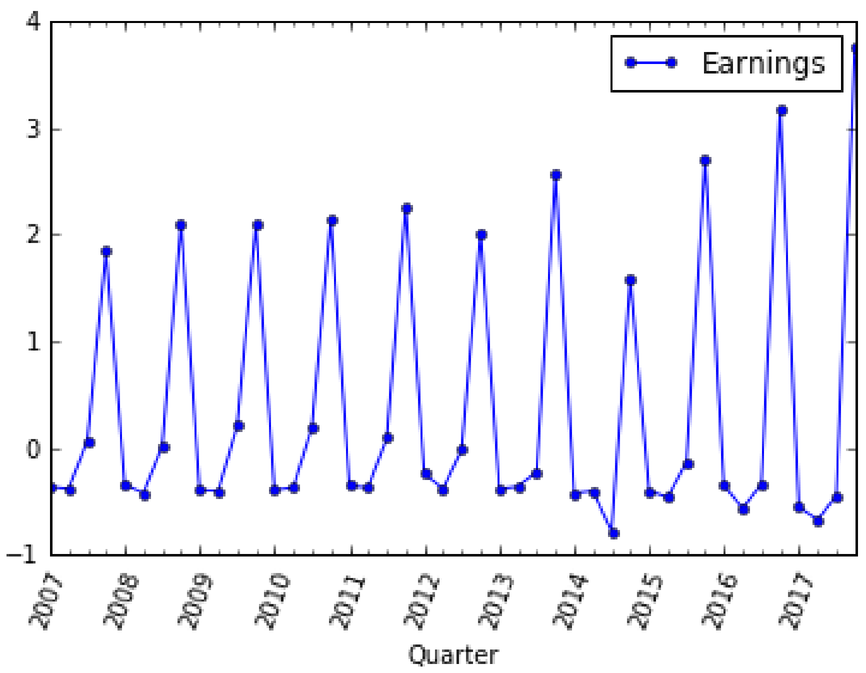
```
plot.plot(SPY.diff())
```



Transforming Nonstationary Series Into Stationary Series

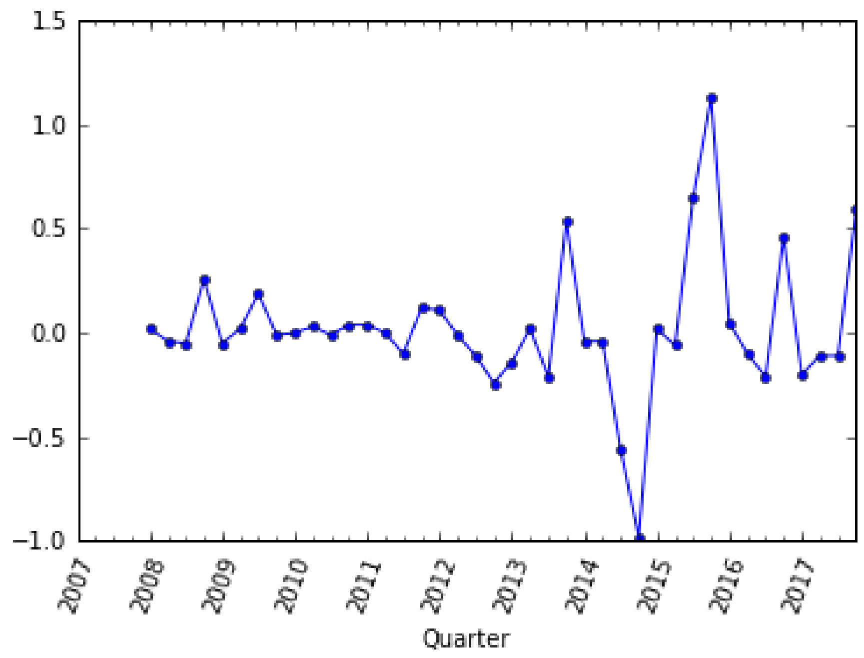
- Seasonality

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



- Seasonal difference

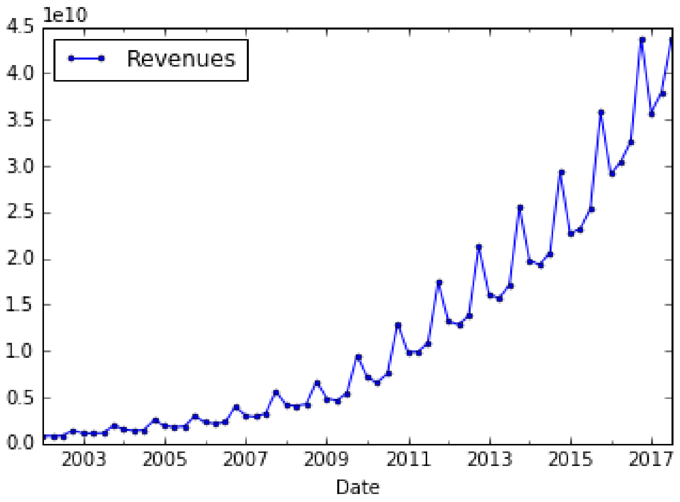
```
plot.plot(HRB.diff(4))
```



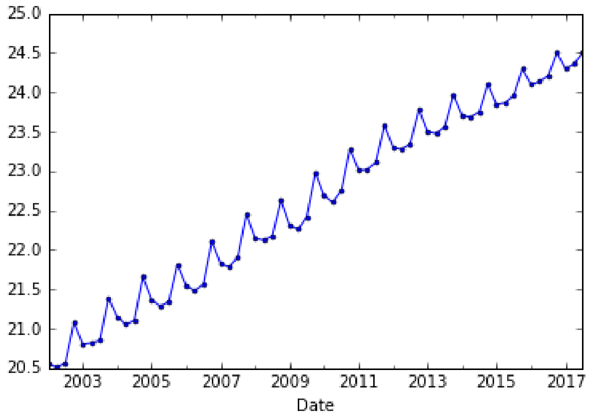
Transforming Nonstationary Series Into Stationary Series

- AMZN Quarterly Revenues

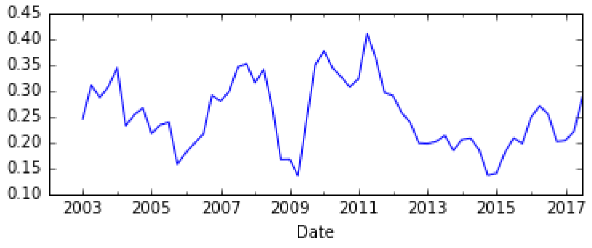
```
plt.plot(AMZN)
```




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




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



Let's practice!

TIME SERIES ANALYSIS IN PYTHON