

# Introduction to the Course

TIME SERIES ANALYSIS IN PYTHON



**Rob Reider**

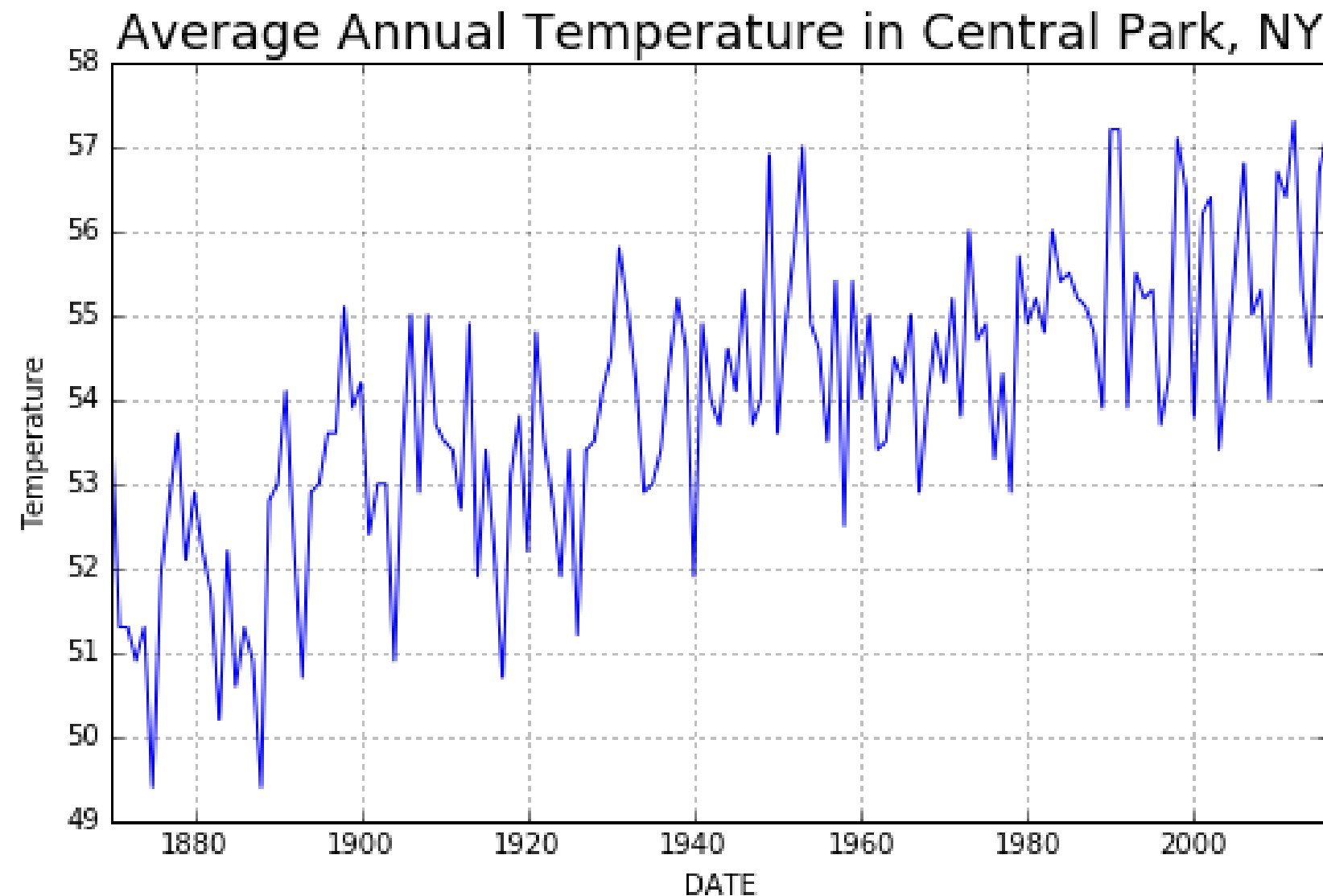
Adjunct Professor, NYU-Courant  
Consultant, Quantopian

Quantopian: Business built on python for analyzing backlog of data & quantitative data strategies.

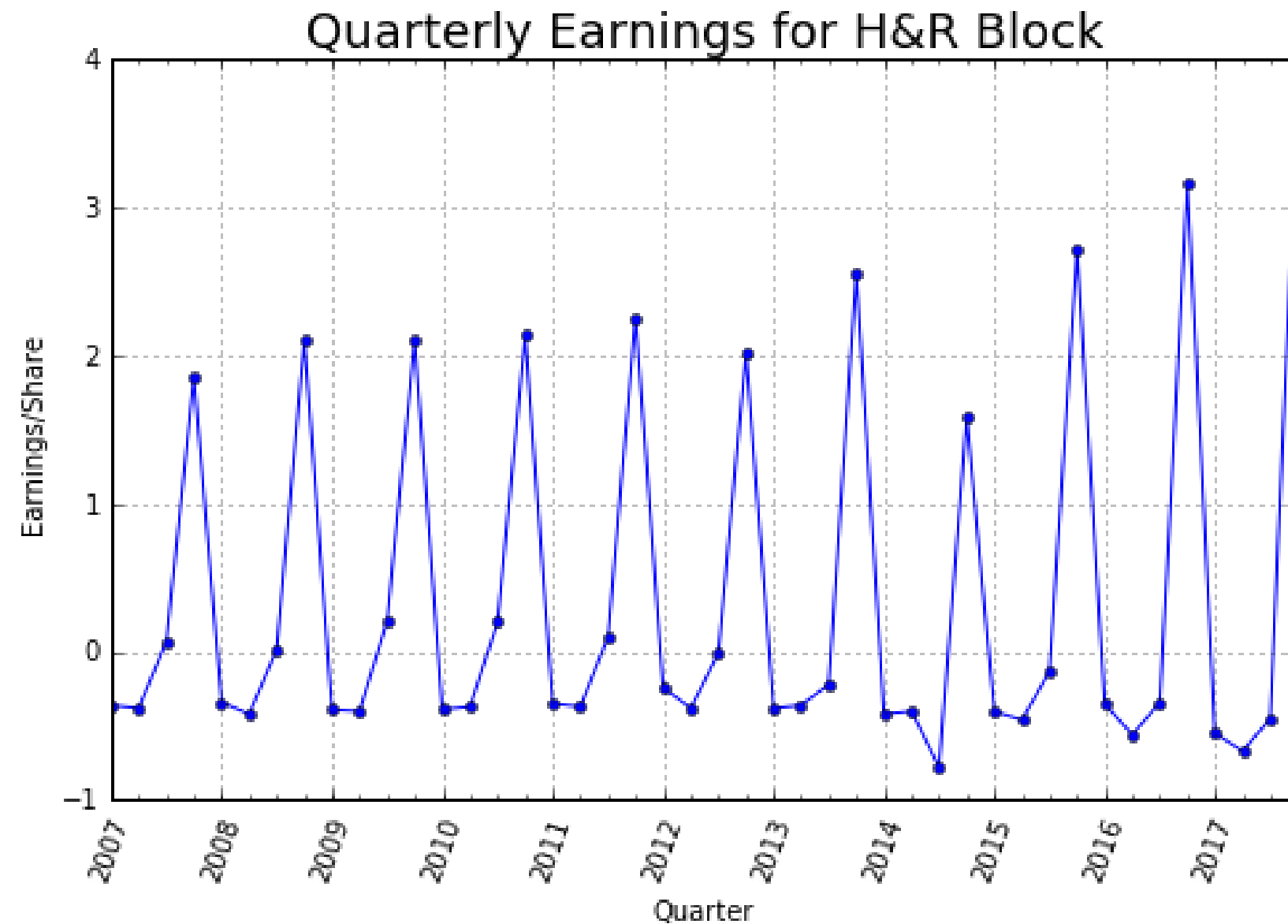
# Example of Time Series: Google Trends



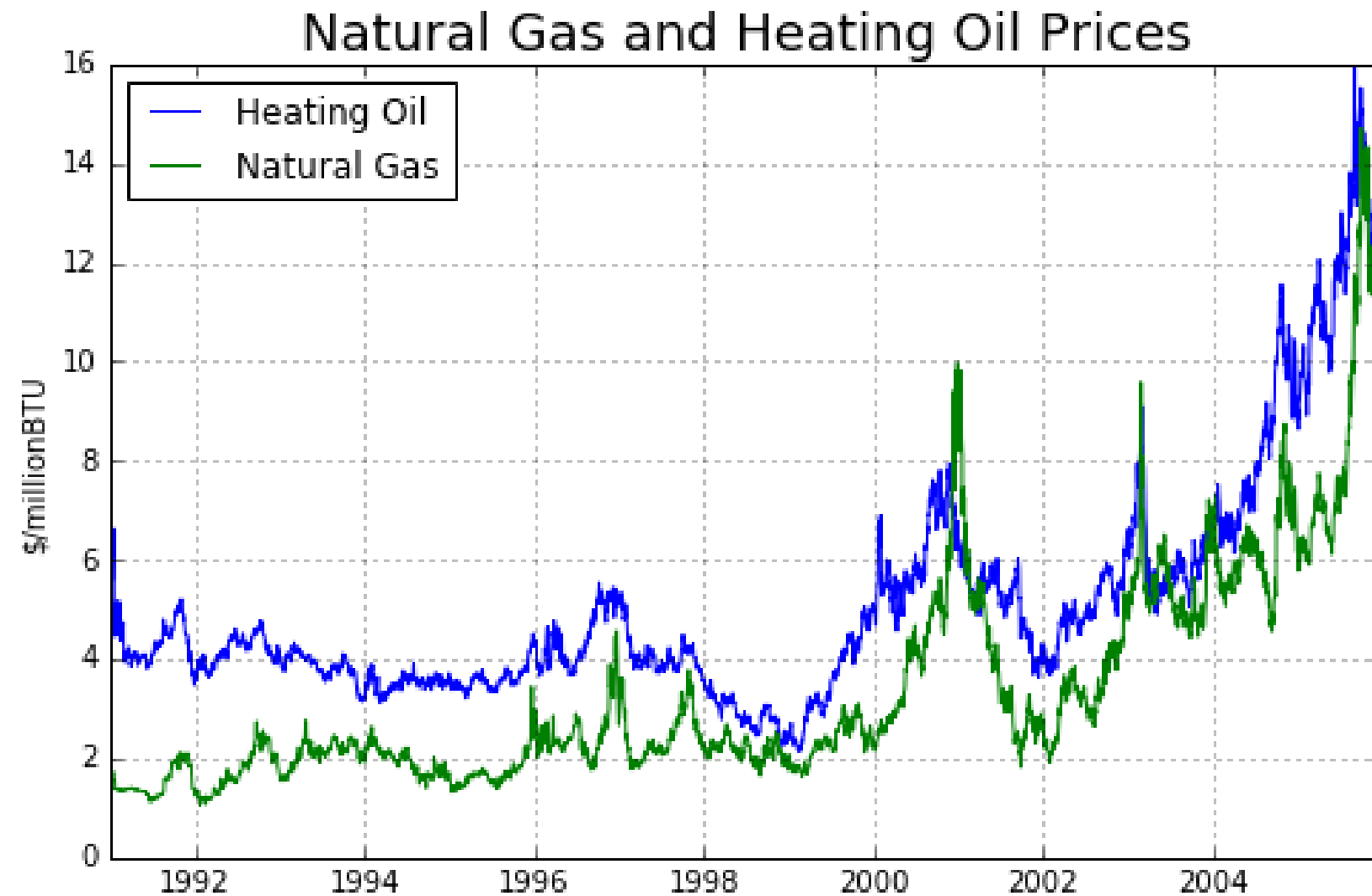
# Example of Time Series: Climate Data



# Example of Time Series: Quarterly Earnings Data



# Example of Multiple Series: Natural Gas and Heating Oil



# Goals of Course

- Learn about time series models
- Fit data to a times series model
- Use the models to make forecasts of the future
- Learn how to use the relevant statistical packages in Python
- Provide concrete examples of how these models are used

# Some Useful Pandas Tools

- Changing an index to datetime

```
df.index = pd.to_datetime(df.index)
```

- Plotting data

```
df.plot()
```

- Slicing data

```
df[ '2012' ]
```

# Some Useful Pandas Tools

- Join two DataFrames

```
df1.join(df2)
```

- Resample data (e.g. from daily to weekly)

```
df = df.resample(rule='W', how='last')
```



# More pandas Functions

- Computing percent changes and differences of a time series

```
df[ 'col' ].pct_change()  
df[ 'col' ].diff()
```

- pandas correlation method of Series

```
df[ 'ABC' ].corr(df[ 'XYZ' ])
```

- pandas autocorrelation

```
df[ 'ABC' ].autocorr()
```

# Let's practice!

TIME SERIES ANALYSIS IN PYTHON

# Correlation of Two Time Series

TIME SERIES ANALYSIS IN PYTHON

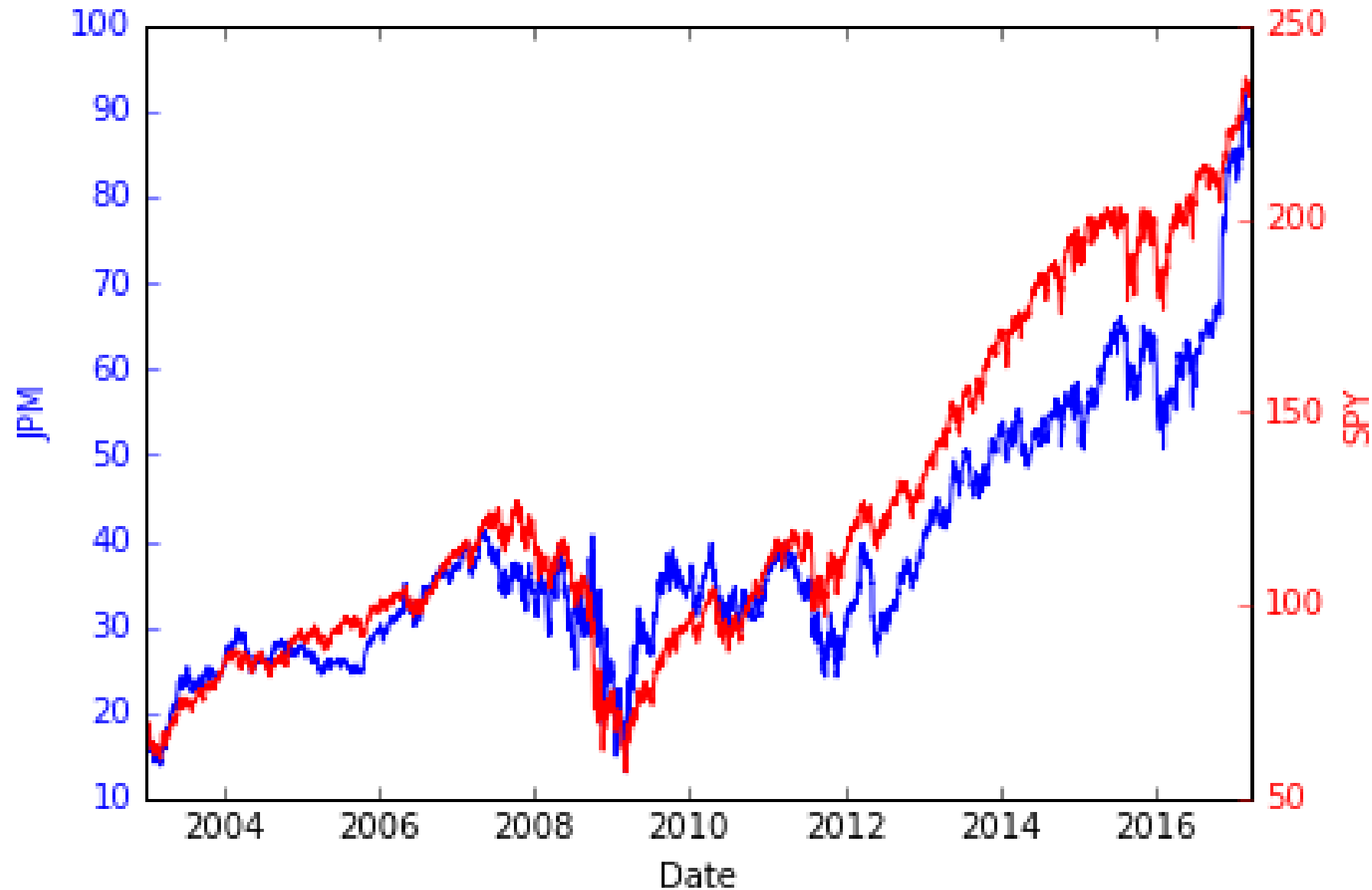


**Rob Reider**

Adjunct Professor, NYU-Courant  
Consultant, Quantopian

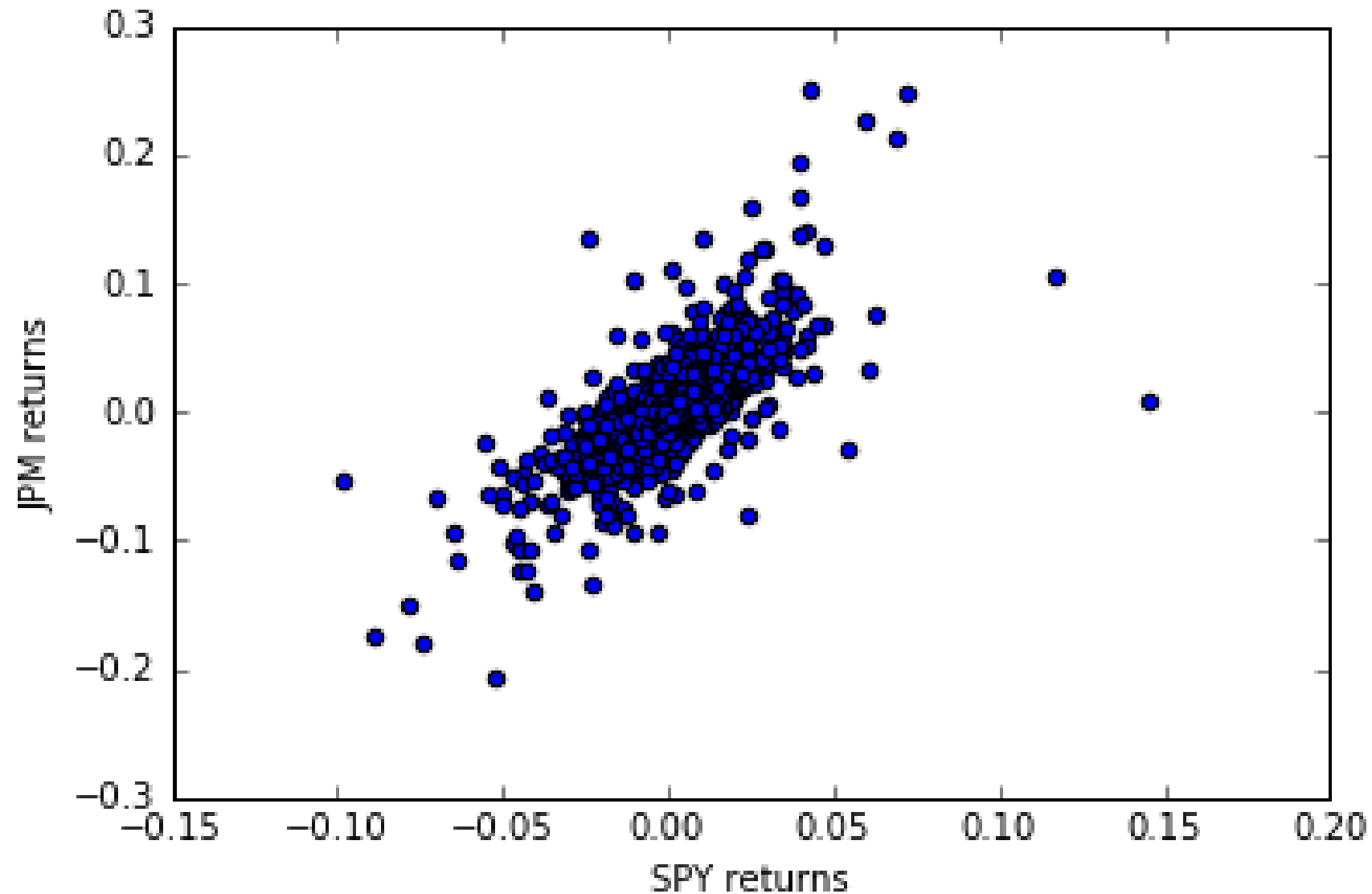
# Correlation of Two Time Series

- Plot of S&P500 and JPMorgan stock



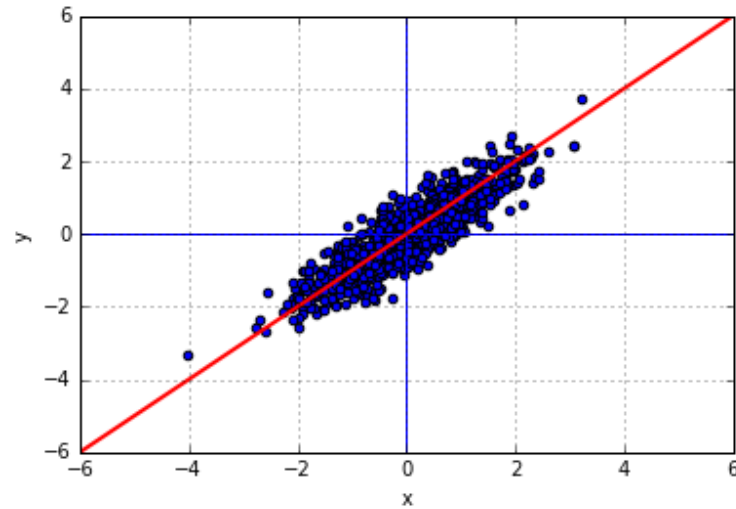
# Correlation of Two Time Series

- Scatter plot of S&P500 and JP Morgan returns

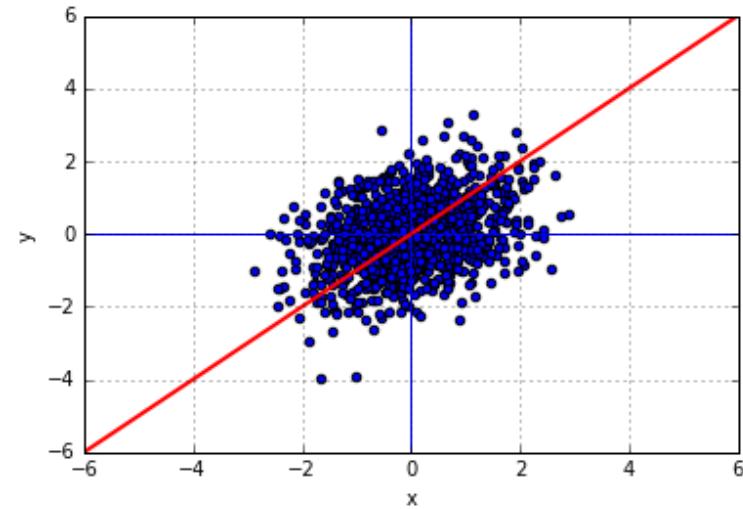


# More Scatter Plots

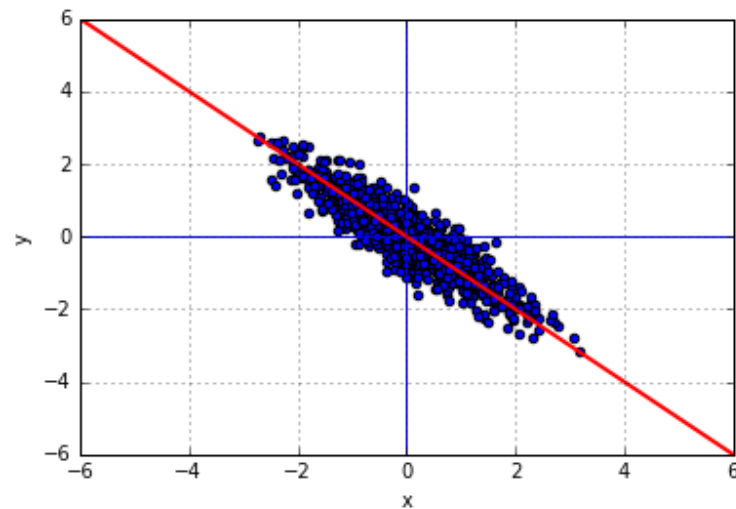
- Correlation = 0.9



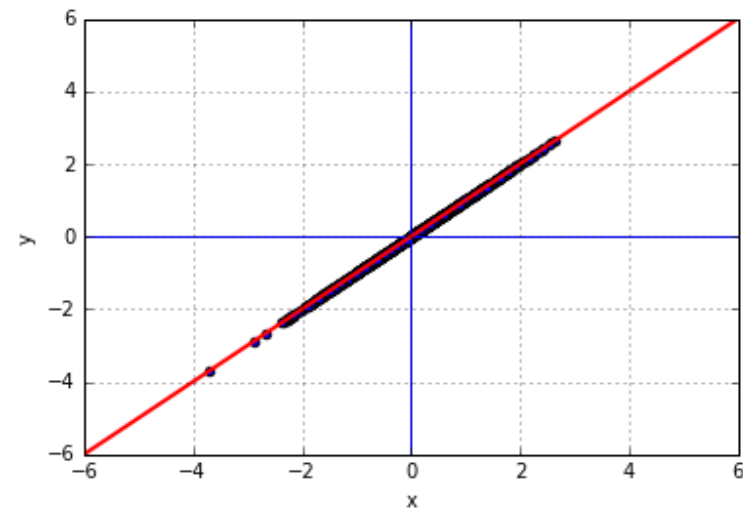
- Correlation = 0.4



- Correlation = -0.9

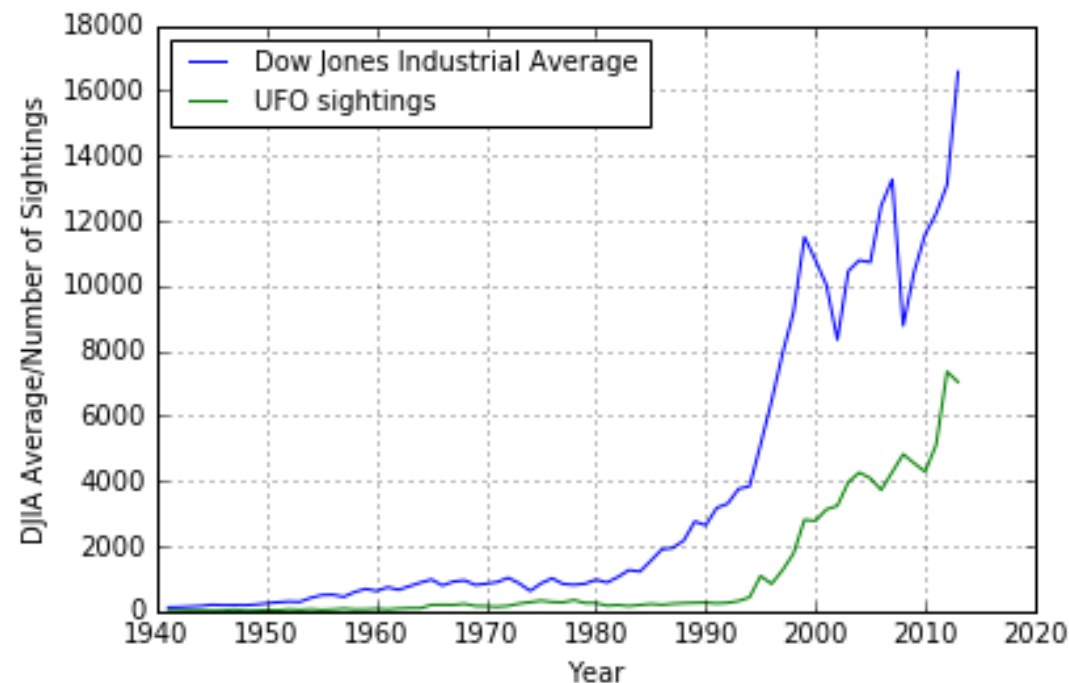


- Correlation = 1.0



# Common Mistake: Correlation of Two Trending Series

- Dow Jones Industrial Average and UFO Sightings ([www.nuforc.org](http://www.nuforc.org))



- Correlation of levels: 0.94

# Example: Correlation of Large Cap and Small Cap Stocks

- Start with stock prices of SPX (large cap) and R2000 (small cap)
- First step: Compute percentage changes of both series

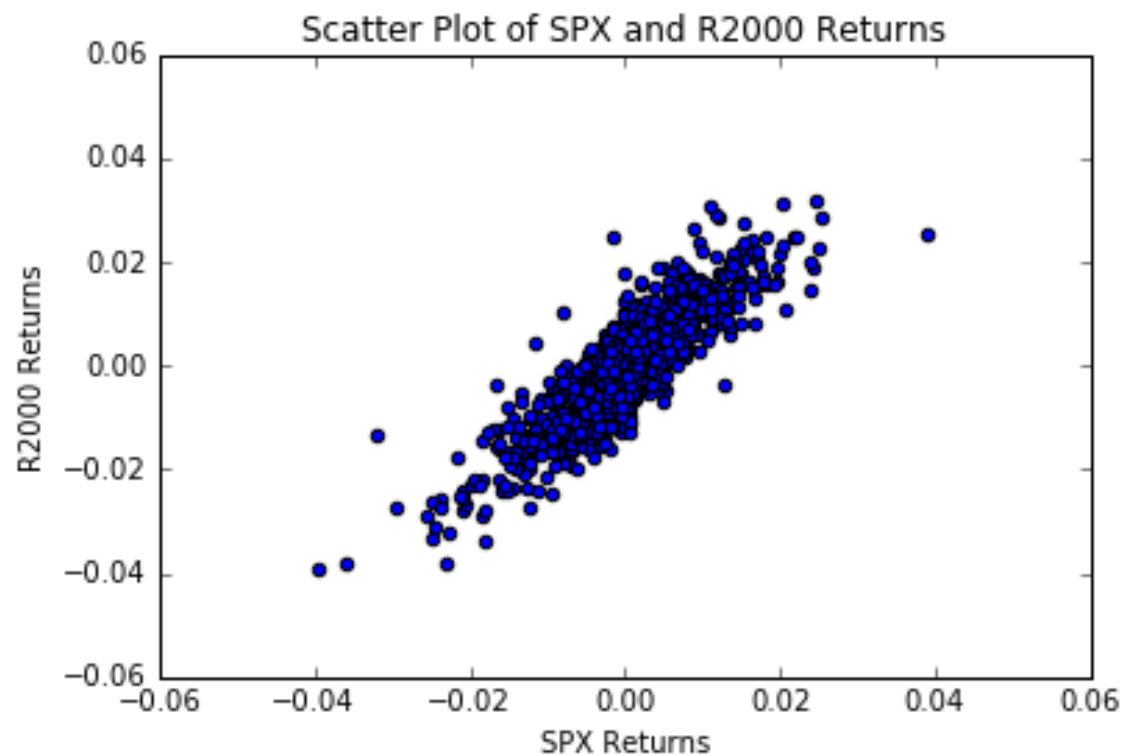
```
df['SPX_Ret'] = df['SPX_Prices'].pct_change()  
df['R2000_Ret'] = df['R2000_Prices'].pct_change()
```



# Example: Correlation of Large Cap and Small Cap Stocks

- Visualize correlation with scatter plot

```
plt.scatter(df['SPX_Return'], df['R2000_Return'])  
plt.show()
```



# Example: Correlation of Large Cap and Small Cap Stocks

- Use pandas correlation method for Series

```
correlation = df['SPX_Ret'].corr(df['R2000_Ret'])  
print("Correlation is: ", correlation)
```

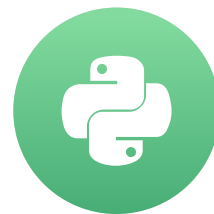
```
Correlation is: 0.868
```

# Let's practice!

TIME SERIES ANALYSIS IN PYTHON

# Simple Linear Regressions

TIME SERIES ANALYSIS IN PYTHON



**Rob Reider**

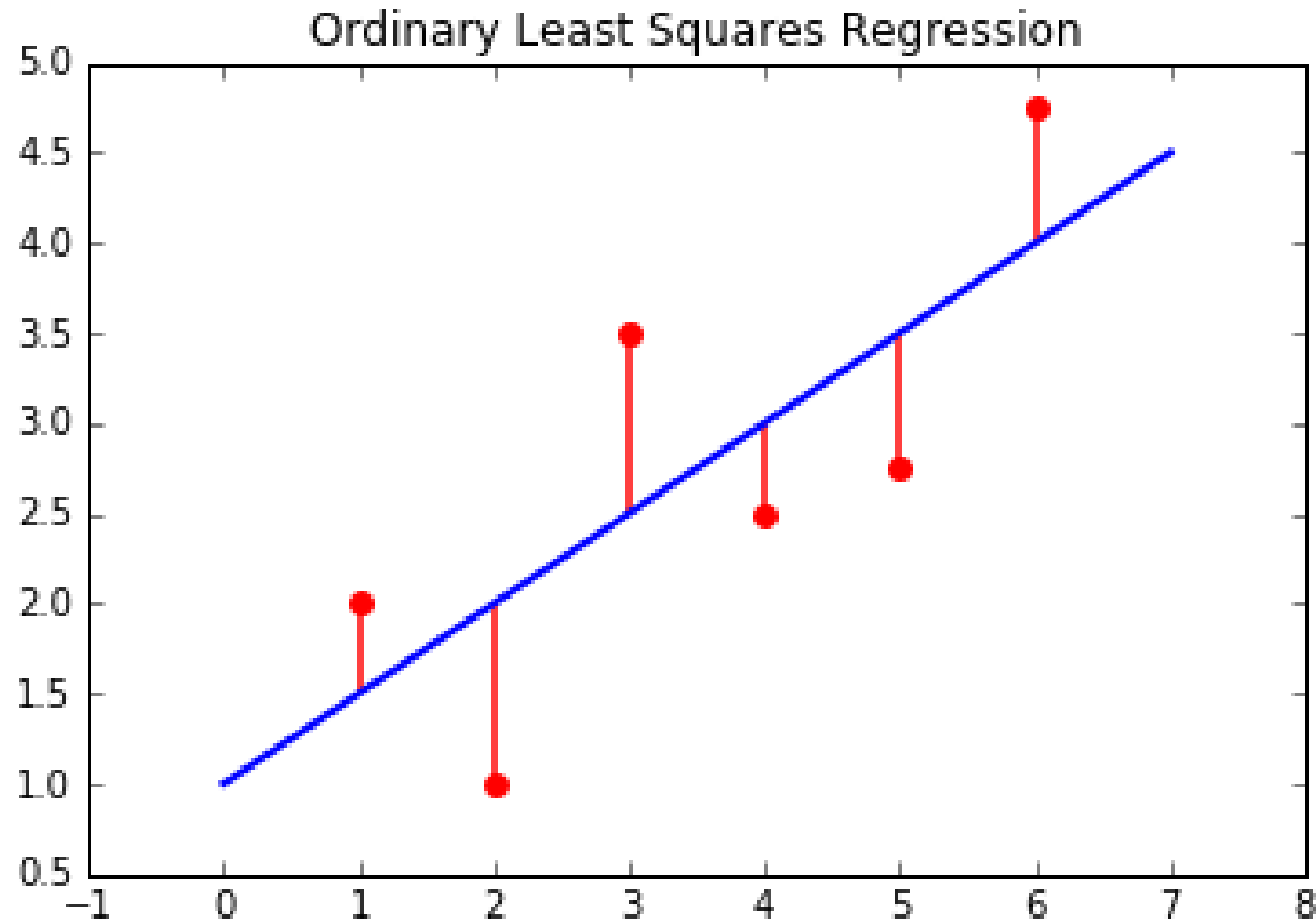
Adjunct Professor, NYU-Courant  
Consultant, Quantopian

# What is a Regression?

- Simple linear regression:  $y_t = \alpha + \beta x_t + \epsilon_t$

# What is a Regression?

- Ordinary Least Squares (OLS)



# Python Packages to Perform Regressions

- In statsmodels:

```
import statsmodels.api as sm
sm.OLS(y, x).fit()
```

Warning: the order of `x` and `y` is not consistent across packages

- In numpy:

```
np.polyfit(x, y, deg=1)
```

- In pandas:

```
pd.ols(y, x)
```

- In scipy:

```
from scipy import stats
stats.linregress(x, y)
```

# Example: Regression of Small Cap Returns on Large Cap

- Import the statsmodels module

```
import statsmodels.api as sm
```

- As before, compute percentage changes in both series

```
df['SPX_Ret'] = df['SPX_Prices'].pct_change()  
df['R2000_Ret'] = df['R2000_Prices'].pct_change()
```

- Add a constant to the DataFrame for the regression intercept

```
df = sm.add_constant(df)
```



# Regression Example (continued)

- Notice that the first row of returns is NaN

	SPX_Price	R2000_Price	SPX_Ret	R2000_Ret
Date				
2012-11-01	1427.589966	827.849976	NaN	NaN
2012-11-02	1414.199951	814.369995	-0.009379	-0.016283

- Delete the row of NaN

```
df = df.dropna()
```

- Run the regression

```
results = sm.OLS(df['R2000_Ret'], df[['const', 'SPX_Ret']]).fit()  
print(results.summary())
```

# Regression Example (continued)

- Regression output

```
=====
                        OLS Regression Results
=====
Dep. Variable:          R2000_Ret      R-squared:                0.753
Model:                  OLS           Adj. R-squared:           0.753
Method:                 Least Squares   F-statistic:              3829.
Date:                  Fri, 26 Jan 2018   Prob (F-statistic):       0.00
Time:                  13:29:55          Log-Likelihood:           4882.4
No. Observations:      1257             AIC:                     -9761.
Df Residuals:          1255             BIC:                     -9751.
Df Model:               1
Covariance Type:       nonrobust
=====

```

	coef	std err	t	P> t	[95.0% Conf. Int.]
const	-4.964e-05	0.000	-0.353	0.724	-0.000 0.000
SPX_Ret	1.1412	0.018	61.877	0.000	1.105 1.177

```
=====
Omnibus:                61.950      Durbin-Watson:           1.991
Prob(Omnibus):           0.000      Jarque-Bera (JB):        148.100
Skew:                   0.266      Prob(JB):                6.93e-33
Kurtosis:                4.595      Cond. No.                131.
=====
```

- Intercept in `results.params[0]`
- Slope in `results.params[1]`

# Regression Example (continued)

- Regression output

```

=====
                        OLS Regression Results
=====
Dep. Variable:          R2000_Ret      R-squared:                0.753
Model:                  OLS           Adj. R-squared:           0.753
Method:                 Least Squares   F-statistic:              3829.
Date:                  Fri, 26 Jan 2018   Prob (F-statistic):       0.00
Time:                  13:29:55          Log-Likelihood:           4882.4
No. Observations:      1257             AIC:                     -9761.
Df Residuals:          1255             BIC:                     -9751.
Df Model:               1
Covariance Type:       nonrobust
=====

```

	coef	std err	t	P> t	[95.0% Conf. Int.]
const	-4.964e-05	0.000	-0.353	0.724	-0.000 0.000
SPX_Ret	1.1412	0.018	61.877	0.000	1.105 1.177

```

=====
Omnibus:                61.950   Durbin-Watson:           1.991
Prob(Omnibus):           0.000   Jarque-Bera (JB):        148.100
Skew:                    0.266   Prob(JB):                 6.93e-33
Kurtosis:                4.595   Cond. No.                  131.
=====

```

# Relationship Between R-Squared and Correlation

- $[\text{corr}(x, y)]^2 = R^2$  (or R-squared)
- $\text{sign}(\text{corr}) = \text{sign}(\text{regression slope})$
- In last example:
  - R-Squared = 0.753
  - Slope is positive
  - correlation =  $+\sqrt{0.753} = 0.868$

# Let's practice!

TIME SERIES ANALYSIS IN PYTHON

# Autocorrelation

TIME SERIES ANALYSIS IN PYTHON



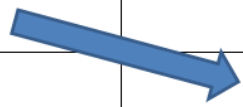
**Rob Reider**

Adjunct Professor, NYU-Courant  
Consultant, Quantopian

# What is Autocorrelation?

- Correlation of a time series with a lagged copy of itself

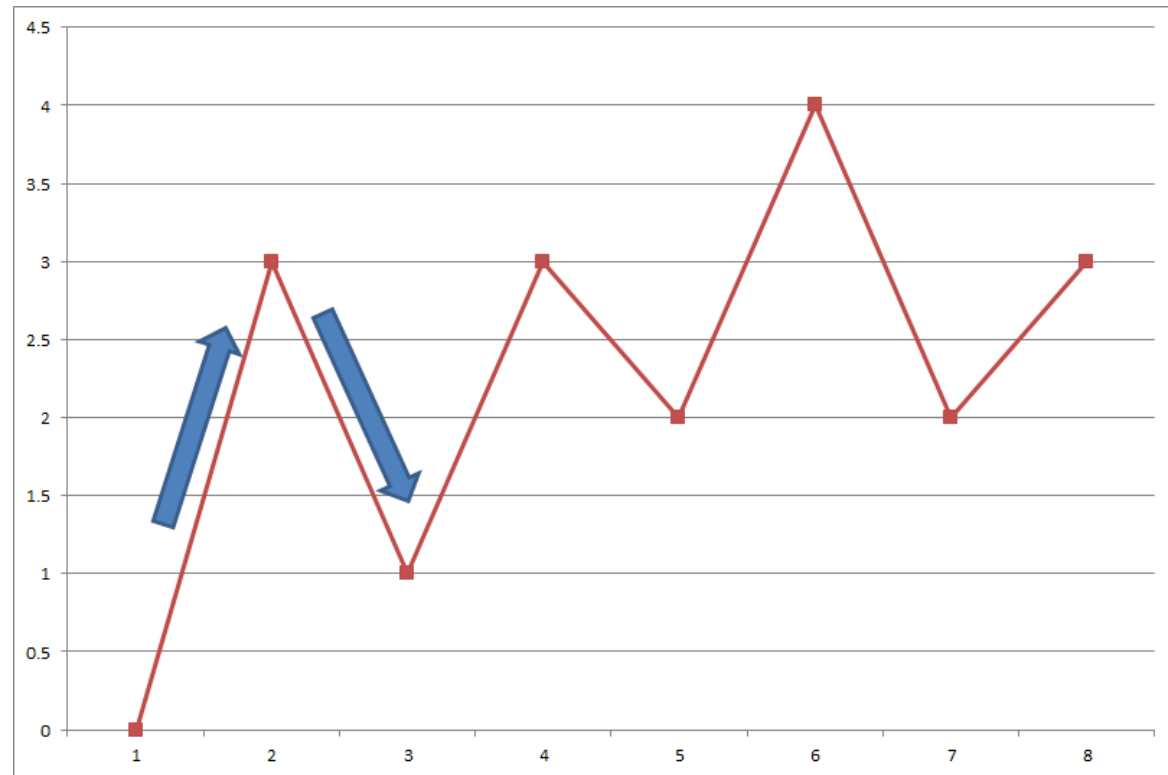
Series	Lagged Series
5	
10	5
15	10
20	15
25	20
⋮	⋮



- Lag-one autocorrelation
- Also called **serial correlation**

# Interpretation of Autocorrelation

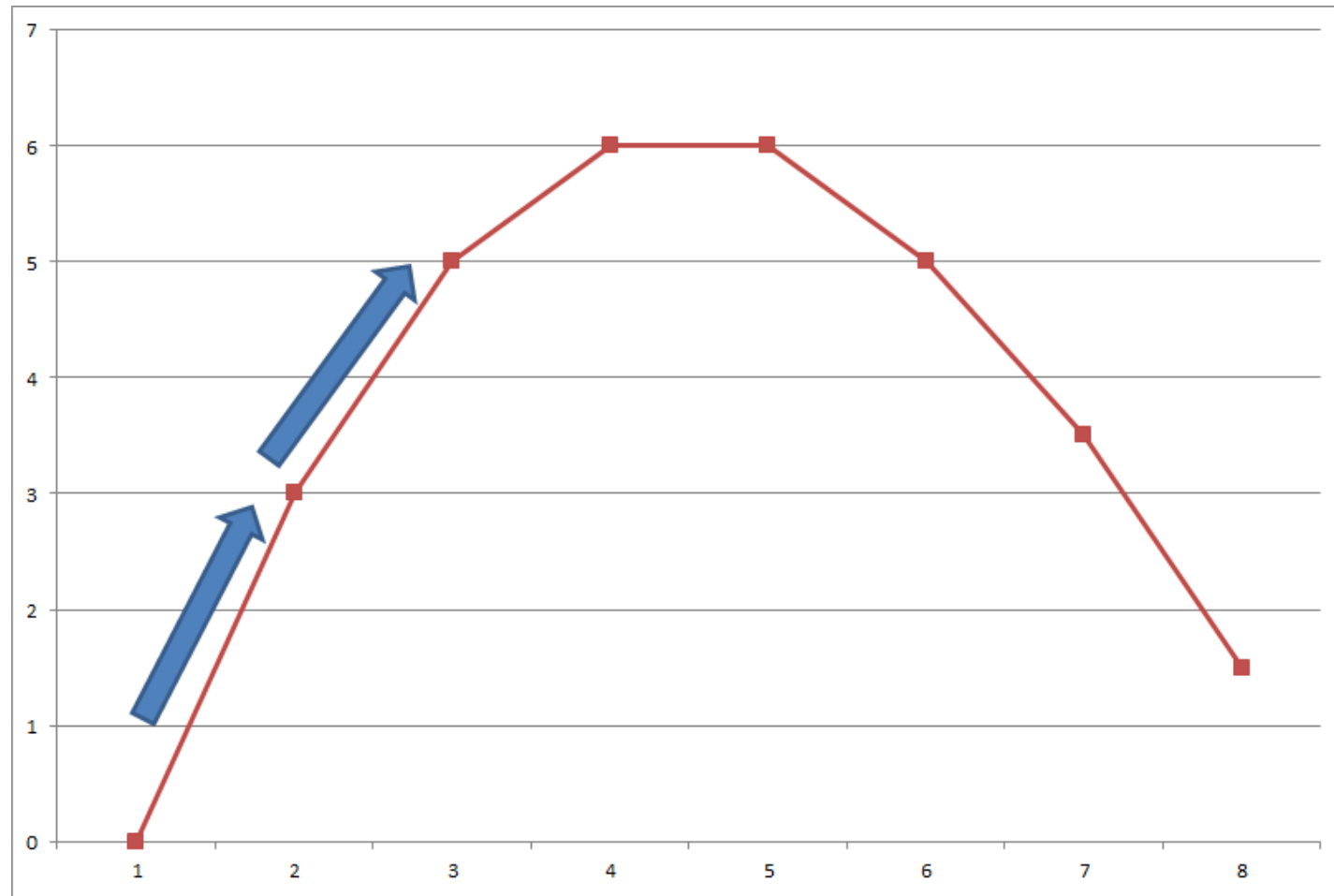
- Mean Reversion - Negative autocorrelation





# Interpretation of Autocorrelation

- Momentum, or Trend Following - Positive autocorrelation



# Traders Use Autocorrelation to Make Money

- Individual stocks
  - Historically have negative autocorrelation
  - Measured over short horizons (days)
  - Trading strategy: Buy losers and sell winners
- Commodities and currencies
  - Historically have positive autocorrelation
  - Measured over longer horizons (months)
  - Trading strategy: Buy winners and sell losers

# Example of Positive Autocorrelation: Exchange Rates

- Use daily ¥/\$ exchange rates in DataFrame `df` from [FRED](#)
- Convert index to datetime

```
# Convert index to datetime
df.index = pd.to_datetime(df.index)
# Downsample from daily to monthly data
df = df.resample(rule='M', how='last')
# Compute returns from prices
df['Return'] = df['Price'].pct_change()
# Compute autocorrelation
autocorrelation = df['Return'].autocorr()
print("The autocorrelation is: ", autocorrelation)
```

```
The autocorrelation is: 0.0567
```

# Let's practice!

TIME SERIES ANALYSIS IN PYTHON