

# Introduction to the Course

TIME SERIES ANALYSIS IN PYTHON



**Rob Reider**

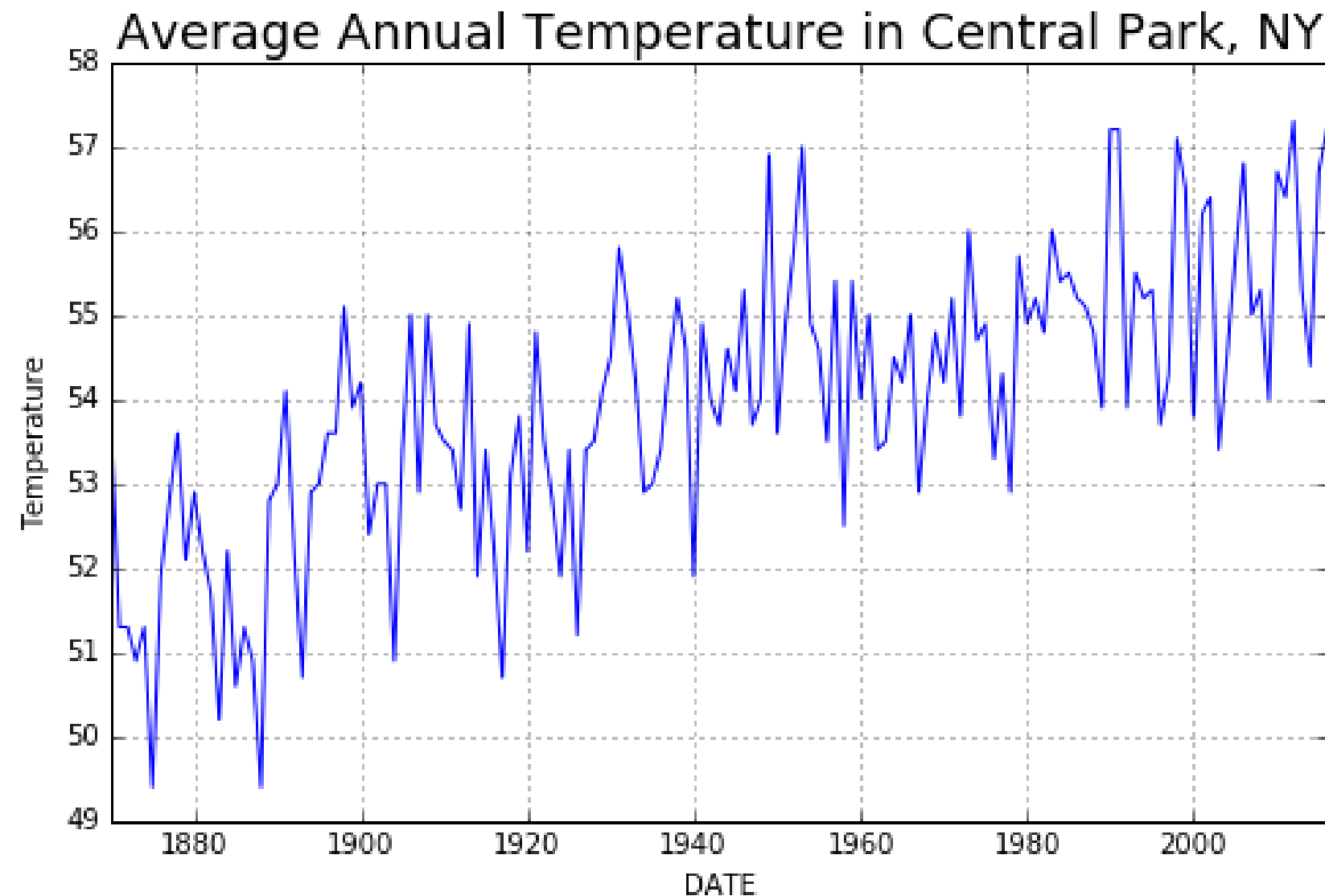
Adjunct Professor, NYU-Courant  
Consultant, Quantopian

Quantopian: Business built on python for analyzing backlog of data & quantitative data strategies.  
Quantopian is also a community where code is shared and questions are asked/answered.

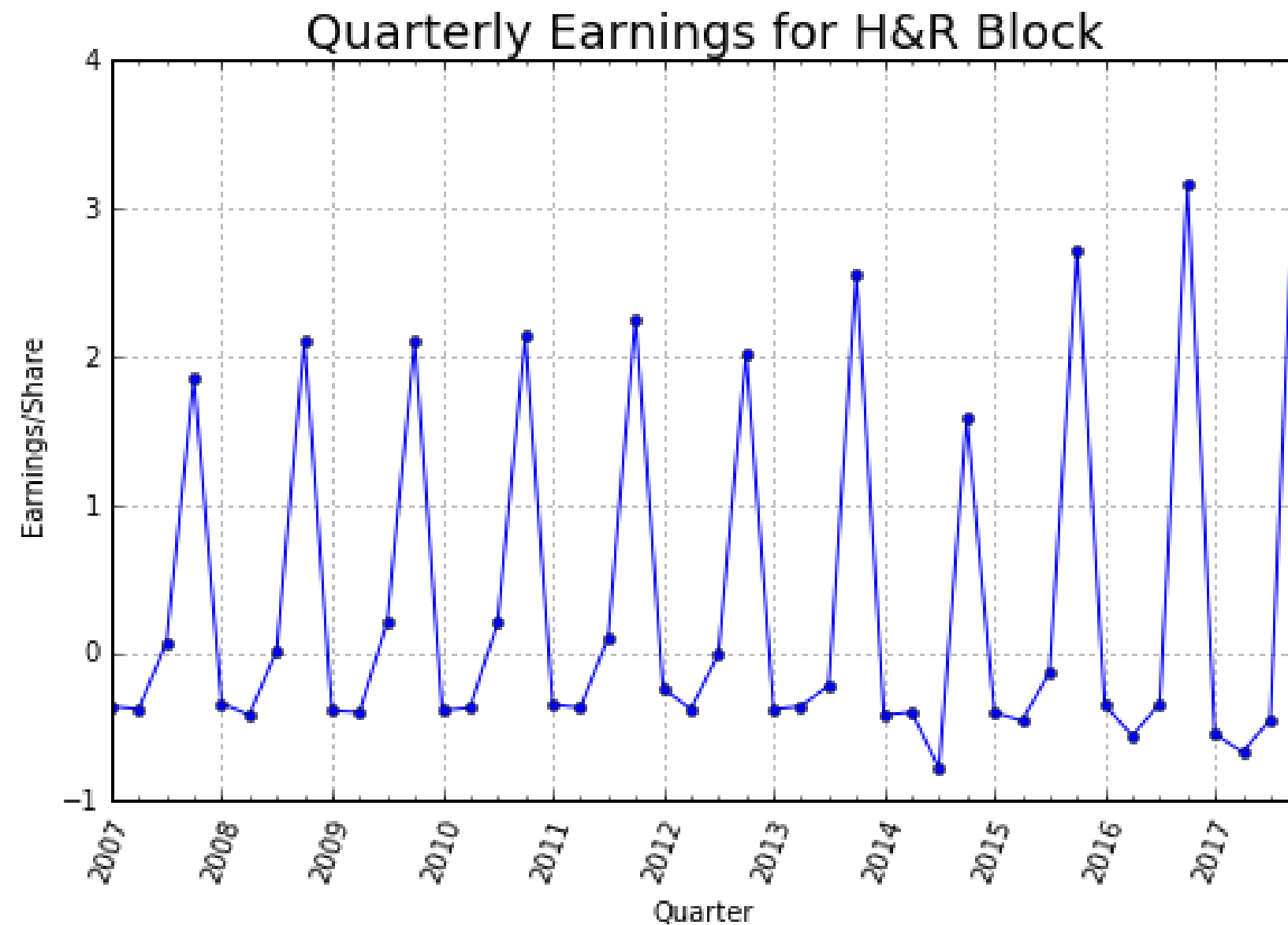
# Example of Time Series: Google Trends



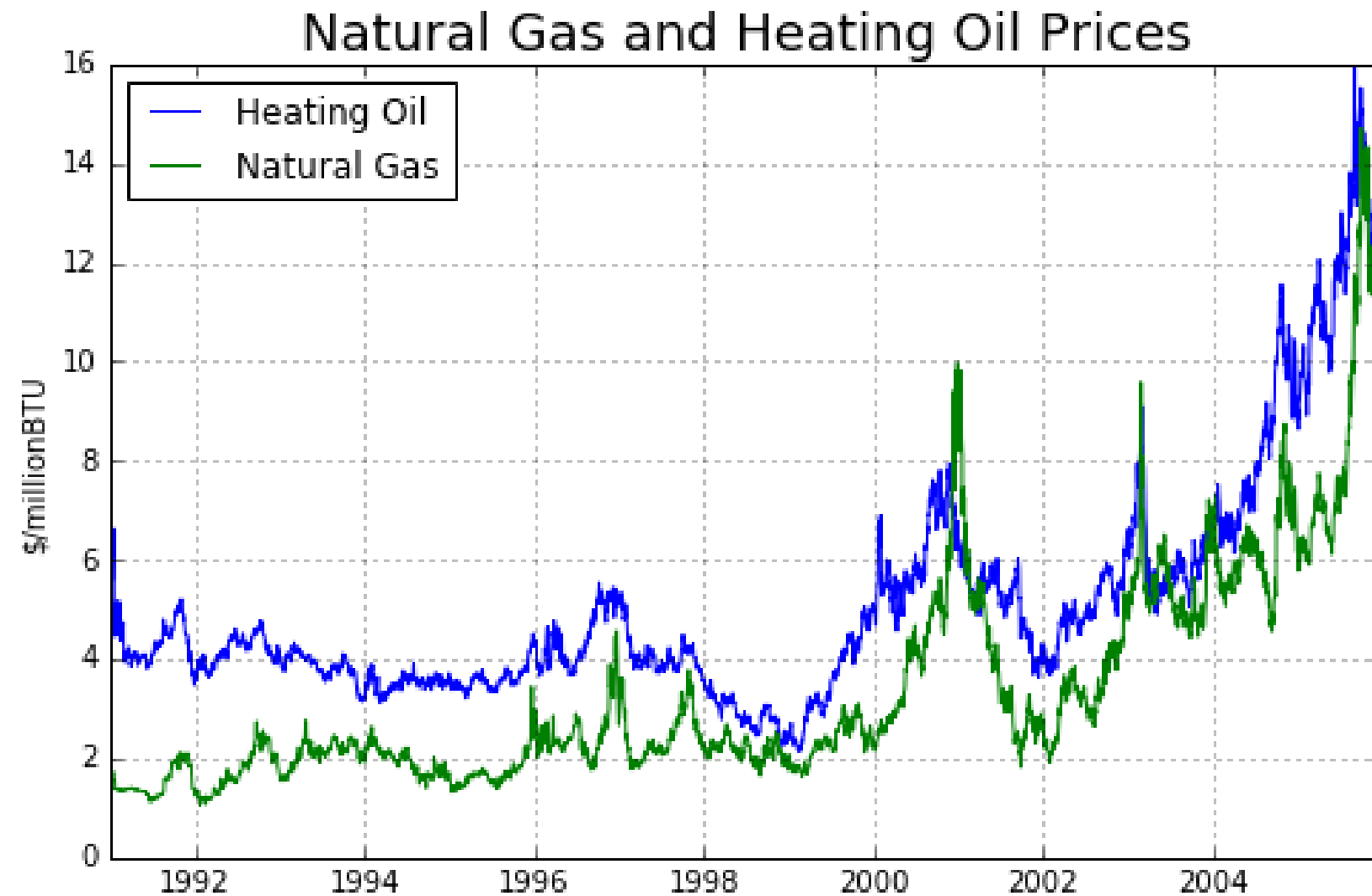
# Example of Time Series: Climate Data



# 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

Many of the models and data will be applicable in the field of finance.

# 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 Say, if one dataframe contains stock prices while another one contains bond prices.

```
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() Say, if you wanted to convert prices to returns.  
df[ 'col' ].diff()
```

- pandas correlation method of Series

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

- pandas autocorrelation

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

```
# From previous step
diet.index = pd.to_datetime(diet.index)
```

```
# Slice the dataset to keep only 2012
diet2012 = diet['2012']
```

```
# Plot 2012 data
diet2012.plot(grid = True)
plt.show()
```

```
# Import pandas
import pandas as pd
```

```
# Convert the stock index and bond index into sets
set_stock_dates = set(stocks.index)
set_bond_dates = set(bonds.index)
```

```
# Take the difference between the sets and print
print(set_stock_dates - set_bond_dates)
```

```
# Merge stocks and bonds DataFrames using join()
stocks_and_bonds = stocks.join(bonds, how = 'inner')
```

# Let's practice!

## TIME SERIES ANALYSIS IN PYTHON

# Correlation of Two Time Series

TIME SERIES ANALYSIS IN PYTHON

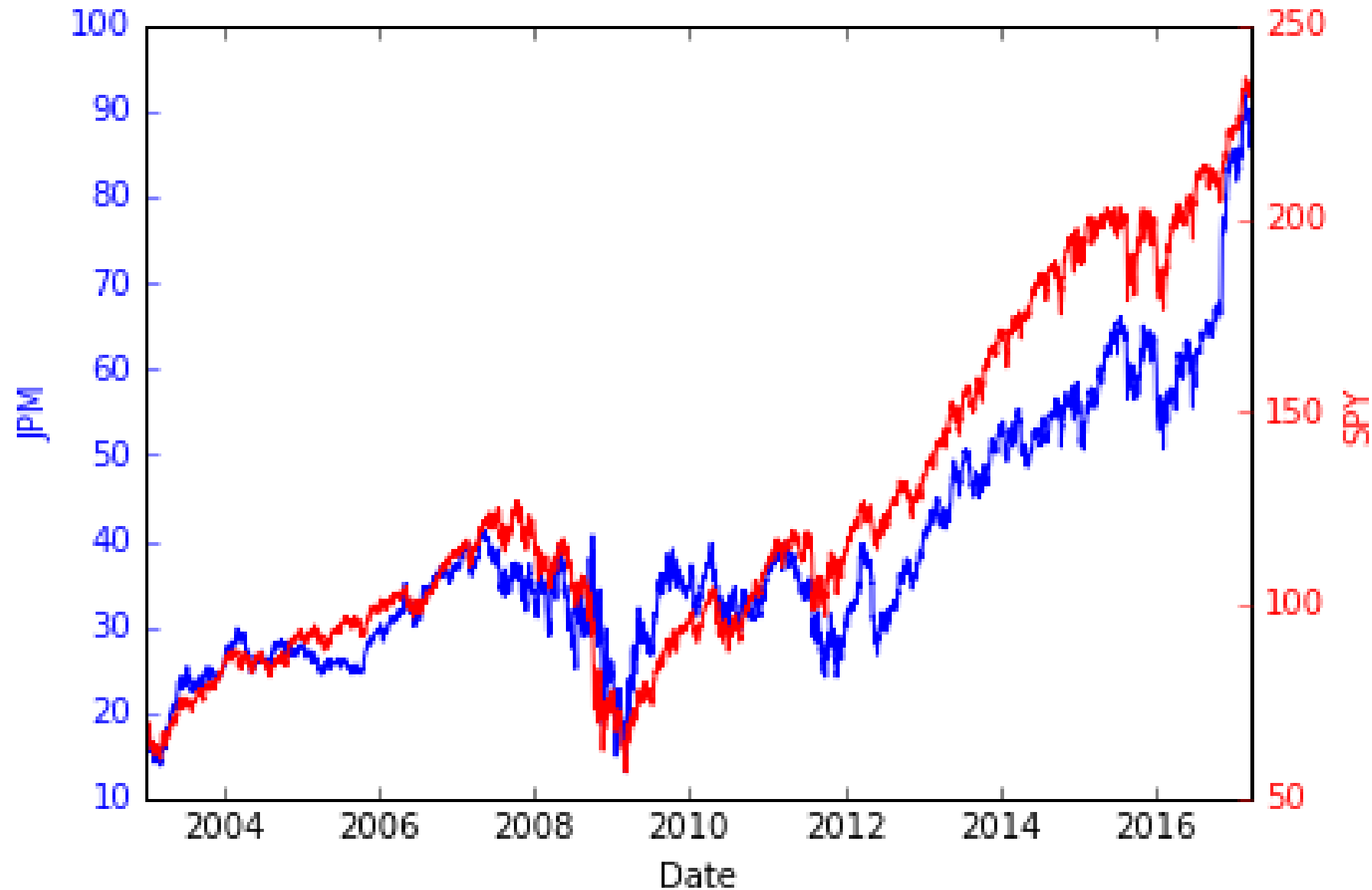


**Rob Reider**

Adjunct Professor, NYU-Courant  
Consultant, Quantopian

# Correlation of Two Time Series

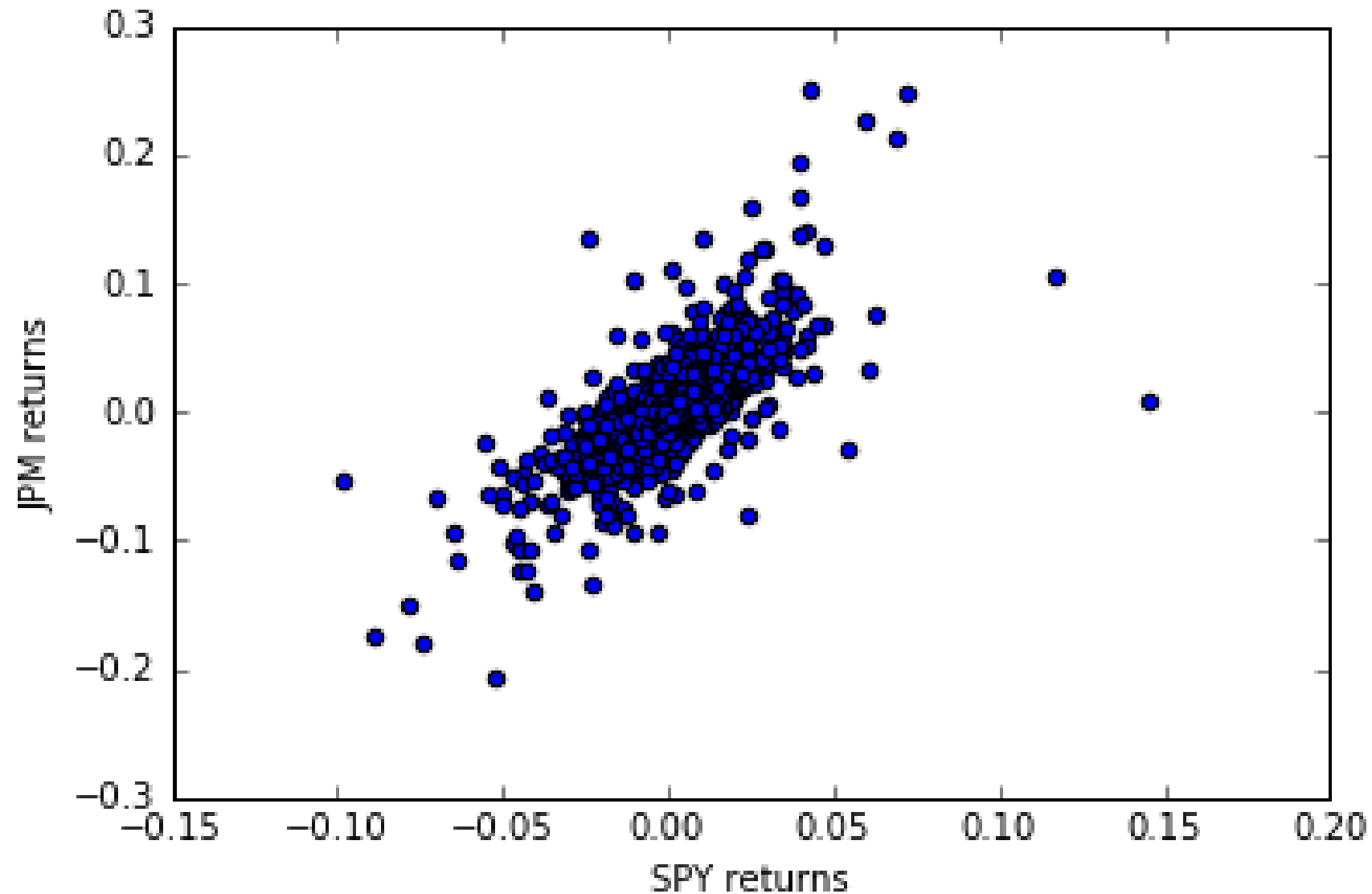
- Plot of S&P500 and JPMorgan stock



Here you see that, generally, when the market dips, so does JPM, and when the market (S&P500) rises, so too does JPM.

# Correlation of Two Time Series

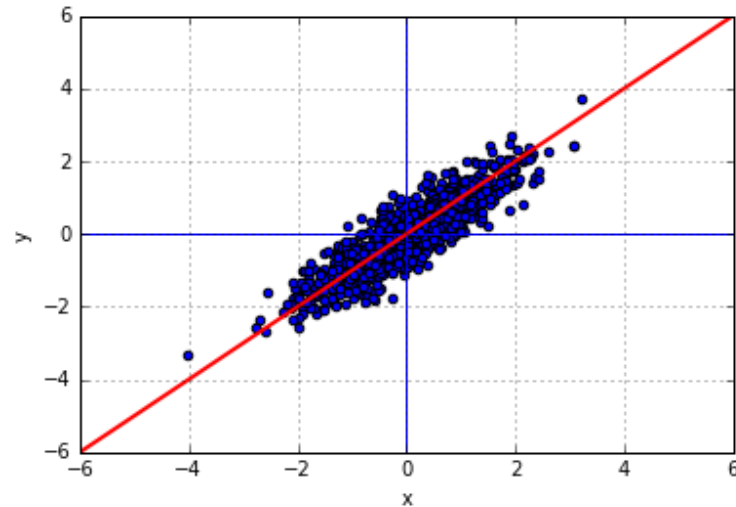
- Scatter plot of S&P500 and JP Morgan returns



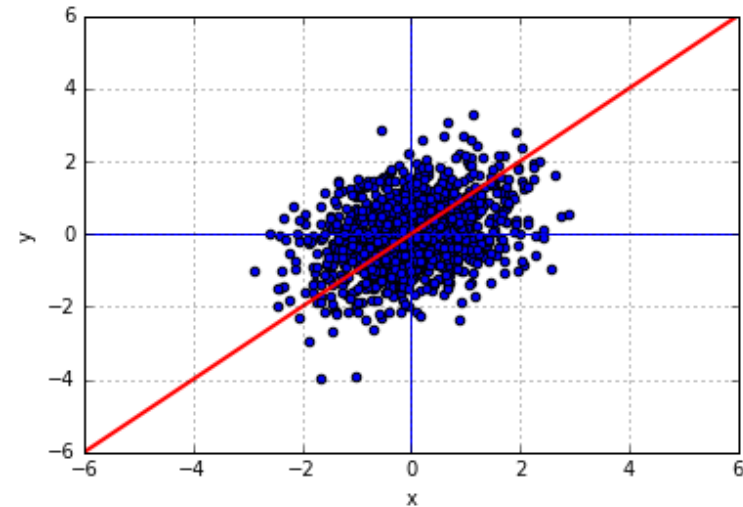
# More Scatter Plots

The correlation coefficient is a measure of how much two series vary together.

- Correlation = 0.9



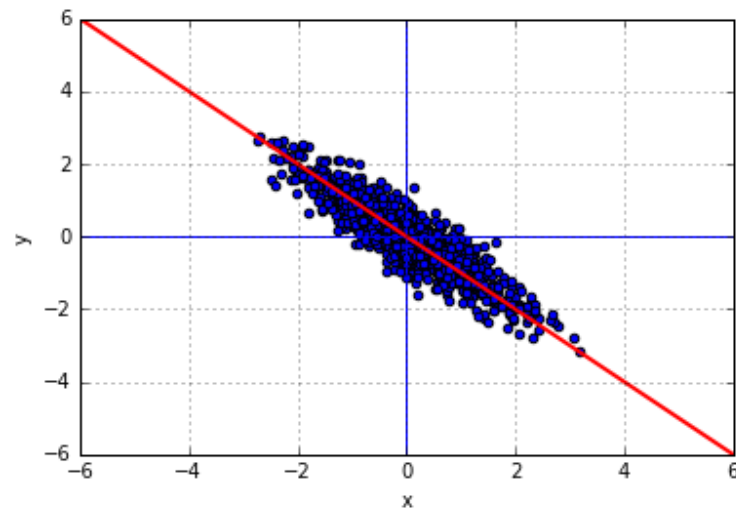
- Correlation = 0.4



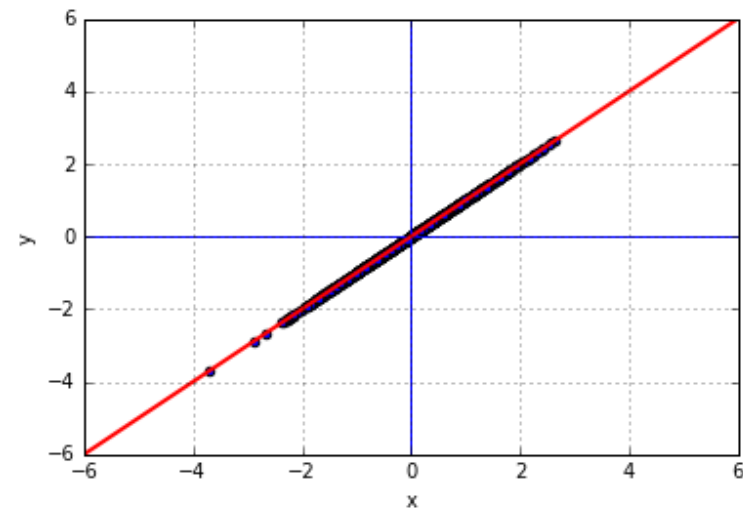
High correlation = the two series strongly vary together.

Low correlation = the two series vary together but with a weak association.

- Correlation = -0.9



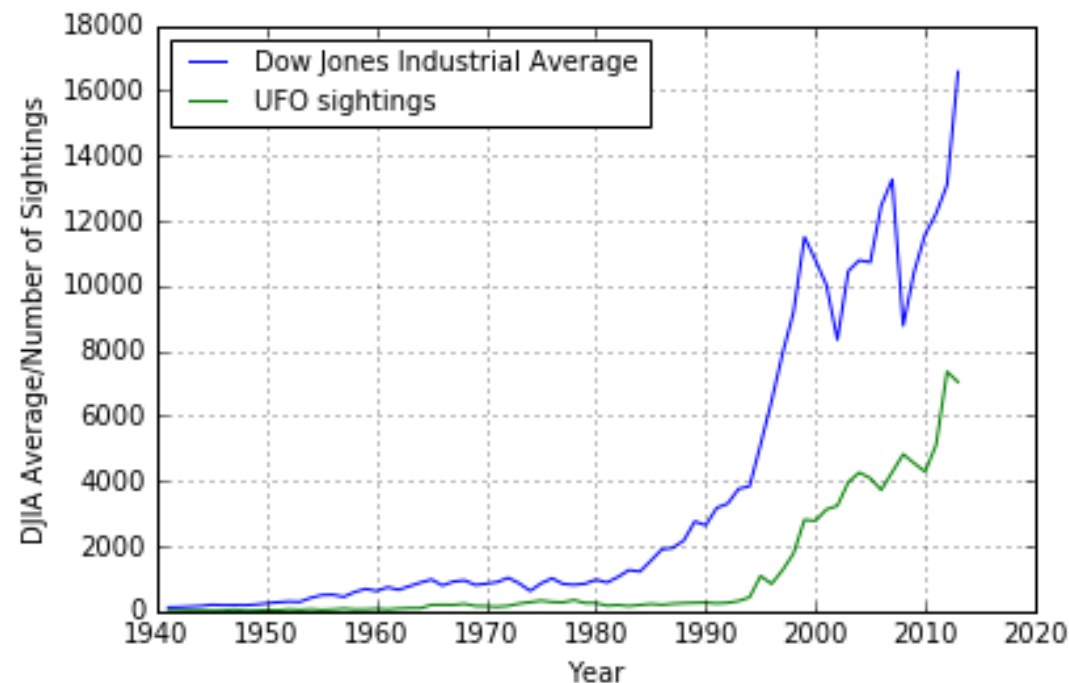
- Correlation = 1.0



High (negative) correlation: the two series vary together, but in opposite directions, but still, with a linear relationship.

# Common Mistake: Correlation of Two Trending Series

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



Even if the two series are totally unrelated you can still get a strong correlation.

In the situation where you're looking at the correlation of two stocks, it's important to look at the correlations of their returns rather than their levels.

- Correlation of levels: 0.94

Here, when you look at the correlations of their levels you get the above coefficient, but when you correlate their `pct_change`, you get 0.

# 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()
```

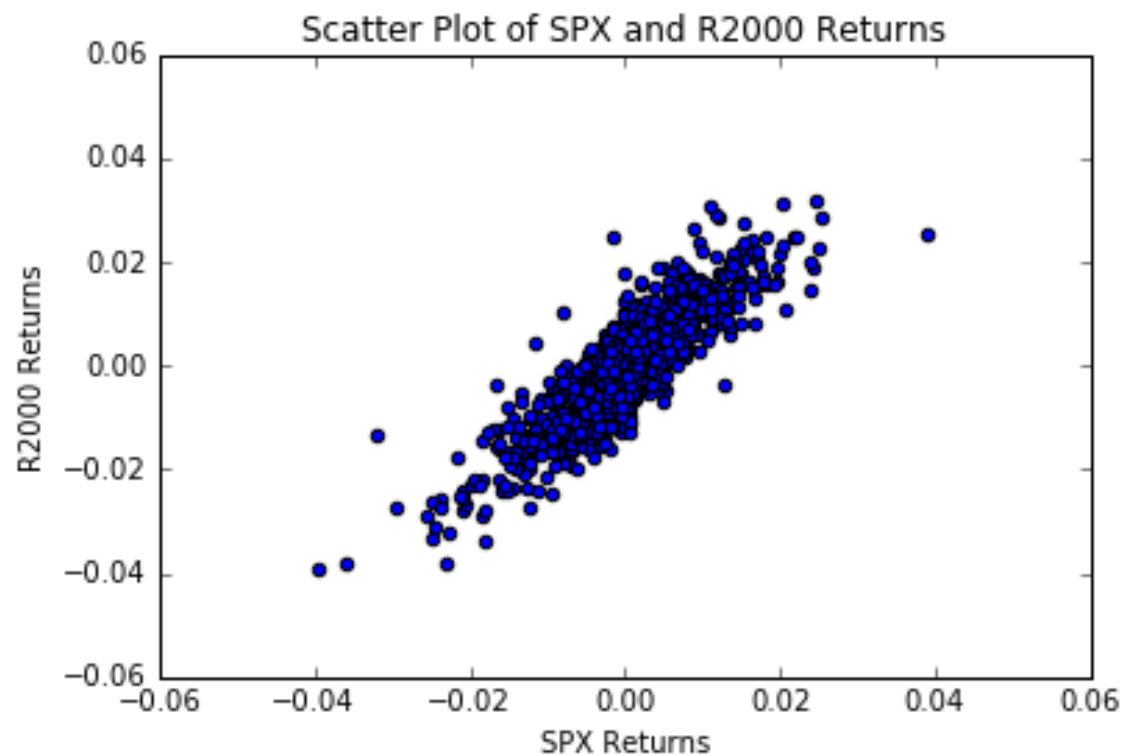
This gives the returns of these series instead of prices.



# 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
```

```
# Compute percent change using pct_change()
returns = stocks_and_bonds.pct_change()

# Compute correlation using corr()
correlation = returns['SP500'].corr(returns['US10Y'])
print("Correlation of stocks and interest rates: ",
      correlation)

# Make scatter plot
plt.scatter(returns['SP500'],returns['US10Y'])
plt.show()
```

The positive correlation means that when interest rates go down, stock prices go down. For example, during crises like 9/11, investors sold stocks and moved their money to less risky bonds (this is sometimes referred to as a 'flight to quality'). During these periods, stocks drop and interest rates drop as well. Of course, there are times when the opposite relationship holds too.

# Let's practice!

## TIME SERIES ANALYSIS IN PYTHON

```
# Compute correlation of levels
correlation1 = levels.DJI.corr(levels.UFO)
print("Correlation of levels: ", correlation1)

# Compute correlation of percent changes
changes = levels.pct_change()
correlation2 = changes.DJI.corr(changes.UFO)
print("Correlation of changes: ", correlation2)
```

Notice that the correlation on levels is high but the correlation on changes is close to zero.

# Simple Linear Regressions

TIME SERIES ANALYSIS IN PYTHON



**Rob Reider**

Adjunct Professor, NYU-Courant  
Consultant, Quantopian

# What is a Regression?

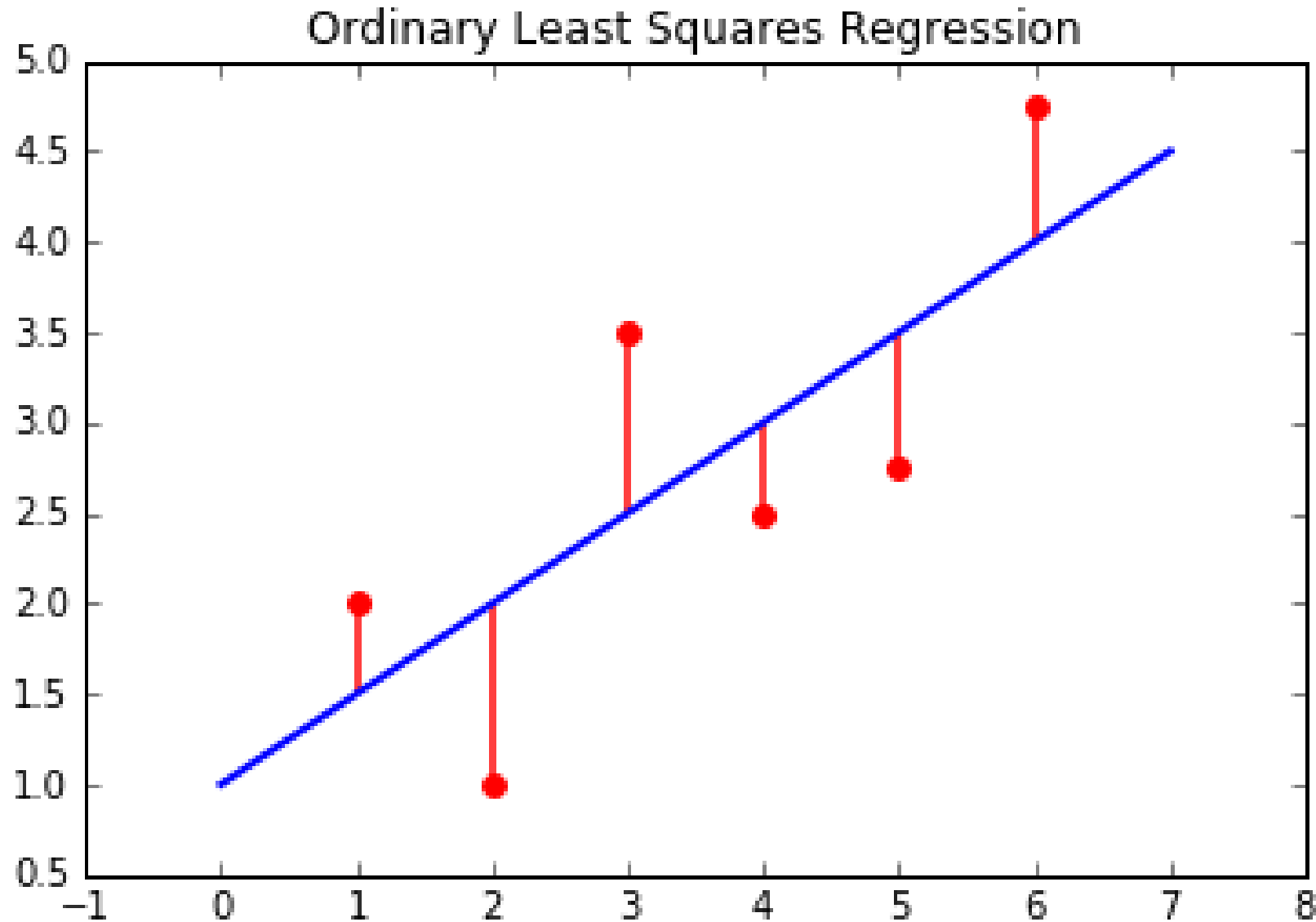
- Simple linear regression:  $y_t = \alpha + \beta x_t + \epsilon_t$

A linear regression finds the slope, Beta, and the intercept, Alpha, of a line, that's the best fit between a dependent variable y and an independent variable x.

Xs & Ys can be two time series.

# What is a Regression?

- Ordinary Least Squares (OLS)



A linear regression is also known as a Ordinary Least Squares:

It aims to minimize the sum of the squared distances between the data points and the regression line.

# 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)
```

<— — setting 'deg' parameter to 1 is the command for a linear regression.

- In pandas:

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

- In scipy:

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

Beware, that the order of x & y is not consistent across packages.

# 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)
```

If you don't have a constant column then the packages assumed you don't want to run the regression with an intercept.

By adding a column of 1s, statsmodels will compute the regression coefficient of that column as well, which can be interpreted as the intercept of the line.



# 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

Because the return is calculated from a previous price, the first line has no previous price and therefore can't produce a `pct_change()` value.

- 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())
```

The first argument is your dependent variable `y`, the next argument is your independent variable or variables for `x`.

# 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.
=====
```

const is the intercept or Alpha  
SPX\_Ret is the Slope or Beta

- 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 =  $+\sqrt{0.753} = 0.868$

In the same way that a correlation measures how closely the data are clustered along a line, so too does R-squared.

R-squared measures how well the linear regression line fits the data.

There is a relationship between correlation and r-squared: The magnitude of the correlation is the square root of the r-squared. The sign of the correlation is the sign of the slope or Beta

If the regression line is positive then the correlation is positive and vice-versa.

```
# Import the statsmodels module
import statsmodels.api as sm

# Compute correlation of x and y
correlation = x.corr(y)
print("The correlation between x and y is %4.2f" %(correlation))

# Convert the Series x to a DataFrame and name the column x
dfx = pd.DataFrame(x, columns=['x'])

# Add a constant to the DataFrame dfx
dfx1 = sm.add_constant(dfx)

# Regress y on dfx1
result = sm.OLS(y, dfx1).fit()

# Print out the results and look at the relationship between R-squared and the correlation above
print(result.summary())
```

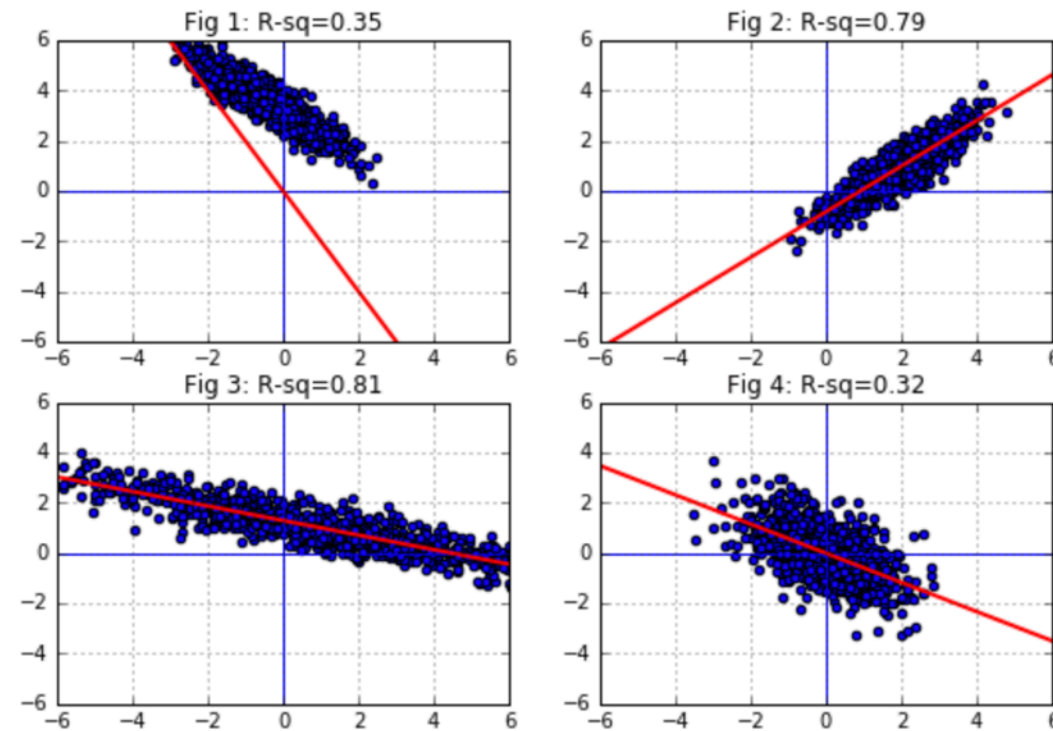
Notice that the two different methods of computing correlation give the same result. The correlation is about -0.9 and the R-squared is about 0.81

# Let's practice!

## TIME SERIES ANALYSIS IN PYTHON

## Match Correlation with Regression Output

Here are four scatter plots, each showing a linear regression line and an R-squared:



Which correlation is correct?

✓ Answer the question

50 XP

### Possible Answers

- ☐ Fig 1: correlation = -0.6
- ☐ Fig 2: correlation = -0.9
- ☒ Fig 3: correlation = -0.9
- ☐ Fig 4: correlation = -0.32

💡 Take Hint (-15xp)

press 1

press 2

press 3

press 4

Submit Answer

# Autocorrelation

TIME SERIES ANALYSIS IN PYTHON



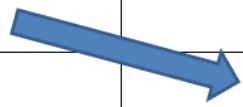
**Rob Reider**

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
⋮	⋮

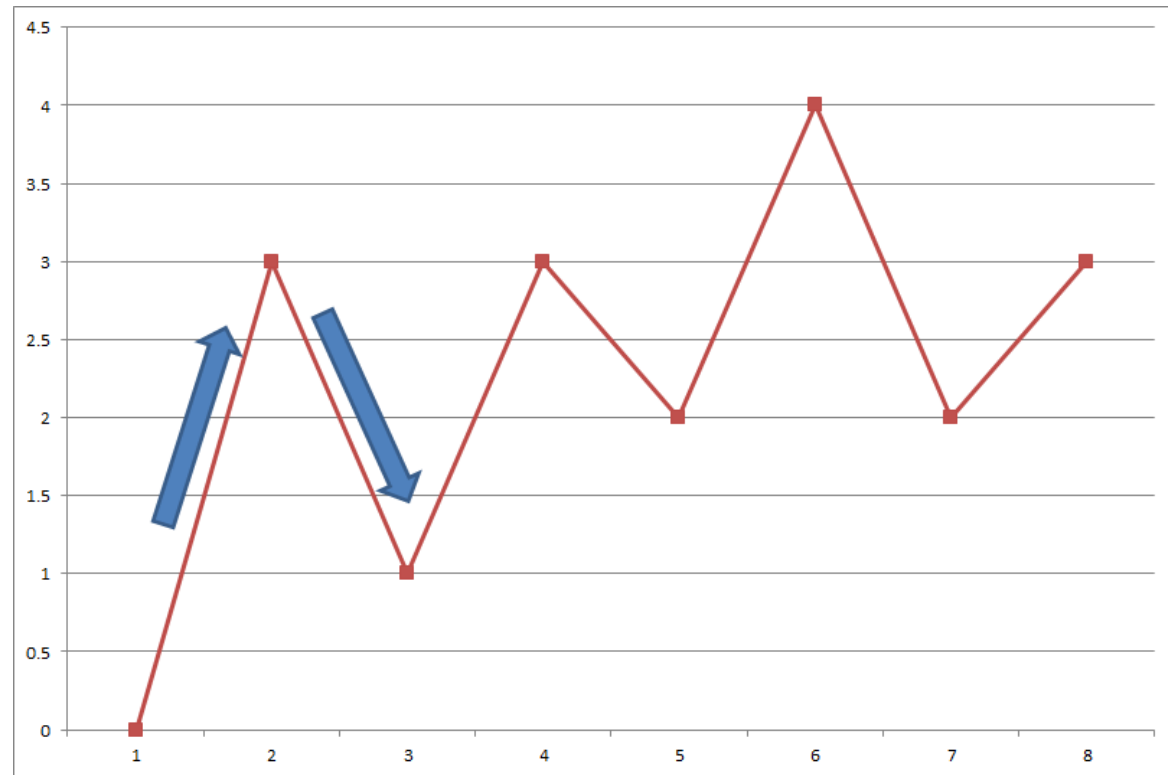


- Lag-one autocorrelation
- Also called **serial correlation**



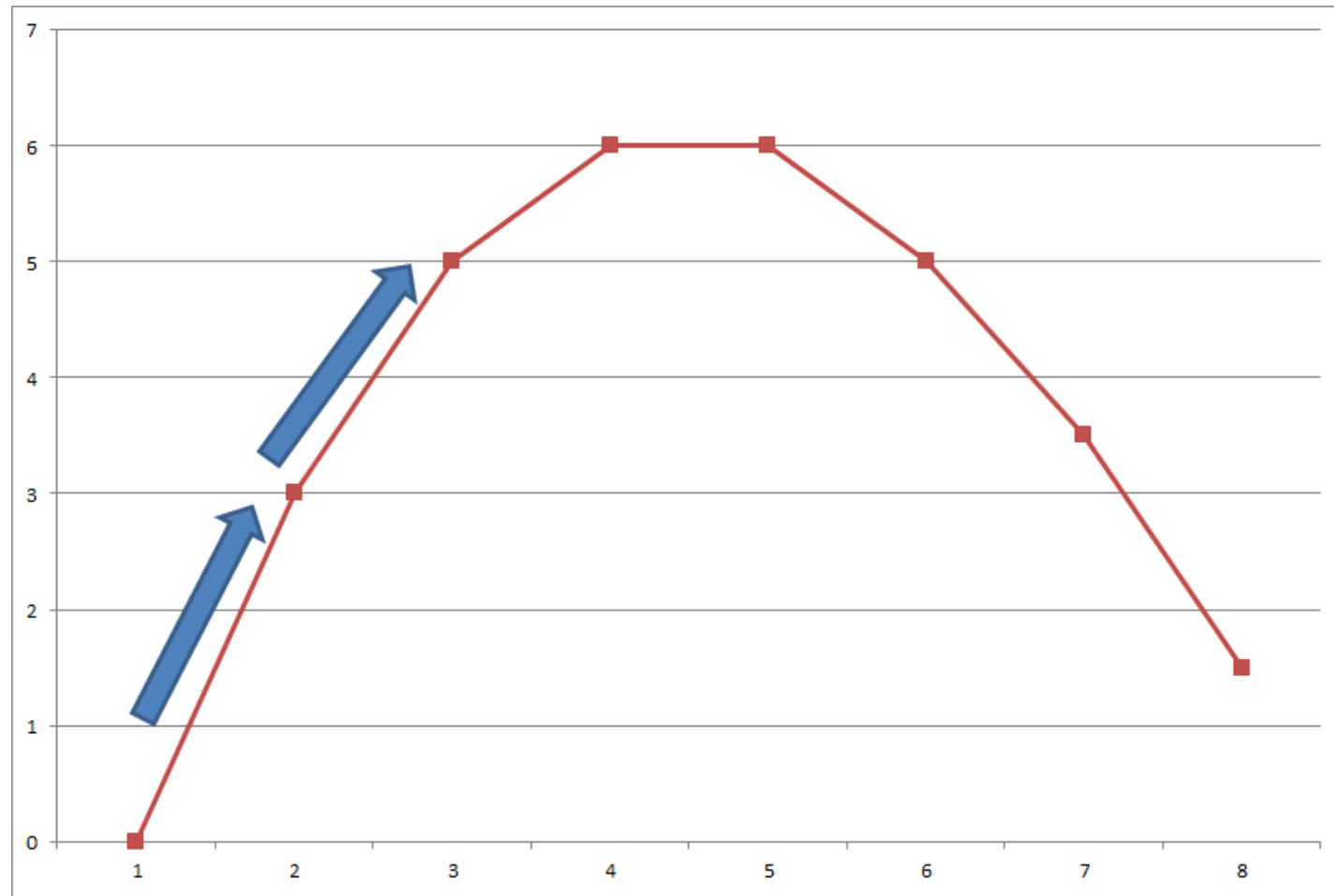
# Interpretation of Autocorrelation

- 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