



The DataReader: Access financial data online





pandas_datareader

- Easy access to various financial Internet data sources
- Little code needed to import into a pandas DataFrame
- Available sources include:
 - Yahoo! and Google Finance (including derivatives)
 - Federal Reserve
 - World Bank, OECD, Eurostat
 - OANDA



Stock prices: Google Finance

```
In [1]: from pandas_datareader.data import DataReader
In [2]: from datetime import date # Date & time functionality
In [3]: start = date(2015, 1, 1) # Default: Jan 1, 2010
In [4]: end = date(2016, 12, 31) # Default: today
In [5]: ticker = 'G00G'
In [6]: data_source = 'google'
In [7]: stock_data = DataReader(ticker, data_source, start, end)
```





Stock prices: Google Finance (2)



Stock prices: Google Finance (3)

```
In [10]: pd.concat([stock_data.head(3), stock_data.tail(3)])
Out[10]:
                      High
                                      Close
                                              Volume
              0pen
                                Low
Date
            529.01
2015-01-02
                            524.10
                                     524.81
                    531.27
                                             1446662
2015-01-05
            523.26
                    524.33
                            513.06
                                     513.87
                                             2054238
2015-01-06
            515.00
                    516.18
                            501.05
                                     501.96
                                             2891950
2016-12-28
                                     785.05
            793.70
                    794.23
                            783.20
                                             1153824
2016-12-29
            783.33
                                             744272
                    785.93
                            778.92
                                     782.79
2016-12-30
            782.75
                    782.78
                             770.41
                                     771.82
                                             1769950
```

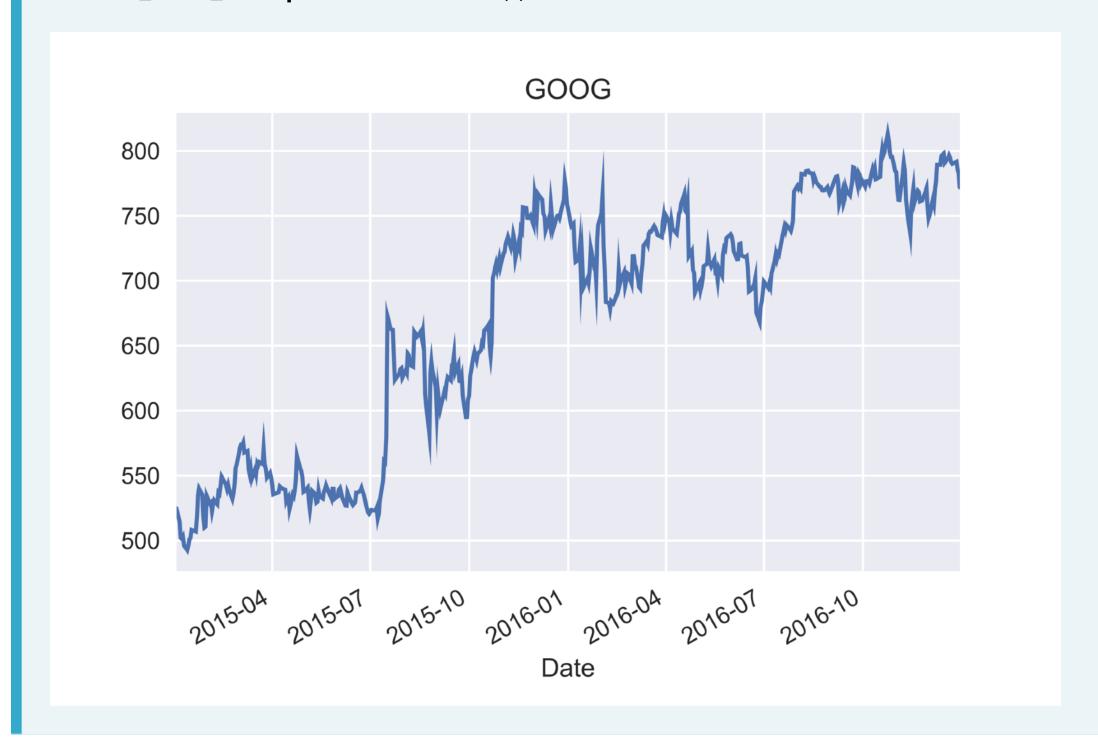


Stock prices: Visualization

```
In [11]: import matplotlib.pyplot as plt
```

In [12]: stock_data['Close'].plot(title=ticker)

In [13]: plt.show()







Let's practice!





Economic data from the Federal Reserve





Economic data from FRED

Federal Reserve Economic Data

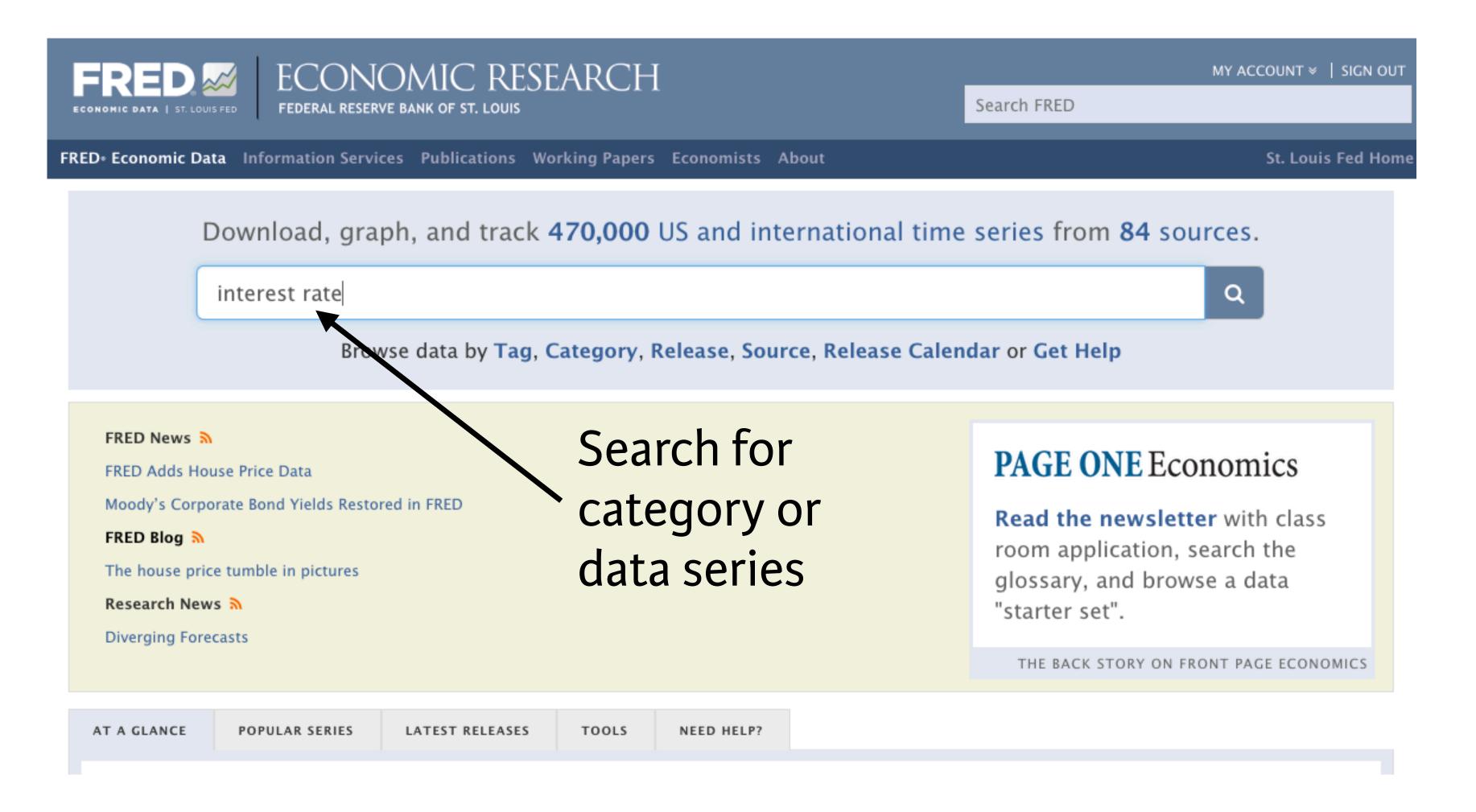


- 500,000 series covering a range of categories:
 - Economic growth & employment
 - Monetary & fiscal policy
 - Demographics, industries, commodity prices
 - Daily, monthly, annual frequencies





Get data from FRED







Get data from FRED (2)







Get data from FRED (3)







Interest rates:

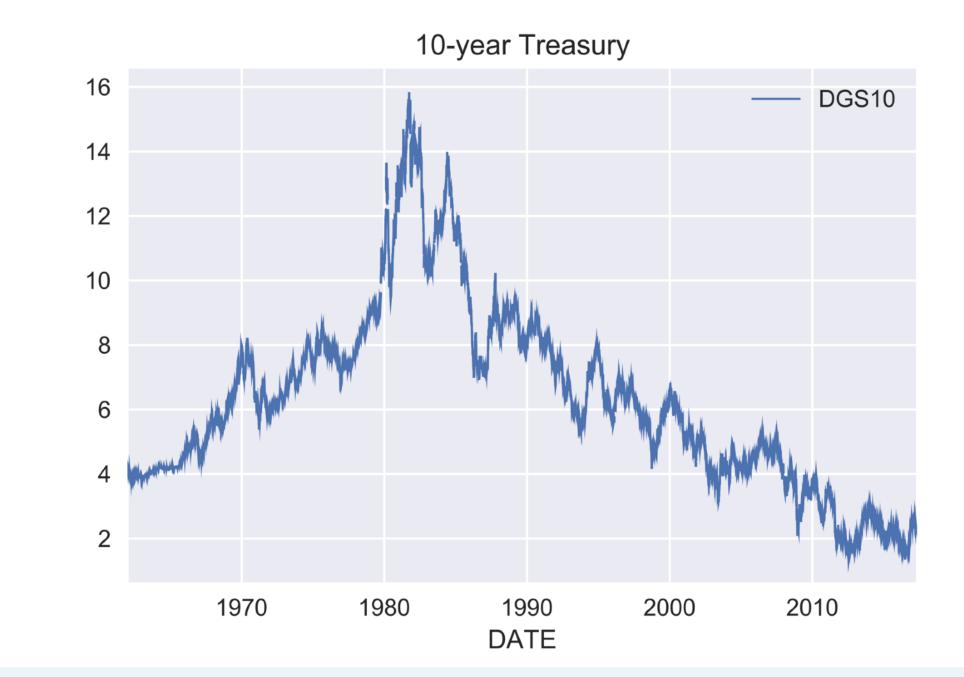


```
In [1]: from pandas_datareader.data import DataReader
In [2]: from datetime import date
In [3]: series_code = 'DGS10' # 10-year Treasury Rate
In [4]: data_source = 'fred' # FED Economic Data Service
In [5]: start = date(1962, 1, 1)
In [7]: data = DataReader(series_code, data_source, start)
In [8]: data.info()
DatetimeIndex: 14439 entries, 1962-01-02 to 2017-05-05 # latest
Data columns (total 1 columns):
       13821 non-null float64
DGS10
dtypes: float64(1)
```



Stock prices: Visualization

```
In [9]: series_name = '10-year Treasury'
In [10]: data = data.rename(columns={series_code: series_name})
In [11]: data.plot(title=series_name); plt.show()
```



.rename():

- Columns or index
- dictionary: {old_name: new_name}



Combine stock & economic data

```
In [13]: start = date(2000, 1, 1)
In [14]: series = 'DCOILWTICO' # West Texas Intermediate Oil Price
In [15]: oil = DataReader(series, 'fred', start)
In [16]: ticker = 'XOM' # Exxon Mobile Corporation
In [17]: stock = DataReader(ticker, 'google', start)
In [18]: data = pd.concat([stock[['Close']], oil], axis=1)
In [19]: data.info()
DatetimeIndex: 4526 entries, 2000-01-03 to 2017-05-08
Data columns (total 2 columns):
           4364 non-null float64
Close
DCOILWTICO 4352 non-null float64
```



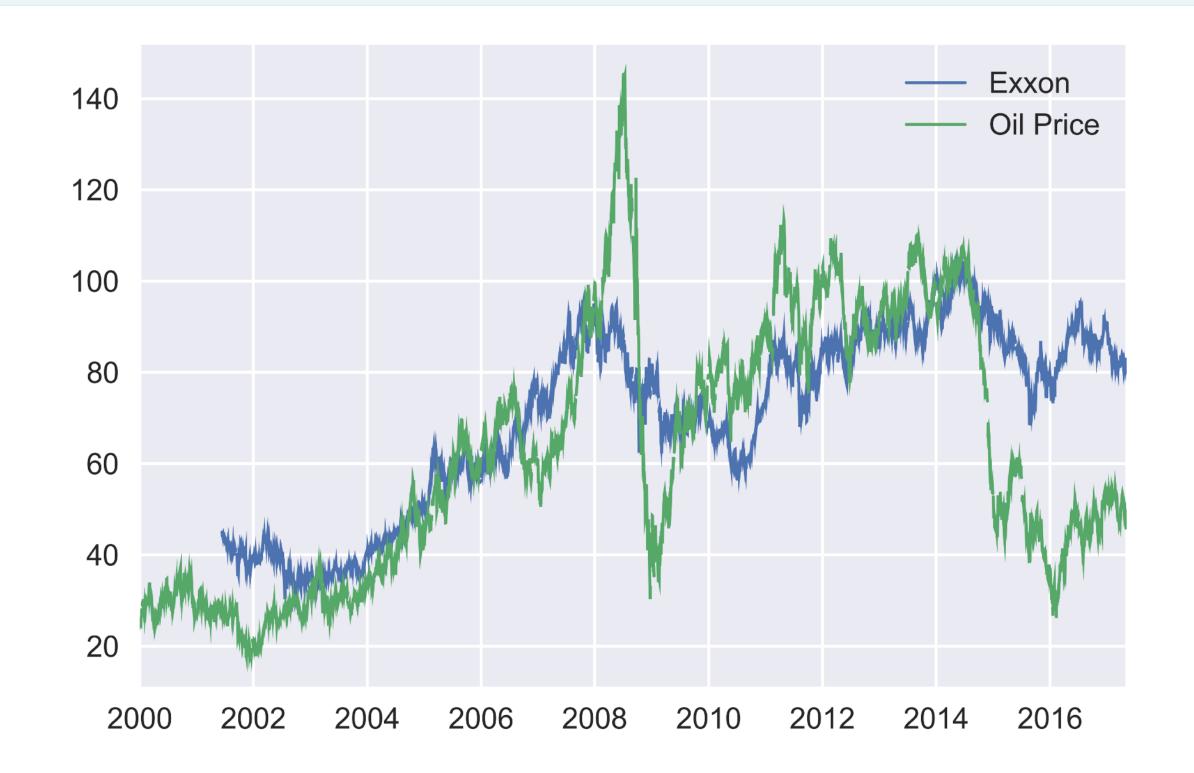


Combine stock & economic data (2)

```
In [13]: data.columns = ['Exxon', 'Oil Price']
```

In [14]: data.plot(); plt.show()

.columns: Assign labels as list







Let's practice!





Select stocks and get data from Google Finance





Select stocks based on criteria

- Use the listing information to select specific stocks
- As criteria:
 - Stock Exchange
 - Sector or Industry
 - IPO Year
 - Market Capitalization





Get ticker for largest company

```
In [1]: nyse = pd.read_excel('listings.xlsx', sheetname='nyse',
                            na_values='n/a')
In [2]: nyse = nyse.sort_values('Market Capitalization', ascending=False)
In [3]: nyse[['Stock Symbol', 'Company Name']].head(3)
Out[4]:
     Stock Symbol
                             Company Name
                        Johnson & Johnson
             JNJ
1586
             XOM Exxon Mobil Corporation
1125
             JPM
                    J P Morgan Chase & Co
1548
In [5]: largest_by_market_cap = nyse.iloc[0] # 1st row
In [6]: largest_by_market_cap['Stock Symbol'] # Select row label
'CNC'
```





Get ticker for largest company (2)

```
In [7]: nyse = nyse.set_index('Stock Symbol') # Stock ticker as index
In [8]: nyse.info()
Out[9]:
Index: 3147 entries, JNJ to EAE
Data columns (total 6 columns):
Company Name 3147 non-null object
Last Sale
         3147 non-null object
Market Capitalization 3147 non-null float64
         3147 non-null object
IPO Year
          3147 non-null object
Sector
         3147 non-null object
Industry
dtypes: float64(1), object(5)
In [10]: nyse['Market Capitalization'].idxmax() # Index of max value
'JNJ'
```





Get ticker for largest tech company

```
In [11]: nyse['Sector'].unique() # Unique values as numpy array
Out[11]:
array(['Technology', 'Health Care', 'Consumer Services', 'Capital
       Goods', 'Consumer Durables', nan, 'Finance', ..., 'Energy',
       'Transportation'], dtype=object)
In [12]: tech = nyse.loc[nyse.Sector=='Technology']
                                                       Filter condition:
                                                         <=, <
In [13]: tech['Company Name'].head(2)
Out[13]:
                                                         >=, >
                                    Company Name
Stock Symbol
ORCL
                              Oracle Corporation
             Taiwan Semiconductor Manufacturing
TSM
In [14]: nyse.loc[nyse.Sector=='Technology',
                                'Market Capitalization'].idxmax()
'ORCL'
```



Get data for largest tech company with 2017 IPO

Multiple conditions:

- Parentheses
- Logical operators





Visualize price & volume on two axes

```
[20]: import matplotlib.pyplot as plt
   [21]: data.plot(title=ticker, secondary_y='Volume')
In [22]: plt.tight_layout(); plt.show()
                   SNAP
                                               secondary_y:
                                              Column on right axis
26
                                               with different scale
                                        1.5
24
                                               plt.tight_layout:
                                        1.0
22
                                               Reduce whitespace
20
                                        0.5
```





Let's practice!





Get several stocks & manage a MultiIndex





Get data for several stocks

- Use the listing information to select multiple stocks
 - E.g. largest 3 stocks per sector
- Use Google Finance to retrieve data for several stocks
- Learn how to manage a pandas MultiIndex, a powerful tool to deal with more complex data sets



Load prices for top 5 companies

```
In [1]: nasdaq = pd.read_excel('listings.xlsx', sheetname='nasdaq', na_values='n/a')
In [2]: nasdaq.set_index('Stock Symbol', inplace=True)
In [3]: top_5 = nasdaq['Market Capitalization'].nlargest(n=5) # Top 5
In [4]: top_5.div(1000000) # Market Cap in million USD
Stock Symbol
AAPL
        740024.467000
GOOG
        569426.124504
MSFT 501903.061809
AMZN 422138.530626
FB
   402834.498146
Name: Market Capitalization, dtype: float64
In [5]: tickers = top_5.index.tolist() # Convert index to list
   ['AAPL', 'GOOG', 'MSFT', 'AMZN', 'FB']
```



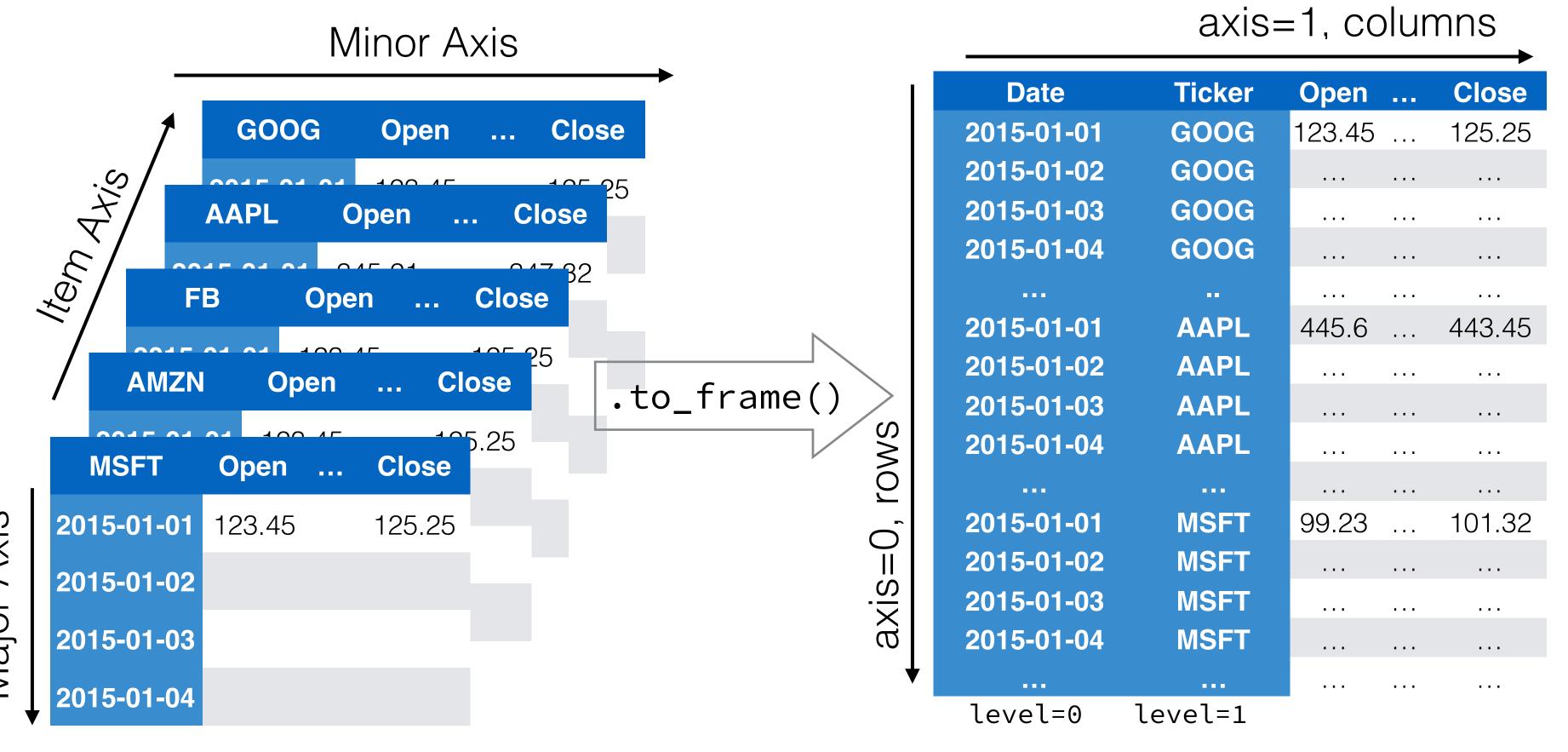
Load prices for top 5 companies (2)

```
In [6]: panel = DataReader(tickers, 'google', start=date(2015, 1, 1))
<class 'pandas.core.panel.Panel'>
Dimensions: 5 (items) x 591 (major_axis) x 5 (minor_axis)
Items axis: Open to Volume
Major_axis axis: 2015-01-02 to 2017-05-08
Minor_axis axis: AAPL to MSFT
In [7]: data = panel.to_frame()
In [9]: data.info()
MultiIndex: 2955 entries, (2015-01-02, AAPL) to (2017-05-08, MSFT)
Data columns (total 5 columns):
         2955 non-null float64
0pen
High
       2955 non-null float64
Low
         2955 non-null float64
          2955 non-null float64
Close
Volume
          2955 non-null float64
```



Into higher dimensions: MultiIndex

.to_frame():from pd.Panel() to pd.DataFrame()



pd.MultiIndex()





Reshape your data: .unstack()

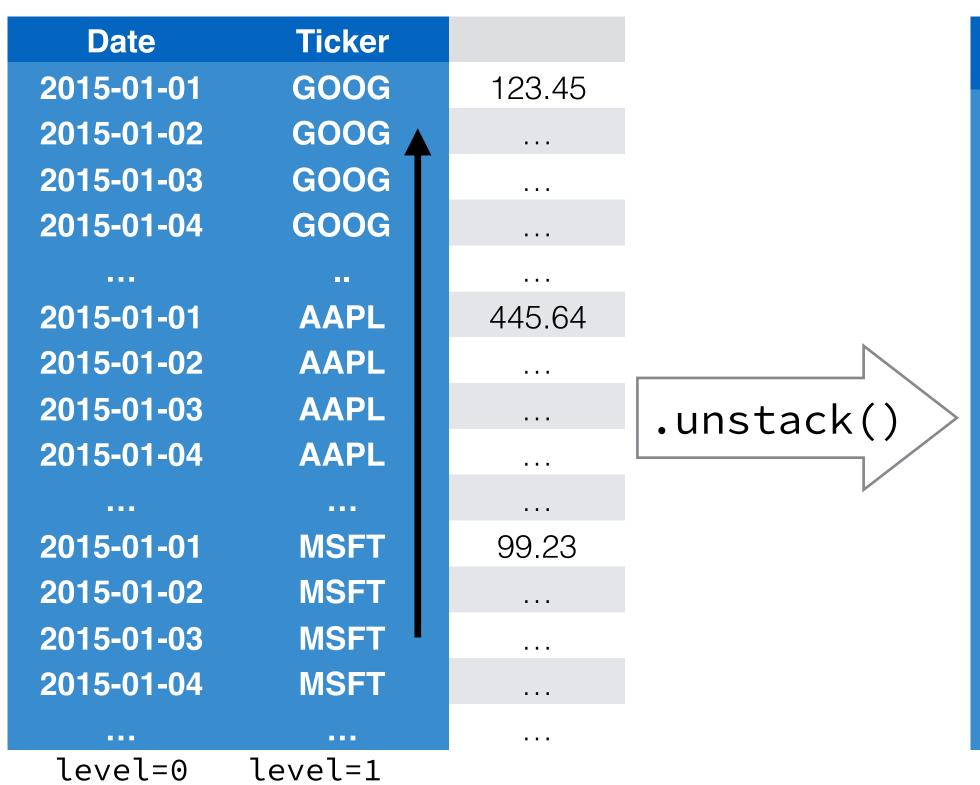
```
In [10]: unstacked = data['Close'].unstack()
In [11]: unstacked.info()
DatetimeIndex: 591 entries, 2015-01-02 to 2017-05-08
Data columns (total 5 columns):
AAPL 591 non-null float64
AMZN 591 non-null float64
FB 591 non-null float64
GOOG 591 non-null float64
MSFT 591 non-null float64
dtypes: float64(5)
memory usage: 27.7 KB
```





From long to wide format

In [10]: unstacked = data['Close'].unstack() # Results in pd.DataFrame()



					
Date	GOOG	AAPL	AMZN	FB	MSFT
2015-01-01	123.45	445.64			99.23
2015-01-02					
2015-01-03					
2015-01-04					
2017-05-01					
2017-05-02					
2017-05-03					
2017-05-04					

pd.MultiIndex()



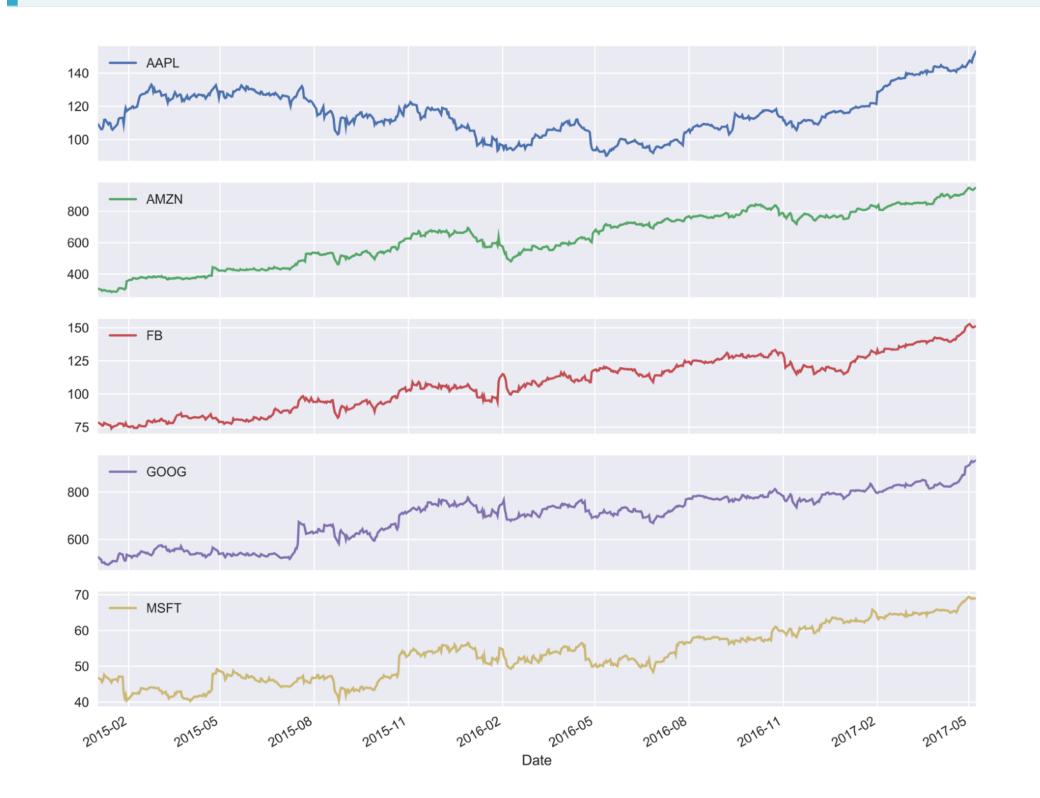
Stock prices: Visualization

In [19]: unstacked.plot(subplots=True)

In [20]: plt.tight_layout(); plt.show()

subplots:

Display each column on separate chart







Let's practice!