Select index components & import data

MANIPULATING TIME SERIES DATA IN PYTHON

Stefan Jansen
Founder & Lead Data Scientist at Applied Artificial Intelligence

Market value-weighted index

- Composite performance of various stocks
- Components weighted by market capitalization
  - `Share Price` x `Number of Shares` => `Market Value`
- Larger components get higher percentage weightings
- Key market indexes are value-weighted:
  - `S&P 500` , `NASDAQ` , `Wilshire 5000` , `Hang Seng`

Build a cap-weighted Index

- Apply new skills to construct value-weighted index
  - Select components from exchange listing data
  - Get component number of shares and stock prices
  - Calculate component weights
  - Calculate index
  - Evaluate performance of components and index

Load stock listing data

```
nyse = pd.read_excel('listings.xlsx', sheetname='nyse',
                     na_values='n/a')

nyse.info()
```

```
RangeIndex: 3147 entries, 0 to 3146
Data columns (total 7 columns):
Stock Symbol              3147 non-null object   # Stock Ticker
Company Name              3147 non-null object
Last Sale                 3079 non-null float64  # Latest Stock Price
Market Capitalization     3147 non-null float64
IPO Year                  1361 non-null float64  # Year of listing
Sector                    2177 non-null object
Industry                  2177 non-null object
dtypes: float64(3), object(4)
```

Load & prepare listing data

```
nyse.set_index('Stock Symbol', inplace=True)
nyse.dropna(subset=['Sector'], inplace=True)  remove companies without sector information
nyse['Market Capitalization'] /= 1e6  # in Million USD
```

```
Index: 2177 entries, DDD to ZTO
Data columns (total 6 columns):
Company Name              2177 non-null object
Last Sale                 2175 non-null float64
Market Capitalization     2177 non-null float64
IPO Year                  967 non-null float64
Sector                    2177 non-null object
Industry                  2177 non-null object
dtypes: float64(3), object(3)
```

Select index components

```
components = nyse.groupby(['Sector'])['Market Capitalization'].nlargest(1)
components.sort_values(ascending=False)
```

```
Sector                 Stock Symbol
Health Care            JNJ             338834.390080
Energy                 XOM             338728.713874
Finance                JPM             300283.250479
Miscellaneous          BABA            275525.000000
Public Utilities       T               247339.517272
Basic Industries       PG              230159.644117
Consumer Services      WMT             221864.614129
Consumer Non-Durables  KO              183655.305119
Technology             ORCL            181046.096000
Capital Goods          TM              155660.252483
Transportation         UPS              90180.886756
Consumer Durables      ABB              48398.935676
Name: Market Capitalization, dtype: float64
```

Import & prepare listing data

```
tickers = components.index.get_level_values('Stock Symbol')
tickers
```

select ‘Stock Symbol’ from the index of components

```
Index(['PG', 'TM', 'ABB', 'KO', 'WMT', 'XOM', 'JPM', 'JNJ', 'BABA', 'T',
       'ORCL', ‘UPS'], dtype='object', name='Stock Symbol’)
```

```
tickers.tolist()
```

```
['PG',
 'TM',
 'ABB',
 'KO',
 'WMT',
...
 'T',
 'ORCL',
 'UPS']
```

Stock index components

```
columns = ['Company Name', 'Market Capitalization', 'Last Sale']
component_info = nyse.loc[tickers, columns]
pd.options.display.float_format = '{:,.2f}'.format
```

Stock Symbol	Company Name	Market Capitalization	Last Sale
PG	Procter & Gamble Company (The)	230,159.64	90.03
TM	Toyota Motor Corp Ltd Ord	155,660.25	104.18
ABB	ABB Ltd	48,398.94	22.63
KO	Coca-Cola Company (The)	183,655.31	42.79
WMT	Wal-Mart Stores, Inc.	221,864.61	73.15
XOM	Exxon Mobil Corporation	338,728.71	81.69
JPM	J P Morgan Chase & Co	300,283.25	84.40
JNJ	Johnson & Johnson	338,834.39	124.99
BABA	Alibaba Group Holding Limited	275,525.00	110.21
T	AT&T Inc.	247,339.52	40.28
ORCL	Oracle Corporation	181,046.10	44.00
UPS	United Parcel Service, Inc.	90,180.89	103.74

Import & prepare listing data

```
data = pd.read_csv('stocks.csv', parse_dates=['Date'],
                               index_col='Date').loc[:, tickers.tolist()]
data.info()
```

```
DatetimeIndex: 252 entries, 2016-01-04 to 2016-12-30
Data columns (total 12 columns):
ABB     252 non-null float64
BABA    252 non-null float64
JNJ     252 non-null float64
JPM     252 non-null float64
KO      252 non-null float64
ORCL    252 non-null float64
PG      252 non-null float64
T       252 non-null float64
TM      252 non-null float64
UPS     252 non-null float64
WMT     252 non-null float64
XOM     252 non-null float64
dtypes: float64(12)
```

Let's practice!

MANIPULATING TIME SERIES DATA IN PYTHON

Build a market-cap weighted index

MANIPULATING TIME SERIES DATA IN PYTHON

Stefan Jansen

Founder & Lead Data Scientist at Applied Artificial Intelligence

Build your value-weighted index

- Key inputs:
  - number of shares
  - stock price series

Build your value-weighted index

- Key inputs:
  - number of shares
  - stock price series
  - Normalize index to start at 100

Stock index components

```
components
```

Stock Symbol	Company Name	Market Capitalization	Last Sale
PG	Procter & Gamble Company (The)	230,159.64	90.03
TM	Toyota Motor Corp Ltd Ord	155,660.25	104.18
ABB	ABB Ltd	48,398.94	22.63
KO	Coca-Cola Company (The)	183,655.31	42.79
WMT	Wal-Mart Stores, Inc.	221,864.61	73.15
XOM	Exxon Mobil Corporation	338,728.71	81.69
JPM	J P Morgan Chase & Co	300,283.25	84.40
JNJ	Johnson & Johnson	338,834.39	124.99
BABA	Alibaba Group Holding Limited	275,525.00	110.21
T	AT&T Inc.	247,339.52	40.28
ORCL	Oracle Corporation	181,046.10	44.00
UPS	United Parcel Service, Inc.	90,180.89	103.74

Number of shares outstanding

```
shares = components['Market Capitalization'].div(components['Last Sale'])
```

```
Stock Symbol
PG     2,556.48 # Outstanding shares in million
TM     1,494.15
ABB    2,138.71
KO     4,292.01
WMT    3,033.01
XOM    4,146.51
JPM    3,557.86
JNJ    2,710.89
BABA   2,500.00
T      6,140.50
ORCL   4,114.68
UPS      869.30
dtype: float64
```

- Market Capitalization = Number of Shares x Share Price

Historical stock prices

```
data = pd.read_csv('stocks.csv', parse_dates=['Date'],
                           index_col='Date').loc[:, tickers.tolist()]
market_cap_series = data.mul(no_shares)
market_series.info()
```

```
DatetimeIndex: 252 entries, 2016-01-04 to 2016-12-30
Data columns (total 12 columns):
ABB     252 non-null float64
BABA    252 non-null float64
JNJ     252 non-null float64
JPM     252 non-null float64
...
TM      252 non-null float64
UPS     252 non-null float64
WMT     252 non-null float64
XOM     252 non-null float64
dtypes: float64(12)
```

From stock prices to market value

```
market_cap_series.first('D').append(market_cap_series.last('D'))
```

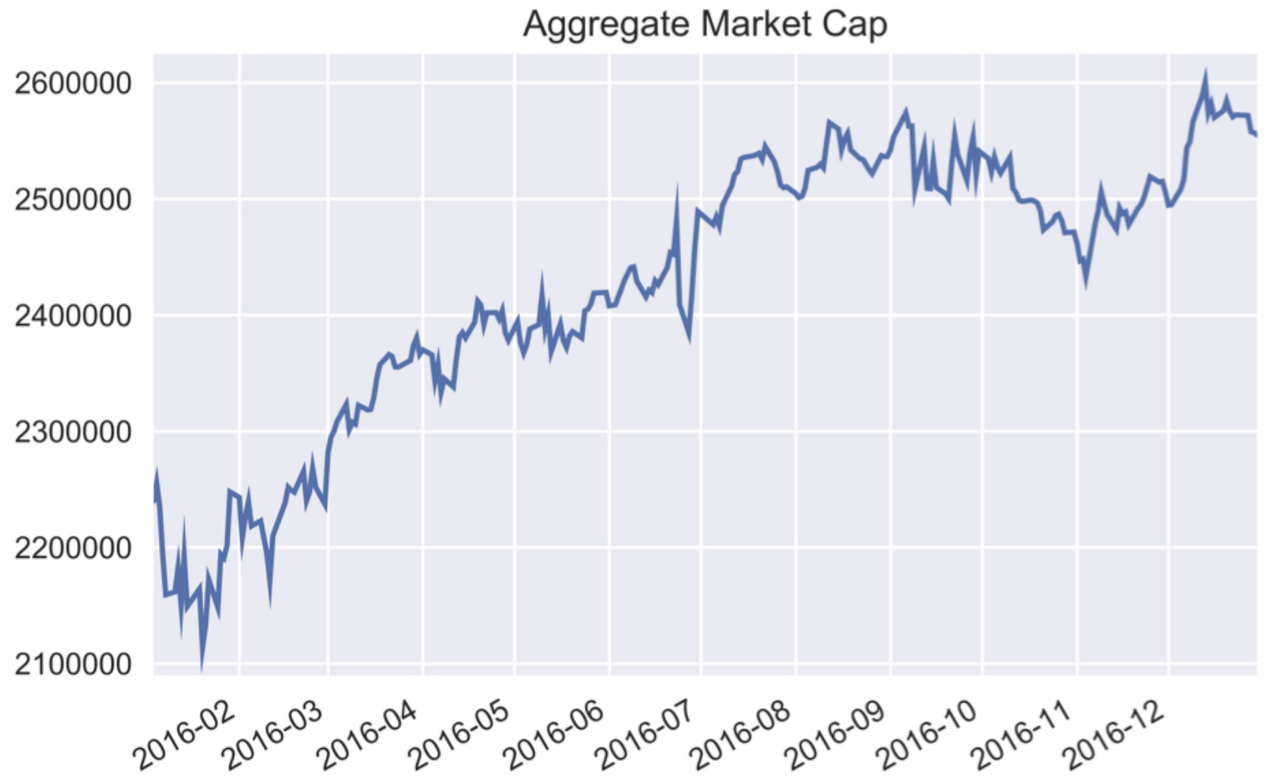
```
                ABB       BABA        JNJ        JPM         KO       ORCL  \\
Date
2016-01-04 37,470.14 191,725.00 272,390.43 226,350.95 181,981.42 147,099.95
2016-12-30 45,062.55 219,525.00 312,321.87 307,007.60 177,946.93 158,209.60
                 PG          T          TM       UPS        WMT         XOM
Date
2016-01-04 200,351.12 210,926.33 181,479.12 82,444.14 186,408.74 321,188.96
2016-12-30 214,948.60 261,155.65 175,114.05 99,656.23 209,641.59 374,264.34
```

Aggregate market value per period

sum each row

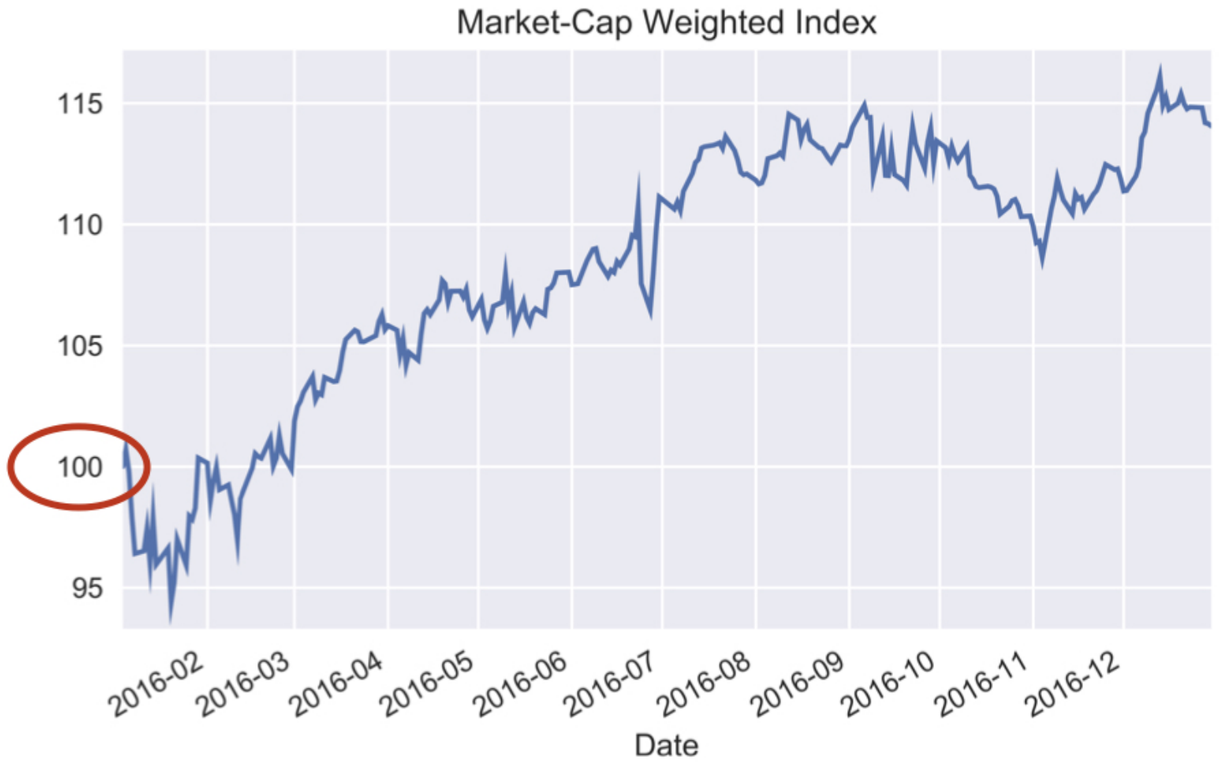
```
agg_mcap = market_cap_series.sum(axis=1) # Total market cap

agg_mcap(title='Aggregate Market Cap')
```



Value-based index

```
index = agg_mcap.div(agg_mcap.iloc[0]).mul(100) # Divide by 1st valu
index.plot(title='Market-Cap Weighted Index')
```



Let's practice!

MANIPULATING TIME SERIES DATA IN PYTHON

Evaluate index performance

MANIPULATING TIME SERIES DATA IN PYTHON

Stefan Jansen

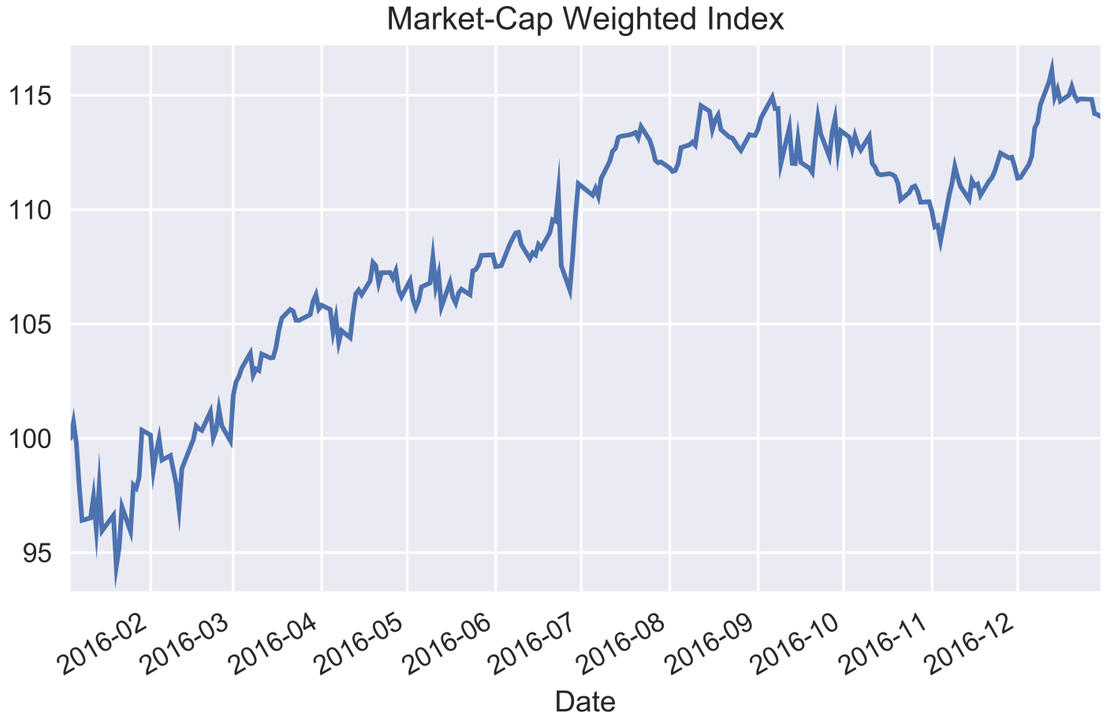
Founder & Lead Data Scientist at Applied Artificial Intelligence

Evaluate your value-weighted index

- Index return:
  - Total index return
  - Contribution by component
- Performance vs Benchmark
  - Total period return
  - Rolling returns for sub periods

Value-based index - recap

```
agg_market_cap = market_cap_series.sum(axis=1)
index = agg_market_cap.div(agg_market_cap.iloc[0]).mul(100)
index.plot(title='Market-Cap Weighted Index')
```



Value contribution by stock

```
agg_market_cap.iloc[-1] - agg_market_cap.iloc[0]
```

```
315,037.71
```

Value contribution by stock

```
change = market_cap_series.first('D').append(market_cap_series.last('D'))
change.diff().iloc[-1].sort_values() # or: .loc['2016-12-30']
```

```
TM     -6,365.07
KO     -4,034.49
ABB     7,592.41
ORCL   11,109.65
PG     14,597.48
UPS    17,212.08
WMT    23,232.85
BABA   27,800.00
JNJ    39,931.44
T      50,229.33
XOM    53,075.38
JPM    80,656.65
Name: 2016-12-30 00:00:00, dtype: float64
```

Market-cap based weights

```
market_cap = components['Market Capitalization']
weights = market_cap.div(market_cap.sum())
weights.sort_values().mul(100)
```

```
Stock Symbol
ABB     1.85
UPS     3.45
TM      5.96
ORCL    6.93
KO      7.03
WMT     8.50
PG      8.81
T       9.47
BABA   10.55
JPM    11.50
XOM    12.97
JNJ    12.97
Name: Market Capitalization, dtype: float64
```

Value-weighted component returns

```
index_return = (index.iloc[-1] / index.iloc[0] - 1) * 100
```

```
14.06
```

```
weighted_returns = weights.mul(index_return)
weighted_returns.sort_values().plot(kind='barh')
```

Let's practice!

MANIPULATING TIME SERIES DATA IN PYTHON

Index correlation & exporting to Excel

MANIPULATING TIME SERIES DATA IN PYTHON

Stefan Jansen

Founder & Lead Data Scientist at Applied Artificial Intelligence

Some additional analysis of your index

- Daily return correlations:
- Calculate among all components
- Visualize the result as heatmap
- Write results to excel using `.xls` and `.xlsx` formats:
- Single worksheet
- Multiple worksheets

Index components - price data

```
data = DataReader(tickers, 'google', start='2016', end='2017')['Close']
data.info()
```

```
DatetimeIndex: 252 entries, 2016-01-04 to 2016-12-30
Data columns (total 12 columns):
ABB     252 non-null float64
BABA    252 non-null float64
JNJ     252 non-null float64
JPM     252 non-null float64
KO      252 non-null float64
ORCL    252 non-null float64
PG      252 non-null float64
T       252 non-null float64
TM      252 non-null float64
UPS     252 non-null float64
WMT     252 non-null float64
XOM     252 non-null float64
```

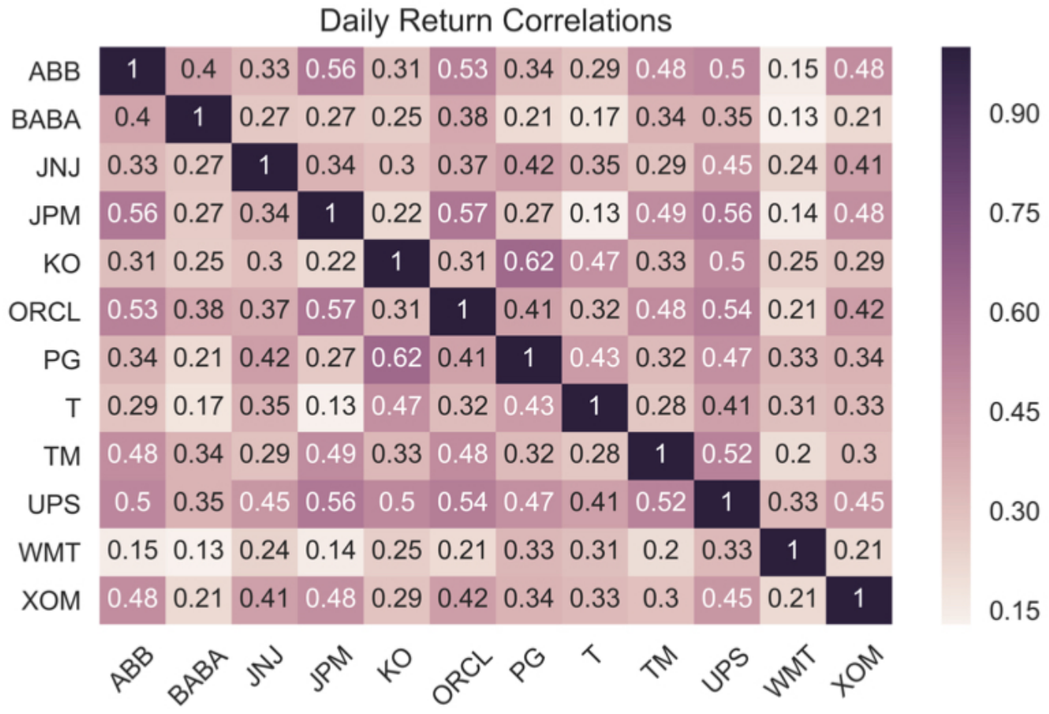
Index components: return correlations

```
daily_returns = data.pct_change()
correlations = daily_returns.corr()
```

```
ABB   BABA   JNJ   JPM    KO   ORCL    PG     T    TM   UPS   WMT   XOM
ABB   1.00  0.40 0.33 0.56 0.31  0.53 0.34 0.29 0.48 0.50 0.15 0.48
BABA 0.40  1.00 0.27 0.27 0.25  0.38 0.21 0.17 0.34 0.35 0.13 0.21
JNJ   0.33  0.27 1.00 0.34 0.30  0.37 0.42 0.35 0.29 0.45 0.24 0.41
JPM   0.56  0.27 0.34 1.00 0.22  0.57 0.27 0.13 0.49 0.56 0.14 0.48
KO    0.31  0.25 0.30 0.22 1.00  0.31 0.62 0.47 0.33 0.50 0.25 0.29
ORCL 0.53  0.38 0.37 0.57 0.31  1.00 0.41 0.32 0.48 0.54 0.21 0.42
PG    0.34  0.21 0.42 0.27 0.62  0.41 1.00 0.43 0.32 0.47 0.33 0.34
T     0.29  0.17 0.35 0.13 0.47  0.32 0.43 1.00 0.28 0.41 0.31 0.33
TM    0.48  0.34 0.29 0.49 0.33  0.48 0.32 0.28 1.00 0.52 0.20 0.30
UPS   0.50  0.35 0.45 0.56 0.50  0.54 0.47 0.41 0.52 1.00 0.33 0.45
WMT   0.15  0.13 0.24 0.14 0.25  0.21 0.33 0.31 0.20 0.33 1.00 0.21
XOM   0.48  0.21 0.41 0.48 0.29  0.42 0.34 0.33 0.30 0.45 0.21 1.00
```

Index components: return correlations

```
sns.heatmap(correlations, annot=True)
plt.xticks(rotation=45)
plt.title('Daily Return Correlations')
```



Saving to a single Excel worksheet

```
correlations.to_excel(excel_writer= 'correlations.xls',
                      sheet_name='correlations',
                      startrow=1,
                      startcol=1)
```

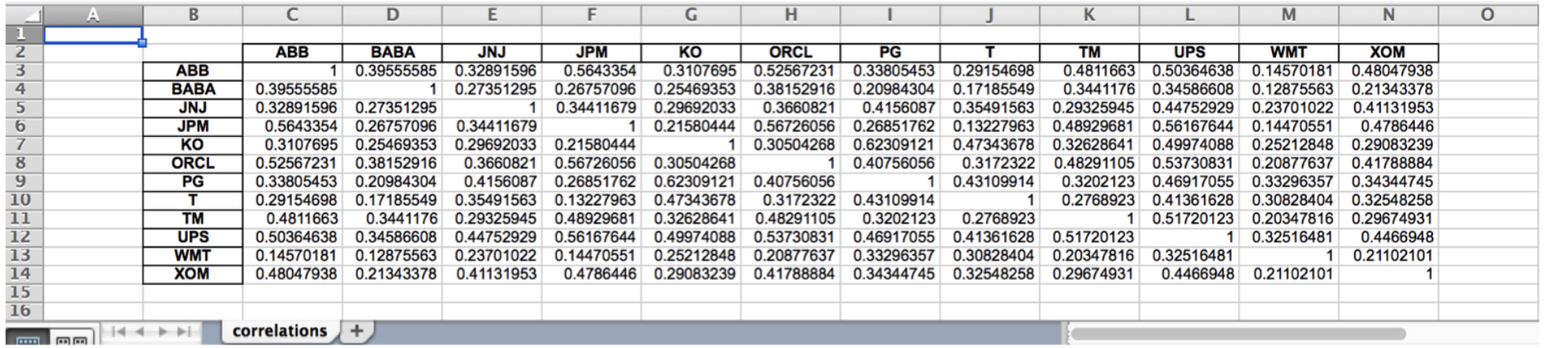


	ABB	BABA	JNJ	JPM	KO	ORCL	PG	T	TM	UPS	WMT	XOM
ABB	1	0.39555585	0.32891596	0.5643354	0.3107695	0.52567231	0.33805453	0.29154698	0.4811663	0.50364638	0.14570181	0.48047938
BABA	0.39555585	1	0.27351295	0.26757096	0.25469353	0.38152916	0.20984304	0.17185549	0.3441176	0.34586608	0.12875563	0.21343378
JNJ	0.32891596	0.27351295	1	0.34411679	0.29692033	0.3660821	0.4156087	0.35491563	0.29325945	0.44752929	0.23701022	0.41131953
JPM	0.5643354	0.26757096	0.34411679	1	0.21580444	0.56726056	0.26851762	0.13227963	0.48929681	0.56167644	0.14470551	0.4786446
KO	0.3107695	0.25469353	0.29692033	0.21580444	1	0.30504268	0.62309121	0.47343678	0.32628641	0.49974088	0.25212848	0.29083239
ORCL	0.52567231	0.38152916	0.3660821	0.56726056	0.30504268	1	0.40756056	0.3172322	0.48291105	0.53730831	0.20877637	0.41788884
PG	0.33805453	0.20984304	0.4156087	0.26851762	0.62309121	0.40756056	1	0.43109914	0.3202123	0.46917055	0.33296357	0.34344745
T	0.29154698	0.17185549	0.35491563	0.13227963	0.47343678	0.3172322	0.43109914	1	0.2768923	0.41361628	0.30828404	0.32548258
TM	0.4811663	0.3441176	0.29325945	0.48929681	0.32628641	0.48291105	0.3202123	0.2768923	1	0.51720123	0.20347816	0.29674931
UPS	0.50364638	0.34586608	0.44752929	0.56167644	0.49974088	0.53730831	0.46917055	0.41361628	0.51720123	1	0.32516481	0.4466948
WMT	0.14570181	0.12875563	0.23701022	0.14470551	0.25212848	0.20877637	0.33296357	0.30828404	0.20347816	0.32516481	1	0.21102101
XOM	0.48047938	0.21343378	0.41131953	0.4786446	0.29083239	0.41788884	0.34344745	0.32548258	0.29674931	0.4466948	0.21102101	1

Saving to multiple Excel worksheets

```
data.index = data.index.date # Keep only date component
with pd.ExcelWriter('stock_data.xlsx') as writer:
    corr.to_excel(excel_writer=writer, sheet_name='correlations')
    data.to_excel(excel_writer=writer, sheet_name='prices')
    data.pct_change().to_excel(writer, sheet_name='returns')
```

	A	B	C	D	E	F	G	H	I	J	K	L	M
1		ABB	BABA	JNJ	JPM	KO	ORCL	PG	T	TM	UPS	WMT	XOM
2	2016-01-04	17.52	76.69	100.48	63.62	42.4	35.75	78.37	34.35	121.46	94.84	61.46	77.46
3	2016-01-05	17.21	78.63	100.9	63.73	42.55	35.64	78.62	34.59	121.14	95.78	62.92	78.12
4	2016-01-06	16.92	77.33	100.39	62.81	42.32	35.82	77.86	34.06	118.38	94.42	63.55	77.47
5	2016-01-07	16.6	72.72	99.22	60.27	41.62	35.04	77.18	33.51	115.57	92.6	65.03	76.23
6	2016-01-08	16.31	70.8	98.16	58.92	41.51	34.65	75.97	33.54	113.06	91.39	63.54	74.69
7	2016-01-11	16.31	69.92	97.57	58.83	41.58	34.94	76.67	33.95	114.81	91.66	64.22	73.69
8	2016-01-12	16.66	72.68	98.24	58.96	42.12	35.37	76.51	33.9	115.83	93	63.62	75.2
9	2016-01-13	16.3	70.29	97.02	57.34	41.85	34.08	75.85	33.74	114.99	90.61	61.92	75.65
10	2016-01-14	16.73	72.25	98.89	58.2	41.88	34.79	76.15	34.3	116.29	91.15	63.06	79.12
11	2016-01-15	16.06	69.59	97	57.04	41.5	34.12	74.98	33.99	112.6	90.04	61.93	77.58
12	2016-01-19	16.26	70.13	97.5	57.01	41.92	34.55	76.73	34.51	115.02	90.35	62.56	76.4

correlations | prices | returns | +

Let's practice!

MANIPULATING TIME SERIES DATA IN PYTHON

Congratulations!

MANIPULATING TIME SERIES DATA IN PYTHON

Stefan Jansen

Founder & Lead Data Scientist at Applied Artificial Intelligence

Let's practice!

MANIPULATING TIME SERIES DATA IN PYTHON