Financial Data Visualization with Pandas

Preparation

- new libraries
- new pandas methods
- absolute and relative changes
- gross and net returns

Python libraries

- pandas
- matplotlib.pyplot
- pandas_datareader
- yfinance
- scipy.stats

Python libraries

We need to install libraries yfinance and pandas_datareader (not included with pandas) in jupyter notebook

To install them in Windows run

```
!pip install pandas_datareader
```

To install them in Mac OS use

```
pip install pandas_datareader
```

pandas methods

Method	Returns
`abs()`	Object with absolute values taken (of numerical data)
`argmax()`	The index label of the maximum value
`argmin()`	The index label of the minimum value
`count()`	The number of non-null entries
`cumprod()`	The cumulative product over an axis
`cumsum()`	The cumulative sum over an axis
`max()`	The maximum of the entries
`mean()`	The average of the entries
`median()`	The median of the entries
`min()`	The minimum of the entries
`mode()`	The most common element(s)
`prod()`	The product of the elements
`sum()`	The sum of the elements
`var()`	The variance of the elements

Table 7.1: Numerical methods of the `Series` and `DataFrame` pandas classes.

pandas methods

Method	Returns
`abs()`	Object with absolute values taken (of numerical data)
`argmax()`	The index label of the maximum value
`argmin()`	The index label of the minimum value
`count()`	The number of non-null entries
`cumprod()`	The cumulative product over an axis
`cumsum()`	The cumulative sum over an axis
`max()`	The maximum of the entries
`mean()`	The average of the entries
`median()`	The median of the entries
`min()`	The minimum of the entries
`mode()`	The most common element(s)
`prod()`	The product of the elements
`sum()`	The sum of the elements
`var()`	The variance of the elements

Table 7.1: Numerical methods of the `Series` and `DataFrame` pandas classes.

pandas methods

Method	Description
`append()`	Concatenate two or more `Series`.
`drop()`	Remove the entries with the specified label or labels
`drop_duplicates()`	Remove duplicate values
`dropna()`	Drop null entries
`fillna()`	Replace null entries with a specified value or strategy
`reindex()`	Replace the index
`sample()`	Draw a random entry
`shift()`	Shift the index
`unique()`	Return unique values

Table 7.2: Methods for managing or modifying data in a pandas `Series` or `DataFrame`.

pandas methods

Date	Stock Price	shift(1)
4/17/14	522.02	
4/21/14	528.22	522.02
4/22/14	528.75	528.22
4/23/14	521.84	528.75
4/24/14	564.62	521.84
4/25/14	568.76	564.62
4/28/14	590.79	568.76
4/29/14	589.04	590.79
		589.04

pandas methods

	A	B	C	D	E
1					
2		**Date**	**Stock Price**	**cumsum**	
3		4/17/14	522.02		
4		4/21/14	528.22	1050.24	=C3+C4
5		4/22/14	528.75	1578.99	=D4+C5
6		4/23/14	521.84	2100.83	=D5+C6
7		4/24/14	564.62	2665.45	=D6+C7
8		4/25/14	568.76	3234.21	=D7+C8
9		4/28/14	590.79	3825	=D8+C9
10		4/29/14	589.04	4414.04	=D9+C10

pandas methods

	A	B	C	D	E
1					
2		**Date**	**Price**	**cumprod**	
3		4/17/14	5.2202		
4		4/21/14	5.2822	27.574	=C3*C4
5		4/22/14	5.2875	145.798	=D4*C5
6		4/23/14	5.2184	760.834	=D5*C6
7		4/24/14	5.6462	4295.819	=D6*C7
8		4/25/14	5.6876	24432.901	=D7*C8
9		4/28/14	5.9079	144347.135	=D8*C9
10		4/29/14	5.8904	850262.363	=D9*C10

changes

- Absolute change
- Relative change
- Gross change
- Net change

changes

- gross change = $\frac{Value\ in\ period\ 2}{Value\ in\ period\ 1}$

- net change $=$ $\frac{Value\ in\ period\ 2 - Value\ in\ period\ 1}{Value\ in\ period\ 1}$

changes

- gross change = $\frac{Value\ in\ period\ 2}{Value\ in\ period\ 1}$

- net change = $\frac{Value\ in\ period\ 2 - Value\ in\ period\ 1}{Value\ in\ period\ 1}$

(relative change)

(absolute change)

changes

- gross change = $\frac{Value\ in\ period\ 2}{Value\ in\ period\ 1}$

- net change $=$ $\frac{Value\ in\ period\ 2 - Value\ in\ period\ 1}{Value\ in\ period\ 1}$

changes

- gross change = $\frac{Value\ in\ period\ 2}{Value\ in\ period\ 1}$
- net change = $\frac{Value\ in\ period\ 2}{Value\ in\ period\ 1}$ - 1

changes

- gross change = $\frac{Value\ in\ period\ 2}{Value\ in\ period\ 1}$
- net change = $\frac{Value\ in\ period\ 2}{Value\ in\ period\ 1}$ - 1
- gross change = net change - 1

returns

- gross return = $\frac{Value\ in\ period\ 2}{Value\ in\ period\ 1}$
- net return = $\frac{Value\ in\ period\ 2}{Value\ in\ period\ 1}$ - 1
- net return = gross return - 1

Example: stock's data

Example - Overview

- Download stock prices
- Plot time series data
- Find the most profitable stock

  (Using the Cumulative Gross returns)
- Find the most risky stock (Using Net returns)
- Use quantile plots to assess normality of net returns
- Use Net returns to find correlated stocks

Example 1 – Business questions

From a set of stocks, find

- Which one is more risky?
- Which one is more profitable?
- Which ones are highly (cor)related?

Example 1

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import pandas_datareader
import yfinance as yf
```

```
yf.pdr_override()
```

```
start_date = '2007-01-03'
end_date = '2014-05-16'
```

```
tickers = ['AAPL']
df_apple = pandas_datareader.data.get_data_yahoo(tickers,\
                                                 start_date,end_date,\
                                                 interval='1d')
```

Example 1

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import pandas_datareader
import yfinance as yf
```

```
yf.pdr_override()
```

```
start_date = '2007-01-03'
end_date = '2014-05-16'
```

```
tickers = ['AAPL']
df_apple = pandas_datareader.data.get_data_yahoo(tickers,\
                                                 start_date,end_date,\
                                                 interval='1d')

df_apple[:5]
```

daily data

```
[**********************100%**********************]  1 of 1 completed
```

Date	Open	High	Low	Close	Adj Close	Volume
2007-01-03	3.081786	3.092143	2.925000	2.992857	2.573566	1238319600
2007-01-04	3.001786	3.069643	2.993571	3.059286	2.630688	847260400
2007-01-05	3.063214	3.078571	3.014286	3.037500	2.611954	834741600
2007-01-08	3.070000	3.090357	3.045714	3.052500	2.624853	797106800
2007-01-09	3.087500	3.320714	3.041071	3.306071	2.842900	3349298400

Example 1

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import pandas_datareader
import yfinance as yf
```

```
yf.pdr_override()
```

```
start_date = '2007-01-03'
end_date = '2014-05-16'
```

```
tickers = ['AAPL']
df_apple = pandas_datareader.data.get_data_yahoo(tickers,\
                                                 start_date,end_date,\
                                                 interval='1d')

df_apple[:5]
```

```
[**********************100%************************]  1 of 1 completed
```

index

Date	Open	High	Low	Close	Adj Close	Volume
2007-01-03	3.081786	3.092143	2.925000	2.992857	2.573566	1238319600
2007-01-04	3.001786	3.069643	2.993571	3.059286	2.630688	847260400
2007-01-05	3.063214	3.078571	3.014286	3.037500	2.611954	834741600
2007-01-08	3.070000	3.090357	3.045714	3.052500	2.624853	797106800
2007-01-09	3.087500	3.320714	3.041071	3.306071	2.842900	3349298400

Example 1 – Download a single stock

```
start_date = '2007-01-03'
end_date = '2014-05-16'
```

```
tickers = ['AAPL']
df_apple = pandas_datareader.data.get_data_yahoo(tickers,\
                                                 start_date,end_date,\
                                                 interval='1d')

df_apple[:5]
```

```
[**********************100%**********************]  1 of 1 completed
```

Date	Open	High	Low	Close	Adj Close	Volume
2007-01-03	3.081786	3.092143	2.925000	2.992857	2.573566	1238319600
2007-01-04	3.001786	3.069643	2.993571	3.059286	2.630688	847260400
2007-01-05	3.063214	3.078571	3.014286	3.037500	2.611954	834741600
2007-01-08	3.070000	3.090357	3.045714	3.052500	2.624853	797106800
2007-01-09	3.087500	3.320714	3.041071	3.306071	2.842900	3349298400

Example 1 – Download a set of stocks

```
tickers = ['AAPL', 'CVX', 'IBM', 'XOM', 'GS', 'BA', 'TEVA', 'CME']
df = pandas_datareader.data.get_data_yahoo(tickers,start_date,
                                           end_date,interval='1d')
df.shape
```

```
[**********************100%************************]  8 of 8 completed

(1856, 48)
```

```
df[:5]
```

	Adj Close						...	Volume	
	AAPL	BA	CME	CVX	GS	IBM	...	CME	CVX
Date									
2007-01-03	2.573566	64.405731	63.408928	40.954422	164.774277	63.854336	...	3981500.0	12719600.0
2007-01-04	2.630688	64.665741	63.869690	40.556221	163.239136	64.537102	...	2590000.0	10825300.0
2007-01-05	2.611954	64.391273	64.169128	40.712025	163.403351	63.952831	...	2196500.0	9618000.0
2007-01-08	2.624853	64.239609	64.851746	41.231388	167.245239	64.924393	...	2040000.0	9435100.0
2007-01-09	2.842900	63.560661	65.226357	40.758183	167.532547	65.692467	...	2144500.0	10500000.0

Example 1

```
tickers = ['AAPL', 'CVX', 'IBM', 'XOM', 'GS', 'BA', 'TEVA', 'CME']
df = pandas_datareader.data.get_data_yahoo(tickers,start_date,
                                           end_date,interval='1d')
df.shape
```

```
[**********************100%**********************]  8 of 8 completed

(1856, 48)
```

```
df[:5]
```

multi-index

	Adj Close						...	Volume	
	AAPL	BA	CME	CVX	GS	IBM	...	CME	CVX
Date									
2007-01-03	2.573566	64.405731	63.408928	40.954422	164.774277	63.854336	...	3981500.0	12719600.0
2007-01-04	2.630688	64.665741	63.869690	40.556221	163.239136	64.537102	...	2590000.0	10825300.0
2007-01-05	2.611954	64.391273	64.169128	40.712025	163.403351	63.952831	...	2196500.0	9618000.0
2007-01-08	2.624853	64.239609	64.851746	41.231388	167.245239	64.924393	...	2040000.0	9435100.0
2007-01-09	2.842900	63.560661	65.226357	40.758183	167.532547	65.692467	...	2144500.0	10500000.0

Example 1

```
tickers = ['AAPL', 'CVX', 'IBM', 'XOM', 'GS', 'BA', 'TEVA', 'CME']
df = pandas_datareader.data.get_data_yahoo(tickers,start_date,
                                           end_date,interval='1d')
df.shape
```

```
[**********************100%**********************]  8 of 8 completed

(1856, 48)
```

```
df[:5]
```

	Adj Close						..	Volume	
	AAPL	BA	CME	CVX	GS	IBM	..	CME	CVX
Date									
2007-01-03	2.573566	64.405731	63.408928	40.954422	164.774277	63.854336	..	3981500.0	12719600.0
2007-01-04	2.630688	64.665741	63.869690	40.556221	163.239136	64.537102	..	2590000.0	10825300.0
2007-01-05	2.611954	64.391273	64.169128	40.712025	163.403351	63.952831	..	2196500.0	9618000.0
2007-01-08	2.624853	64.239609	64.851746	41.231388	167.245239	64.924393	..	2040000.0	9435100.0
2007-01-09	2.842900	63.560661	65.226357	40.758183	167.532547	65.692467	..	2144500.0	10500000.0

Example 1 – NaN entries in the last row

```
tickers = ['AAPL', 'CVX', 'IBM', 'XOM', 'GS', 'BA', 'TEVA', 'CME']
df = pandas_datareader.data.get_data_yahoo(tickers,start_date,
                                           end_date,interval='1d')
df.shape
```

```
[**********100%**********]  8 of 8 completed

(1856, 48)
```

```
df[-5:]
```

	Adj Close						...	Volume	
	AAPL	BA	CME	CVX	GS	IBM	...	CME	CVX
Date									
2014-05-12	19.019421	114.608109	53.570065	90.447502	141.399826	142.896942	...	1314700.0	4914700.0
2014-05-13	19.049248	115.342781	53.328297	90.888016	142.046829	142.614944	...	812700.0	4415200.0
2014-05-14	19.052780	114.945206	52.739052	91.299683	141.311249	140.040024	...	1120400.0	4961400.0
2014-05-15	18.890764	113.406708	52.489746	90.178009	138.820892	138.363037	...	1443800.0	5939500.0
2014-05-16	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN

Example 1 – drop last row

```
df.drop(pd.to_datetime('2014-05-16'),inplace = True)
df
```

	Adj Close						...	Volume	
	AAPL	BA	CME	CVX	GS	IBM	...	CME	CVX
Date									
2007-01-03	2.569716	64.405731	63.408928	40.409588	163.977493	63.127567	...	3981500.0	12719600.0
2007-01-04	2.626753	64.665741	63.869690	40.016708	162.449783	63.802544	...	2590000.0	10825300.0
2007-01-05	2.608047	64.391273	64.169128	40.170444	162.613205	63.224930	...	2196500.0	9618000.0
2007-01-08	2.620926	64.239609	64.851746	40.682899	166.436493	64.185463	...	2040000.0	9435100.0
2007-01-09	2.838647	63.560661	65.226357	40.215988	166.722427	64.944771	...	2144500.0	10500000.0
...	...	...	...	...	...	...	...	...	...
2014-05-09	18.785536	113.311661	52.572842	90.295837	139.317200	141.049255	...	1407200.0	4826100.0
2014-05-12	19.019421	114.608109	53.570065	90.447502	141.399826	142.896942	...	1314700.0	4914700.0
2014-05-13	19.049248	115.342781	53.328297	90.888016	142.046829	142.614944	...	812700.0	4415200.0
2014-05-14	19.052780	114.945206	52.739052	91.299683	141.311249	140.040024	...	1120400.0	4961400.0
2014-05-15	18.890764	113.406708	52.489746	90.178009	138.820892	138.363037	...	1443800.0	5939500.0

1855 rows × 48 columns

Example 1 – select Adj Close columns

```
close_px = df['Adj Close']
close_px
```

Date	AAPL	BA	CME	CVX	GS	IBM	TEVA	XOM
2007-01-03	2.569716	64.405731	63.408928	40.409588	163.977493	63.127567	24.707600	44.713043
2007-01-04	2.626753	64.665741	63.869690	40.016708	162.449783	63.802544	25.007948	43.874420
2007-01-05	2.608047	64.391273	64.169128	40.170444	162.613205	63.224930	25.458473	44.188141
2007-01-08	2.620926	64.239609	64.851746	40.682899	166.436493	64.185463	25.988035	43.832180
2007-01-09	2.838647	63.560661	65.226357	40.215988	166.722427	64.944771	25.940605	43.494301
...	...	...	...	...	...	...	...	...
2014-05-09	18.785536	113.311661	52.572842	90.295837	139.317200	141.049255	44.431610	73.258736
2014-05-12	19.019421	114.608109	53.570065	90.447502	141.399826	142.896942	45.099201	73.459953
2014-05-13	19.049248	115.342781	53.328297	90.888016	142.046829	142.614944	46.190830	73.553375
2014-05-14	19.052780	114.945206	52.739052	91.299683	141.311249	140.040024	45.586372	73.503075
2014-05-15	18.890764	113.406708	52.489746	90.178009	138.820892	138.363037	45.171379	72.418022

Most profitable stock

Example 1 – Which stock is most profitable?

```
close_px = df['Adj Close']
close_px
```

Date	AAPL	BA	CME	CVX	GS	IBM	TEVA	XOM
2007-01-03	2.569716	64.405731	63.408928	40.409588	163.977493	63.127567	24.707600	44.713043
2007-01-04	2.626753	64.665741	63.869690	40.016708	162.449783	63.802544	25.007948	43.874420
2007-01-05	2.608047	64.391273	64.169128	40.170444	162.613205	63.224930	25.458473	44.188141
2007-01-08	2.620926	64.239609	64.851746	40.682899	166.436493	64.185463	25.988035	43.832180
2007-01-09	2.838647	63.560661	65.226357	40.215988	166.722427	64.944771	25.940605	43.494301
...	...	...	...	...	...	...	...	...
2014-05-09	18.785536	113.311661	52.572842	90.295837	139.317200	141.049255	44.431610	73.258736
2014-05-12	19.019421	114.608109	53.570065	90.447502	141.399826	142.896942	45.099201	73.459953
2014-05-13	19.049248	115.342781	53.328297	90.888016	142.046829	142.614944	46.190830	73.553375
2014-05-14	19.052780	114.945206	52.739052	91.299683	141.311249	140.040024	45.586372	73.503075
2014-05-15	18.890764	113.406708	52.489746	90.178009	138.820892	138.363037	45.171379	72.418022

Example 1 – Which stock is most profitable?

```
close_px = df['Adj Close']
close_px
```

Date	AAPL	BA	CME	CVX	GS	IBM	TEVA	XOM
2007-01-03	2.569716	64.405731	63.408928	40.409588	163.977493	63.127567	24.707600	44.713043
2007-01-04	2.626753	64.665741	63.869690	40.016708	162.449783	63.802544	25.007948	43.874420
2007-01-05	2.608047	64.391273	64.169128	40.170444	162.613205	63.224930	25.458473	44.188141
2007-01-08	2.620926	64.239609	64.851746	40.682899	166.436493	64.185463	25.988035	43.832180
2007-01-09	2.838647	63.560661	65.226357	40.215988	166.722427	64.944771	25.940605	43.494301
...	...	...	...	...	...	...	...	...
2014-05-09	18.785536	113.311661	52.572842	90.295837	139.317200	141.049255	44.431610	73.258736
2014-05-12	19.019421	114.608109	53.570065	90.447502	141.399826	142.896942	45.099201	73.459953
2014-05-13	19.049248	115.342781	53.328297	90.888016	142.046829	142.614944	46.190830	73.553375
2014-05-14	19.052780	114.945206	52.739052	91.299683	141.311249	140.040024	45.586372	73.503075
2014-05-15	18.890764	113.406708	52.489746	90.178009	138.820892	138.363037	45.171379	72.418022

find the ratio

Example 1 – Which stock is most profitable?

```
close_px = df['Adj Close']
close_px
```

Date	AAPL	BA	CME
2007-01-03	2.569716	64.405731	63.408928
2007-01-04	2.626753	64.665741	63.869690
2007-01-05	2.608047	64.391273	64.169128
2007-01-08	2.620926	64.239609	64.851746
2007-01-09	2.838647	63.560661	65.226357
...	...	...	...
2014-05-09	18.785536	113.311661	52.572842
2014-05-12	19.019421	114.608109	53.570065
2014-05-13	19.049248	115.342781	53.328297
2014-05-14	19.052780	114.945206	52.739052
2014-05-15	18.890764	113.406708	52.489746

```
ratios = close_px.iloc[-1] / close_px.iloc[0]
ratios
```

```
AAPL    7.351303
BA      1.760817
CME     0.827797
CVX     2.231599
GS      0.846585
IBM     2.191801
TEVA    1.828238
XOM     1.619617
dtype: float64
```

= 18.890 / 2.569

Example 1 – Which stock is most profitable?

```
close_px = df['Adj Close']
close_px
```

Date	AAPL	BA	CME
2007-01-03	2.569716	64.405731	63.408928
2007-01-04	2.626753	64.665741	63.869690
2007-01-05	2.608047	64.391273	64.169128
2007-01-08	2.620926	64.239609	64.851746
2007-01-09	2.838647	63.560661	65.226357
...	...	...	...
2014-05-09	18.785536	113.311661	52.572842
2014-05-12	19.019421	114.608109	53.570065
2014-05-13	19.049248	115.342781	53.328297
2014-05-14	19.052780	114.945206	52.739052
2014-05-15	18.890764	113.406708	52.489746

```
ratios = close_px.iloc[-1] / close_px.iloc[0]
ratios
```

```
AAPL    7.351303
BA      1.760817
CME     0.827797
CVX     2.231599
GS      0.846585
IBM     2.191801
TEVA    1.828238
XOM     1.619617
dtype: float64
```

= 113.4067 / 64.4057

Example 1 – Which stock is most profitable?

```
close_px = df['Adj Close']
close_px
```

Date	AAPL	BA	CME
2007-01-03	2.569716	64.405731	63.408928
2007-01-04	2.626753	64.665741	63.869690
2007-01-05	2.608047	64.391273	64.169128
2007-01-08	2.620926	64.239609	64.851746
2007-01-09	2.838647	63.560661	65.226357
...	...	...	...
2014-05-09	18.785536	113.311661	52.572842
2014-05-12	19.019421	114.608109	53.570065
2014-05-13	19.049248	115.342781	53.328297
2014-05-14	19.052780	114.945206	52.739052
2014-05-15	18.890764	113.406708	52.489746

```
ratios = close_px.iloc[-1] / close_px.iloc[0]
ratios
```

```
AAPL    7.351303
BA      1.760817
CME     0.827797
CVX     2.231599
GS      0.846585
IBM     2.191801
TEVA    1.828238
XOM     1.619617
dtype: float64
```

= 52.4897 / 63.4089

Example 1 – Which stock is most profitable?

```
close_px = df['Adj Close']
close_px
```

Date	AAPL	BA	CME
2007-01-03	2.569716	64.405731	63.408928
2007-01-04	2.626753	64.665741	63.869690
2007-01-05	2.608047	64.391273	64.169128
2007-01-08	2.620926	64.239609	64.851746
2007-01-09	2.838647	63.560661	65.226357
...	...	...	...
2014-05-09	18.785536	113.311661	52.572842
2014-05-12	19.019421	114.608109	53.570065
2014-05-13	19.049248	115.342781	53.328297
2014-05-14	19.052780	114.945206	52.739052
2014-05-15	18.890764	113.406708	52.489746

```
ratios = close_px.iloc[-1] / close_px.iloc[0]
ratios
```

```
AAPL    7.351303
BA      1.760817
CME     0.827797
CVX     2.231599
GS      0.846585
IBM     2.191801
TEVA    1.828238
XOM     1.619617
dtype: float64
```

= 52.4897 / 63.4089

```
# AAPL is most profitable stock
```

AAPL stock price in 2014
is 7.35 times
the stock price in 2007

Plotting with pandas

Plot time series prices

- pandas.plot( ) method makes quite simple to create plots
- It is a wrapper function of the plot( ) function from the matplotlib library
- It makes easy the labeling, axis generation, formatting of series plots

Example 1 – Plot AAPL prices

```
close_px = df['Adj Close']
close_px[:5]
```

Symbols	AAPL	CVX	IBM	XOM	GS	BA	TEVA	CME
Date								
2007-01-03	2.577937	42.041698	65.430618	46.833088	165.980194	64.405731	24.707600	63.941113
2007-01-04	2.635158	41.632954	66.130203	45.954712	164.433838	64.665741	25.007948	64.405716
2007-01-05	2.616391	41.792900	65.531532	46.283306	164.599274	64.391273	25.458473	64.707672
2007-01-08	2.629312	42.326057	66.527077	45.910465	168.469238	64.239609	25.988035	65.395996
2007-01-09	2.847729	41.840282	67.314095	45.556561	168.758636	63.560661	25.940605	65.773788

```
close_px.shape
```

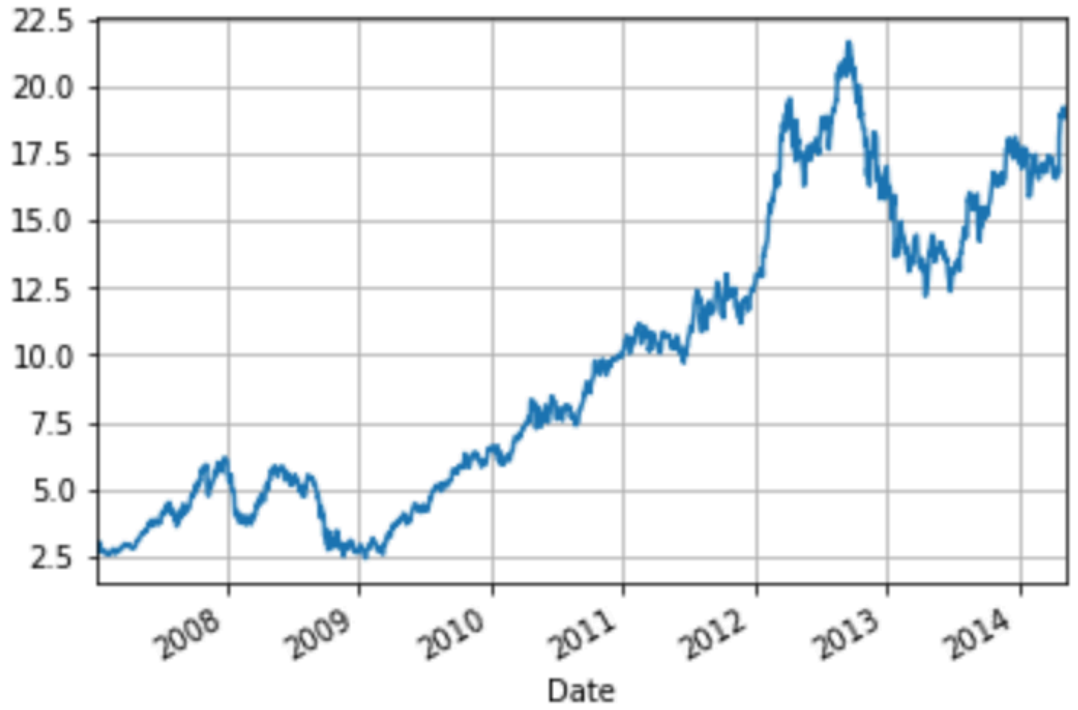
```
(1856, 8)
```

Plot AAPL prices with Matplotlib

```
# matplotlib to plot AAPL closing prices
plt.figure()
plt.plot(close_px.AAPL)
plt.grid();
```

Plot AAPL prices with pandas

```
# pandas to plot AAPL closing prices
close_px['AAPL'].plot(grid=True);
```

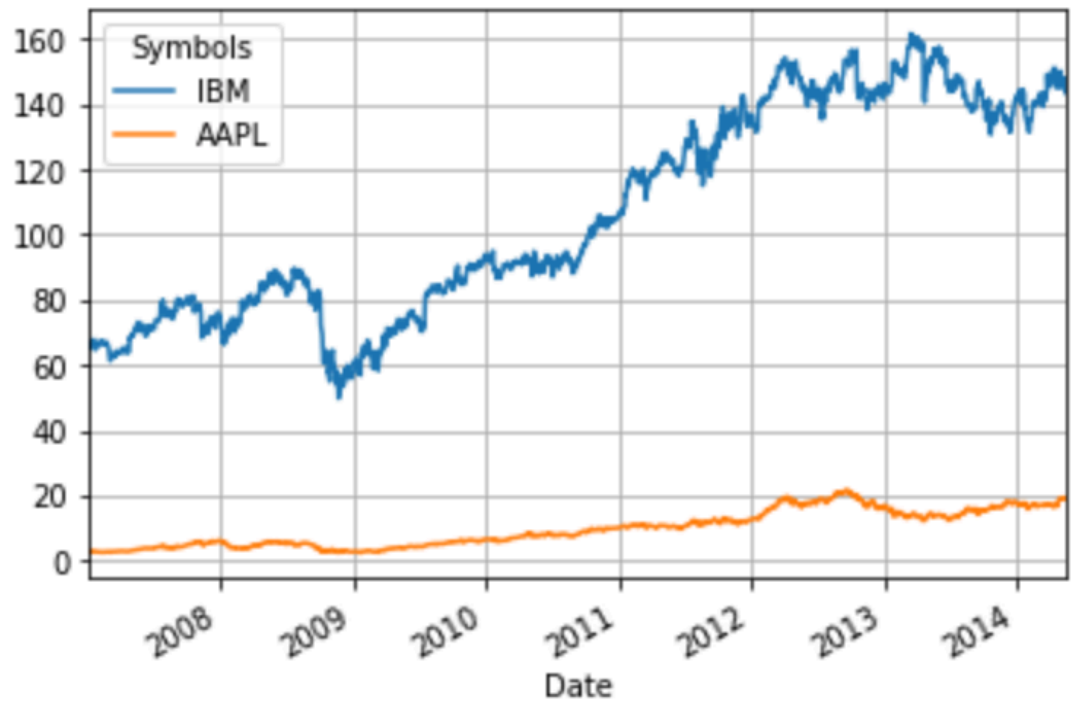


Plot IBM prices with pandas

```
# plot closing prices of IBM
close_px['IBM'].plot(grid=True);
```

Plot IBM and AAPL prices

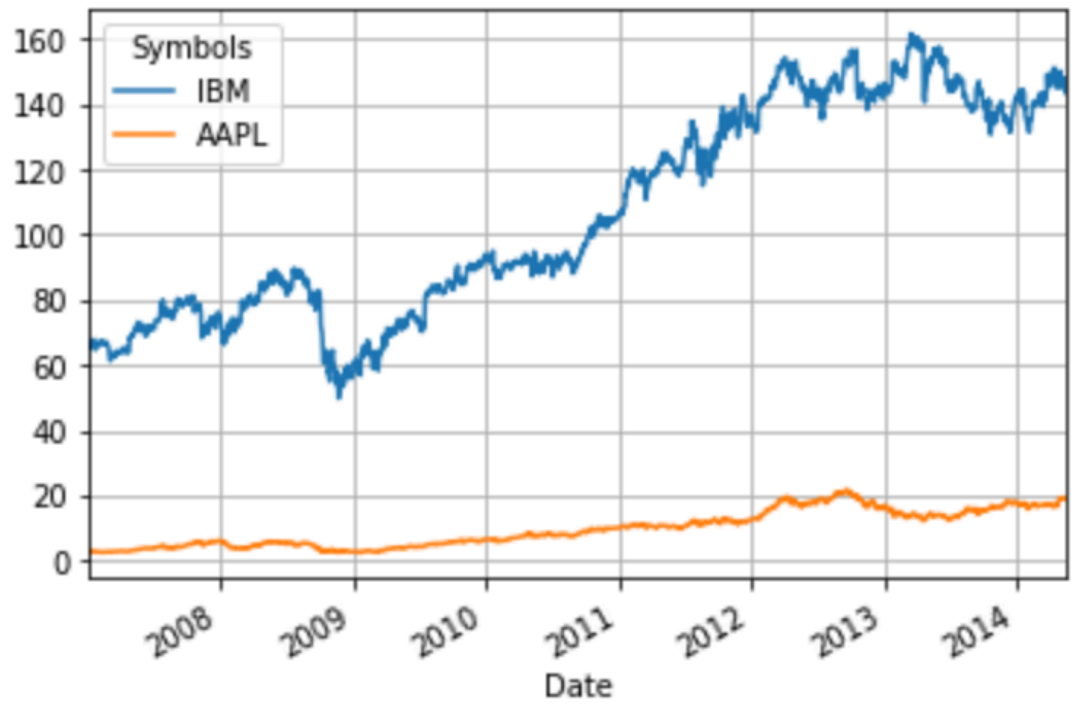
```
# Compare IBM, AAPL on the same chart
close_px[['IBM', 'AAPL']].plot(grid=True);
```



Gross Returns (IBM and AAPL)

Plot IBM and AAPL prices

```
# Compare IBM, AAPL on the same chart
close_px[['IBM', 'AAPL']].plot(grid=True);
```



Which stock was more profitable?

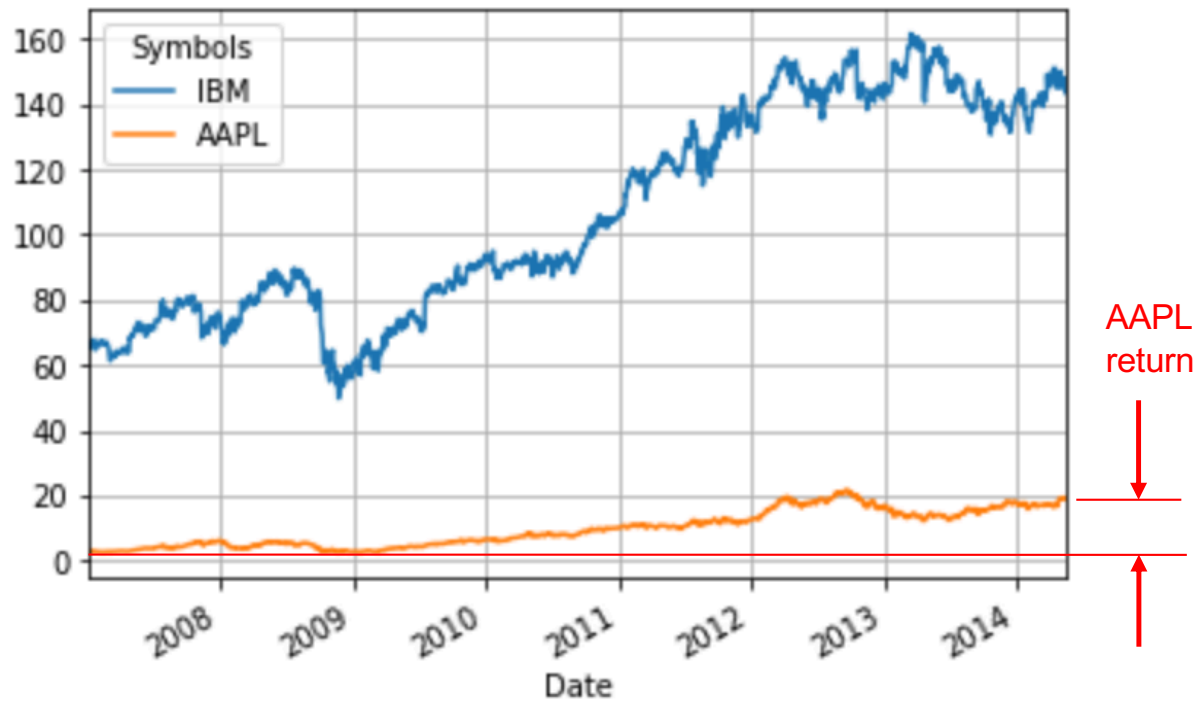
Plot IBM and AAPL prices

```
# Compare IBM, AAPL on the same chart
close_px[['IBM', 'AAPL']].plot(grid=True);
```

Which stock was more profitable?

Plot IBM and AAPL prices

```
# Compare IBM, AAPL on the same chart
close_px[['IBM', 'AAPL']].plot(grid=True);
```



Which stock was more profitable?

IBM and AAPL - Gross returns

```
df2 = close_px[['IBM', 'AAPL']]
df2[:5]
```

Date	IBM	AAPL
2007-01-03	63.127567	2.569716
2007-01-04	63.802544	2.626753
2007-01-05	63.224930	2.608047
2007-01-08	64.185463	2.620926
2007-01-09	64.944771	2.838647

```
gross_returns = df2 / df2.shift(1)
gross_returns
```

Date	IBM	AAPL
2007-01-03	NaN	NaN
2007-01-04	1.010692	1.022196
2007-01-05	0.990947	0.992879
2007-01-08	1.015192	1.004938
2007-01-09	1.011830	1.083070

IBM and AAPL - Gross returns

```
df2 = close_px[['IBM', 'AAPL']]
df2[:5]
```

Date	IBM	AAPL
2007-01-03	63.127567	2.569716
2007-01-04	63.802544	2.626753
2007-01-05	63.224930	2.608047
2007-01-08	64.185463	2.620926
2007-01-09	64.944771	2.838647

```
gross_returns = df2 / df2.shift(1)
gross_returns
```

Date	IBM	AAPL
2007-01-03	NaN	NaN
2007-01-04	1.010692	1.022196
2007-01-05	0.990947	0.992879
2007-01-08	1.015192	1.004938
2007-01-09	1.011830	1.083070

= 2.626753 / 2.569716

IBM and AAPL - Gross returns

```
df2 = close_px[['IBM', 'AAPL']]
df2[:5]
```

Date	IBM	AAPL
2007-01-03	63.127567	2.569716
2007-01-04	63.802544	2.626753
2007-01-05	63.224930	2.608047
2007-01-08	64.185463	2.620926
2007-01-09	64.944771	2.838647

```
gross_returns = df2 / df2.shift(1)
gross_returns
```

Date	IBM	AAPL
2007-01-03	NaN	NaN
2007-01-04	1.010692	1.022196
2007-01-05	0.990947	0.992879
2007-01-08	1.015192	1.004938
2007-01-09	1.011830	1.083070

= 2.608047 / 2.626753

IBM and AAPL - Gross returns

```
gross_returns = df2 / df2.shift(1)
gross_returns
```

Date	IBM	AAPL
2007-01-03	NaN	NaN
2007-01-04	1.010692	1.022196
2007-01-05	0.990947	0.992879
2007-01-08	1.015192	1.004938
2007-01-09	1.011830	1.083070

```
cum_gross_returns = gross_returns.cumprod()
cum_gross_returns[:5]
```

Date	IBM	AAPL
2007-01-03	NaN	NaN
2007-01-04	1.010692	1.022196
2007-01-05	1.001542	1.014916
2007-01-08	1.016758	1.019928
2007-01-09	1.028786	1.104654

IBM and AAPL - Gross returns

```
gross_returns = df2 / df2.shift(1)
gross_returns
```

Date	IBM	AAPL
2007-01-03	NaN	NaN
2007-01-04	1.010692	1.022196
2007-01-05	0.990947	0.992879
2007-01-08	1.015192	1.004938
2007-01-09	1.011830	1.083070

```
cum_gross_returns = gross_returns.cumprod()
cum_gross_returns[:5]
```

Date	IBM	AAPL
2007-01-03	NaN	NaN
2007-01-04	1.010692	1.022196
2007-01-05	1.001542	1.014916
2007-01-08	1.016758	1.019928
2007-01-09	1.028786	1.104654

= 1.022196 * 0.992879

IBM and AAPL - Gross returns

```
gross_returns = df2 / df2.shift(1)
gross_returns
```

Date	IBM	AAPL
2007-01-03	NaN	NaN
2007-01-04	1.010692	1.022196
2007-01-05	0.990947	0.992879
2007-01-08	1.015192	1.004938
2007-01-09	1.011830	1.083070

```
cum_gross_returns = gross_returns.cumprod()
cum_gross_returns[:5]
```

Date	IBM	AAPL
2007-01-03	NaN	NaN
2007-01-04	1.010692	1.022196
2007-01-05	1.001542	1.014916
2007-01-08	1.016758	1.019928
2007-01-09	1.028786	1.104654

= 1.014916 * 1.004938

IBM and AAPL - Gross returns

```
gross_returns = df2 / df2.shift(1)
gross_returns
```

Date	IBM	AAPL
2007-01-03	NaN	NaN
2007-01-04	1.010692	1.022196
2007-01-05	0.990947	0.992879
2007-01-08	1.015192	1.004938
2007-01-09	1.011830	1.083070

```
cum_gross_returns = gross_returns.cumprod()
cum_gross_returns[:5]
```

Date	IBM	AAPL
2007-01-03	NaN	NaN
2007-01-04	1.010692	1.022196
2007-01-05	1.001542	1.014916
2007-01-08	1.016758	1.019928
2007-01-09	1.028786	1.104654

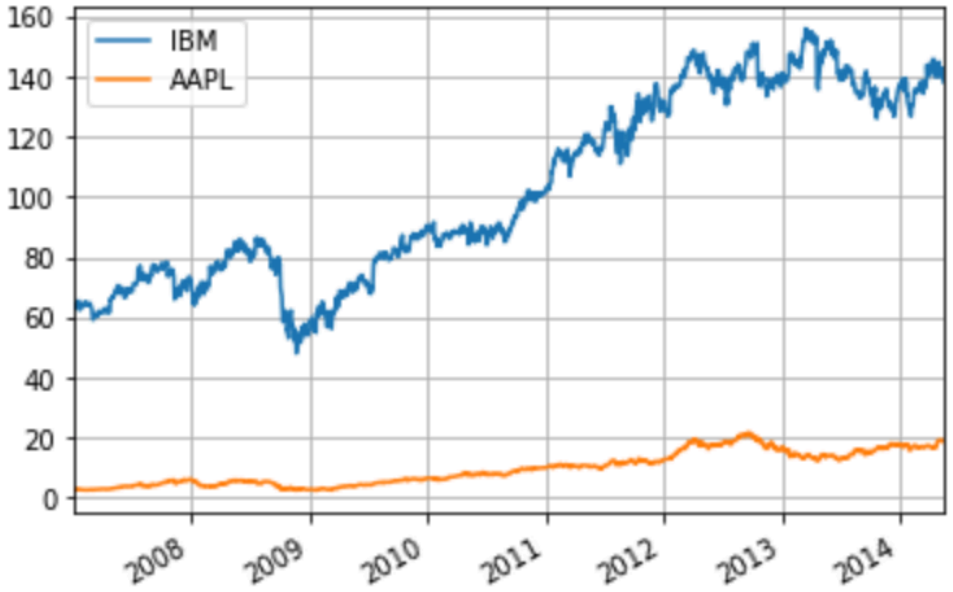
= 1.019928 * 1.083070

Plot IBM and AAPL Cumulative Gross returns

```
cum_gross_returns.plot(grid=True)
plt.xlabel('')
plt.legend(loc='upper left');
```

Plot IBM and AAPL – One-dollar investment

Evolution of Stock Prices




```
close_px[['IBM', 'AAPL']].plot(grid=True)
```

Evolution of one-dollar investment


```
cum_gross_returns.plot(grid=True)
```

Plot IBM and AAPL – One-dollar investment

```
ratios = close_px.iloc[-1] / close_px.iloc[0]
ratios
```

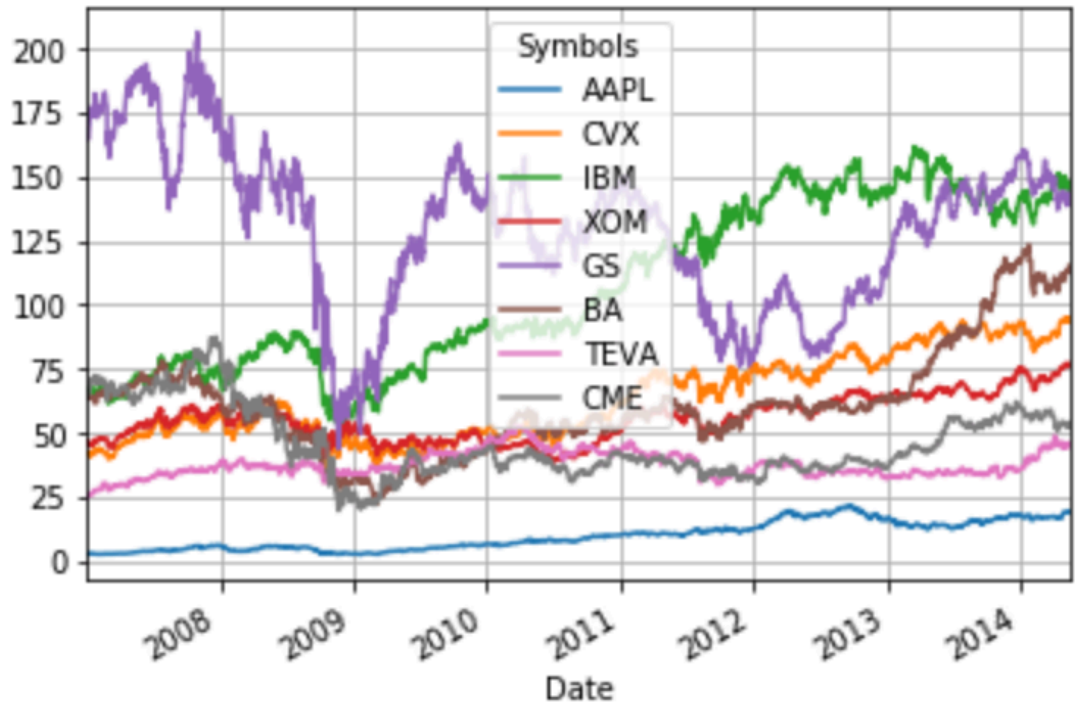
```
AAPL     7.351303
BA       1.760817
CME      0.827797
CVX      2.231599
GS       0.846585
IBM      2.191801
TEVA     1.828238
XOM      1.619617
dtype: float64
```

```
# AAPL is most profitable stock
```

Evolution of one-dollar investment

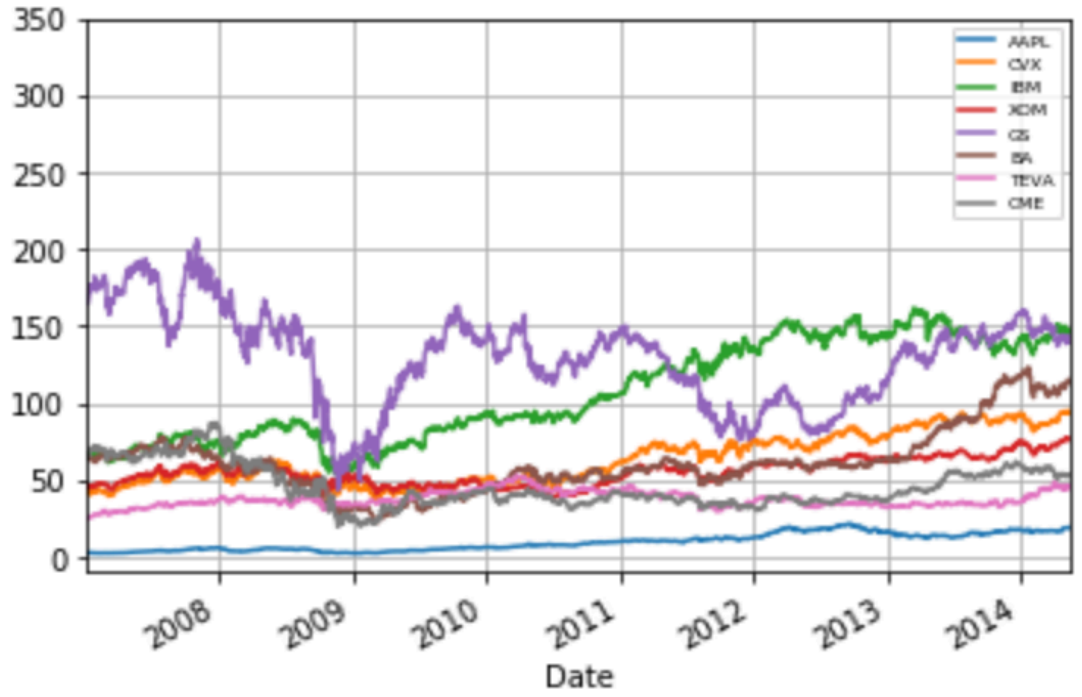
Plot all prices

```
# plot all prices
close_px.plot(grid=True);
```



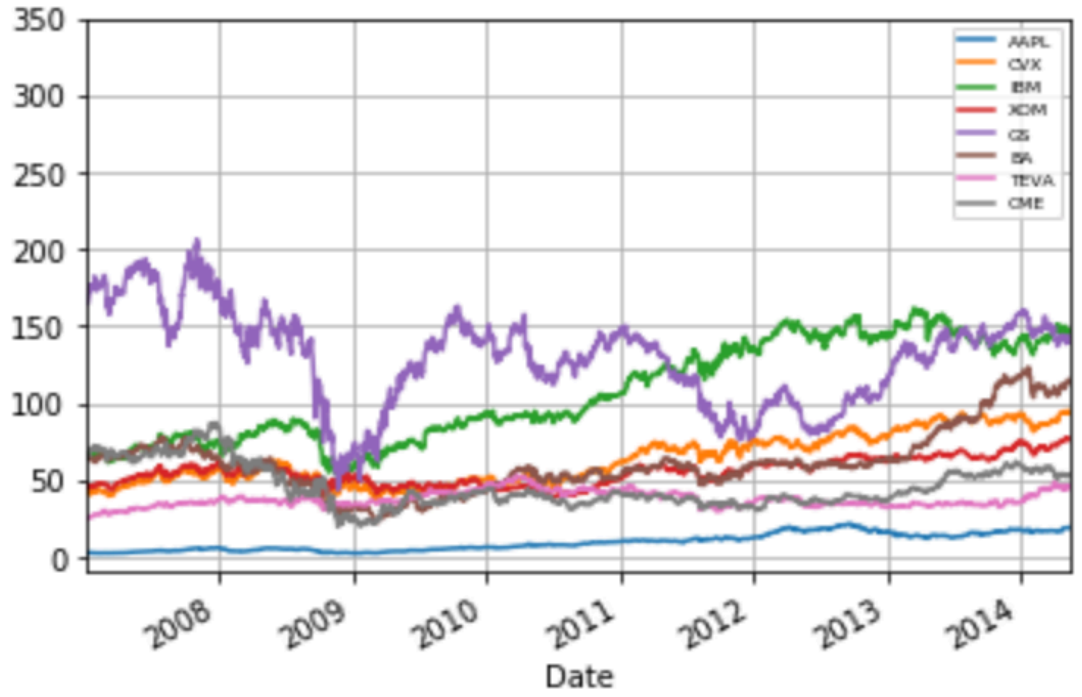
Plot all prices

```
# plot all prices
close_px.plot(grid=True)
plt.ylim(-10,350)
plt.legend(loc=1, prop={'size': 6});
```



Plot all prices

```
# plot all prices
close_px.plot(grid=True)
plt.ylim(-10,350)
plt.legend(loc=1, prop={'size': 6});
```




```
# starting prices (2007) are different
```

Net Returns

Net changes in prices

$$\text{net return} = \frac{Value\ in\ period\ 2}{Value\ in\ period\ 1} - 1$$

net return = gross return - 1

```
pd.DataFrame(close_px['AAPL'])
```

Date	AAPL
2007-01-03	2.577937
2007-01-04	2.635158
2007-01-05	2.616391
2007-01-08	2.629312
2007-01-09	2.847729

Net changes in prices

$$\text{net return} = \frac{Value\ in\ period\ 2}{Value\ in\ period\ 1} - 1$$

net return = gross return - 1

```
pd.DataFrame(close_px['AAPL'])
```

	AAPL
Date	
2007-01-03	2.577937
2007-01-04	2.635158
2007-01-05	2.616391
2007-01-08	2.629312
2007-01-09	2.847729

```
net_returns = close_px / close_px.shift(1) - 1
net_returns[:5]
```

Symbols	AAPL
Date	
2007-01-03	NaN
2007-01-04	0.022196
2007-01-05	-0.007122
2007-01-08	0.004938
2007-01-09	0.083070

Net changes in prices

$$\text{net return} = \frac{Value\ in\ period\ 2}{Value\ in\ period\ 1} - 1$$

net return = gross return - 1

```
pd.DataFrame(close_px['AAPL'])
```

	AAPL
Date	
2007-01-03	2.577937
2007-01-04	2.635158
2007-01-05	2.616391
2007-01-08	2.629312
2007-01-09	2.847729

```
net_returns = close_px / close_px.shift(1) - 1
net_returns[:5]
```

Symbols	AAPL
Date	
2007-01-03	NaN
2007-01-04	0.022196
2007-01-05	-0.007122
2007-01-08	0.004938
2007-01-09	0.083070

$$= \frac{2.635158}{2.577937} - 1$$

Net changes in prices

$$\text{net change} = \frac{Value\ in\ period\ 2}{Value\ in\ period\ 1} - 1$$

```
pd.DataFrame(close_px['AAPL'])
```

Date	AAPL
2007-01-03	2.577937
2007-01-04	2.635158
2007-01-05	2.616391
2007-01-08	2.629312
2007-01-09	2.847729

```
net_returns = close_px / close_px.shift(1) - 1
net_returns[:5]
```

Date	Symbols AAPL
2007-01-03	NaN
2007-01-04	0.022196
2007-01-05	-0.007122
2007-01-08	0.004938
2007-01-09	0.083070

$$= \frac{2.616391}{2.635158} - 1$$

Net changes in prices

$$\text{net change} = \frac{Value\ in\ period\ 2}{Value\ in\ period\ 1} - 1$$

```
pd.DataFrame(close_px['AAPL'])
```

Date	AAPL
2007-01-03	2.577937
2007-01-04	2.635158
2007-01-05	2.616391
2007-01-08	2.629312
2007-01-09	2.847729

```
net_returns = close_px / close_px.shift(1) - 1
net_returns[:5]
```

Symbols / Date	AAPL
2007-01-03	NaN
2007-01-04	0.022196
2007-01-05	-0.007122
2007-01-08	0.004938
2007-01-09	0.083070

$$= \frac{2.629312}{2.616391} - 1$$

Net changes in prices

$$\text{net change} = \frac{Value\ in\ period\ 2}{Value\ in\ period\ 1} - 1$$

```
pd.DataFrame(close_px['AAPL'])
```

	AAPL
Date	
2007-01-03	2.577937
2007-01-04	2.635158
2007-01-05	2.616391
2007-01-08	2.629312
2007-01-09	2.847729

```
net_returns = close_px / close_px.shift(1) - 1
net_returns[:5]
```

Symbols	AAPL	CVX	IBM	XOM	GS
Date					
2007-01-03	NaN	NaN	NaN	NaN	NaN
2007-01-04	0.022196	-0.009722	0.010692	-0.018755	-0.009317
2007-01-05	-0.007122	0.003842	-0.009053	0.007150	0.001006
2007-01-08	0.004938	0.012757	0.015192	-0.008056	0.023511
2007-01-09	0.083070	-0.011477	0.011830	-0.007709	0.001718

Net changes in prices

$$\text{net change} = \frac{Value\ in\ period\ 2}{Value\ in\ period\ 1} - 1$$

```
pd.DataFrame(close_px['AAPL'])
```

Date	AAPL
2007-01-03	2.577937
2007-01-04	2.635158
2007-01-05	2.616391
2007-01-08	2.629312
2007-01-09	2.847729

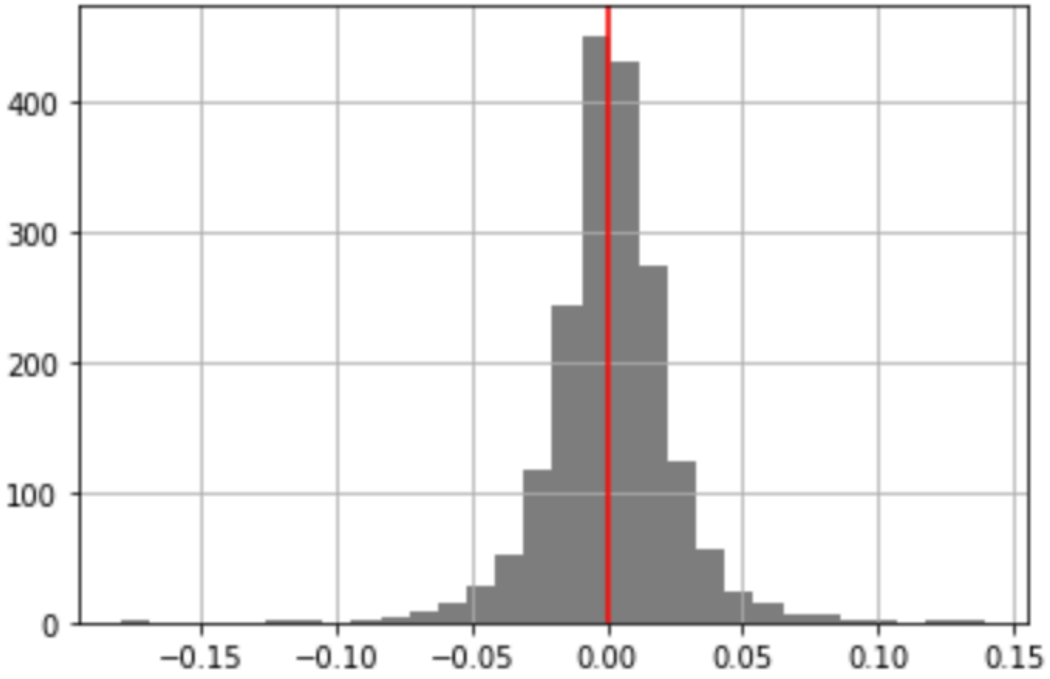
```
net_returns = close_px / close_px.shift(1) - 1
net_returns[:5]
```

Symbols	AAPL	CVX	IBM	XOM	GS
Date					
2007-01-03	NaN	NaN	NaN	NaN	NaN
2007-01-04	0.022196	-0.009722	0.010692	-0.018755	-0.009317
2007-01-05	-0.007122	0.003842	-0.009053	0.007150	0.001006
2007-01-08	0.004938	0.012757	0.015192	-0.008056	0.023511
2007-01-09	0.083070	-0.011477	0.011830	-0.007709	0.001718

plot AAPL net returns

Net changes distribution

```
x = net_returns["AAPL"]
# histogram of net returns
x.hist(color = 'k',bins=30,alpha=0.5,grid=True)
plt.axvline(linewidth=1.5,color='r');
```

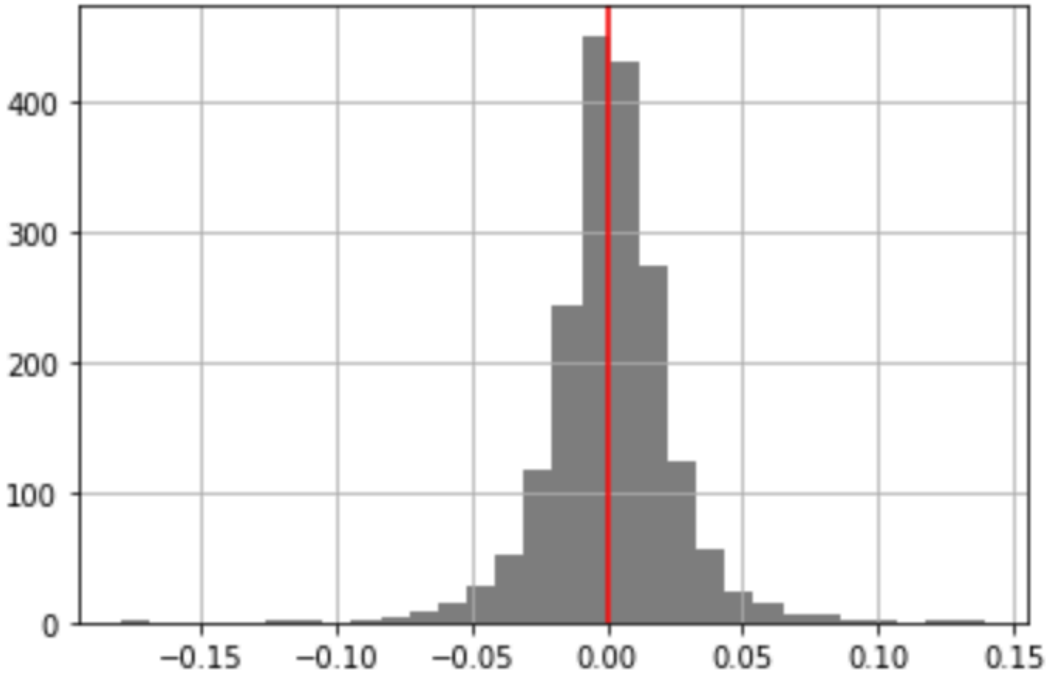


Net changes distribution

```
x = net_returns["AAPL"]
# histogram of net returns
x.hist(color = 'k',bins=30,alpha=0.5,grid=True)
plt.axvline(linewidth=1.5,color='r');
```

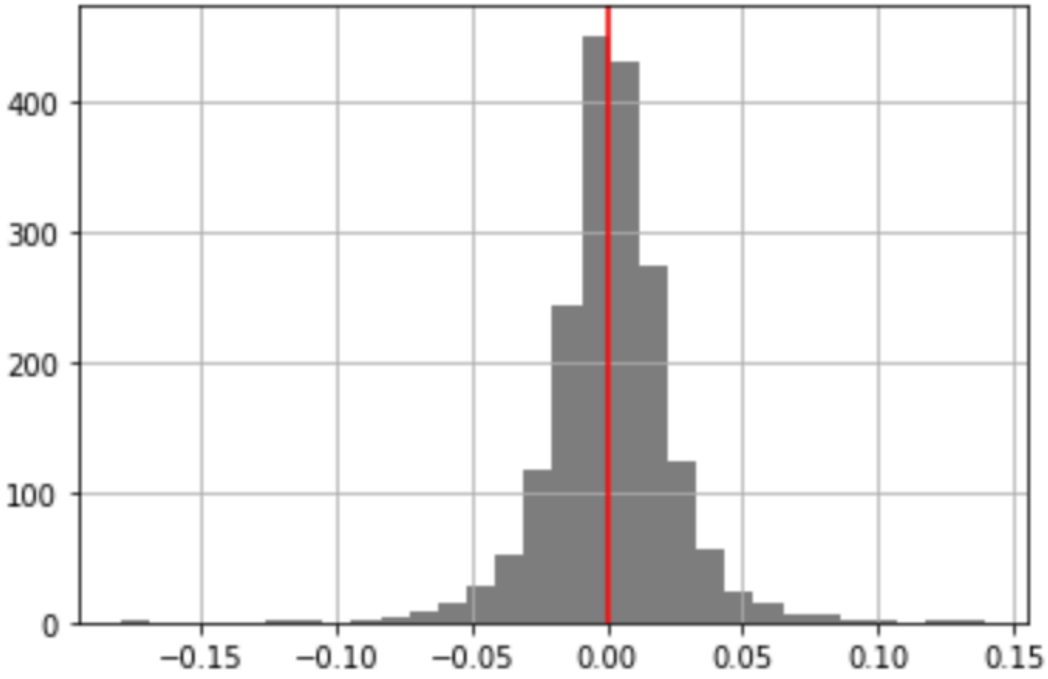
pandas

matplotlib



Net changes distribution

```
x = net_returns["AAPL"]
# histogram of net returns
x.hist(color = 'k',bins=30,alpha=0.5,grid=True)
plt.axvline(linewidth=1.5,color='r');
```



how risky is this stock?

Risk of stocks

```
net_returns = close_px / close_px.shift(1) - 1
net_returns[:5]
```

Symbols	AAPL	CVX	IBM	XOM	GS
Date					
2007-01-03	NaN	NaN	NaN	NaN	NaN
2007-01-04	0.022196	-0.009722	0.010692	-0.018755	-0.009317
2007-01-05	-0.007122	0.003842	-0.009053	0.007150	0.001006
2007-01-08	0.004938	0.012757	0.015192	-0.008056	0.023511
2007-01-09	0.083070	-0.011477	0.011830	-0.007709	0.001718

```
net_returns.std()
```

```
Symbols
AAPL    0.022422
CVX     0.018282
IBM     0.014744
XOM     0.016941
GS      0.028437
BA      0.019591
TEVA    0.015718
CME     0.027336
```

Risk of stocks

```
net_returns = close_px / close_px.shift(1) - 1
net_returns[:5]
```

Symbols	AAPL	CVX	IBM	XOM	GS
Date					
2007-01-03	NaN	NaN	NaN	NaN	NaN
2007-01-04	0.022196	-0.009722	0.010692	-0.018755	-0.009317
2007-01-05	-0.007122	0.003842	-0.009053	0.007150	0.001006
2007-01-08	0.004938	0.012757	0.015192	-0.008056	0.023511
2007-01-09	0.083070	-0.011477	0.011830	-0.007709	0.001718

```
net_returns.std()
```

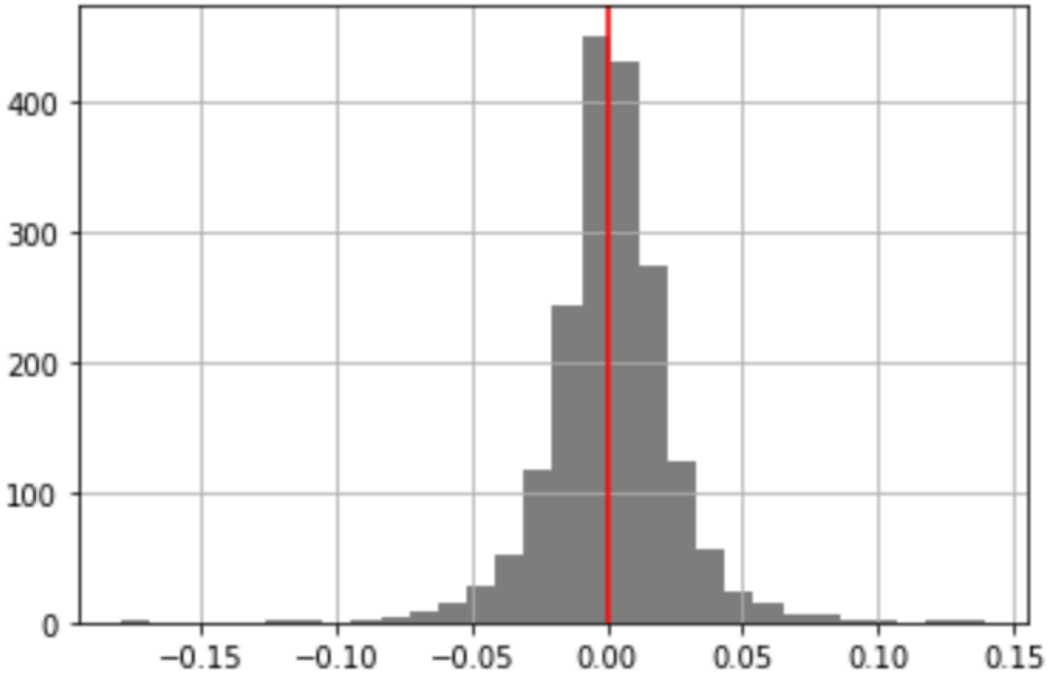
```
Symbols
AAPL    0.022422
CVX     0.018282
IBM     0.014744
XOM     0.016941
GS      0.028437
BA      0.019591
TEVA    0.015718
CME     0.027336
```

least

most risky

Distribution of net changes

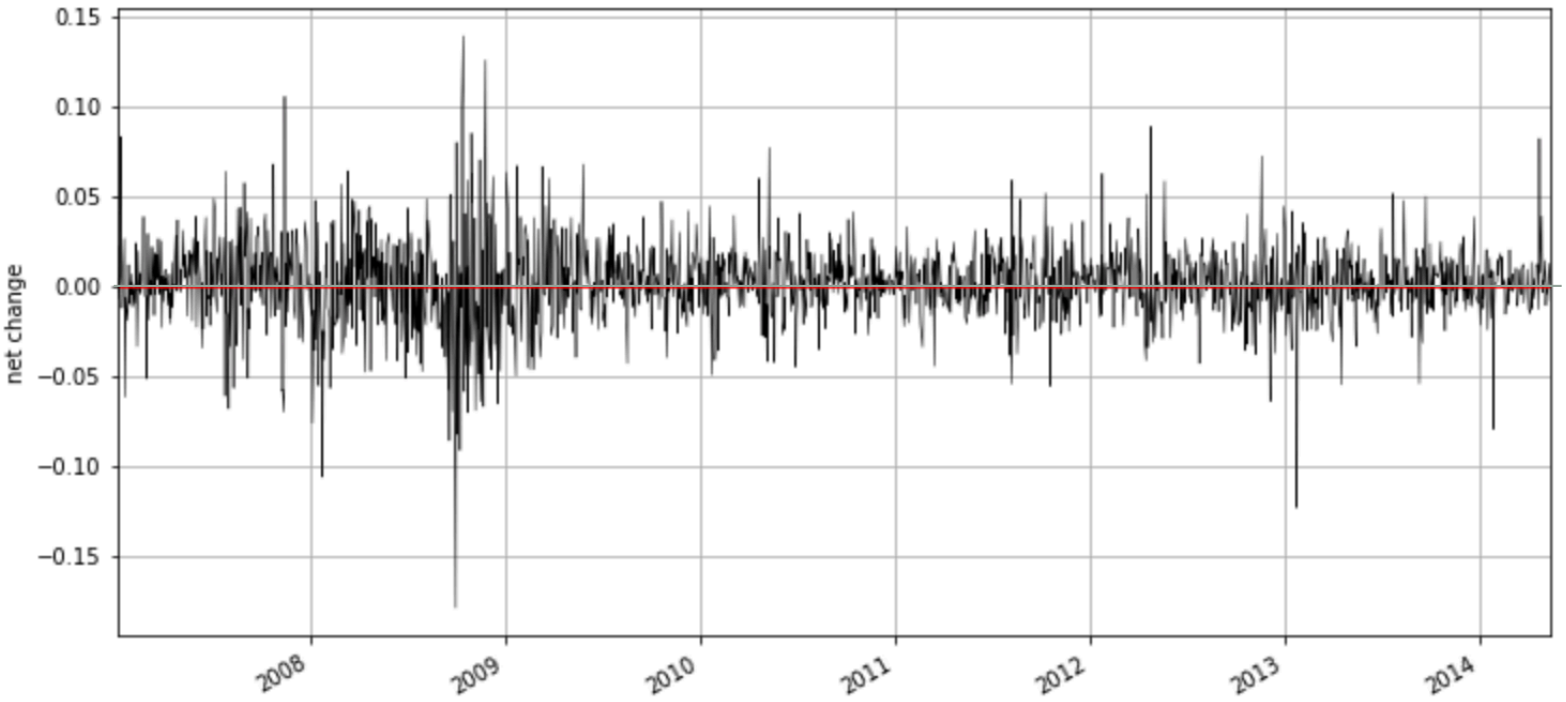
```
x = net_returns["AAPL"]
# histogram of net returns
x.hist(color = 'k',bins=30,alpha=0.5,grid=True)
plt.axvline(linewidth=1.5,color='r');
```



Net changes in AAPL prices over time

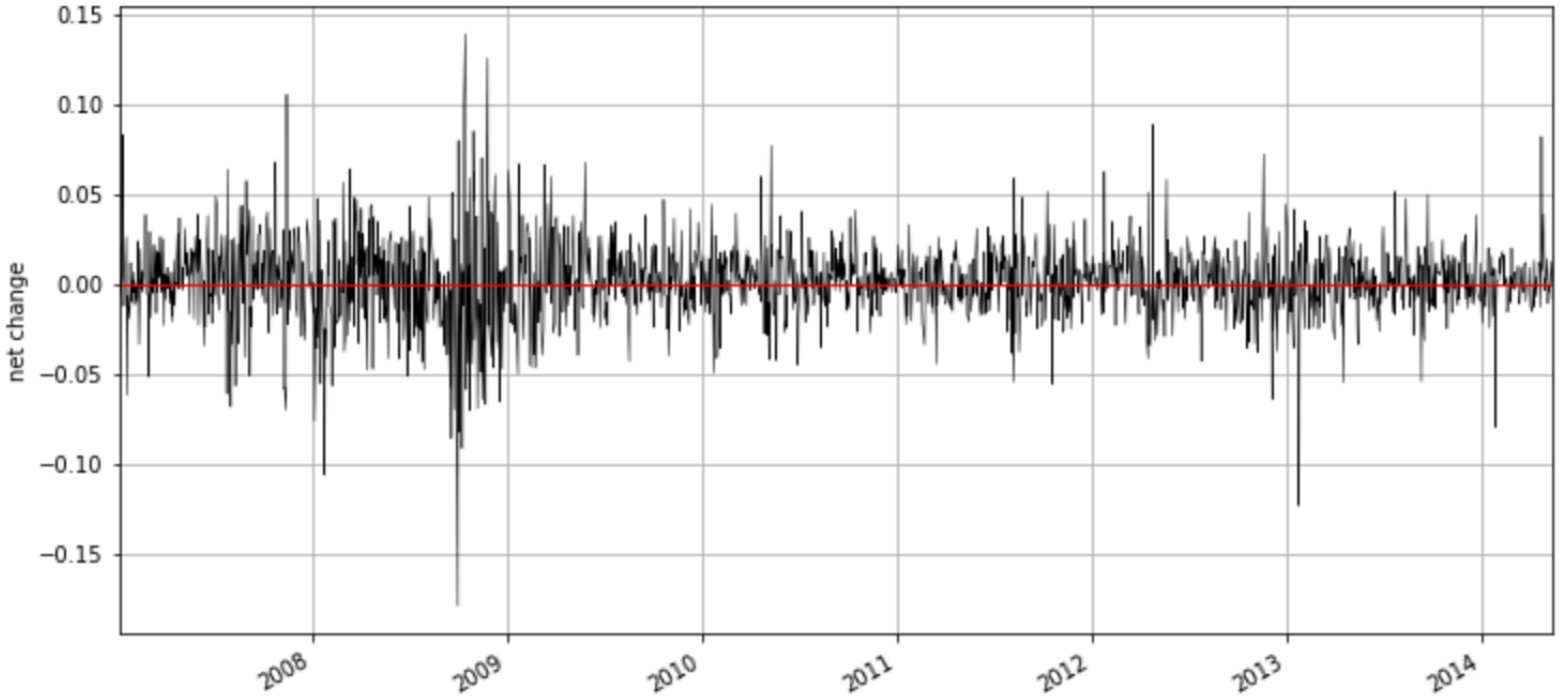
```
# plot daily changes
net_returns["AAPL"].plot(figsize = (12,6),grid=True,color = 'k',linewidth = 0.5)
```

lineplot



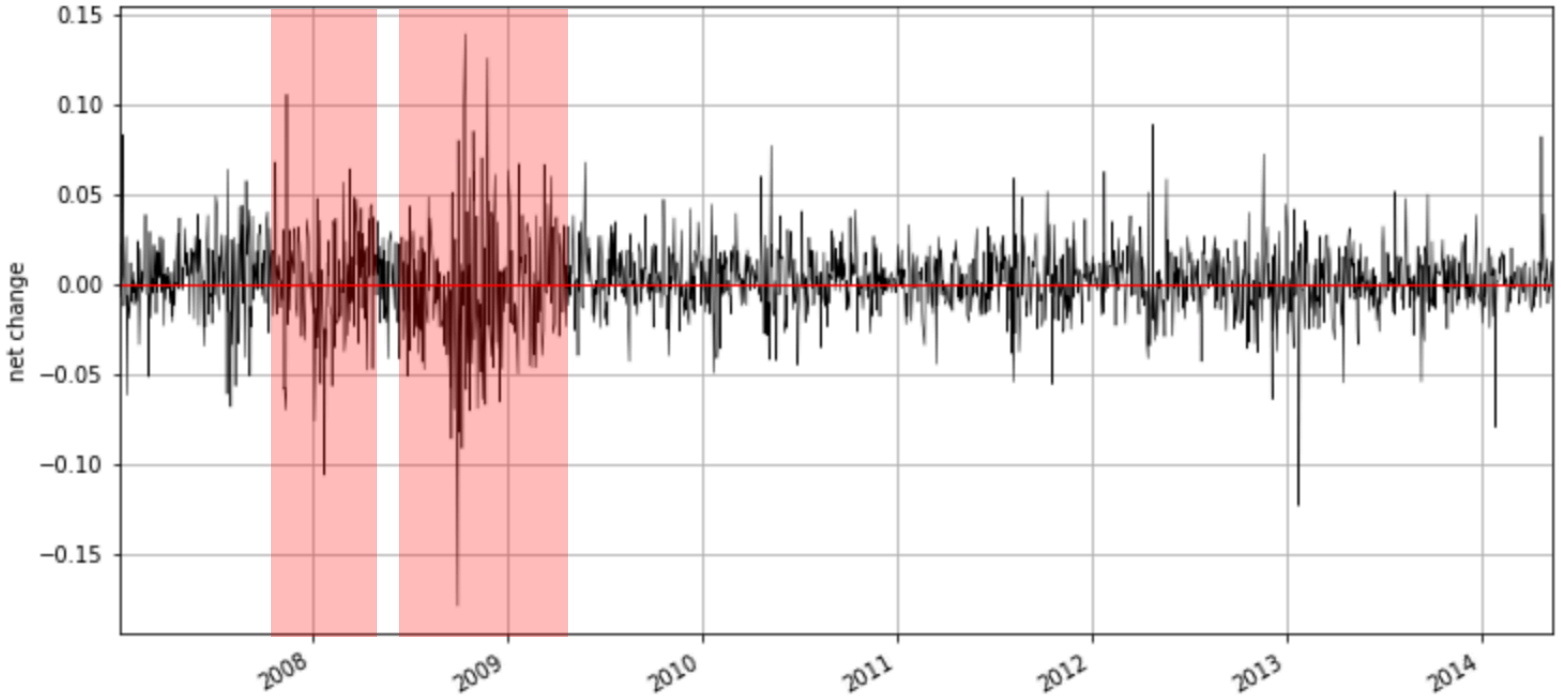
Net changes in AAPL prices over time

```
# plot daily changes
net_returns["AAPL"].plot(figsize = (12,6),grid=True,color = 'k',linewidth = 0.5)
# add line at y=0
plt.axhline(linewidth=1, color='r')
plt.xlabel('')
plt.ylabel('net change');
```

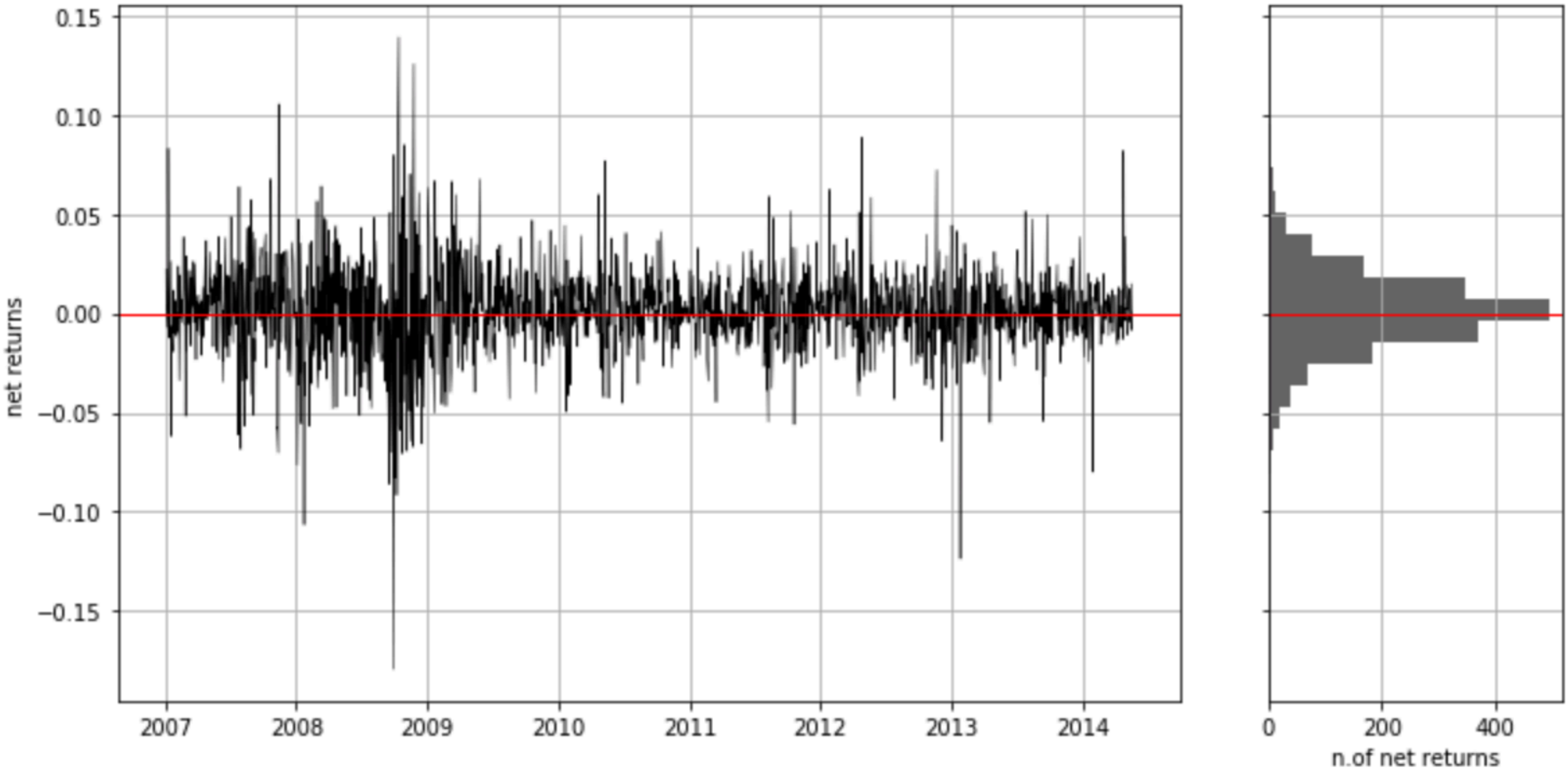


Periods of high volatility (high risk periods)

```
# plot daily changes
net_returns["AAPL"].plot(figsize = (12,6),grid=True,color = 'k',linewidth = 0.5)
# add line at y=0
plt.axhline(linewidth=1, color='r')
plt.xlabel('')
plt.ylabel('net change');
```



Net changes in prices over time

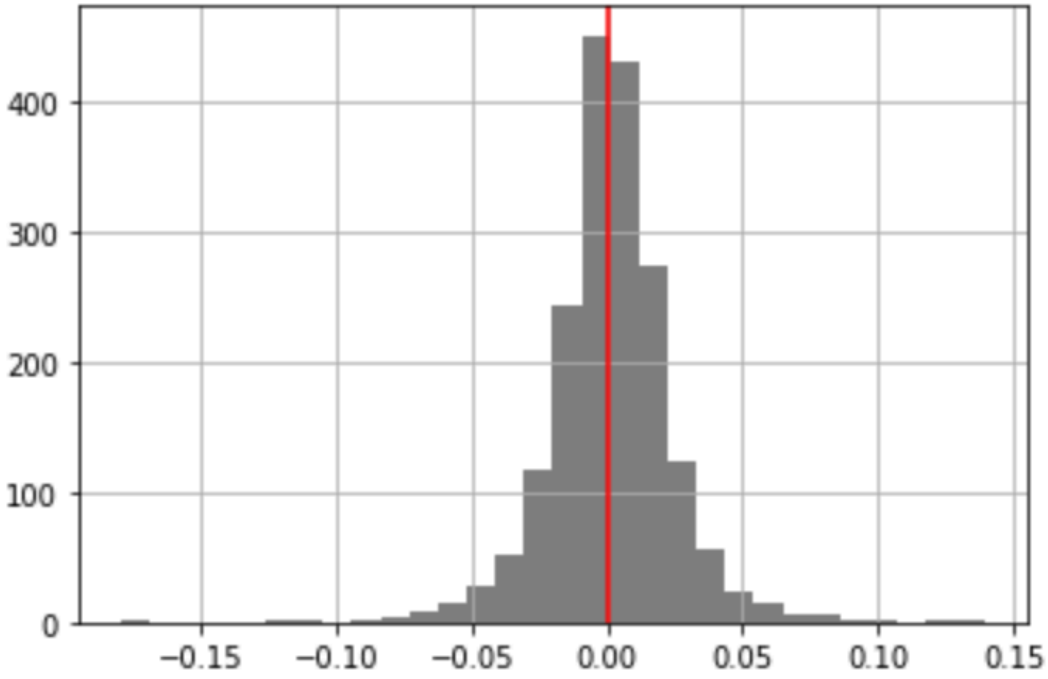


evolution

distribution

Distribution of net changes

```
x = net_returns["AAPL"]
# histogram of net returns
x.hist(color = 'k',bins=30,alpha=0.5,grid=True)
plt.axvline(linewidth=1.5,color='r');
```



Is this a sample from a normal distribution?

Quantile-Quantile plot

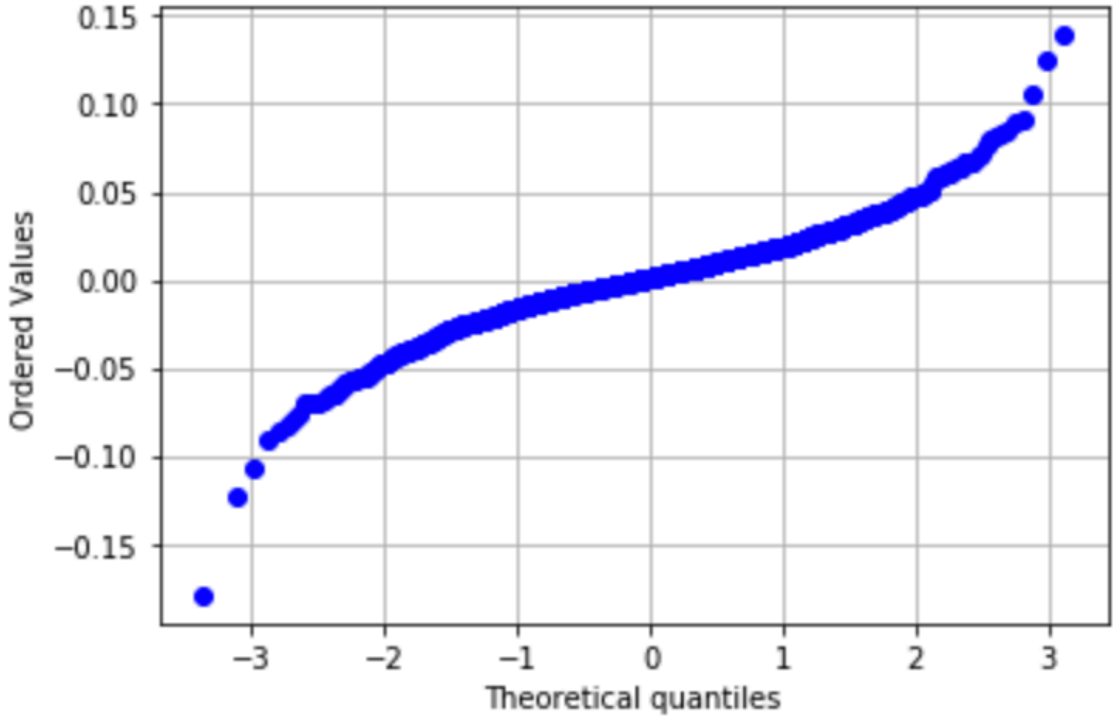
```
import scipy.stats as stats
x = net_returns["AAPL"]
stats.probplot(x, dist="norm", plot=plt)
```

assumed distribution

QQ plot of net changes

scipy

```
import scipy.stats as stats
x = net_returns["AAPL"]
stats.probplot(x, dist="norm", plot=plt)
plt.title('')
plt.grid();
```



Plot is not linear ⇒ The assumed distribution is not the right one

Gross Returns (all stocks)

Gross returns in prices

$$\text{gross return} = \frac{Value\ in\ period\ 2}{Value\ in\ period\ 1}$$

```
gross_returns = close_px / close_px.shift(1)
gross_returns
```

Symbols	AAPL	CVX	IBM	XOM	GS
Date					
2007-01-03	NaN	NaN	NaN	NaN	NaN
2007-01-04	1.022196	0.990278	1.010692	0.981245	0.990683
2007-01-05	0.992878	1.003842	0.990947	1.007150	1.001006
2007-01-08	1.004938	1.012757	1.015192	0.991944	1.023511
2007-01-09	1.083070	0.988523	1.011830	0.992291	1.001718

Gross returns in prices

$$\text{gross return} = \frac{Value\ in\ period\ 2}{Value\ in\ period\ 1}$$

```
gross_returns = close_px / close_px.shift(1)
gross_returns
```

Symbols	AAPL	CVX	IBM	XOM	GS
Date					
2007-01-03	NaN	NaN	NaN	NaN	NaN
2007-01-04	1.022196	0.990278	1.010692	0.981245	0.990683
2007-01-05	0.992878	1.003842	0.990947	1.007150	1.001006
2007-01-08	1.004938	1.012757	1.015192	0.991944	1.023511
2007-01-09	1.083070	0.988523	1.011830	0.992291	1.001718

Gross returns above 1.0 show an increase in stock price from previous day

Gross returns in prices

$$\text{gross return} = \frac{Value\ in\ period\ 2}{Value\ in\ period\ 1}$$

```
gross_returns = close_px / close_px.shift(1)
gross_returns
```

Symbols	AAPL	CVX	IBM	XOM	GS
Date					
2007-01-03	NaN	NaN	NaN	NaN	NaN
2007-01-04	1.022196	0.990278	1.010692	0.981245	0.990683
2007-01-05	0.992878	1.003842	0.990947	1.007150	1.001006
2007-01-08	1.004938	1.012757	1.015192	0.991944	1.023511
2007-01-09	1.083070	0.988523	1.011830	0.992291	1.001718

Cumulative Gross returns

```
gross_returns.cumsum()
```

AAPL	CVX	IBM	XOM
NaN	NaN	NaN	NaN
1.022196	0.990278	1.010692	0.981245
2.015075	1.994119	2.001639	1.988395
3.020013	3.006877	3.016831	2.980339
4.103083	3.995400	4.028661	3.972631

Gross returns in prices

$$\text{gross return} = \frac{Value\ in\ period\ 2}{Value\ in\ period\ 1}$$

```
gross_returns = close_px / close_px.shift(1)
gross_returns
```

Symbols	AAPL	CVX	IBM	XOM	GS
Date					
2007-01-03	NaN	NaN	NaN	NaN	NaN
2007-01-04	1.022196	0.990278	1.010692	0.981245	0.990683
2007-01-05	0.992878	1.003842	0.990947	1.007150	1.001006
2007-01-08	1.004938	1.012757	1.015192	0.991944	1.023511
2007-01-09	1.083070	0.988523	1.011830	0.992291	1.001718

```
gross_returns.cumsum()
```

AAPL	CVX	IBM	XOM
NaN	NaN	NaN	NaN
1.022196	0.990278	1.010692	0.981245
2.015075	1.994119	2.001639	1.988395
3.020013	3.006877	3.016831	2.980339
4.103083	3.995400	4.028661	3.972631

Gross returns in prices

$$\text{gross return} = \frac{Value\ in\ period\ 2}{Value\ in\ period\ 1}$$

```
gross_returns = close_px / close_px.shift(1)
gross_returns
```

Symbols	AAPL	CVX	IBM	XOM	GS
Date					
2007-01-03	NaN	NaN	NaN	NaN	NaN
2007-01-04	1.022196	0.990278	1.010692	0.981245	0.990683
2007-01-05	0.992878	1.003842	0.990947	1.007150	1.001006
2007-01-08	1.004938	1.012757	1.015192	0.991944	1.023511
2007-01-09	1.083070	0.988523	1.011830	0.992291	1.001718

```
gross_returns.cumsum()
```

AAPL	CVX	IBM	XOM
NaN	NaN	NaN	NaN
1.022196	0.990278	1.010692	0.981245
2.015075	1.994119	2.001639	1.988395
3.020013	3.006877	3.016831	2.980339
4.103083	3.995400	4.028661	3.972631

Cumulative Gross returns in prices

```
gross_returns = close_px / close_px.shift(1)
gross_returns
```

Symbols	AAPL	CVX	IBM	XOM	GS
Date					
2007-01-03	NaN	NaN	NaN	NaN	NaN
2007-01-04	1.022196	0.990278	1.010692	0.981245	0.990683
2007-01-05	0.992878	1.003842	0.990947	1.007150	1.001006
2007-01-08	1.004938	1.012757	1.015192	0.991944	1.023511
2007-01-09	1.083070	0.988523	1.011830	0.992291	1.001718

```
gross_returns.cumprod()
```

AAPL	CVX	IBM	XOM	GS
NaN	NaN	NaN	NaN	NaN
1.022196	0.990278	1.010692	0.981245	0.990683
1.014917	0.994082	1.001542	0.988261	0.991680
1.019929	1.006764	1.016758	0.980300	1.014996
1.104654	0.995209	1.028786	0.972743	1.016740

Cumulative Gross returns in prices

```
gross_returns = close_px / close_px.shift(1)
gross_returns
```

Symbols	AAPL	CVX	IBM	XOM	GS
Date					
2007-01-03	NaN	NaN	NaN	NaN	NaN
2007-01-04	1.022196	0.990278	1.010692	0.981245	0.990683
2007-01-05	0.992878	1.003842	0.990947	1.007150	1.001006
2007-01-08	1.004938	1.012757	1.015192	0.991944	1.023511
2007-01-09	1.083070	0.988523	1.011830	0.992291	1.001718

```
gross_returns.cumprod()
```

AAPL	CVX	IBM	XOM	GS
NaN	NaN	NaN	NaN	NaN
1.022196	0.990278	1.010692	0.981245	0.990683
1.014917	0.994082	1.001542	0.988261	0.991680
1.019929	1.006764	1.016758	0.980300	1.014996
1.104654	0.995209	1.028786	0.972743	1.016740

Cumulative Gross returns in prices

```
gross_returns = close_px / close_px.shift(1)
gross_returns
```

Symbols	AAPL	CVX	IBM	XOM	GS
Date					
2007-01-03	NaN	NaN	NaN	NaN	NaN
2007-01-04	1.022196	0.990278	1.010692	0.981245	0.990683
2007-01-05	0.992878	1.003842	0.990947	1.007150	1.001006
2007-01-08	1.004938	1.012757	1.015192	0.991944	1.023511
2007-01-09	1.083070	0.988523	1.011830	0.992291	1.001718

```
gross_returns.cumprod()
```

AAPL	CVX	IBM	XOM	GS
NaN	NaN	NaN	NaN	NaN
1.022196	0.990278	1.010692	0.981245	0.990683
1.014917	0.994082	1.001542	0.988261	0.991680
1.019929	1.006764	1.016758	0.980300	1.014996
1.104654	0.995209	1.028786	0.972743	1.016740

Cumulative Gross returns in prices

```
gross_returns = close_px / close_px.shift(1)
gross_returns
```

Symbols	AAPL	CVX	IBM	XOM	GS
Date					
2007-01-03	NaN	NaN	NaN	NaN	NaN
2007-01-04	1.022196	0.990278	1.010692	0.981245	0.990683
2007-01-05	0.992878	1.003842	0.990947	1.007150	1.001006
2007-01-08	1.004938	1.012757	1.015192	0.991944	1.023511
2007-01-09	1.083070	0.988523	1.011830	0.992291	1.001718

```
gross_returns.cumprod()
```

AAPL	CVX	IBM	XOM	GS
NaN	NaN	NaN	NaN	NaN
1.022196	0.990278	1.010692	0.981245	0.990683
1.014917	0.994082	1.001542	0.988261	0.991680
1.019929	1.006764	1.016758	0.980300	1.014996
1.104654	0.995209	1.028786	0.972743	1.016740

Cumulative Gross returns in prices

```
gross_returns = close_px / close_px.shift(1)
gross_returns
```

Symbols	AAPL	CVX	IBM	XOM	GS
Date					
2007-01-03	NaN	NaN	NaN	NaN	NaN
2007-01-04	1.022196	0.990278	1.010692	0.981245	0.990683
2007-01-05	0.992878	1.003842	0.990947	1.007150	1.001006
2007-01-08	1.004938	1.012757	1.015192	0.991944	1.023511
2007-01-09	1.083070	0.988523	1.011830	0.992291	1.001718

```
gross_returns.cumprod()
```

AAPL	CVX	IBM	XOM	GS
NaN	NaN	NaN	NaN	NaN
1.022196	0.990278	1.010692	0.981245	0.990683
1.014917	0.994082	1.001542	0.988261	0.991680
1.019929	1.006764	1.016758	0.980300	1.014996
1.104654	0.995209	1.028786	0.972743	1.016740

one dollar in Jan 3rd becomes 1.022196 dollars in Jan 4th

Cumulative Gross returns in prices

```
gross_returns = close_px / close_px.shift(1)
gross_returns
```

Symbols	AAPL	CVX	IBM	XOM	GS
Date					
2007-01-03	NaN	NaN	NaN	NaN	NaN
2007-01-04	1.022196	0.990278	1.010692	0.981245	0.990683
2007-01-05	0.992878	1.003842	0.990947	1.007150	1.001006
2007-01-08	1.004938	1.012757	1.015192	0.991944	1.023511
2007-01-09	1.083070	0.988523	1.011830	0.992291	1.001718

```
gross_returns.cumprod()
```

AAPL	CVX	IBM	XOM	GS
NaN	NaN	NaN	NaN	NaN
1.022196	0.990278	1.010692	0.981245	0.990683
1.014917	0.994082	1.001542	0.988261	0.991680
1.019929	1.006764	1.016758	0.980300	1.014996
1.104654	0.995209	1.028786	0.972743	1.016740

one dollar in Jan 3rd becomes 1.022196 dollars in Jan 4th

one dollar in Jan 3rd becomes 1.014917 dollars in Jan 5th

Cumulative Gross returns in prices

```
gross_returns = close_px / close_px.shift(1)
gross_returns
```

Symbols	AAPL	CVX	IBM	XOM	GS
Date					
2007-01-03	NaN	NaN	NaN	NaN	NaN
2007-01-04	1.022196	0.990278	1.010692	0.981245	0.990683
2007-01-05	0.992878	1.003842	0.990947	1.007150	1.001006
2007-01-08	1.004938	1.012757	1.015192	0.991944	1.023511
2007-01-09	1.083070	0.988523	1.011830	0.992291	1.001718

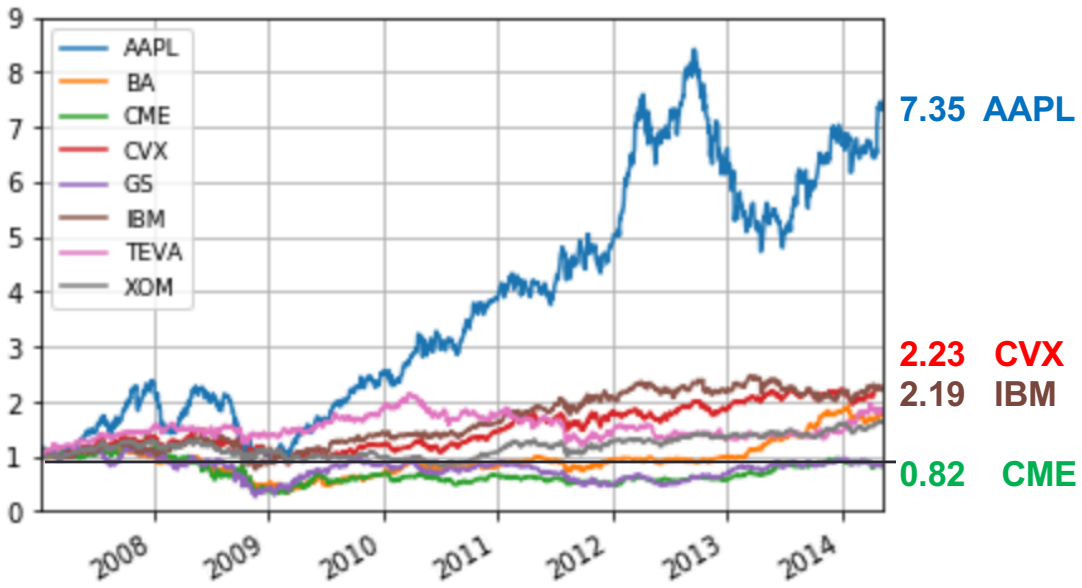
```
gross_returns.cumprod()
```

AAPL	CVX	IBM	XOM	GS
NaN	NaN	NaN	NaN	NaN
1.022196	0.990278	1.010692	0.981245	0.990683
1.014917	0.994082	1.001542	0.988261	0.991680
1.019929	1.006764	1.016758	0.980300	1.014996
1.104654	0.995209	1.028786	0.972743	1.016740

```
# give name to this dataframe
cum_gross_returns = gross_returns.cumprod()
```

Plotting Daily Cumulative Gross returns

```
cum_gross_returns.plot(grid=True)
plt.ylim(0,9)
plt.xlabel('')
plt.legend(prop={'size': 9});
```



Most profitable stocks

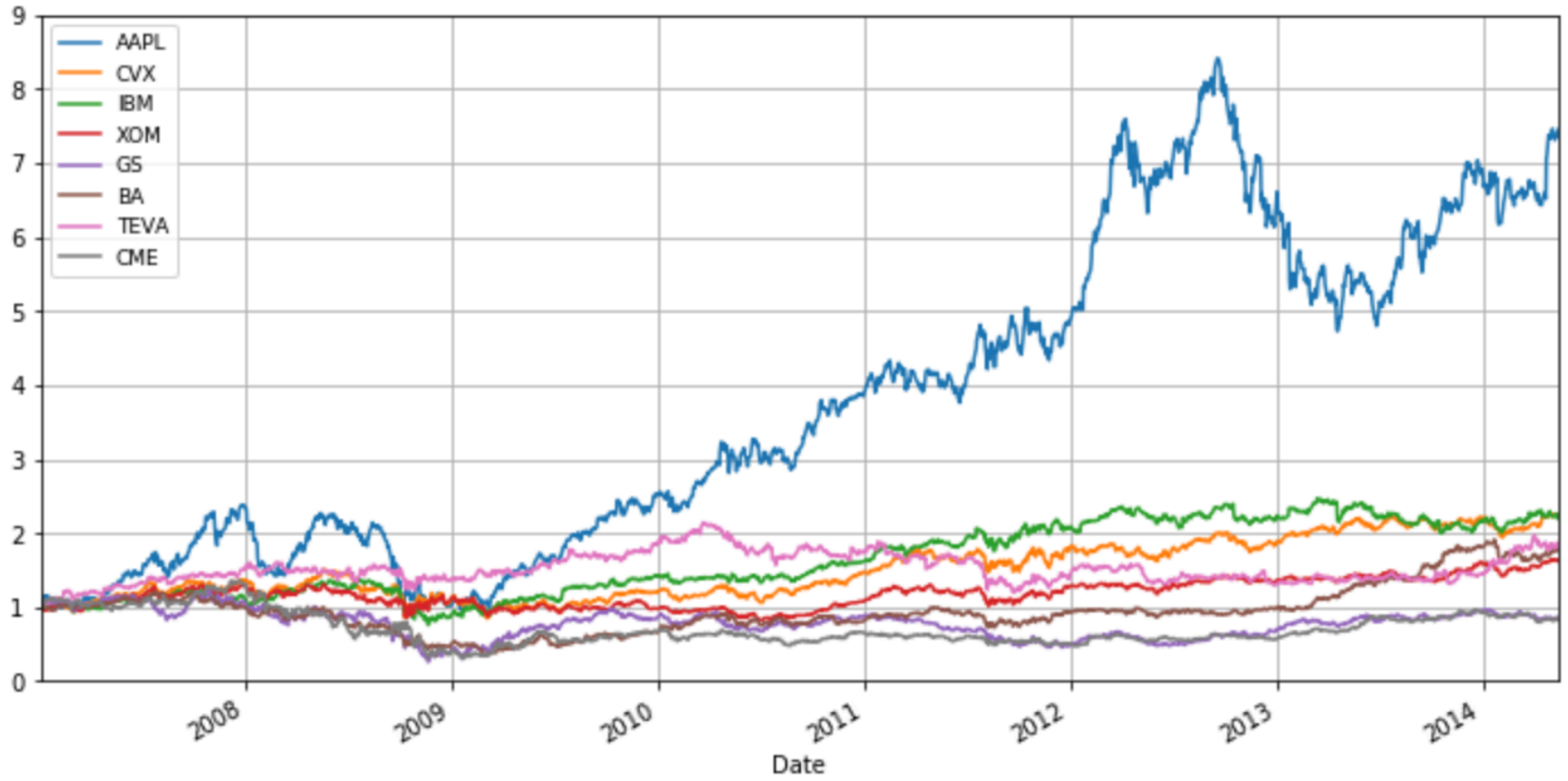
```
ratios = close_px.iloc[-1]\
             / close_px.iloc[0]
ratios
```

```
AAPL    7.351302
BA      1.760818
CME     0.827797
CVX     2.231599
GS      0.846585
IBM     2.191799
TEVA    1.828238
XOM     1.619617
```

Monthly Returns

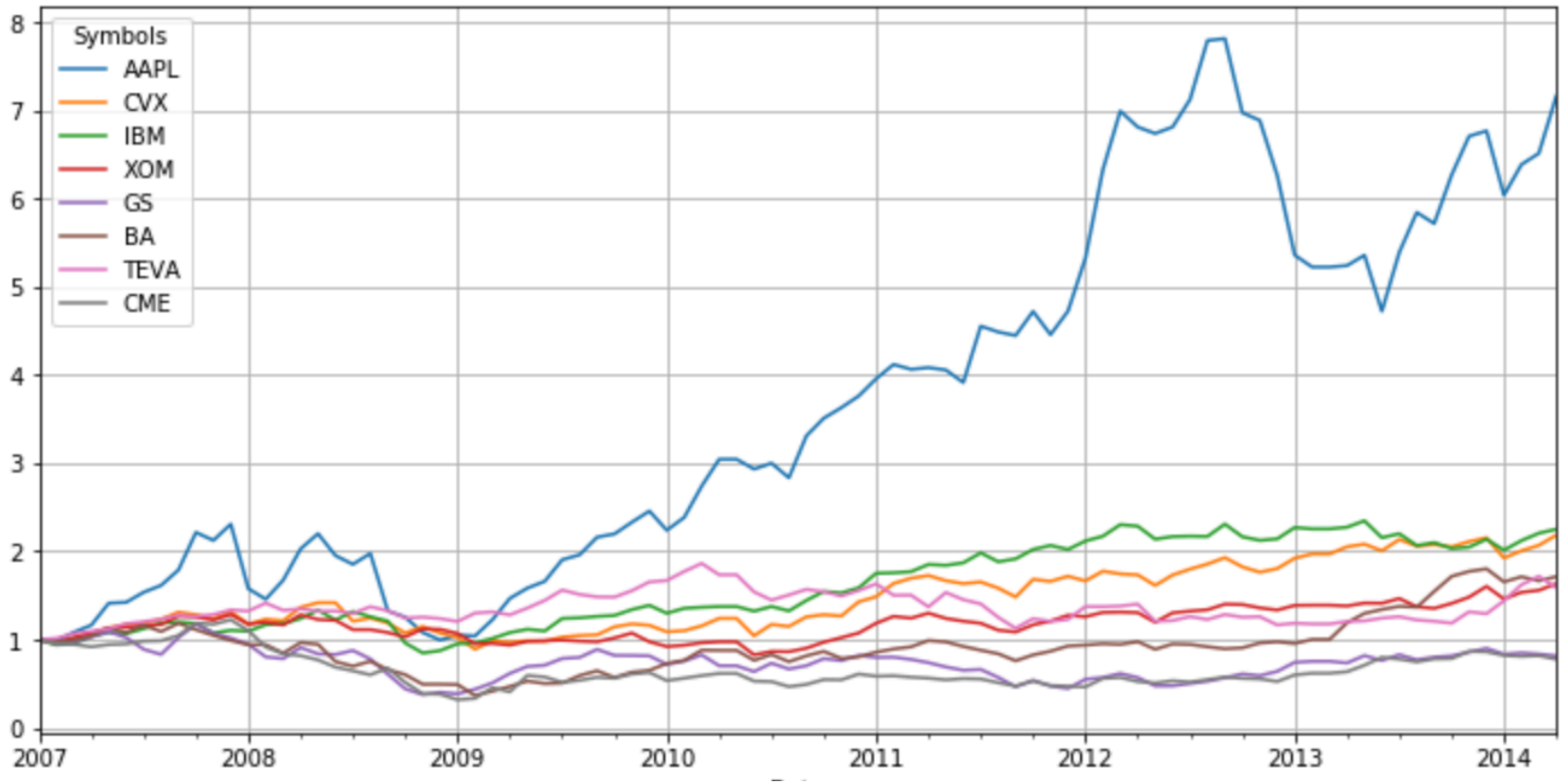
Daily Cumulative Gross returns

```
# plot cumulative gross returns
cum_gross_returns.plot(figsize=(12,6),grid=True)
plt.ylim(0,9)
plt.legend(prop={'size': 9});
```



Monthly Cumulative Gross returns

```
# plot the monthly cumulative returns
monthly_cr.plot(figsize=(12,6),grid=True);
```



Possible interval (frequency) values

Alias	Description
B	Business day frequency
C	Custom business day frequency
D	Calendar day frequency (the default)
W	Weekly frequency
M	Month end frequency
BM	Business month end frequency
CBM	Custom business month end frequency

MS	Month start frequency
BMS	Business month start frequency
CBMS	Custom business month start frequency
Q	Quarter end frequency
BQ	Business quarter frequency
QS	Quarter start frequency
BQS	Business quarter start frequency
A	Year-end frequency
BA	Business year-end frequency
AS	Year start frequency
BAS	Business year start frequency
H	Hourly frequency
T	Minute-by-minute frequency
S	Second-by-second frequency
L	Milliseconds
U	Microseconds

Moving Averages

AAPL Moving Averages

```
close_px.AAPL
```

```
Date
2007-01-03     2.577937
2007-01-04     2.635158
2007-01-05     2.616391
2007-01-08     2.629312
2007-01-09     2.847729
                  ...
2014-05-12    19.080269
2014-05-13    19.110191
2014-05-14    19.113733
2014-05-15    18.951204
2014-05-16    19.230890
Name: AAPL, Length: 1856, dtype: float64
```

store AAPL prices

```
aapl_close = close_px.AAPL
```

AAPL Moving Averages – pandas rolling( )

Compute Moving Averages (MA)

```
ma_30 = aapl_close.rolling(window=30).mean()
ma_90 = aapl_close.rolling(window=90).mean()
```

Store MAs in a new DataFrame df2

```
df2 = pd.DataFrame()
df2['Price'] = aapl_close
df2['30 days m.a.'] = ma_30
df2['90 days m.a.'] = ma_90
```

AAPL Moving Averages

```
df2.iloc[26:33,]
```

Date	Price	30 days m.a.	90 days m.a.
2007-02-09	2.557290	NaN	NaN
2007-02-12	2.606735	NaN	NaN
2007-02-13	2.601206	NaN	NaN
2007-02-14	2.619633	2.687903	NaN
2007-02-15	2.616868	2.689346	NaN
2007-02-16	2.605198	2.688497	NaN
2007-02-20	2.638059	2.689367	NaN

first 30-day
moving average

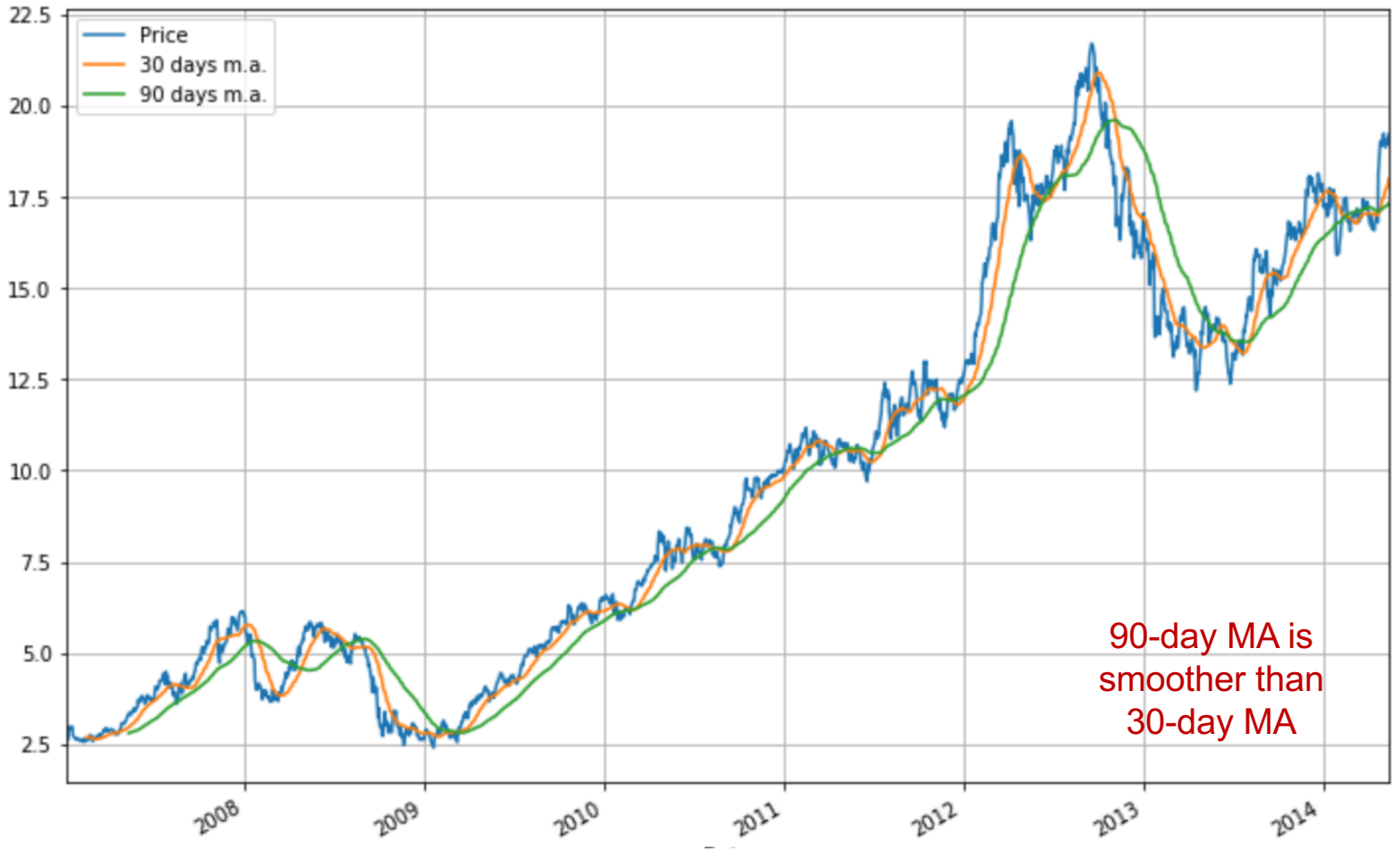
```
df2.iloc[86:93,]
```

Date	Price	30 days m.a.	90 days m.a.
2007-05-08	3.226478	2.927263	NaN
2007-05-09	3.282372	2.938953	NaN
2007-05-10	3.296499	2.953387	NaN
2007-05-11	3.339493	2.968733	2.800431
2007-05-14	3.358535	2.985572	2.809152
2007-05-15	3.302027	2.999771	2.816612
2007-05-16	3.296499	3.012915	2.824218

first 90-day
moving average

AAPL Moving Averages

```
# plot prices and moving averages
df2.plot(figsize=(12,8))
plt.legend(loc='upper left')
plt.grid()
```

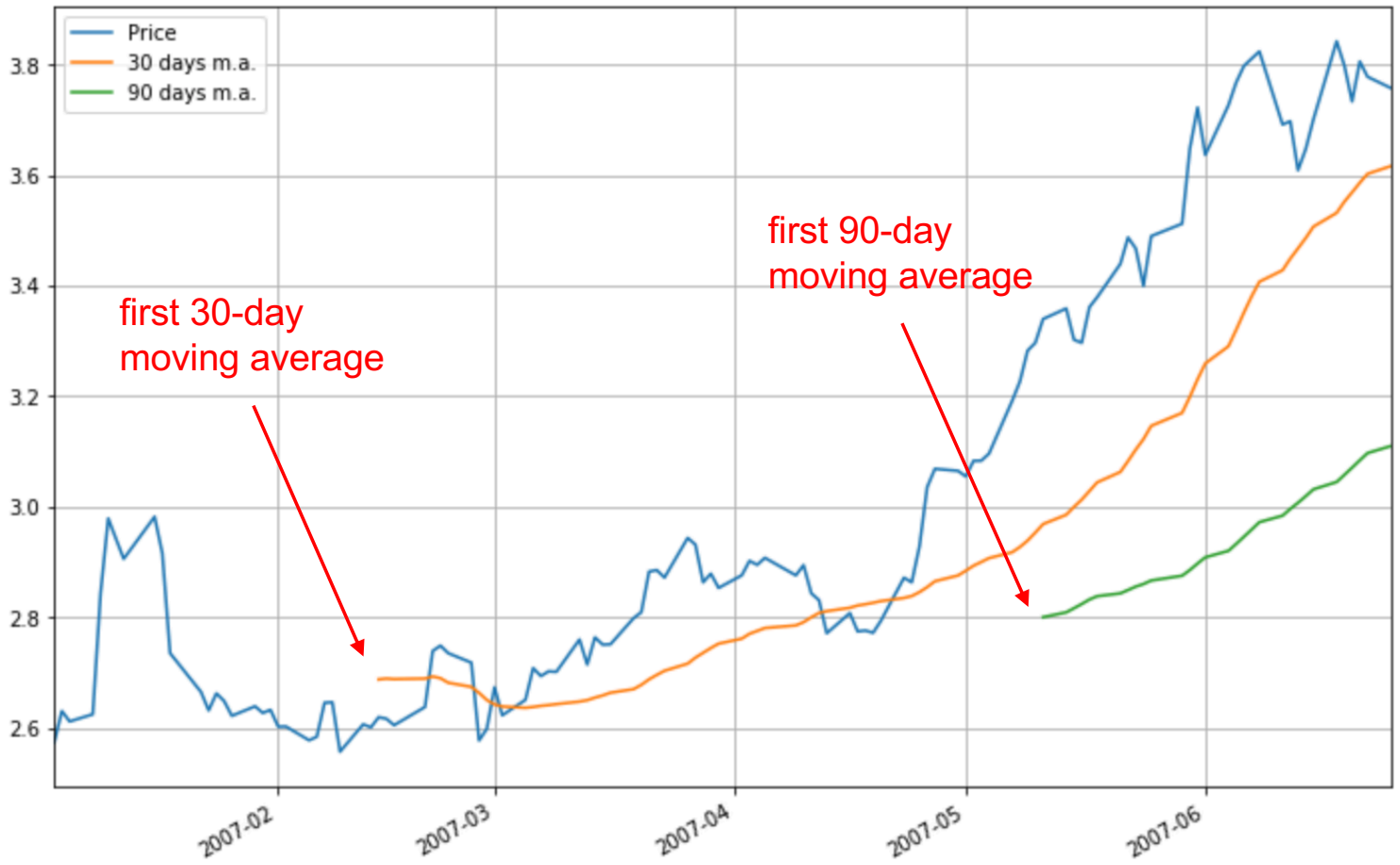


AAPL Moving Averages

```
# plot prices and moving averages
df2.plot(figsize=(12,8))
plt.legend(loc='upper left')
plt.grid()
```

AAPL Moving Averages

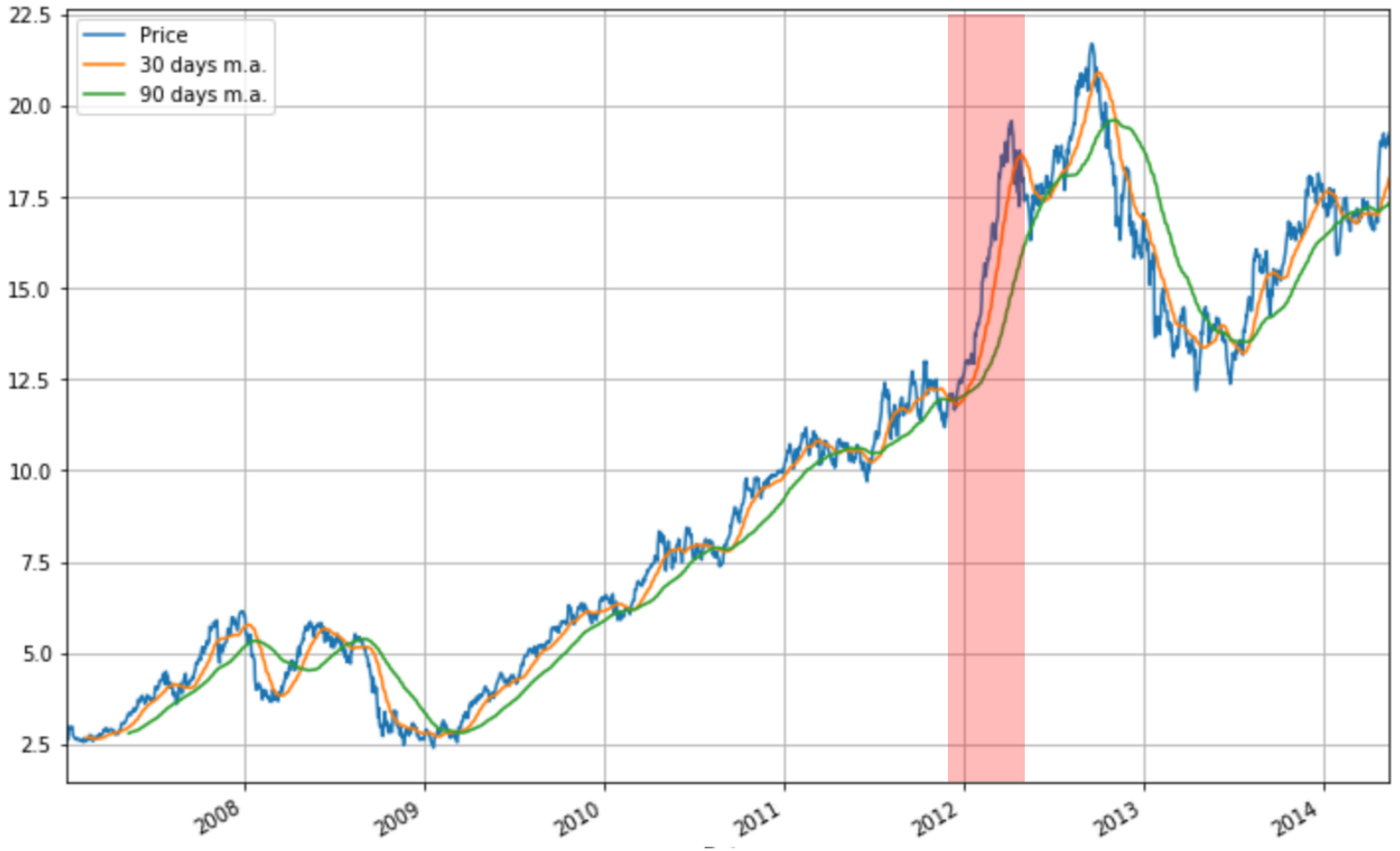
```
# plot first 120 days
df2.iloc[0:120,].plot(figsize=(12,8))
plt.legend(loc='upper left')
plt.xlabel('')
plt.grid()
```



AAPL Moving Averages

```
# plot prices and moving averages
df2.plot(figsize=(12,8))
plt.legend(loc='upper left')
plt.grid()
```

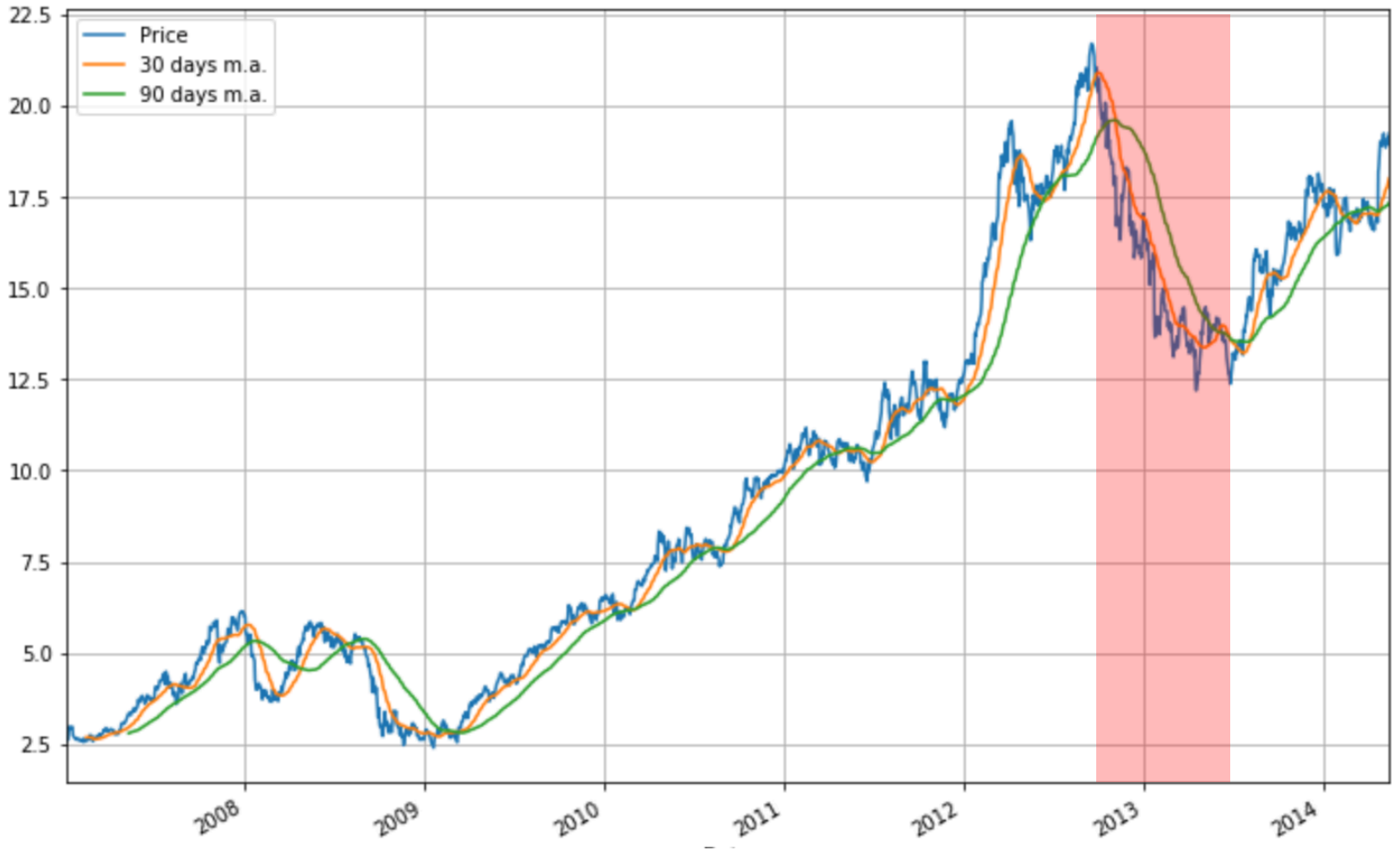
Lags below when data is increasing



AAPL Moving Averages

```
# plot prices and moving averages
df2.plot(figsize=(12,8))
plt.legend(loc='upper left')
plt.grid()
```

Lags above when data is decreasing



Correlation of net returns

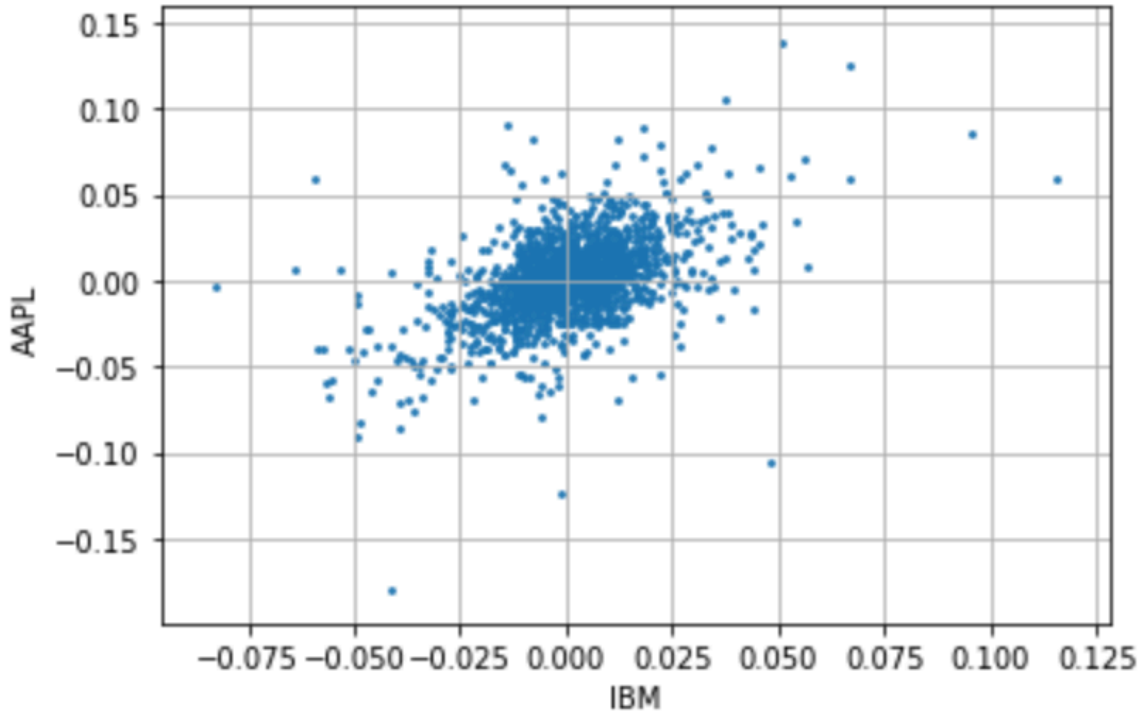
Net returns

```
net_returns = close_px / close_px.shift(1) - 1
net_returns[:9]
```

Date	AAPL	BA	CME	CVX	GS	IBM	TEVA	XOM
2007-01-03	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2007-01-04	0.022196	0.004037	0.007267	-0.009723	-0.009317	0.010693	0.012156	-0.018756
2007-01-05	-0.007121	-0.004244	0.004688	0.003842	0.001006	-0.009053	0.018015	0.007150
2007-01-08	0.004938	-0.002355	0.010638	0.012757	0.023512	0.015192	0.020801	-0.008056
2007-01-09	0.083070	-0.010569	0.005776	-0.011477	0.001718	0.011830	-0.001825	-0.007708
2007-01-10	0.047856	0.014431	0.028588	-0.017273	0.019747	-0.011791	0.002133	-0.015259
2007-01-11	-0.012371	-0.004817	0.005963	-0.010374	0.018116	-0.002427	-0.007905	-0.000141
2007-01-12	-0.012317	-0.007992	0.031132	0.024167	0.009958	0.006995	0.003065	0.023669
2007-01-16	0.026210	-0.001475	-0.007405	-0.009666	-0.001869	0.014898	0.005194	-0.014175

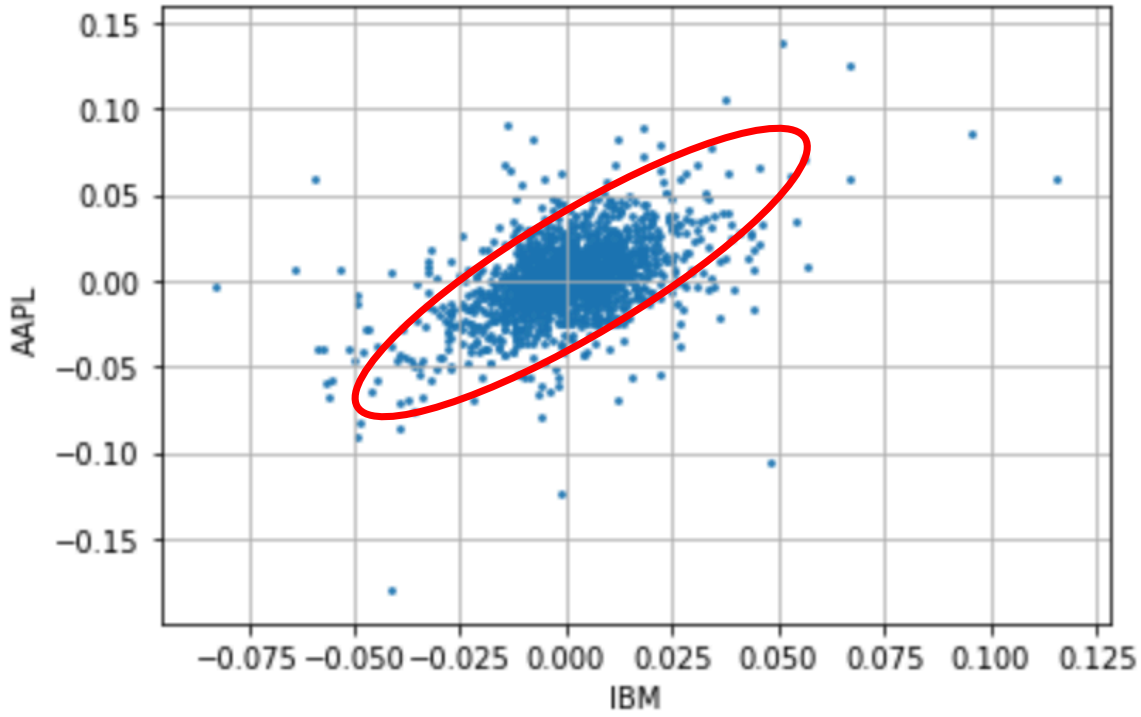

Are IBM & APPL stocks related?

```
plt.scatter(net_returns['IBM'], net_returns['AAPL'],s=3)
plt.xlabel('IBM')
plt.ylabel('AAPL')
plt.grid()
```



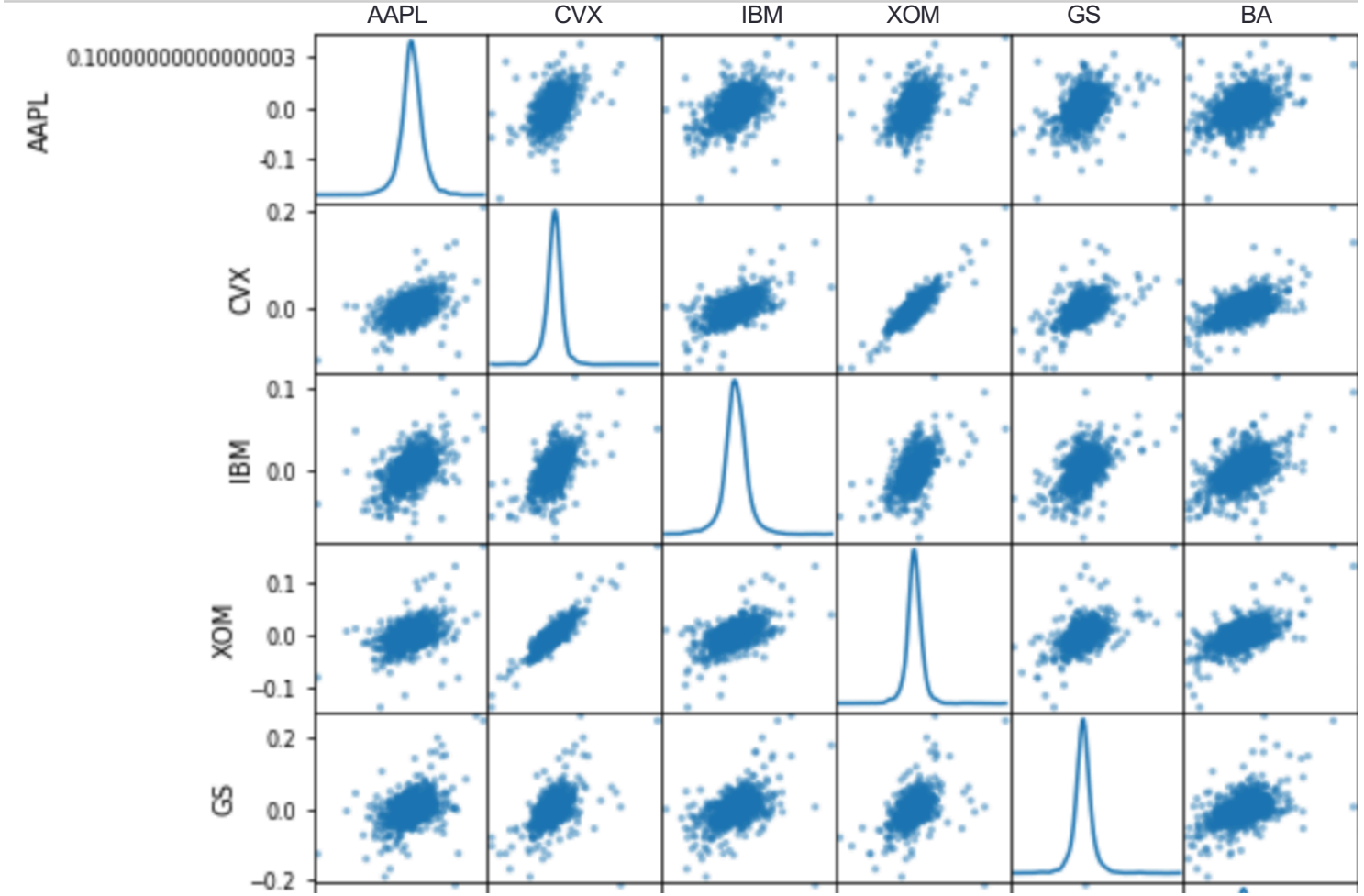
Comparing IBM & APPL net returns

```
plt.scatter(net_returns['IBM'], net_returns['AAPL'],s=3)
plt.xlabel('IBM')
plt.ylabel('AAPL')
plt.grid()
```



Comparing all stocks' net returns

```
# scatterplot for ALL stocks returns
pd.plotting.scatter_matrix(net_returns, diagonal='kde',
                           figsize=(10,10));
```

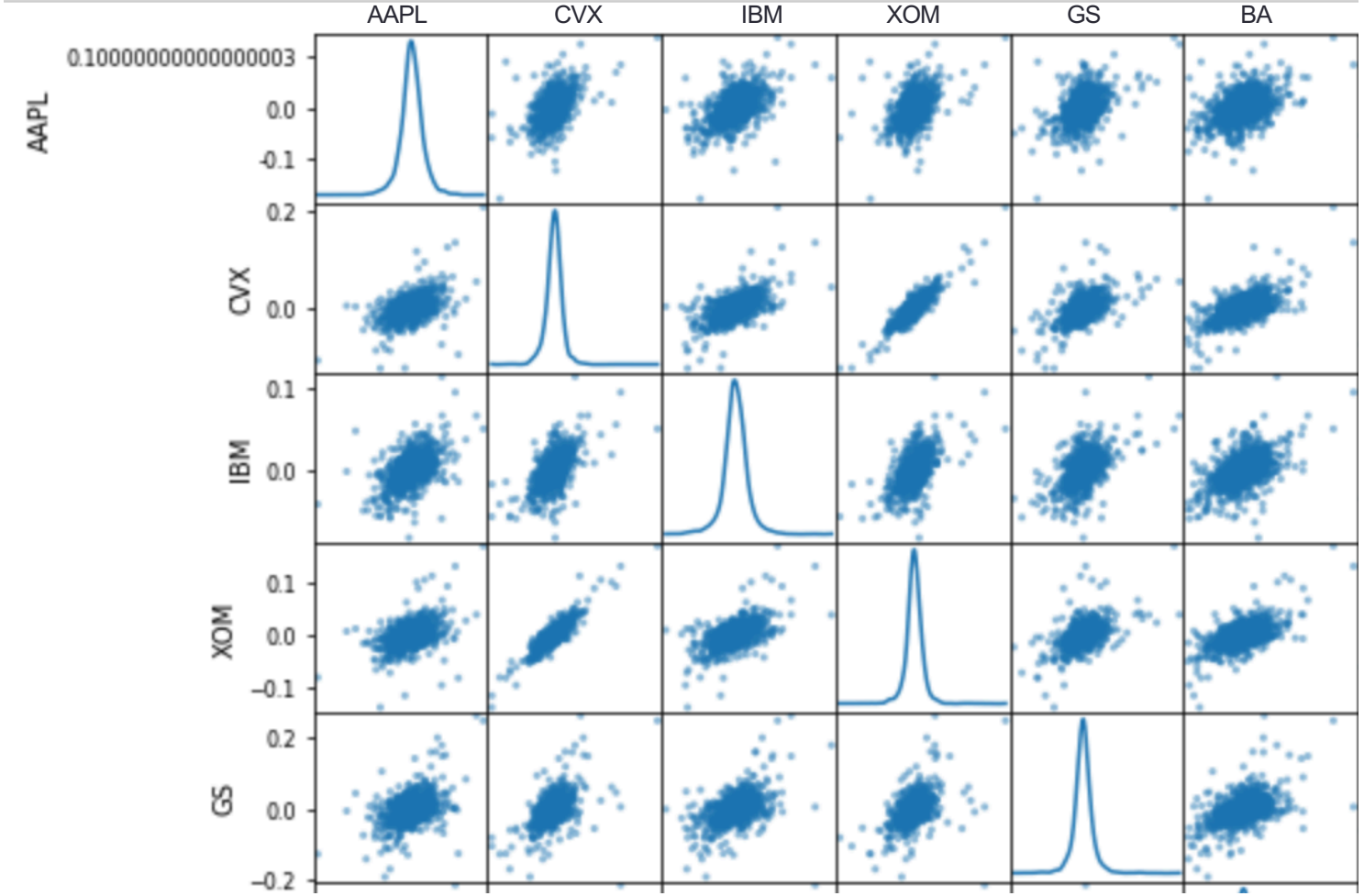


Comparing all net returns

kernel density estimate

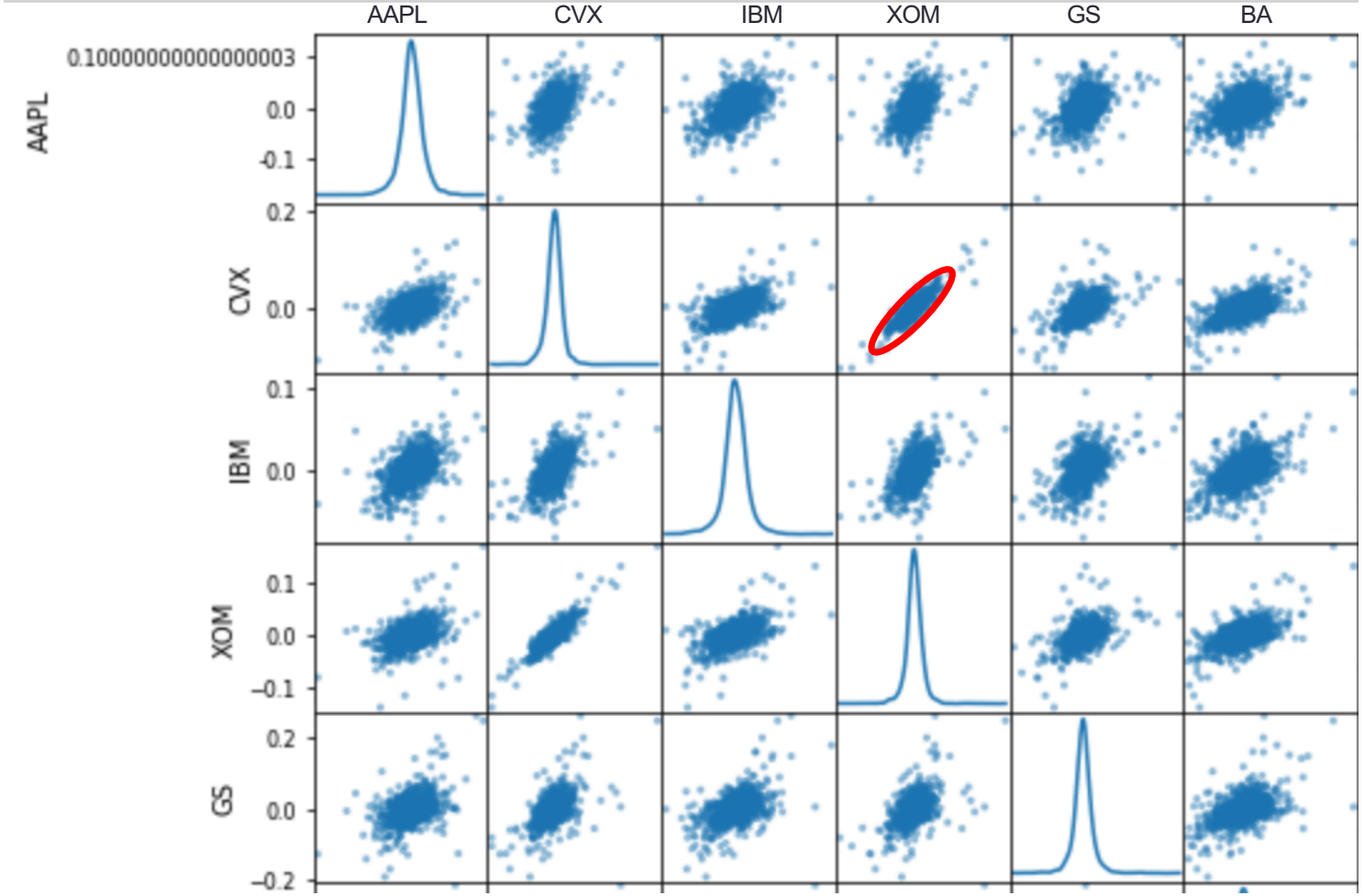
a smooth curve replacing a histogram

```
# scatterplot for ALL stocks returns
pd.plotting.scatter_matrix(net_returns, diagonal='kde',
                           figsize=(10,10));
```



Comparing all net returns

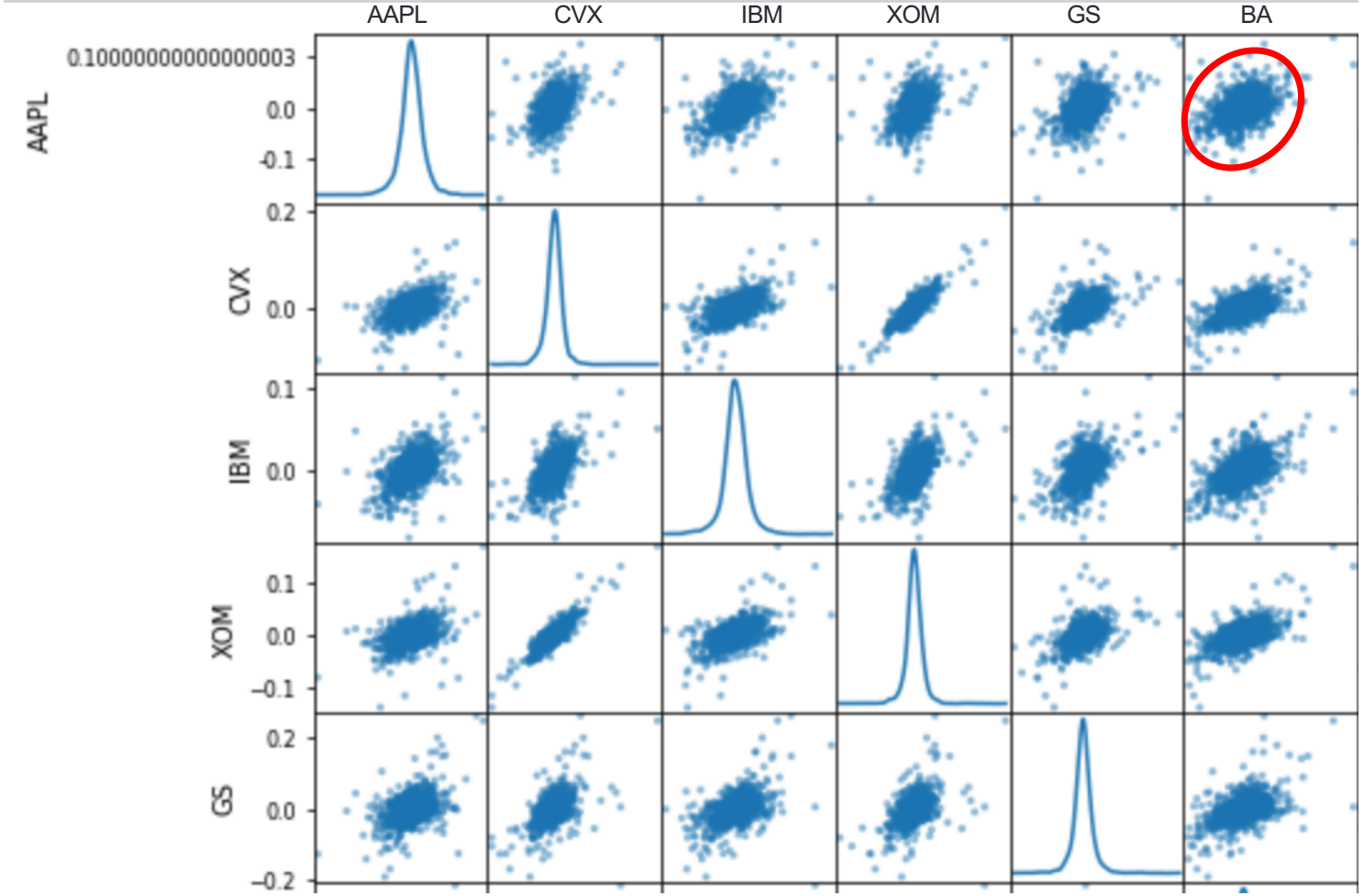
```
# scatterplot for ALL stocks returns
pd.plotting.scatter_matrix(net_returns, diagonal='kde',
                           figsize=(10,10));
```



stocks highly related

Comparing all net returns

```
# scatterplot for ALL stocks returns
pd.plotting.scatter_matrix(net_returns, diagonal='kde',
                           figsize=(10,10));
```



stocks not related

Find stocks with smallest correlation

```
net_returns = close_px / close_px.shift(1) - 1
net_returns[:9]
```

Date	AAPL	BA	CME	CVX	GS	IBM	TEVA	XOM
2007-01-03	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2007-01-04	0.022196	0.004037	0.007267	-0.009723	-0.009317	0.010693	0.012156	-0.018756
2007-01-05	-0.007121	-0.004244	0.004688	0.003842	0.001006	-0.009053	0.018015	0.007150
2007-01-08	0.004938	-0.002355	0.010638	0.012757	0.023512	0.015192	0.020801	-0.008056
2007-01-09	0.083070	-0.010569	0.005776	-0.011477	0.001718	0.011830	-0.001825	-0.007708
2007-01-10	0.047856	0.014431	0.028588	-0.017273	0.019747	-0.011791	0.002133	-0.015259
2007-01-11	-0.012371	-0.004817	0.005963	-0.010374	0.018116	-0.002427	-0.007905	-0.000141
2007-01-12	-0.012317	-0.007992	0.031132	0.024167	0.009958	0.006995	0.003065	0.023669
2007-01-16	0.026210	-0.001475	-0.007405	-0.009666	-0.001869	0.014898	0.005194	-0.014175

```
net_returns.shape
```

```
(1855, 8)
```

Correlation matrix of net returns

```
corrs = net_returns.corr()
corrs
```

	AAPL	BA	CME	CVX	GS	IBM	TEVA	XOM
AAPL	1.000000	0.423271	0.439085	0.457008	0.443804	0.498449	0.280115	0.417115
BA	0.423271	1.000000	0.473988	0.600928	0.495654	0.521861	0.347606	0.597275
CME	0.439085	0.473988	1.000000	0.519919	0.585881	0.490923	0.323112	0.504936
CVX	0.457008	0.600928	0.519919	1.000000	0.556283	0.578764	0.421735	0.891371
GS	0.443804	0.495654	0.585881	0.556283	1.000000	0.523909	0.314261	0.512294
IBM	0.498449	0.521861	0.490923	0.578764	0.523909	1.000000	0.322244	0.567272
TEVA	0.280115	0.347606	0.323112	0.421735	0.314261	0.322244	1.000000	0.410558
XOM	0.417115	0.597275	0.504936	0.891371	0.512294	0.567272	0.410558	1.000000

```
type(corrs)
```

```
pandas.core.frame.DataFrame
```

```
corrs.shape
```

```
(8, 8)
```

In this set of stocks, all correlations are positive

Correlation matrix of net returns

```
corrs = net_returns.corr()
corrs
```

	AAPL	BA	CME	CVX	GS	IBM	TEVA	XOM
AAPL	1.000000	0.423271	0.439085	0.457008	0.443804	0.498449	0.280115	0.417115
BA	0.423271	1.000000	0.473988	0.600928	0.495654	0.521861	0.347606	0.597275
CME	0.439085	0.473988	1.000000	0.519919	0.585881	0.490923	0.323112	0.504936
CVX	0.457008	0.600928	0.519919	1.000000	0.556283	0.578764	0.421735	0.891371
GS	0.443804	0.495654	0.585881	0.556283	1.000000	0.523909	0.314261	0.512294
IBM	0.498449	0.521861	0.490923	0.578764	0.523909	1.000000	0.322244	0.567272
TEVA	0.280115	0.347606	0.323112	0.421735	0.314261	0.322244	1.000000	0.410558
XOM	0.417115	0.597275	0.504936	0.891371	0.512294	0.567272	0.410558	1.000000

```
type(corrs)
```

```
pandas.core.frame.DataFrame
```

```
corrs.shape
```

```
(8, 8)
```

Convert DataFrame to a numpy array

```
corrs.values
```

```
array([[0.        , 0.42327032, 0.43908419, 0.45700733, 0.44380395,
        0.49844826, 0.28011472, 0.41711523],
       [0.42327032, 0.        , 0.47398807, 0.60092704, 0.49565394,
        0.52186103, 0.34760586, 0.59727443],
       [0.43908419, 0.47398807, 0.        , 0.51991886, 0.58588144,
        0.49092268, 0.32311235, 0.50493638],
       [0.45700733, 0.60092704, 0.51991886, 0.        , 0.55628218,
        0.57876373, 0.42173468, 0.8913708 ],
       [0.44380395, 0.49565394, 0.58588144, 0.55628218, 0.        ,
        0.52390858, 0.31426153, 0.51229353],
       [0.49844826, 0.52186103, 0.49092268, 0.57876373, 0.52390858,
        0.        , 0.32224369, 0.5672723 ],
       [0.28011472, 0.34760586, 0.32311235, 0.42173468, 0.31426153,
        0.32224369, 0.        , 0.41055742],
       [0.41711523, 0.59727443, 0.50493638, 0.8913708 , 0.51229353,
        0.5672723 , 0.41055742, 0.        ]])
```

```
type(corrs.values)
```

```
numpy.ndarray
```

Correlation matrix of net returns

```
corrs = net_returns.corr()
corrs
```

Symbols	AAPL	CVX	IBM	XOM	GS	BA	TEVA	CME
Symbols								
AAPL	1.000000	0.456852	0.498455	0.417059	0.443742	0.423168	0.280114	0.438881
CVX	0.456852	1.000000	0.578713	0.891354	0.556277	0.600937	0.421709	0.519952
IBM	0.498455	0.578713	1.000000	0.567262	0.523898	0.521834	0.322248	0.490851
XOM	0.417059	0.891354	0.567262	1.000000	0.512294	0.597273	0.410555	0.504922
GS	0.443742	0.556277	0.523898	0.512294	1.000000	0.495655	0.314258	0.585865
BA	0.423168	0.600937	0.521834	0.597273	0.495655	1.000000	0.347594	0.474002
TEVA	0.280114	0.421709	0.322248	0.410555	0.314258	0.347594	1.000000	0.323076
CME	0.438881	0.519952	0.490851	0.504922	0.585865	0.474002	0.323076	1.000000

Find stocks with smallest correlation

```
corrs = net_returns.corr()
corrs
```

Symbols	AAPL	CVX	IBM	XOM	GS	BA	TEVA	CME
Symbols								
AAPL	1.000000	0.456852	0.498455	0.417059	0.443742	0.423168	0.280114	0.438881
CVX	0.456852	1.000000	0.578713	0.891354	0.556277	0.600937	0.421709	0.519952
IBM	0.498455	0.578713	1.000000	0.567262	0.523898	0.521834	0.322248	0.490851
XOM	0.417059	0.891354	0.567262	1.000000	0.512294	0.597273	0.410555	0.504922
GS	0.443742	0.556277	0.523898	0.512294	1.000000	0.495655	0.314258	0.585865
BA	0.423168	0.600937	0.521834	0.597273	0.495655	1.000000	0.347594	0.474002
TEVA	0.280114	0.421709	0.322248	0.410555	0.314258	0.347594	1.000000	0.323076
CME	0.438881	0.519952	0.490851	0.504922	0.585865	0.474002	0.323076	1.000000

```
# find stocks with smallest correlation
corrs.values.min()
```

```
0.28011377539952054
```

Find stocks with smallest correlation

```
corrs = net_returns.corr()
corrs
```

Symbols	AAPL	CVX	IBM
Symbols			
AAPL	1.000000	0.456852	0.498455
CVX	0.456852	1.000000	0.578713
IBM	0.498455	0.578713	1.000000
XOM	0.417059	0.891354	0.567262
GS	0.443742	0.556277	0.523898
BA	0.423168	0.600937	0.521834
TEVA	0.280114	0.421709	0.322248
CME	0.438881	0.519952	0.490851

```
# find stocks with smallest c
corrs.values.min()
```

```
0.28011377539952054
```

```
mincorr = corrs.values.min()
mincorr
```

```
0.2801148483844085
```

```
np.where(corrs == mincorr)
```

```
(array([0, 6]), array([6, 0]))
```

```
# display column names
corrs.columns[0],corrs.columns[6]
```

```
('AAPL', 'TEVA')
```

```
# smallest corr from TEVA and AAPL
```

Find stocks with largest correlation

```
# find stocks with largest correlation
matrix1 = corrs.values

# make diagonal values equal to zero
np.fill_diagonal(matrix1,0.0)
print(np.round(matrix1,3))
```

convert dataframe to a numpy array

```
[[0.    0.457 0.498 0.417 0.444 0.423 0.28  0.439]
 [0.457 0.    0.579 0.891 0.556 0.601 0.422 0.52 ]
 [0.498 0.579 0.    0.567 0.524 0.522 0.322 0.491]
 [0.417 0.891 0.567 0.    0.512 0.597 0.411 0.505]
 [0.444 0.556 0.524 0.512 0.    0.496 0.314 0.586]
 [0.423 0.601 0.522 0.597 0.496 0.    0.348 0.474]
 [0.28  0.422 0.322 0.411 0.314 0.348 0.    0.323]
 [0.439 0.52  0.491 0.505 0.586 0.474 0.323 0.   ]]
```

Correlation matrix of net returns

```
matrix1
[[0.    0.457 0.498 0.417 0.444 0.423 0.28  0.439]
 [0.457 0.    0.579 0.891 0.556 0.601 0.422 0.52 ]
 [0.498 0.579 0.    0.567 0.524 0.522 0.322 0.491]
 [0.417 0.891 0.567 0.    0.512 0.597 0.411 0.505]
 [0.444 0.556 0.524 0.512 0.    0.496 0.314 0.586]
 [0.423 0.601 0.522 0.597 0.496 0.    0.348 0.474]
 [0.28  0.422 0.322 0.411 0.314 0.348 0.    0.323]
 [0.439 0.52  0.491 0.505 0.586 0.474 0.323 0.   ]]
```

Correlation matrix of net returns

```
matrix1
```

```
[[0.    0.457 0.498 0.417 0.444 0.423 0.28  0.439]
 [0.457 0.    0.579 0.891 0.556 0.601 0.422 0.52 ]
 [0.498 0.579 0.    0.567 0.524 0.522 0.322 0.491]
 [0.417 0.891 0.567 0.    0.512 0.597 0.411 0.505]
 [0.444 0.556 0.524 0.512 0.    0.496 0.314 0.586]
 [0.423 0.601 0.522 0.597 0.496 0.    0.348 0.474]
 [0.28  0.422 0.322 0.411 0.314 0.348 0.    0.323]
 [0.439 0.52  0.491 0.505 0.586 0.474 0.323 0.   ]]
```

```
np.amax(matrix1)
```

```
0.8913544340738709
```

find largest value in matrix1

Correlation matrix of net returns

```
matrix1
[[0.    0.457 0.498 0.417 0.444 0.423 0.28  0.439]
 [0.457 0.    0.579 0.891 0.556 0.601 0.422 0.52 ]
 [0.498 0.579 0.    0.567 0.524 0.522 0.322 0.491]
 [0.417 0.891 0.567 0.    0.512 0.597 0.411 0.505]
 [0.444 0.556 0.524 0.512 0.    0.496 0.314 0.586]
 [0.423 0.601 0.522 0.597 0.496 0.    0.348 0.474]
 [0.28  0.422 0.322 0.411 0.314 0.348 0.    0.323]
 [0.439 0.52  0.491 0.505 0.586 0.474 0.323 0.   ]]
```

```
np.amax(matrix1)
```

```
0.8913544340738709
```

find largest value in matrix1

Correlation matrix of net returns

```
matrix1
[[0.    0.457 0.498 0.417 0.444 0.423 0.28  0.439]
 [0.457 0.    0.579 0.891 0.556 0.601 0.422 0.52 ]
 [0.498 0.579 0.    0.567 0.524 0.522 0.322 0.491]
 [0.417 0.891 0.567 0.    0.512 0.597 0.411 0.505]
 [0.444 0.556 0.524 0.512 0.    0.496 0.314 0.586]
 [0.423 0.601 0.522 0.597 0.496 0.    0.348 0.474]
 [0.28  0.422 0.322 0.411 0.314 0.348 0.    0.323]
 [0.439 0.52  0.491 0.505 0.586 0.474 0.323 0.   ]]
```

```
np.amax(matrix1)
```

```
0.8913544340738709
```

```
np.where(matrix1 == np.amax(matrix1))
```

```
(array([1, 3]), array([3, 1]))
```

Correlation matrix of net returns

```
matrix1
```

```
[[0.    0.457 0.498 0.417 0.444 0.423 0.28  0.439]
 [0.457 0.    0.579 0.891 0.556 0.601 0.422 0.52 ]
 [0.498 0.579 0.    0.567 0.524 0.522 0.322 0.491]
 [0.417 0.891 0.567 0.    0.512 0.597 0.411 0.505]
 [0.444 0.556 0.524 0.512 0.    0.496 0.314 0.586]
 [0.423 0.601 0.522 0.597 0.496 0.    0.348 0.474]
 [0.28  0.422 0.322 0.411 0.314 0.348 0.    0.323]
 [0.439 0.52  0.491 0.505 0.586 0.474 0.323 0.   ]]
```

```
np.amax(matrix1)
```

```
0.8913544340738709
```

```
np.where(matrix1 == np.amax(matrix1))
```

```
(array([1, 3]), array([3, 1]))
```

```
corrs.columns[1],corrs.columns[3]
```

```
('CVX', 'XOM')
```

Correlation matrix of net returns

```
corrs = net_returns.corr()
corrs
```

Symbols	AAPL	CVX	IBM	XOM	GS	BA	TEVA	CME
Symbols								
AAPL	1.000000	0.456852	0.498455	0.417059	0.443742	0.423168	0.280114	0.438881
CVX	0.456852	1.000000	0.578713	0.891354	0.556277	0.600937	0.421709	0.519952
IBM	0.498455	0.578713	1.000000	0.567262	0.523898	0.521834	0.322248	0.490851
XOM	0.417059	0.891354	0.567262	1.000000	0.512294	0.597273	0.410555	0.504922
GS	0.443742	0.556277	0.523898	0.512294	1.000000	0.495655	0.314258	0.585865
BA	0.423168	0.600937	0.521834	0.597273	0.495655	1.000000	0.347594	0.474002
TEVA	0.280114	0.421709	0.322248	0.410555	0.314258	0.347594	1.000000	0.323076
CME	0.438881	0.519952	0.490851	0.504922	0.585865	0.474002	0.323076	1.000000

```
corrs.columns[1],corrs.columns[3]
```

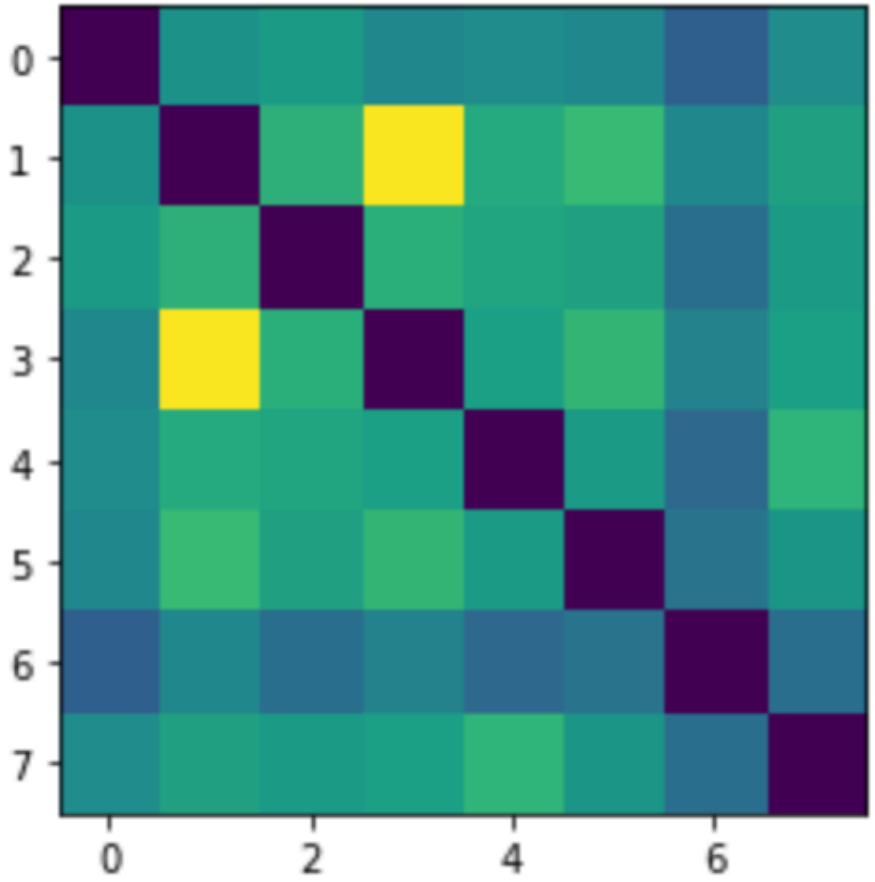
```
('CVX', 'XOM')
```

Heatmap

A 2D display from matplotlib

showing numerical values as colors

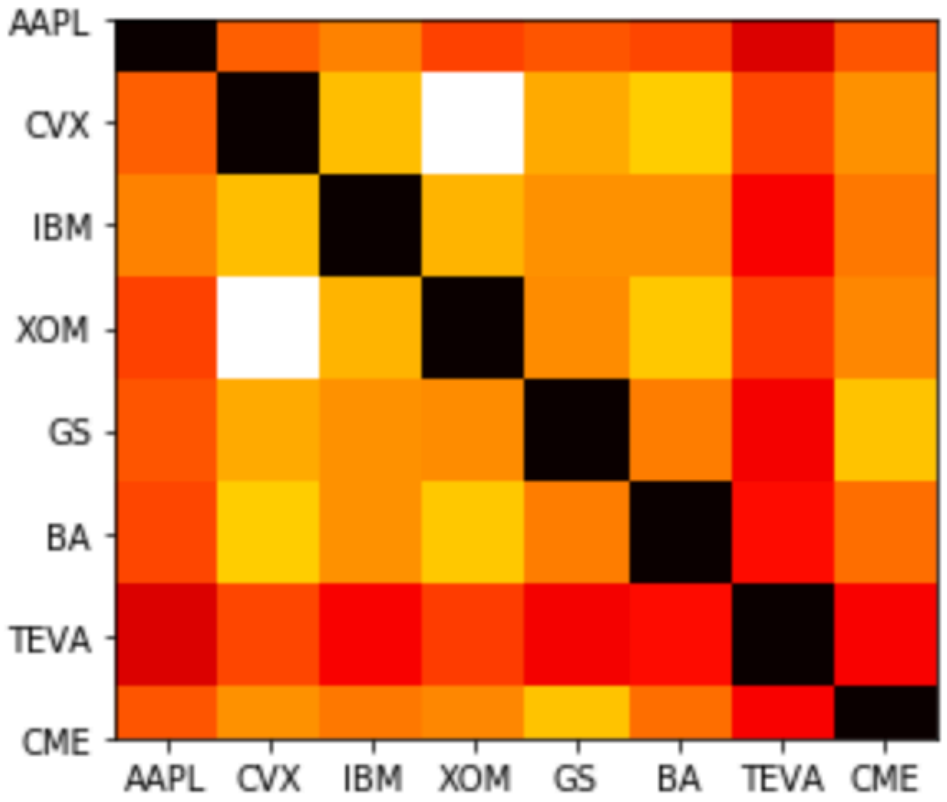
```
# Correlation matrix as a colored-table
plt.imshow(corrs);
```



Heatmap

```
col_names = corrs.columns

plt.imshow(corrs, cmap='hot')
plt.xticks(range(8), col_names)
plt.yticks(range(8), col_names);
```



Heatmap

```
col_names = corrs.columns

plt.imshow(corrs, cmap='hot')
plt.xticks(range(8), col_names)
plt.yticks(range(8), col_names)
plt.colorbar();
```