Major Bank Data

May 27, 2022

1 US Bank Data (2018-Present)

2 Imports

[46]:
```
import datetime as dt
import pandas as pd
from pandas_datareader import data as web
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from matplotlib.pyplot import figure
from matplotlib.pyplot import style
import numpy as np
import requests
from bs4 import BeautifulSoup
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

3 Start - End Date

[47]:
```
end = dt.datetime.now()
start = dt.datetime(2018,1,1)
start, end
```

[47]:
```
(datetime.datetime(2018, 1, 1, 0, 0),
 datetime.datetime(2022, 5, 27, 2, 20, 6, 167265))
```

4 US Major bank Tickers

[48]:
```
tickers = ['JPM', 'BAC', 'WFC', 'C']
tickers
```

[48]:
```
['JPM', 'BAC', 'WFC', 'C']
```

5 Getting Data

[49]:
```
df = web.get_data_yahoo(tickers, start, end)
df.head()
```

[49]:
```
Attributes  Adj Close                                     Close             \
Symbols           JPM        BAC        WFC          C         JPM        BAC
Date
2018-01-02  95.053261  27.295818  53.586853  64.795647  107.949997  29.900000
2018-01-03  95.150116  27.204527  53.999134  64.996048  108.059998  29.799999
2018-01-04  96.513199  27.560560  54.674553  65.797737  109.040001  30.190001
2018-01-05  95.893608  27.688372  55.042969  65.710602  108.339996  30.330000
2018-01-08  96.035248  27.496658  54.420181  64.943771  108.500000  30.120001

Attributes                            High             …        Low  \
Symbols           WFC          C         JPM        BAC  …        WFC
Date                                                    …
2018-01-02  61.090000  74.360001  108.019997  29.900000  …  60.700001
2018-01-03  61.560001  74.589996  108.489998  29.940001  …  61.099998
2018-01-04  62.330002  75.510002  110.029999  30.440001  …  61.910000
2018-01-05  62.750000  75.410004  109.550003  30.420000  …  62.090000
2018-01-08  62.040001  74.529999  108.680000  30.270000  …  61.939999

Attributes                  Open                                    \
Symbols             C         JPM        BAC        WFC          C
Date
2018-01-02  74.019997  107.629997  29.750000  61.040001  75.089996
2018-01-03  73.970001  107.860001  29.900000  61.220001  74.349998
2018-01-04  74.660004  108.360001  29.969999  61.980000  75.010002
2018-01-05  74.959999  109.260002  30.370001  62.759998  75.709999
2018-01-08  74.330002  108.150002  30.230000  62.660000  75.169998

Attributes      Volume
Symbols            JPM         BAC         WFC           C
Date
2018-01-02  13578800.0  57121600.0  13819300.0  15819800.0
2018-01-03  11901000.0  57865700.0  14203700.0  14657900.0
2018-01-04  12953700.0  76512500.0  18740500.0  16864900.0
2018-01-05  14155000.0  56445200.0  14217900.0  15300500.0
2018-01-08  12466500.0  42914800.0  15569400.0  14215700.0

[5 rows x 24 columns]
```

6 Index, Columns

[50]:
```
df.index
```

[50]:
```
DatetimeIndex(['2018-01-02', '2018-01-03', '2018-01-04', '2018-01-05',
               '2018-01-08', '2018-01-09', '2018-01-10', '2018-01-11',
               '2018-01-12', '2018-01-16',
               …
               '2022-05-13', '2022-05-16', '2022-05-17', '2022-05-18',
               '2022-05-19', '2022-05-20', '2022-05-23', '2022-05-24',
               '2022-05-25', '2022-05-26'],
              dtype='datetime64[ns]', name='Date', length=1109, freq=None)
```

[51]:
```
df.columns
```

[51]:
```
MultiIndex([('Adj Close', 'JPM'),
            ('Adj Close', 'BAC'),
            ('Adj Close', 'WFC'),
            ('Adj Close',   'C'),
            (    'Close', 'JPM'),
            (    'Close', 'BAC'),
            (    'Close', 'WFC'),
            (    'Close',   'C'),
            (     'High', 'JPM'),
            (     'High', 'BAC'),
            (     'High', 'WFC'),
            (     'High',   'C'),
            (      'Low', 'JPM'),
            (      'Low', 'BAC'),
            (      'Low', 'WFC'),
            (      'Low',   'C'),
            (     'Open', 'JPM'),
            (     'Open', 'BAC'),
            (     'Open', 'WFC'),
            (     'Open',   'C'),
            (   'Volume', 'JPM'),
            (   'Volume', 'BAC'),
            (   'Volume', 'WFC'),
            (   'Volume',   'C')],
           names=['Attributes', 'Symbols'])
```

7 Closing Prices

7.0.1 Clean data to display only the Date and Closing prices of each ticker

[52]:
```
Close = df.Close
Close.head()
Close.tail()
```

[52]:
```
Symbols            JPM        BAC        WFC          C
Date
2022-05-20  117.339996  33.860001  41.669998  49.750000
2022-05-23  124.599998  35.869999  43.820000  52.770000
2022-05-24  126.360001  35.650002  43.290001  52.680000
2022-05-25  127.239998  35.840000  44.119999  52.700001
2022-05-26  129.440002  36.669998  45.599998  54.090000
```

8 Returns (2018-current)

[60]:
```
tickers = ['JPM', 'BAC', 'WFC', 'C']
end = dt.datetime.now()
start = dt.datetime(2018,1,1)

returns = pd.DataFrame()

for ticker in tickers:
    data = web.DataReader(ticker, 'yahoo', start, end)
    data = pd.DataFrame(data)
    data[ticker] = data['Close'].pct_change()

    if returns.empty:
        returns = data[[ticker]]
    else:
        returns = returns.join(data[[ticker]], how = 'outer')

returns = returns.dropna() * 100
print(returns)
```

```
                JPM       BAC       WFC         C
Date
2018-01-03  0.101900 -0.334449  0.769359  0.309300
2018-01-04  0.906907  1.308729  1.250813  1.233417
2018-01-05 -0.641970  0.463728  0.673830 -0.132431
2018-01-08  0.147687 -0.692381 -1.131473 -1.166960
2018-01-09  0.506915  0.498007  0.354606  1.046557
…                …         …         …         …
2022-05-20 -0.819881 -1.712627 -0.785719 -0.060263
2022-05-23  6.187150  5.936203  5.159591  6.070353
```

```
2022-05-24  1.412522 -0.613319 -1.209491 -0.170552
2022-05-25  0.696421  0.532955  1.917297  0.037966
2022-05-26  1.729020  2.315843  3.354487  2.637570

[1108 rows x 4 columns]
```

9 Statistical Analysis

[31]:
```
Close.describe(percentiles=[0.1,0.5,0.9])
```

[31]:

Symbols	JPM	BAC	WFC	C
count	1109.000000	1109.000000	1109.000000	1109.000000
mean	123.248692	32.455410	45.046501	64.364536
std	22.509720	6.853059	10.651623	9.960617
min	79.029999	18.080000	21.139999	35.389999
10%	98.557999	24.512000	25.707999	49.580002
50%	115.160004	30.320000	47.790001	66.989998
90%	158.301996	43.123999	56.006000	75.164003
max	171.779999	49.380001	65.930000	81.910004

10 Statistical Analysis - Last 365 Days

[32]:
```
Close[Close.index > end - dt.timedelta(days=365)].describe(percentiles=[0.1,0.
↪5,0.9])
```

[32]:

Symbols	JPM	BAC	WFC	C
count	252.000000	252.000000	252.000000	252.000000
mean	151.702500	42.469524	48.589167	64.446825
std	14.393685	3.670118	3.894878	7.890203
min	117.339996	33.860001	41.669998	46.560001
10%	127.400003	37.582000	43.959999	51.337001
50%	155.709999	42.480001	48.109999	66.945000
90%	167.598001	47.499000	54.137000	72.093999
max	171.779999	49.380001	59.060001	79.860001

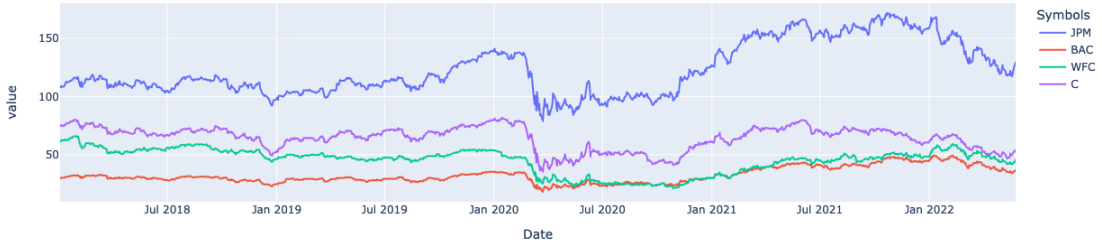
11 Visualizations

[33]:
```
import plotly.offline as pyo
pyo.init_notebook_mode(connected=True)

pd.options.plotting.backend = 'plotly'
```

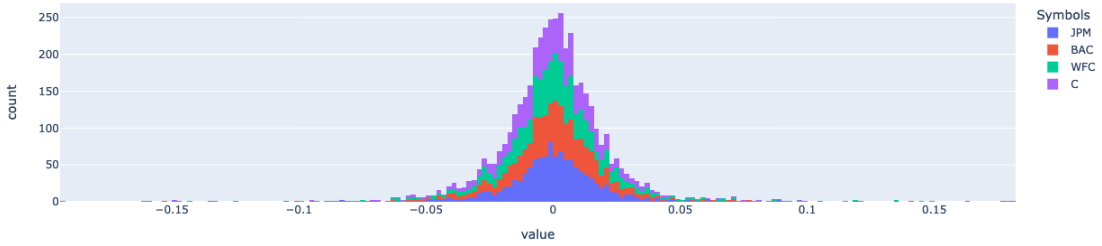
11.1 Closing prices of JPM, BAC, WFC, and C over a period of 12 years

[34]:
```
Close.plot()
```



11.2 Percent change over the 12-year period

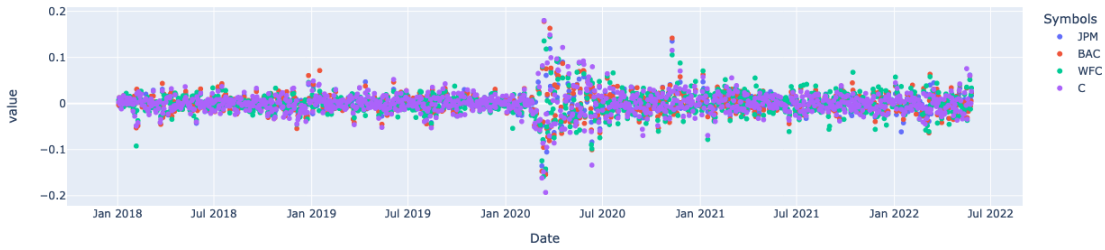
[35]:
```
Close.pct_change().plot(kind = 'hist')
```



11.3 Percent change on scatter plot

11.3.1 Allows to see times of increased volatility outside the normal percentage change

[36]:
```
Close.pct_change().plot(kind = 'scatter')
```



12 JPM Price Over 12-year Period

[37]:
```
Close['JPM'].plot()
```

13 BAC Price Over 12-year Period

[38]:
```
Close['BAC'].plot()
```

14 WFC Price Over 12-year Period

[39]:
```
Close['WFC'].plot()
```

15 C Price Over 12-year Period

[40]:
```
Close['C'].plot()
```