Finance Data Project

In this data project we will focus on exploratory data analysis of stock prices. Keep in mind, this project is just meant to practice your visualization and pandas skills, it is not meant to be a robust financial analysis or be taken as financial advice.

** NOTE: This project is extremely challenging because it will introduce a lot of new concepts and have you looking things up on your own (we'll point you in the right direction) to try to solve the tasks issued. Feel free to just go through the solutions lecture notebook and video as a "walkthrough" project if you don't want to have to look things up yourself. You'll still learn a lot that way! **

We'll focus on bank stocks and see how they progressed throughout the financial crisis all the way to early 2016.

Get the Data

In this section we will learn how to use pandas to directly read data from Google finance using pandas!

First we need to start with the proper imports, which we've already laid out for you here.

*Note: You'll need to install pandas-datareader for this to work! Pandas datareader allows you to read stock information directly from the internet Use these links for install guidance (**pip install pandas-datareader**), or just follow along with the video lecture.*

The Imports

Already filled out for you.

```
from pandas_datareader import data, wb
import pandas as pd
import numpy as np
import datetime
%matplotlib inline
```

```
import warnings
warnings.filterwarnings('ignore')
```

Data

We need to get data using pandas datareader. We will get stock information for the following banks:

- Bank of America
- CitiGroup
- Goldman Sachs
- JPMorgan Chase
- Morgan Stanley
- Wells Fargo

** Figure out how to get the stock data from Jan 1st 2006 to Jan 1st 2016 for each of these banks. Set each bank to be a separate dataframe, with the variable name for that bank being its ticker symbol. This will involve a few steps:**

1. Use datetime to set start and end datetime objects.
2. Figure out the ticker symbol for each bank.
3. Figure out how to use datareader to grab info on the stock.

** Use this documentation page for hints and instructions (it should just be a matter of replacing certain values. Use google finance as a source, for example:**

```
# Bank of America
BAC = data.DataReader("BAC", 'google', start, end)
```

WARNING: MAKE SURE TO CHECK THE LINK ABOVE FOR THE LATEST WORKING API. "google" MAY NOT ALWAYS WORK.

We also provide pickle file in the article lecture right before the video lectures.

```
df = pd.read_pickle('all_banks')
# Read the date the alternative way
```

```
#Bank of America
BAC = df['BAC']
#CitiGroup
C = df['C']
#Goldman Sachs
GS = df['GS']
#JPMorgan Chase
JPM = df ['JPM']
#Morgan Stanley
MS = df['MS']
#Wells Fargo
WFC = df['WFC']
```

MS

Stock Info	Open	High	Low	Close	Volume
Date					
2006-01-03	57.17	58.49	56.74	58.31	5377000
2006-01-04	58.70	59.28	58.35	58.35	7977800
2006-01-05	58.55	58.59	58.02	58.51	5778000
2006-01-06	58.77	58.85	58.05	58.57	6889800
2006-01-09	58.63	59.29	58.62	59.19	4144500
...	...	...	...	...	...
2015-12-24	32.57	32.71	32.44	32.48	2798163
2015-12-28	32.36	32.36	31.95	32.17	5420280
2015-12-29	32.44	32.70	32.32	32.55	6388244
2015-12-30	32.50	32.64	32.20	32.23	5057162
2015-12-31	31.91	32.30	31.77	31.81	8154307

2517 rows × 5 columns

** Create a list of the ticker symbols (as strings) in alphabetical order. Call this list: tickers**

```
tickers=['BAC','C','GS','JPM','MS','WFC']
```

** Use pd.concat to concatenate the bank dataframes together to a single data frame called bank_stocks. Set the keys argument equal to the tickers list. Also pay attention to what axis you concatenate on.**

```
bank_stocks=pd.concat([BAC,C,GS,JPM,MS,WFC],axis=1,keys=tickers)
bank_stocks.head(5)
```

	BAC					C					..
Stock Info	Open	High	Low	Close	Volume	Open	High	Low	Close	Volume	..
Date											
2006-01-03	46.92	47.18	46.15	47.08	16296700	490.0	493.8	481.1	492.9	1537660	.
2006-01-04	47.00	47.24	46.45	46.58	17757900	488.6	491.0	483.5	483.8	1871020	.
2006-01-05	46.58	46.83	46.32	46.64	14970900	484.4	487.8	484.0	486.2	1143160	.
2006-01-06	46.80	46.91	46.35	46.57	12599800	488.8	489.0	482.0	486.2	1370250	.
2006-01-09	46.72	46.97	46.36	46.60	15620000	486.0	487.4	483.0	483.9	1680740	.

5 rows × 30 columns

** Set the column name levels (this is filled out for you):**

```
bank_stocks.columns.names = ['Bank Ticker','Stock Info']
```

** Check the head of the bank_stocks dataframe.**

```
bank_stocks.head(5)
```

Bank Ticker	BAC						C
Stock Info	High	Low	Open	Close	Volume	Adj Close	High
Date							
2006-01-03	47.180000	46.150002	46.919998	47.080002	16296700.0	34.425114	493.799988
2006-01-04	47.240002	46.450001	47.000000	46.580002	17757900.0	34.059509	491.000000
2006-01-05	46.830002	46.320000	46.580002	46.639999	14970700.0	34.103382	487.799988
2006-01-06	46.910000	46.349998	46.799999	46.570000	12599800.0	34.052204	489.000000
2006-01-09	46.970001	46.360001	46.720001	46.599998	15619400.0	34.074108	487.399994

5 rows × 36 columns

EDA

Let's explore the data a bit! Before continuing, I encourage you to check out the documentation on Multi-Level Indexing and Using .xs. Reference the solutions if you can not figure out how to use .xs(), since that will be a major part of this project.

** What is the max Close price for each bank's stock throughout the time period?**

```
bank_stocks.xs('Close', level='Stock Info', axis = 1).max()
```

```
Bank Ticker
BAC     54.90
C      564.10
GS     247.92
JPM     70.08
MS      89.30
WFC     58.52
dtype: float64
```

** Create a new empty DataFrame called returns. This dataframe will contain the returns for each bank's stock. returns are typically defined by:**

$$r_t = \frac{p_t - p_{t-1}}{p_{t-1}} = \frac{p_t}{p_{t-1}} - 1$$

```
returns=pd.DataFrame()
```

** We can use pandas pct_change() method on the Close column to create a column representing this return value. Create a for loop that goes and for each Bank Stock Ticker creates this returns column and set's it as a column in the returns DataFrame.**

```
for tick in tickers:
    returns[tick+'Return']=bank_stocks[tick]['Close'].pct_change()
```

```
returns.head()
```

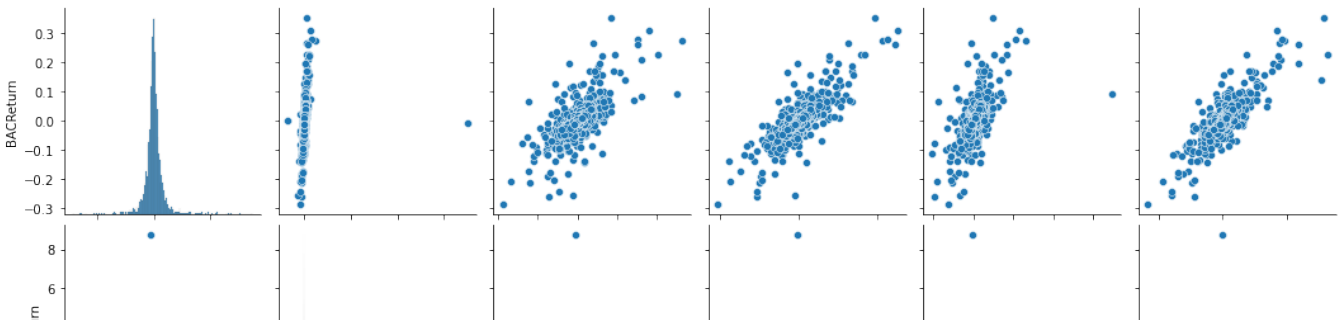
Date	BACReturn	CReturn	GSReturn	JPMReturn	MSReturn	WFCReturn
2006-01-03	NaN	NaN	NaN	NaN	NaN	NaN
2006-01-04	-0.010620	-0.018462	-0.013812	-0.014183	0.000686	-0.011599
2006-01-05	0.001288	0.004961	-0.000393	0.003029	0.002742	-0.000951
2006-01-06	-0.001501	0.000000	0.014169	0.007046	0.001025	0.005714
2006-01-09	0.000644	-0.004731	0.012030	0.016242	0.010586	0.000000

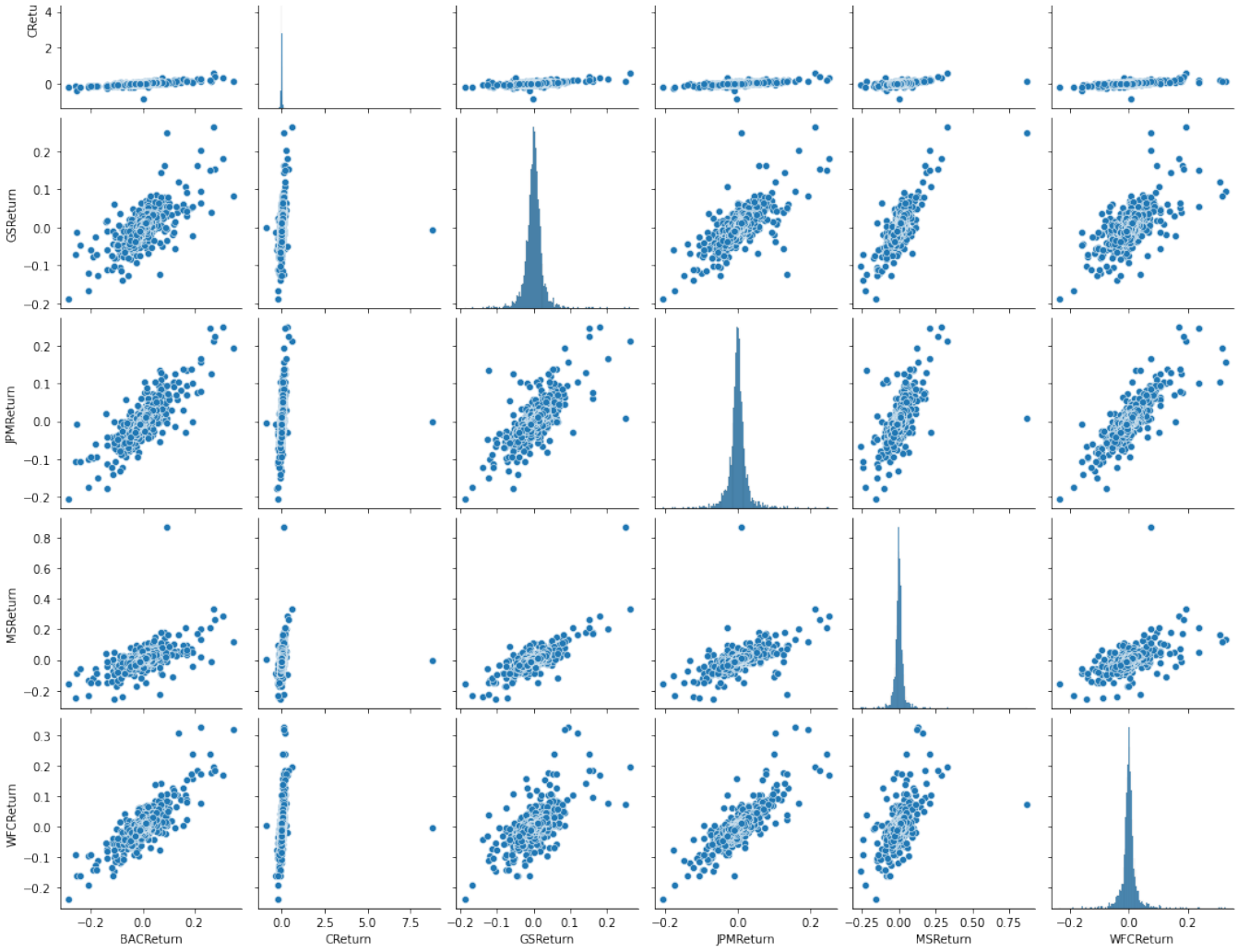
** Create a pairplot using seaborn of the returns dataframe. What stock stands out to you? Can you figure out why?**

```
import seaborn as sns
```

```
sns.pairplot(returns)
```

```
<seaborn.axisgrid.PairGrid at 0x7fd7389d8370>
```





- See solution for details about Citigroup behavior....

You'll also see the enormous crash in value if you take a look a the stock price plot (which we do later in the visualizations.)

** Using this returns DataFrame, figure out on what dates each bank stock had the best and worst single day returns. You should notice that 4 of the banks share the same day for the worst drop, did anything significant happen that day?**

```
returns.idxmin()# Worst Drop (4 of them on Inauguration day)
```

```
BACReturn   2009-01-20
CReturn     2011-05-06
GSReturn    2009-01-20
JPMReturn   2009-01-20
MSReturn    2008-10-09
WFCReturn   2009-01-20
dtype: datetime64[ns]
```

```
returns.idxmax()# Best Single Day Gain
# citigroup stock split in May 2011, but also JPM day after inauguration.
```

```
BACReturn   2009-04-09
CReturn     2011-05-09
GSReturn    2008-11-24
JPMReturn   2009-01-21
MSReturn    2008-10-13
WFCReturn   2008-07-16
dtype: datetime64[ns]
```

** Take a look at the standard deviation of the returns, which stock would you classify as the riskiest over the entire time period? Which would you classify as the riskiest for the year 2015?**

```
returns.std()# Citigroup riskiest
```

```
BACReturn    0.036650
CReturn      0.179969
GSReturn     0.025346
JPMReturn    0.027656
MSReturn     0.037820
WFCReturn    0.030233
dtype: float64
```

```
returns.reset_index()[returns.reset_index()['Date'].apply(lambda x: x.year == 2
```

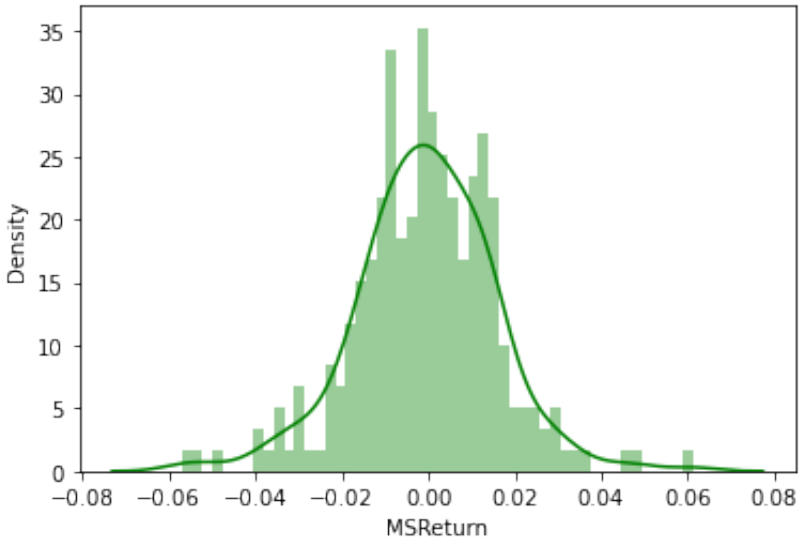
```
BACReturn    0.016163
CReturn      0.015289
GSReturn     0.014046
JPMReturn    0.014017
MSReturn     0.016249
WFCReturn    0.012591
dtype: float64
```

** Create a distplot using seaborn of the 2015 returns for Morgan Stanley **

```
sns.distplot(returns.loc['2015-01-01':'2015-12-31']['MSReturn'],color='green',b

#.ix is deprecated use .loc
```

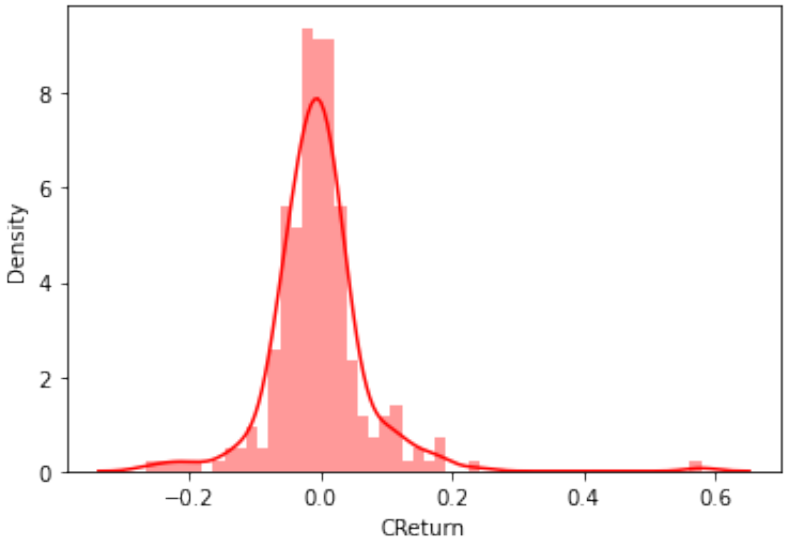
```
<AxesSubplot:xlabel='MSReturn', ylabel='Density'>
```



** Create a distplot using seaborn of the 2008 returns for CitiGroup **

```
sns.distplot(returns.loc['2008-01-01':'2008-12-31']['CReturn'],color='red',bins
```

```
<AxesSubplot:xlabel='CReturn', ylabel='Density'>
```



More Visualization

A lot of this project will focus on visualizations. Feel free to use any of your preferred visualization libraries to try to recreate the described plots below, seaborn, matplotlib, plotly and cufflinks, or just pandas.

Imports

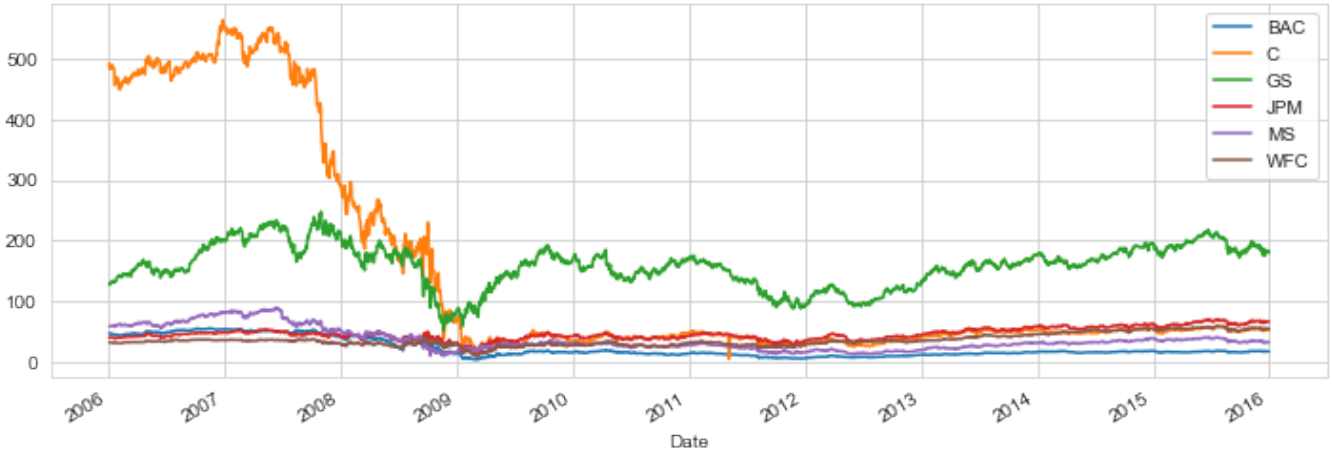
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
%matplotlib inline

# Optional Plotly Method Imports
import plotly
import cufflinks as cf
cf.go_offline()
```

** Create a line plot showing Close price for each bank for the entire index of time. (Hint: Try using a for loop, or use .xs to get a cross section of the data.)**

```
for tick in tickers:
    bank_stocks[tick]['Close'].plot(label=tick,figsize=(12,4))
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fd72651a730>
```




```
bank_stocks.xs('Close', level='Stock Info', axis = 1).iplot()#alternative way t
```

Moving Averages

Let's analyze the moving averages for these stocks in the year 2008.

** Plot the rolling 30 day average against the Close Price for Bank Of America's stock for the year 2008**

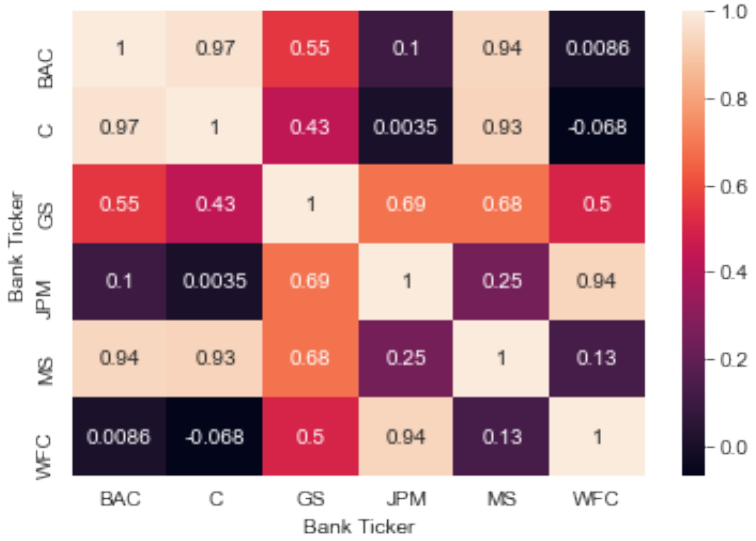
```
BAC['Close'].loc['2008-01-01':'2009-01-01'].rolling(window=30).mean().plot(labe
BAC['Close'].loc['2008-01-01':'2009-01-01'].plot(label='BAC Close')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fd726f866a0>
```

** Create a heatmap of the correlation between the stocks Close Price.**

```
sns.heatmap(bank_stocks.xs(key='Close',axis=1,level='Stock Info').corr(),annot=
```

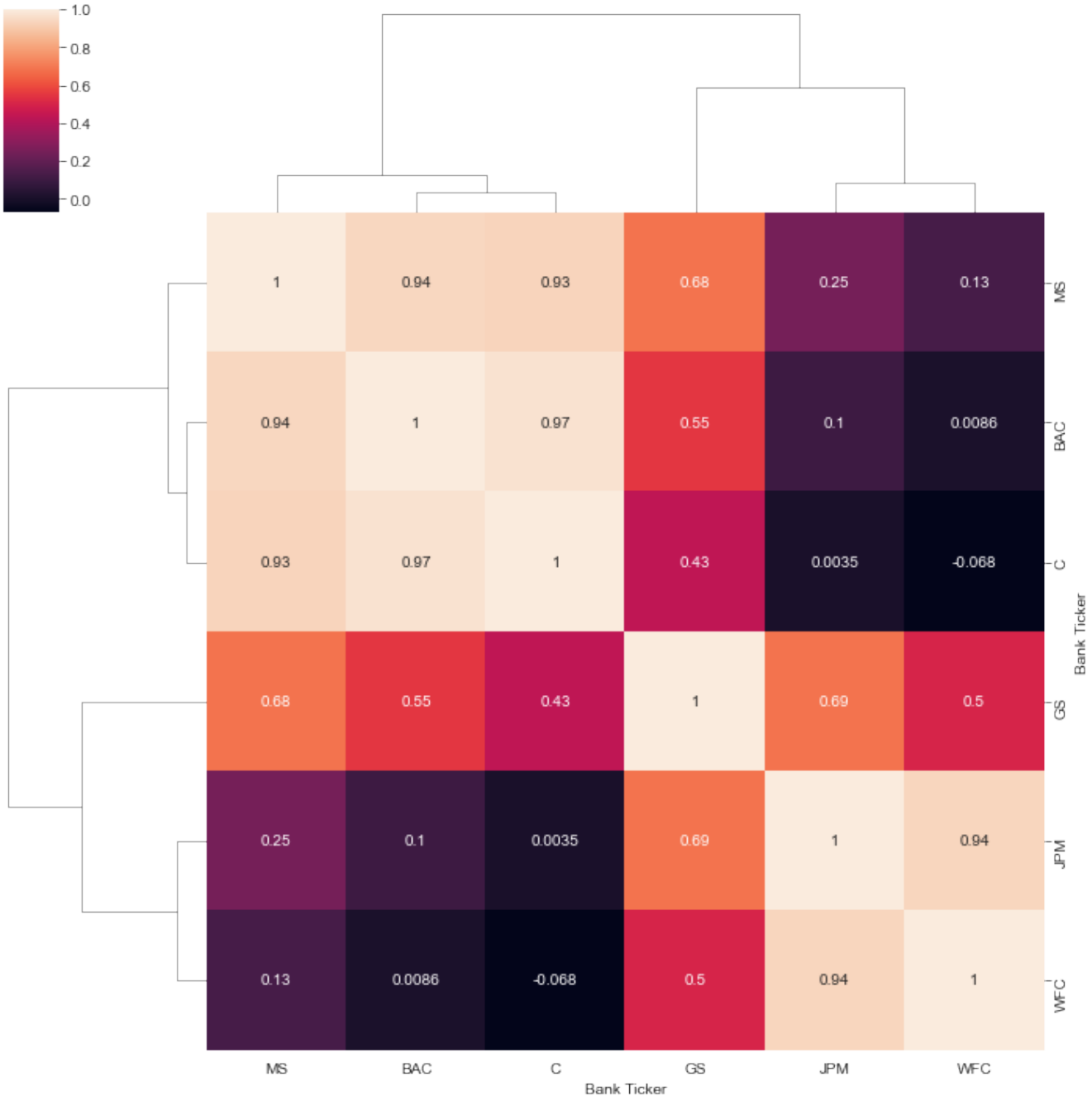
```
<AxesSubplot:xlabel='Bank Ticker', ylabel='Bank Ticker'>
```



** Optional: Use seaborn's clustermap to cluster the correlations together:**

```
sns.clustermap(bank_stocks.xs(key='Close',axis=1,level='Stock Info').corr(),anr
```

```
<seaborn.matrix.ClusterGrid at 0x7fd723cb1c10>
```



Part 2 (Optional)

In this second part of the project we will rely on the cufflinks library to create some Technical Analysis plots. This part of the project is experimental due to its heavy reliance on the cuffinks project, so feel free to skip it if any functionality is broken in the future.

** Use .iplot(kind='candle) to create a candle plot of Bank of America's stock from Jan 1st 2015 to Jan 1st 2016.**

```
bac=BAC[['Open','High','Low','Close']].loc['2015-01-01':'2016-01-01']
bac.iplot(kind='candle')
```

** Use .ta_plot(study='sma') to create a Simple Moving Averages plot of Morgan Stanley for the year 2015.**

```
MS['Close'].loc['2015-01-01':'2016-01-01'].ta_plot(study='sma',periods=[13,21,5
```

Use .ta_plot(study='boll') to create a Bollinger Band Plot for Bank of America for the year 2015.

```
BAC['Close'].loc['2015-01-01':'2016-01-01'].ta_plot(study='boll')
```

Great Job!

Definitely a lot of more specific finance topics here, so don't worry if you didn't understand them all! The only thing you should be concerned with understanding are the basic pandas and visualization oeprations.