In [1]:
```
from pandas_datareader import data, wb
import pandas as pd
import numpy as np
import datetime
%matplotlib inline
```

In [2]:
```
start = datetime.datetime(2007, 1,1)
end = datetime.datetime(2017,1,1)
BAC = data.DataReader("BAC", 'yahoo', start, end)
```

In [3]:
```
BAC
```

Out[3]:

Date	High	Low	Open	Close	Volume	Adj Close
2007-01-03	54.180000	52.990002	53.400002	53.330002	16028200.0	39.919727
2007-01-04	53.889999	53.049999	53.330002	53.669998	13175000.0	40.174225
2007-01-05	53.590000	53.029999	53.590000	53.240002	10205000.0	39.852348
2007-01-08	53.639999	52.799999	53.459999	53.450001	9685900.0	40.009548
2007-01-09	53.709999	52.970001	53.599998	53.500000	12546500.0	40.046978
...	...	...	...	...	...	...
2016-12-23	22.650000	22.430000	22.510000	22.600000	38187800.0	20.310703
2016-12-27	22.740000	22.540001	22.709999	22.610001	39988600.0	20.319695
2016-12-28	22.670000	22.260000	22.620001	22.330000	52652900.0	20.068056
2016-12-29	22.389999	21.770000	22.330000	22.000000	79188400.0	19.771486
2016-12-30	22.260000	21.950001	22.020000	22.100000	72605900.0	19.861357

2518 rows × 6 columns

In [4]:
```
#Goldman Sachs
start = datetime.datetime(2007, 1,1)
end = datetime.datetime(2017,1,1)
GS = data.DataReader("GS", 'yahoo', start, end)

#MorganStanley
start = datetime.datetime(2007, 1,1)
end = datetime.datetime(2017,1,1)
MS = data.DataReader("MS", 'yahoo', start, end)

#JPmorgan
start = datetime.datetime(2007, 1,1)
end = datetime.datetime(2017,1,1)
JPM = data.DataReader("JPM", 'yahoo', start, end)

#citigroup
start = datetime.datetime(2007, 1,1)
end = datetime.datetime(2017,1,1)
C = data.DataReader("C", 'yahoo', start, end)

#wellsfargo
start = datetime.datetime(2007, 1,1)
end = datetime.datetime(2017,1,1)
WFC = data.DataReader("WFC", 'yahoo', start, end)
```

In [5]:
```
tickers=['BAC','C','JPM','MS','GS','WFC']
```

In [7]:
```
#concatenating all the dataframes together , took axis=1 as i want to concat along the column
bank_allstocks=pd.concat([BAC,C,GS,JPM,MS,WFC],axis=1,keys=tickers)
bank_allstocks.head()
```

Out[7]:

	BAC						C				...	GS	
	High	Low	Open	Close	Volume	Adj Close	High	Low	Open	Close	...	Open	Close
Date													
2007-01-03	54.180000	52.990002	53.400002	53.330002	16028200.0	39.919727	562.799988	547.200012	556.599976	552.500000	...	81.930000	81.6200
2007-01-04	53.889999	53.049999	53.330002	53.669998	13175000.0	40.174225	561.500000	547.200012	552.500000	550.599976	...	81.269997	81.9100
2007-01-05	53.590000	53.029999	53.590000	53.240002	10205000.0	39.852348	550.500000	544.599976	550.000000	547.700012	...	81.349998	80.8600
2007-01-08	53.639999	52.799999	53.459999	53.450001	9685900.0	40.009548	551.500000	543.000000	546.000000	550.500000	...	80.610001	81.3499
2007-01-09	53.709999	52.970001	53.599998	53.500000	12546500.0	40.046978	551.500000	541.900024	550.099976	545.700012	...	81.199997	81.1600

5 rows × 36 columns

In [8]:
```
bank_allstocks.columns.names = ['Bank Ticker','Stock Info']
bank_allstocks.head()
```

Out[8]:

Bank Ticker	BAC						C				...	GS	
Stock Info	High	Low	Open	Close	Volume	Adj Close	High	Low	Open	Close	...	Open	Close
Date													
2007-01-03	54.180000	52.990002	53.400002	53.330002	16028200.0	39.919727	562.799988	547.200012	556.599976	552.500000	...	81.930000	81.620(
2007-01-04	53.889999	53.049999	53.330002	53.669998	13175000.0	40.174225	561.500000	547.200012	552.500000	550.599976	...	81.269997	81.910(
2007-01-05	53.590000	53.029999	53.590000	53.240002	10205000.0	39.852348	550.500000	544.599976	550.000000	547.700012	...	81.349998	80.860(
2007-01-08	53.639999	52.799999	53.459999	53.450001	9685900.0	40.009548	551.500000	543.000000	546.000000	550.500000	...	80.610001	81.349(
2007-01-09	53.709999	52.970001	53.599998	53.500000	12546500.0	40.046978	551.500000	541.900024	550.099976	545.700012	...	81.199997	81.160(

5 rows × 36 columns

In [36]:
```
#Moving on to some exploratory data analysis(using xs for selecting within multilevel indexing)
#Now finding : max Close price for each bank's stock throughout the time period
for i in tickers:
    print(i,bank_allstocks[i]['Close'].max())
```

```
BAC 47244.07001757622
C 254826.1899471283
JPM 401285.9198875427
MS 121255.25004196167
GS 81449.30007362366
WFC 93385.80001449585
BAC 54.04999923706055
C 552.5
JPM 247.9199981689453
MS 87.12999725341797
GS 89.30000305175781
WFC 58.52000045776367
```

In [10]:
```
#The same prev thing can be done with in a bit more panda savvy way
bank_allstocks.xs(key='Close',axis=1,level='Stock Info').max()
```

Out[10]:
```
Bank Ticker
BAC     54.049999
C      552.500000
JPM    247.919998
MS      87.129997
GS      89.300003
WFC     58.520000
dtype: float64
```

In [11]:
```
#a new empty DataFrame called returns. This dataframe will contain the returns for each bank's stock. returns are typi
cally defined by:**rt=pt-pt-1/pt-1=pt/pt-1-1
returns=pd.DataFrame()
for i in tickers:
    returns[i+' Return']=bank_allstocks[i]['Close'].pct_change()

#pct_change from pandas gave us the percent change
```

In [12]:
```
returns.head()
```

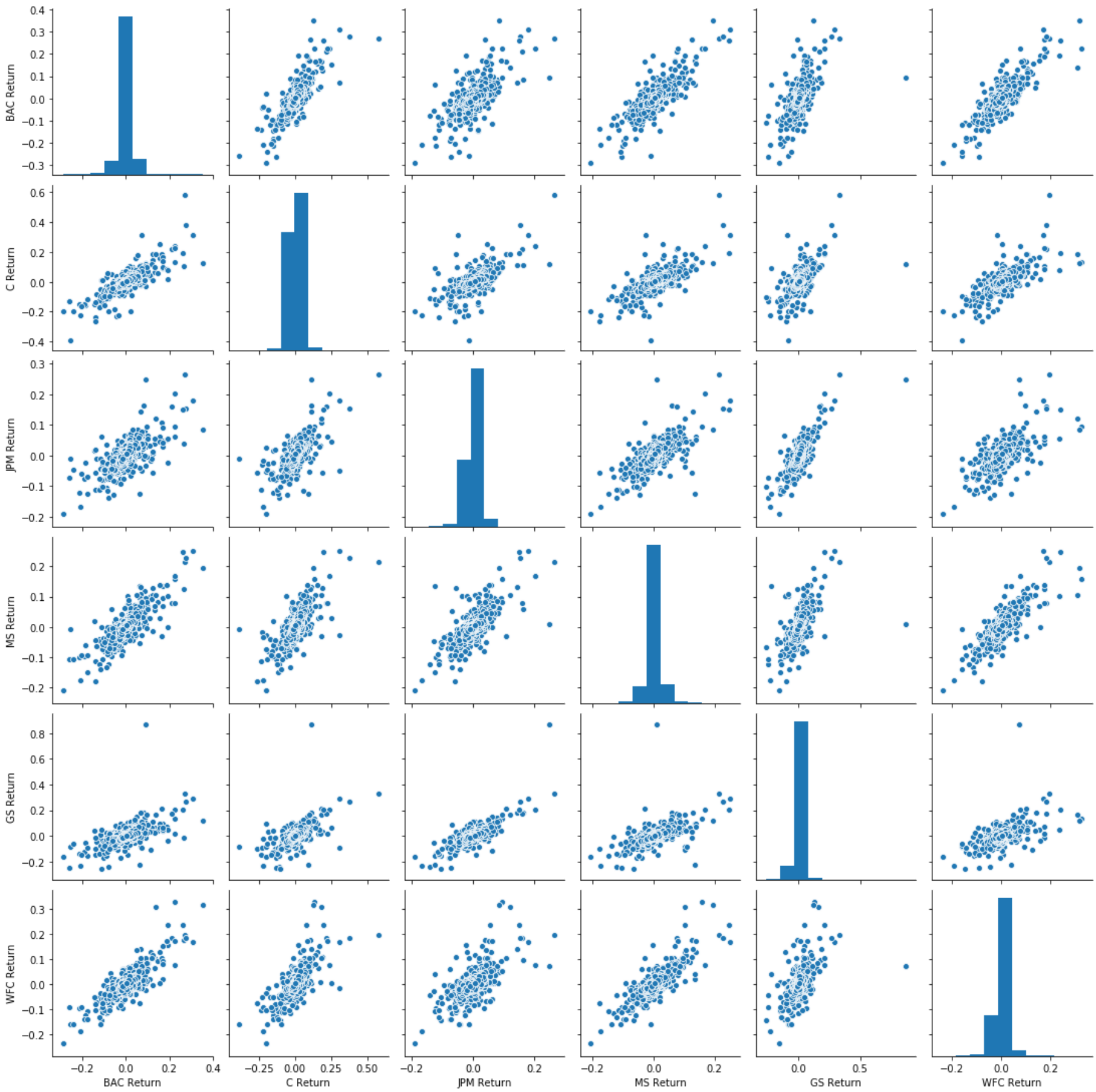
Out[12]:

Date	BAC Return	C Return	JPM Return	MS Return	GS Return	WFC Return
2007-01-03	NaN	NaN	NaN	NaN	NaN	NaN
2007-01-04	0.006375	-0.003439	-0.009316	0.002496	0.003553	0.001679
2007-01-05	-0.008012	-0.005267	0.001006	-0.008300	-0.012819	-0.005587
2007-01-08	0.003944	0.005112	0.023512	0.003348	0.006060	-0.002809
2007-01-09	0.000935	-0.008719	0.001718	-0.004171	-0.002336	0.002535

In [13]:
```
#plotting a pair plot for returns (plotted it from 1 as the first row is NULL,it will throw an error)
import matplotlib.pyplot as plt
import seaborn as sns
```

In [14]:
```
sns.pairplot(returns[1:])
```

Out[14]:
```
<seaborn.axisgrid.PairGrid at 0x20ea3223fc8>
```



In [15]:
```
returns.idxmin()
```

Out[15]:
```
BAC Return   2009-01-20
C Return     2009-02-27
JPM Return   2009-01-20
MS Return    2009-01-20
GS Return    2008-10-09
WFC Return   2009-01-20
dtype: datetime64[ns]
```

In [36]:
```
returns.idxmax()
```

Out[36]:
```
BAC Return   2009-04-09
C Return     2008-11-24
JPM Return   2008-11-24
MS Return    2009-01-21
GS Return    2008-10-13
WFC Return   2008-07-16
dtype: datetime64[ns]
```

In [37]:
```
#looking at the minimum returns date, we see a common date for multiple banks. Reason : The famous Wall street crash
```

In [38]:
```
#which stock would you classify as the riskiest over the entire time period? Which would you classify as the riskiest
 for the year 2015
#now riskiness of a stock can be looked upon by using standard deviation of the return of the stock(more std of return
s means more risky)

returns.std()
```

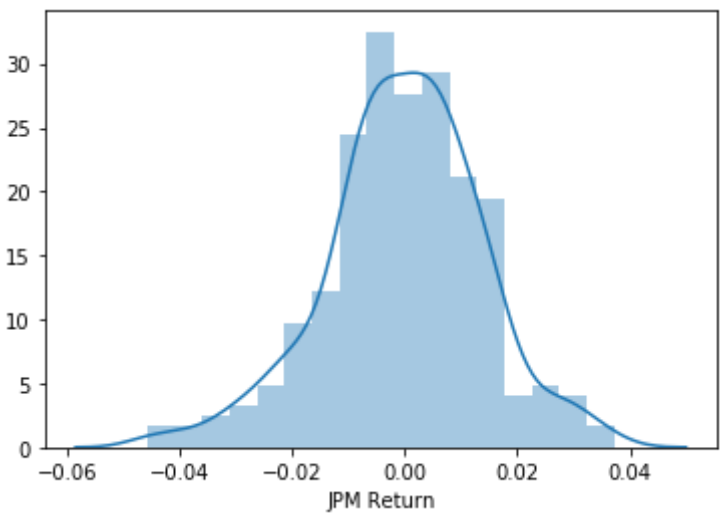
Out[38]:
```
BAC Return    0.037109
C Return      0.039075
JPM Return    0.025513
MS Return     0.027895
GS Return     0.038156
WFC Return    0.030497
dtype: float64
```

In [46]:
```
#std for a particular time period(used loc as ix is deprecated)
returns.loc['2015-1-1':'2015-12-31'].std()
```

Out[46]:
```
BAC Return    0.016163
C Return      0.015289
JPM Return    0.014046
MS Return     0.014017
GS Return     0.016249
WFC Return    0.012591
dtype: float64
```

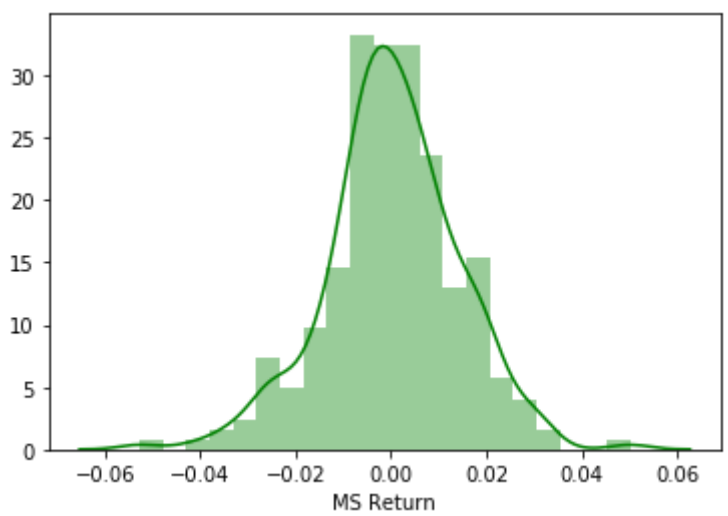
In [16]:
```
#distplot for some of the returns in particular years(say 2015)
sns.distplot(returns.loc['2015-1-1':'2015-12-31']['JPM Return'])
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x20ea49e2508>
```



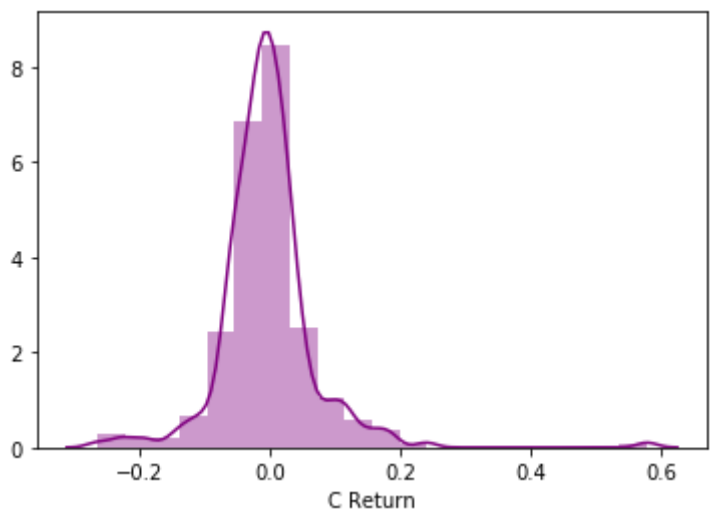
In [17]:
```
sns.distplot(returns.loc['2015-1-1':'2015-12-31']['MS Return'],color='green')
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x20ea62cd148>



In [24]:
```
#distplot using seaborn of the 2008 returns for CitiGroup
sns.distplot(returns.loc['2008-1-1':'2008-12-31']['C Return'],color='purple',bins=20)
```

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x20ea69d98c8>



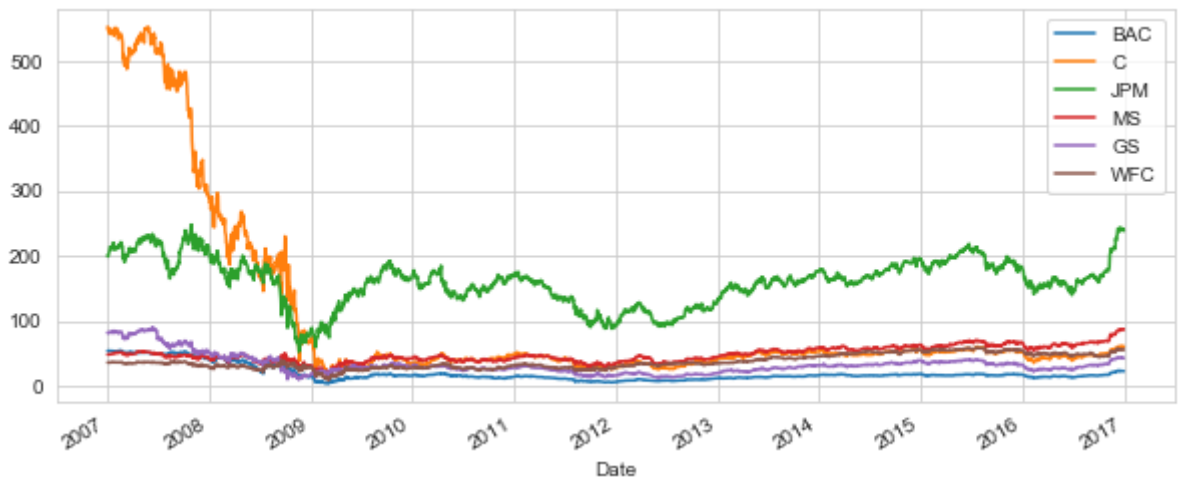
In [25]:
```
#This previous graph shows the std of Citigroup has bigger extremes than other banks( see 0.6 of citigroup vs 0.06 of
 Morgan stanley)
```

In [26]:
```
#Moving to a bit more Visualizations
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()
```

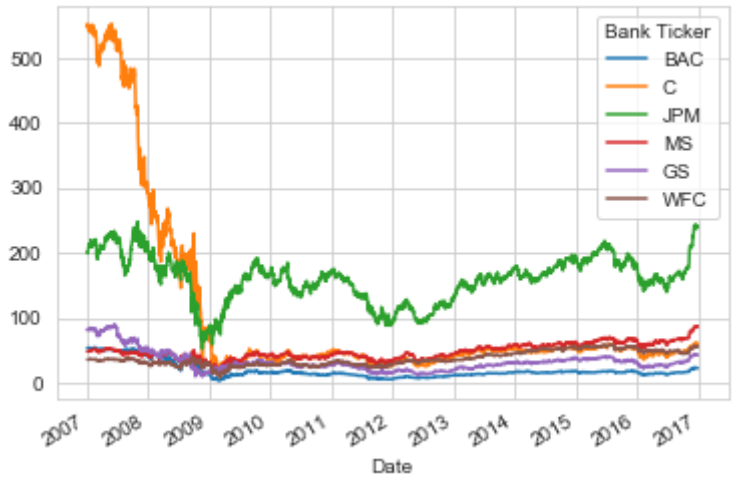
In [28]:
```
# A line plot showing Close price for each bank for the entire index of time
for i in tickers:
    bank_allstocks[i]['Close'].plot(label=i,figsize=(10,4))
plt.legend()
#we can see the dip around 2009(the great recession)
```

Out[28]: <matplotlib.legend.Legend at 0x20ea7614688>

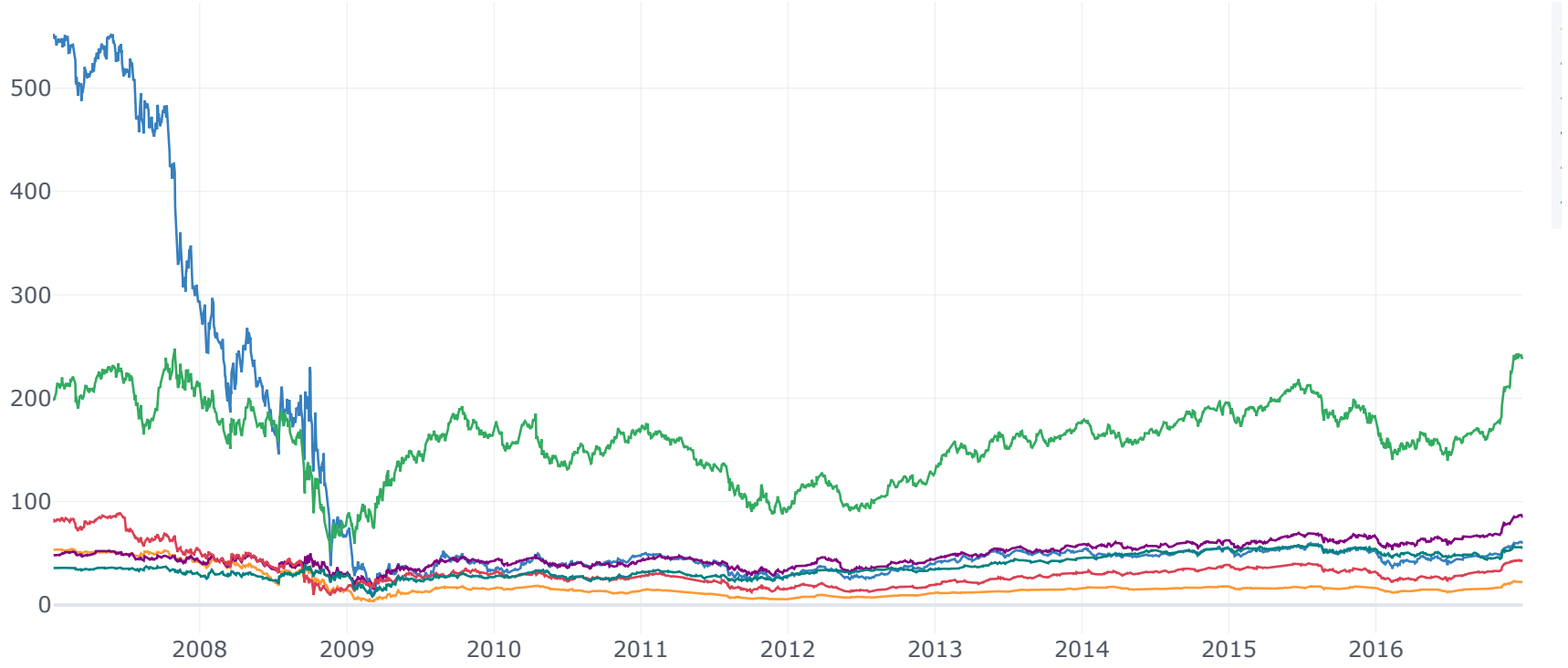


In [29]:
```
#prev plot with xs
bank_allstocks.xs(key='Close',axis=1,level='Stock Info').plot()
```

Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x20ea86c1548>

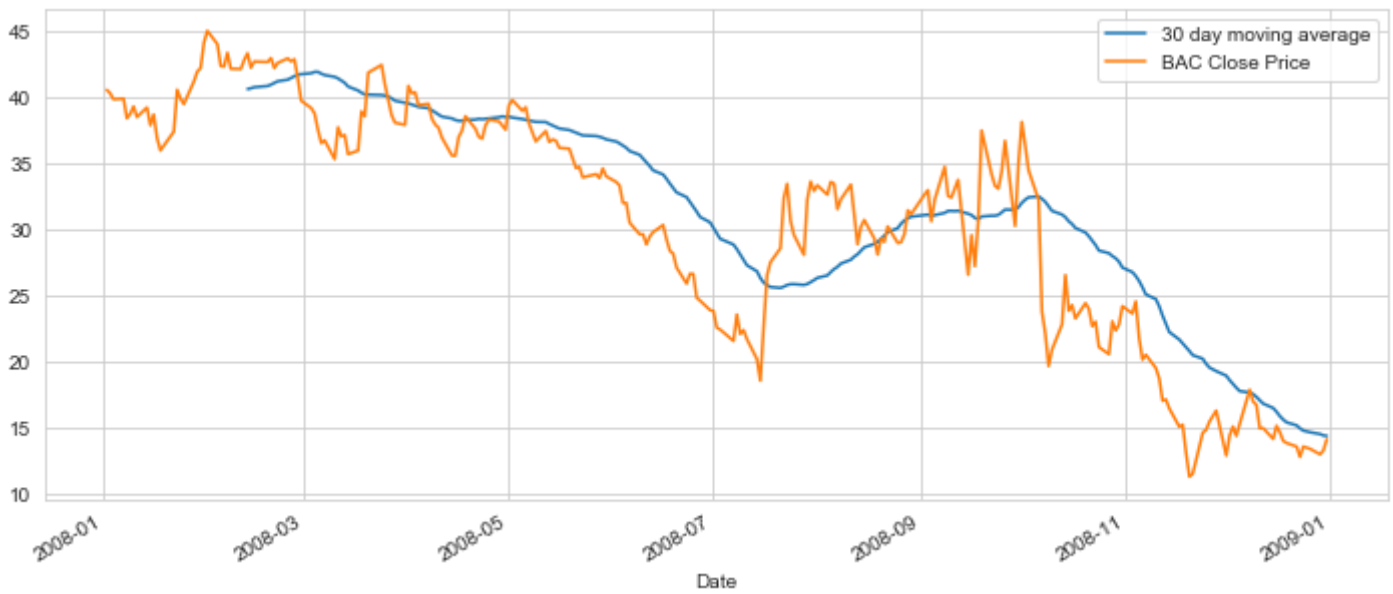


In [30]:
```
#plotly way to do the same thing
bank_allstocks.xs(key='Close',axis=1,level='Stock Info').iplot()
```



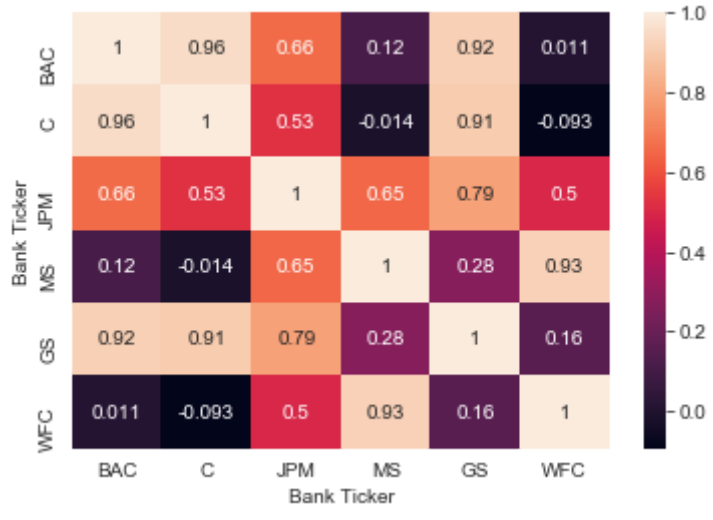
In [39]:
```
#The rolling 30 day average against the Close Price for Bank Of America's stock for the year 2008
plt.figure(figsize=(12,5))
BAC['Close'].loc['2008-1-1':'2009-1-1'].rolling(window=30).mean().plot(label='30 day moving average')
BAC['Close'].loc['2008-1-1':'2009-1-1'].plot(label='BAC Close Price')
plt.legend()
```

Out[39]: <matplotlib.legend.Legend at 0x20eaa04c2c8>



In [43]:
```
#Heatmap of the correlation between the stocks Close Price
sns.heatmap(bank_allstocks.xs(key='Close',axis=1,level='Stock Info').corr(),annot=True)
```

Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x20ea6446388>



In [ ]: