Projeto de dados do mercado financeiro

Neste projeto de dados, nos concentraremos na análise de dados exploratórios dos preços das ações.

Vamos nos concentrar nas ações dos banco e ver como eles performaram durante a crise financeira até o início de 2016.

Obter dados

Nesta seção, usaremos pandas para ler diretamente os dados das finanças do Google. É necessário instalar o pandas-datareader (pip install pandas-datareader). O datareader permite que você leia informações de estoque diretamente da internet.

Os Imports

In [3]:
```
from pandas_datareader import data, wb
import pandas as pd
import numpy as np
import datetime
%matplotlib inline
```

Dados

Precisamos obter dados usando o datareader de pandas. Obteremos informações sobre ações para os seguintes bancos:

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

Queremos obter os dados de ações de 1 de janeiro de 2006 a 1º de janeiro de 2016 para cada um desses bancos.

Em seguida:

1. Usar datetime para definir objetos de início e fim de data e hora.
2. Descobrir o símbolo do ticker para cada banco.
3. Descubrir como usar o datareader para pegar as cotações.

In [4]:
```
start = datetime.datetime(2006, 1, 1)
end = datetime.datetime(2016, 1, 1)
```

In [5]:
```
# Bank of America
BAC = data.DataReader("BAC", 'yahoo', start, end)

# CitiGroup
C = data.DataReader("C", 'yahoo', start, end)

# Goldman Sachs
GS = data.DataReader("GS", 'yahoo', start, end)

# JPMorgan Chase
JPM = data.DataReader("JPM", 'yahoo', start, end)

# Morgan Stanley
MS = data.DataReader("MS", 'yahoo', start, end)

# Wells Fargo
WFC = data.DataReader("WFC", 'yahoo', start, end)
```

In [6]:
```
tickers = ['BAC', 'C', 'GS', 'JPM', 'MS', 'WFC']
```

In [7]:
```
bank_stocks = pd.concat([BAC, C, GS, JPM, MS, WFC],axis=1,keys=tickers)
```

In [8]:
```
bank_stocks.head(3)
```

Out[8]:

	BAC						C				...	MS
	High	Low	Open	Close	Volume	Adj Close	High	Low	Open	Close	...	Open
Date												
2006-01-03	47.180000	46.150002	46.919998	47.080002	16296700.0	34.596096	493.799988	481.100006	490.000000	492.899994	...	57.169998
2006-01-04	47.240002	46.450001	47.000000	46.580002	17757900.0	34.228691	491.000000	483.500000	488.600006	483.799988	...	58.700001
2006-01-05	46.830002	46.320000	46.580002	46.639999	14970700.0	34.272778	487.799988	484.000000	484.399994	486.200012	...	58.549999

3 rows × 36 columns

In [9]:
```
bank_stocks.columns.names = ['Bank Ticker','Stock Info']
```

In [11]:
```
bank_stocks.head(3)
```

Out[11]:

Bank Ticker	BAC						C				...	MS
Stock Info	High	Low	Open	Close	Volume	Adj Close	High	Low	Open	Close	...	Open
Date												
2006-01-03	47.180000	46.150002	46.919998	47.080002	16296700.0	34.596096	493.799988	481.100006	490.000000	492.899994	...	57.169998
2006-01-04	47.240002	46.450001	47.000000	46.580002	17757900.0	34.228691	491.000000	483.500000	488.600006	483.799988	...	58.700001
2006-01-05	46.830002	46.320000	46.580002	46.639999	14970700.0	34.272778	487.799988	484.000000	484.399994	486.200012	...	58.549999

3 rows × 36 columns

Análise exploratória de dados

Qual é o preço máximo de fechamento para o estoque de cada banco durante todo o período?

In [12]:
```
bank_stocks.xs(key='Close',axis=1,level='Stock Info').max()
```

Out[12]:
```
Bank Ticker
BAC     54.900002
C      564.099976
GS     247.919998
JPM     70.080002
MS      89.300003
WFC     58.520000
dtype: float64
```

Calculo de retorno Criarei o DataFrame "returns" que conterá os retornos para o ação de cada banco. Para maiores informações sobre taxas de retorno: https://bit.ly/3bCdEG6. Aqui, calcularemos os retornos definidos por:

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

In [14]:
```
returns = pd.DataFrame()
```

Usarei o método pct_change() que calcula o percentual de mudança entre uma linha e a imediatamente anterior. Usarei como base acoluna close para criar uma coluna que represente esse valor de retorno para cada banco

In [15]:
```
for tick in tickers:
    returns[tick+' Return'] = bank_stocks[tick]['Close'].pct_change()
returns.head()
```

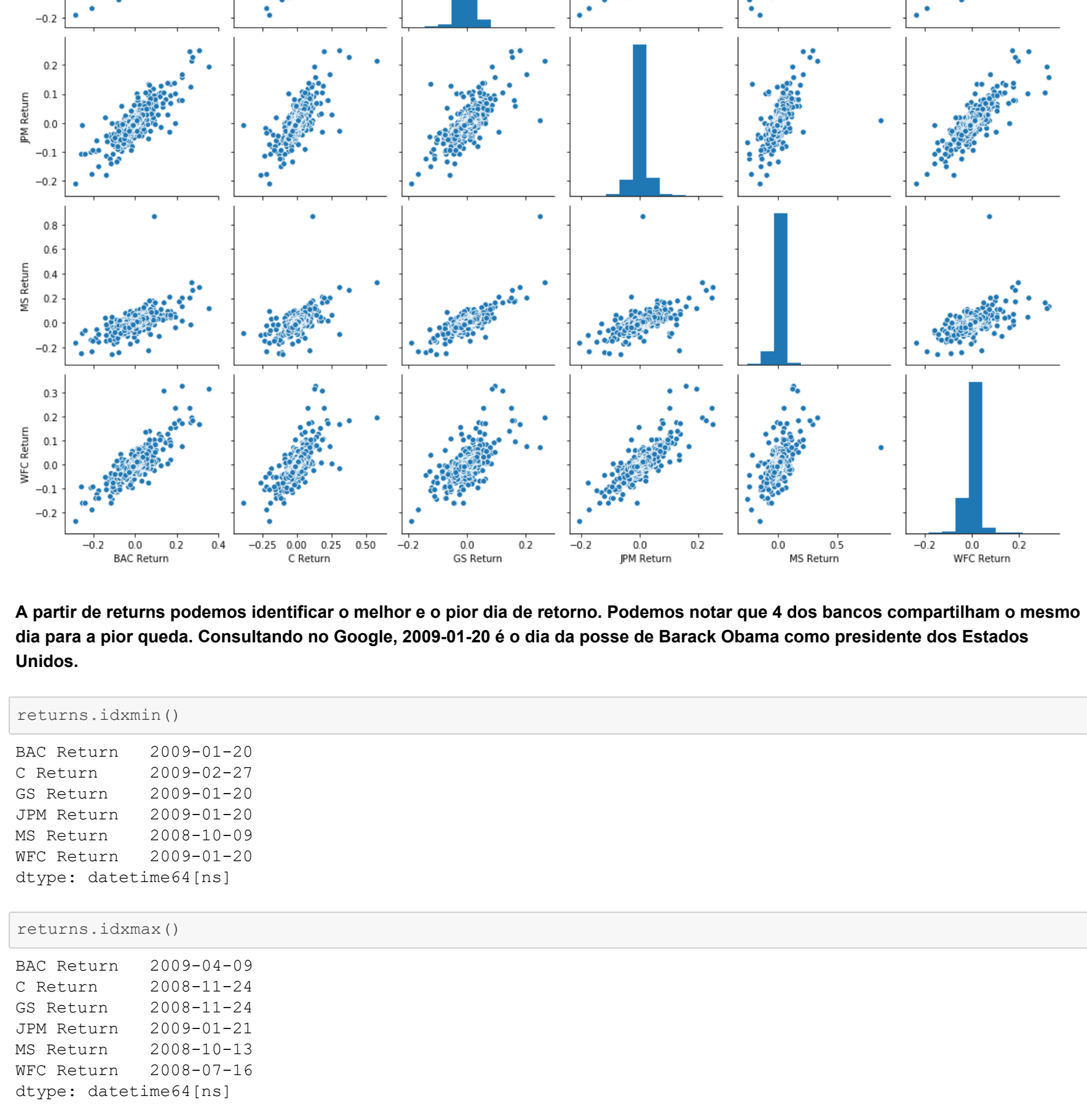
Out[15]:

	BAC Return	C Return	GS Return	JPM Return	MS Return	WFC Return
Date						
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	-0.001110
2006-01-06	-0.001501	0.000000	0.014169	0.007046	0.001025	0.005874
2006-01-09	0.000644	-0.004731	0.012030	0.016242	0.010586	-0.000158

Em cima deste dataframe criarei um pairplot com seaborn

In [16]:
```
#returns[1:]
import seaborn as sns
sns.pairplot(returns[1:])
```

Out[16]: <seaborn.axisgrid.PairGrid at 0x196cc32fc08>



A partir de returns podemos identificar o melhor e o pior dia de retorno. Podemos notar que 4 dos bancos compartilham o mesmo dia para a pior queda. Consultando no Google, 2009-01-20 é o dia da posse de Barack Obama como presidente dos Estados Unidos.

In [13]:
```
returns.idxmin()
```

Out[13]:
```
BAC Return   2009-01-20
C Return     2009-02-27
GS Return    2009-01-20
JPM Return   2009-01-20
MS Return    2008-10-09
WFC Return   2009-01-20
dtype: datetime64[ns]
```

In [14]:
```
returns.idxmax()
```

Out[14]:
```
BAC Return   2009-04-09
C Return     2008-11-24
GS Return    2008-11-24
JPM Return   2009-01-21
MS Return    2008-10-13
WFC Return   2008-07-16
dtype: datetime64[ns]
```

Verificando o desvio padrão para cada um dos retornos, podemos em seguida classificar as ações mais arriscadas durante todo o período de tempo. Em seguida verificarei somente o ano de 2015

In [15]:
```
returns.std()  # Citigroup é a mais arriscada
```

Out[15]:
```
BAC Return    0.036647
C Return      0.038672
GS Return     0.025390
JPM Return    0.027667
MS Return     0.037819
WFC Return    0.030238
dtype: float64
```

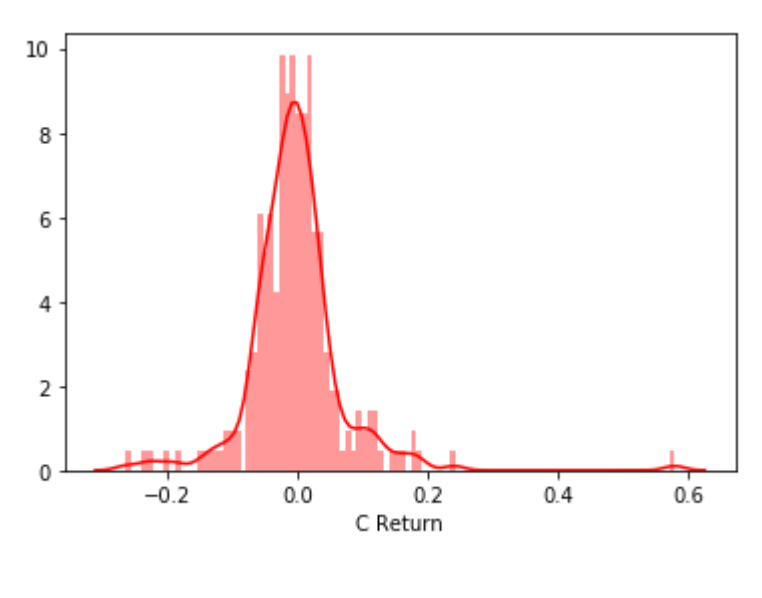
In [22]:
```
returns.loc['2015-01-01':'2015-12-31'].std()  # Muito similares, mas podemos escolher Morgan Stanley ou
 BofA
```

Out[22]:
```
BAC Return    0.016163
C Return      0.015289
GS Return     0.014046
JPM Return    0.014017
MS Return     0.016249
WFC Return    0.012591
dtype: float64
```

Criando um distplot usando seaborn dos retornos de 2015 para Morgan Stanley

In [27]:
```
sns.distplot(returns.loc['2015-01-01':'2015-12-31']['MS Return'],color='green',bins=100)
```

Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x196d1b51cc8>



Criando um distplot dos retornos de 2008 para CitiGroup

In [28]:
```
sns.distplot(returns.loc['2008-01-01':'2008-12-31']['C Return'],color='red',bins=100)
```

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x196d1d26488>

Mais visualização

Importações

In [29]:
```
import matplotlib.pyplot as plt
sns.set_style('whitegrid')
%matplotlib inline
```

Criando um gráfico de linha mostrando o preço de fechamento para cada banco para todo o índice de tempo.

In [31]:
```
for tick in tickers:
    bank_stocks[tick]['Close'].plot(figsize=(12,4),label=tick)
plt.legend()

#ou :  bank_stocks.xs(key='Close',axis=1,level='Stock Info').plot(figsize=(12,4),label=tick)
```

Out[31]: <matplotlib.legend.Legend at 0x196d1fe04c8>

Médias móveis

Vamos analisar as médias móveis para essas ações no ano de 2008. Sobre médias móveis: https://bit.ly/2LNHk8h

Traçando a média de 30 dias para o preço próximo do Bank Of America para o ano de 2008

O parâmentro "window" é o tamanho da janela de deslocamento. Esse é o número de observações usado para calcular a estatística desejada. Sendo no exemplo a média

Na célula abaixo podemos ver como dos dados se comportam

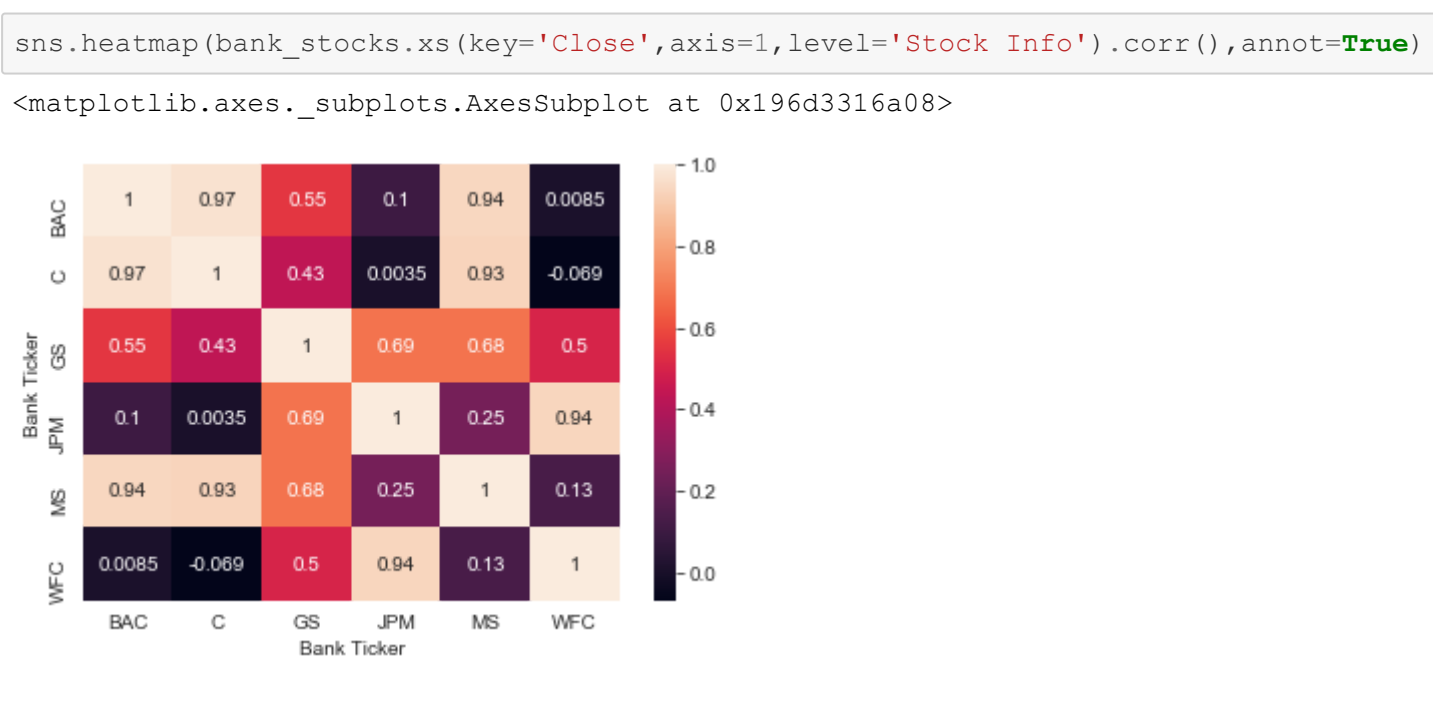
In [45]:
```
pd.set_option('display.max_rows', 35)
BAC['Close'].loc['2008-01-01':'2009-01-01'].rolling(window=30).mean()
```

Out[45]:
```
Date
2008-01-02          NaN
2008-01-03          NaN
2008-01-04          NaN
2008-01-07          NaN
2008-01-08          NaN
                 ...
2008-12-24    14.738667
2008-12-26    14.617333
2008-12-29    14.478667
2008-12-30    14.372667
2008-12-31    14.341000
Name: Close, Length: 253, dtype: float64
```

Abaixo podemos ver um gráfico com os dados de fechamento e as médias móveis para Banc of América entre 2008 e 2009

In [42]:
```
plt.figure(figsize=(12,6))
BAC['Close'].loc['2008-01-01':'2009-01-01'].rolling(window=30).mean().plot(label='30 Day Avg')
BAC['Close'].loc['2008-01-01':'2009-01-01'].plot(label='BAC CLOSE')
plt.legend()
```

Out[42]: <matplotlib.legend.Legend at 0x196d345e488>



Criando um mapa de calor da correlação entre os preços de fechamento das ações.

In [46]:
```
sns.heatmap(bank_stocks.xs(key='Close',axis=1,level='Stock Info').corr(),annot=True)
```

Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x196d3316a08>

Usando o clustermap do seaborn para agrupar as correlações:

```
In [67]: sns.clustermap(bank_stocks.xs(key="Close",axis=1,level="Stock Info").corr(),annot=True)
```

```
Out[67]: <seaborn.matrix.ClusterGrid at 0x19d2d1d7f80>
```