

# Clusterização das ações do IBRX

March 1, 2021

Nesse exercício vamos clusterizar as ações do índice Brasil 100 (IBrX) com o intuito de encontrar pequenos grupos de ações para identificar pares a serem usados em uma estratégia de negociação de pares. A clusterização pode ser usado para dividir ações e outros tipos de ativos em grupos com características semelhantes para vários outros tipos de estratégias de negociação. Também pode ser eficaz na construção de portfólio, ajudando a garantir que escolhamos um pool de ativos com diversificação suficiente entre eles.

Esse trabalho é baseado no livro (Machine Learning & Data Science Blueprints for Finance - Tatsat, Hariom) e nos ajudará para traçar a estratégia de futuros exercícios.

```
[1]: # Load libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pandas import read_csv, set_option
from pandas.plotting import scatter_matrix
import seaborn as sns
from sklearn.preprocessing import StandardScaler
import datetime
import pandas_datareader as dr
import yfinance as yf

# Disable the warnings
import warnings
warnings.filterwarnings('ignore')

# Import Model Packages
from sklearn.cluster import KMeans, \
    ↪ AgglomerativeClustering, AffinityPropagation, DBSCAN
from scipy.cluster.hierarchy import fcluster
from scipy.cluster.hierarchy import dendrogram, linkage, cophenet
from scipy.spatial.distance import pdist
from sklearn.metrics import adjusted_mutual_info_score
from sklearn import cluster, covariance, manifold

# Other Helper Packages and functions
import matplotlib.ticker as ticker
```

```
from itertools import cycle
```

# 1 Acessando e trabalhando os dados

## 1.1 Baixando as ações listadas no IBRX

```
[2]: ibovurl = "http://bvmf.bmfbovespa.com.br/indices/ResumoCarteiraTeorica.aspx?
↳Indice=ibrx&idioma=pt-br"
ibrx = pd.read_html(ibovurl)

# Selecionando apenas a coluna dos tickers

tickers = ibrx[0][0:]['Código'].tolist()
tickers.remove('Quantidade Teórica Total Redutor')

# Adicionando .SA em todas as ações

for i in range(0, len(tickers)):
    tickers[i] = tickers[i] + '.SA'

# Buscando no yahoo finance o preço de fechamento ajustado

dataset = yf.download(tickers = tickers, start = '2020-01-01')['Adj Close']

dataset = pd.DataFrame(dataset)
dataset
```

[\*\*\*\*\*100%\*\*\*\*\*] 100 of 100 completed

```
[2]:
```

	ABEV3.SA	ALPA4.SA	ALSO3.SA	AMAR3.SA	AZUL4.SA	B3SA3.SA	\
Date							
2020-01-02	18.616690	32.529774	50.009998	13.46	58.799999	43.622253	
2020-01-03	18.354891	32.529774	51.419998	13.95	56.759998	42.365135	
2020-01-06	18.442158	32.020557	51.070000	13.63	55.000000	41.958981	
2020-01-07	18.480942	32.749432	50.470001	13.67	56.820000	43.312805	
2020-01-08	18.393677	33.108875	52.000000	13.87	56.919998	43.196762	
...	...	...	...	...	...	...	
2021-02-23	14.690000	35.000000	25.110001	5.50	43.099998	55.340000	
2021-02-24	14.680000	36.070000	25.070000	5.52	44.950001	55.619999	
2021-02-25	14.200000	34.830002	24.930000	5.32	42.419998	54.889999	
2021-02-26	14.110000	34.869999	23.530001	5.02	40.730000	55.310001	
2021-03-01	14.090000	35.889999	23.240000	5.18	40.970001	55.330002	
	BBAS3.SA	BBDC3.SA	BBDC4.SA	BBSE3.SA	...	TIMS3.SA	\
Date					...		

2020-01-02	50.934219	31.287395	33.119057	33.248966	...	NaN
2020-01-03	50.849010	30.799173	33.135822	32.882446	...	NaN
2020-01-06	50.176830	30.622517	32.543949	33.423504	...	NaN
2020-01-07	49.798138	30.101395	31.978584	33.440960	...	NaN
2020-01-08	49.343704	29.642103	31.483898	33.423504	...	NaN
...	...	...	...	...	...	...
2021-02-23	30.430000	21.469999	24.180000	26.830000	...	13.22
2021-02-24	30.309999	21.330000	24.010000	26.540001	...	13.30
2021-02-25	29.459999	20.780001	23.490000	25.950001	...	13.13
2021-02-26	28.120001	20.530001	23.080000	25.549999	...	12.86
2021-03-01	28.490000	20.490000	22.990000	25.240000	...	12.98

	TOTS3.SA	TRPL4.SA	UGPA3.SA	USIM5.SA	VALE3.SA	VIVT3.SA	\
Date							
2020-01-02	22.856628	20.440050	25.279165	9.610517	52.150002	43.943169	
2020-01-03	23.342520	19.928377	24.982924	9.511439	51.766048	44.453289	
2020-01-06	22.810352	20.045073	24.439817	9.333100	51.458889	44.025158	
2020-01-07	22.906208	19.901445	24.982924	9.422270	51.833241	44.999847	
2020-01-08	22.598810	19.748840	24.785431	9.303377	51.842838	45.455315	
...	...	...	...	...	...	...	
2021-02-23	33.520000	24.848221	21.379999	15.740000	96.949997	45.019234	
2021-02-24	33.130001	25.371342	21.280001	17.240000	97.930000	44.300682	
2021-02-25	31.940001	24.983845	19.709999	16.490000	95.709999	44.430420	
2021-02-26	31.540001	24.139999	19.410000	16.190001	94.660004	44.091103	
2021-03-01	32.279999	23.799999	19.389999	16.980000	97.660004	44.169998	

	VVAR3.SA	WEGE3.SA	YDUQ3.SA
Date			
2020-01-02	11.73	34.803658	45.885044
2020-01-03	11.48	34.359226	44.923176
2020-01-06	11.48	34.448112	45.786896
2020-01-07	11.65	34.714771	44.932987
2020-01-08	11.60	33.401234	44.667984
...	...	...	...
2021-02-23	13.19	83.289383	31.680000
2021-02-24	13.27	86.315903	31.450001
2021-02-25	12.61	79.645592	30.420000
2021-02-26	11.92	78.510643	29.770000
2021-03-01	12.47	78.720001	28.930000

[285 rows x 100 columns]

## 1.2 Preparando os dados

Nesta etapa, verificamos os NAs nas linhas e os eliminamos.  
 Vamos nos livrar das colunas com mais de 25% de valores ausentes

```
[3]: missing_fractions = dataset.isnull().mean().sort_values(ascending=False)

missing_fractions.head(10)

drop_list = sorted(list(missing_fractions[missing_fractions > 0.25].index))

dataset.drop(labels=drop_list, axis=1, inplace=True)
dataset.shape
```

```
[3]: (285, 99)
```

```
[4]: # Fill the missing values with the last value available in the dataset.
dataset=dataset.fillna(method='ffill')
```

### 1.3 Transformação dos dados

Para fins de agrupamento, usaremos os retornos anuais e a variância como variáveis, visto que são indicadores primários de desempenho e volatilidade das ações.

```
[5]: returns = dataset.pct_change().mean() * 252
returns = pd.DataFrame(returns)
returns.columns = ['Returns']
returns['Volatility'] = dataset.pct_change().std() * np.sqrt(252)
data=returns
```

Todas as variáveis devem estar na mesma escala antes de aplicar o clustering, caso contrário, um recurso com grandes valores dominará o resultado. Usamos StandardScaler no sklearn para padronizar os recursos do conjunto de dados em escala unitária (média = 0 e variância = 1).

```
[6]: from sklearn.preprocessing import StandardScaler
escala = StandardScaler().fit(data)
reescalando = pd.DataFrame(escala.fit_transform(data),
                           columns = data.columns,
                           index = data.index)

X = reescalando
X.head(5)
```

```
[6]:           Returns  Volatility
ABEV3.SA -0.542219   -1.073463
ALPA4.SA  0.449113    0.036691
ALSO3.SA -1.199285    0.471416
AMAR3.SA -1.121323    1.808360
AZUL4.SA  0.426963    2.650113
```

## 2 Formação e avaliação do modelo

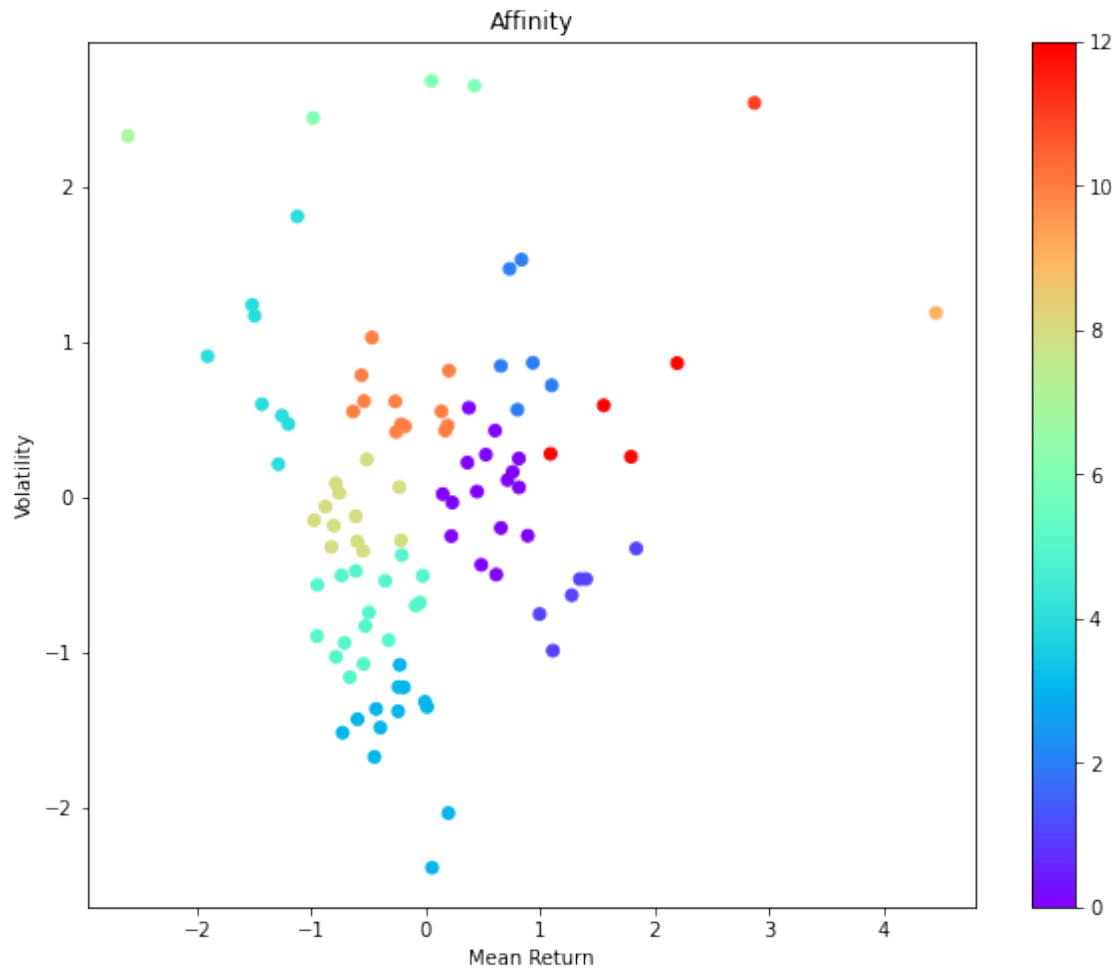
Há algumas formas para avaliar a formação de clusters em ações como, por exemplo, “K-means”, “Hierarchical Clustering” e “Affinity Propagation”. Como já foi analisado previamente em estudos anteriores, nesse exercício vamos encurtar o código usando apenas a terceira opção, a qual se mostrou com melhores resultados.

### 2.1 Affinity Propagation

```
[7]: ap = AffinityPropagation()  
     ap.fit(X)  
     clust_labels = ap.predict(X)
```

```
[8]: fig = plt.figure(figsize=(10,8))  
     ax = fig.add_subplot(111)  
     scatter = ax.scatter(X.iloc[:,0],X.iloc[:,1], c =clust_labels, cmap ="rainbow")  
     ax.set_title('Affinity')  
     ax.set_xlabel('Mean Return')  
     ax.set_ylabel('Volatility')  
     plt.colorbar(scatter)
```

```
[8]: <matplotlib.colorbar.Colorbar at 0x208496b8a30>
```



## 2.2 Vizualização dos Clusters

```
[9]: cluster_centers_indices = ap.cluster_centers_indices_
     labels = ap.labels_
```

```
[10]: no_clusters = len(cluster_centers_indices)
      print('Estimated number of clusters: %d' % no_clusters)
      # Plot exemplars

      X_temp=np.asarray(X)
      plt.close('all')
      plt.figure(1)
      plt.clf()

      fig = plt.figure(figsize=(8,6))
      colors = cycle('bgrcmykbgrcmykbgrcmykbgrcmyk')
```

```

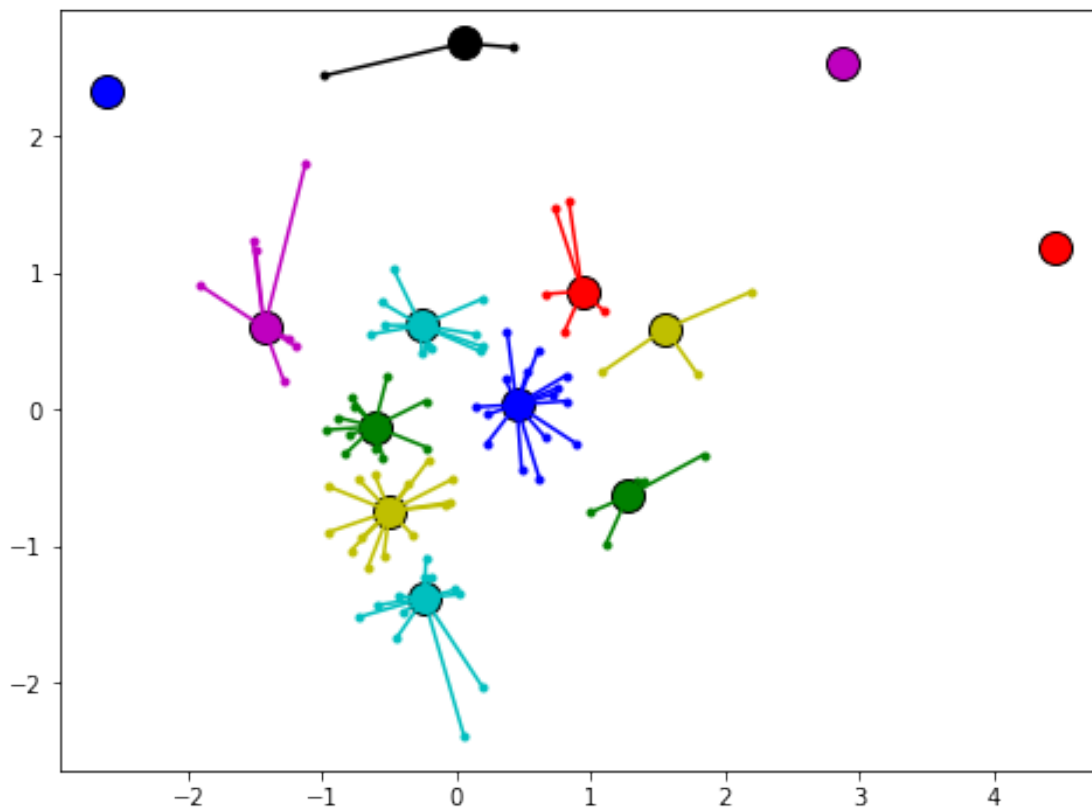
for k, col in zip(range(no_clusters), colors):
    class_members = labels == k
    cluster_center = X_temp[cluster_centers_indices[k]]
    plt.plot(X_temp[class_members, 0], X_temp[class_members, 1], col + '.')
    plt.plot(cluster_center[0], cluster_center[1], 'o', markerfacecolor=col,
    ↪markeredgecolor='k', markersize=14)
    for x in X_temp[class_members]:
        plt.plot([cluster_center[0], x[0]], [cluster_center[1], x[1]], col)

plt.show()

```

Estimated number of clusters: 13

<Figure size 432x288 with 0 Axes>

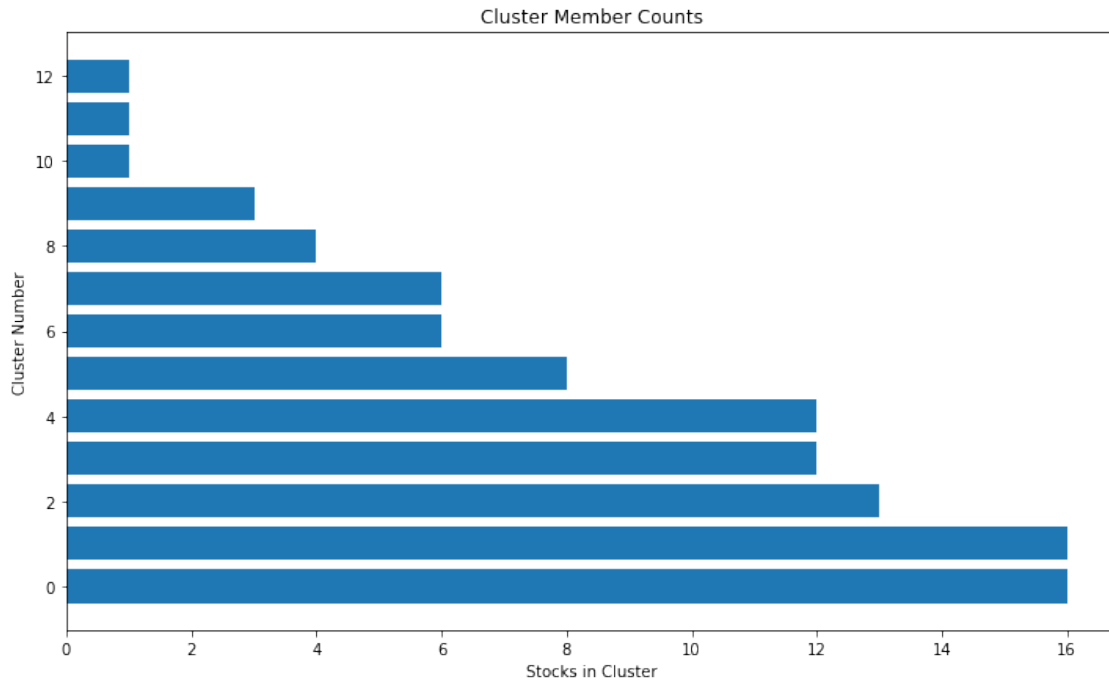


```

[11]: # show number of stocks in each cluster
clustered_series_ap = pd.Series(index=X.index, data=ap.labels_.flatten())
# clustered stock with its cluster label
clustered_series_all_ap = pd.Series(index=X.index, data=ap.labels_.flatten())
clustered_series_ap = clustered_series_ap[clustered_series_ap != -1]

```

```
plt.figure(figsize=(12,7))
plt.barh(
    range(len(clustered_series_ap.value_counts())), # cluster labels, y axis
    clustered_series_ap.value_counts()
)
plt.title('Cluster Member Counts')
plt.xlabel('Stocks in Cluster')
plt.ylabel('Cluster Number')
plt.show()
```



## 2.3 Avaliação dos Clusters

O modelo sugeriu a formação de 13 clusters entre as ações do IBRX.

### 2.3.1 Visualizando o retorno dentro de um cluster

Para entender a intuição por trás do clustering, vamos visualizar os resultados dos clusters.

```
[12]: # all stock with its cluster label (including -1)
clustered_series = pd.Series(index=X.index, data=ap.fit_predict(X).flatten())
# clustered stock with its cluster label
clustered_series_all = pd.Series(index=X.index, data=ap.fit_predict(X).
    ↪flatten())
clustered_series = clustered_series[clustered_series != -1]
```



```
[13]: # get the number of stocks in each cluster
counts = clustered_series_ap.value_counts()

# let's visualize some clusters
cluster_vis_list = list(counts[(counts<25) & (counts>1)].index[::-1])
cluster_vis_list
```

```
[13]: [6, 12, 1, 2, 4, 8, 10, 3, 0, 5]
```

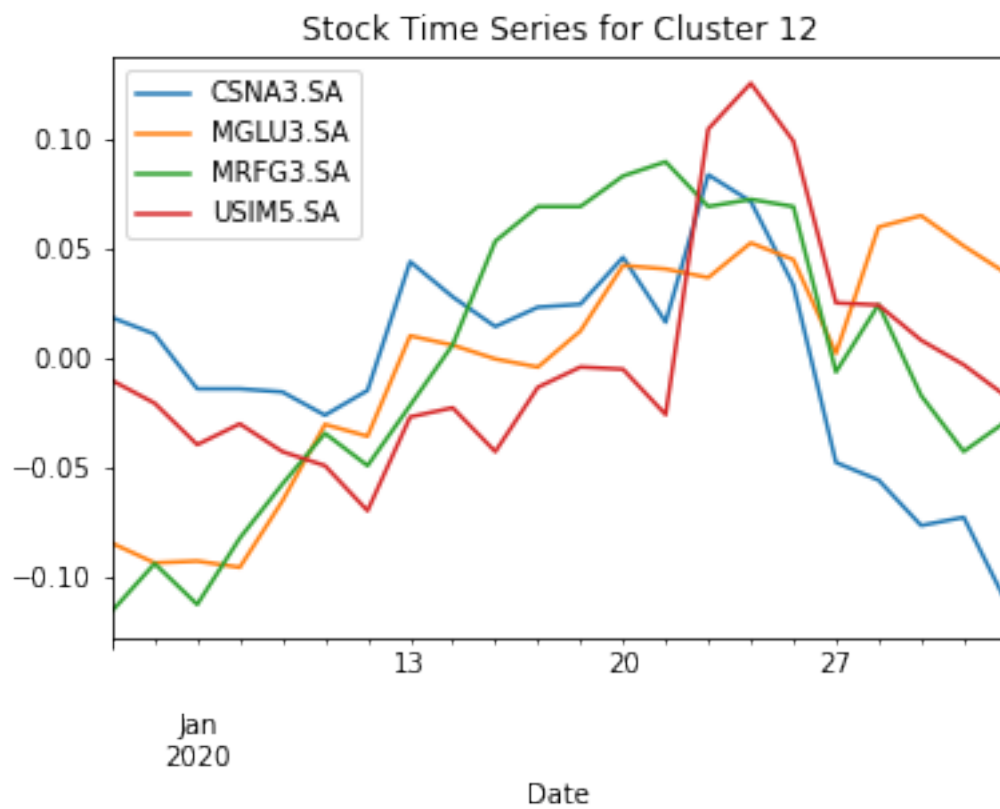
```
[14]: CLUSTER_SIZE_LIMIT = 9999
counts = clustered_series.value_counts()
ticker_count_reduced = counts[(counts>1) & (counts<=CLUSTER_SIZE_LIMIT)]
print ("Clusters formed: %d" % len(ticker_count_reduced))
print ("Pairs to evaluate: %d" % len(ticker_count_reduced*(ticker_count_reduced-1)))
```

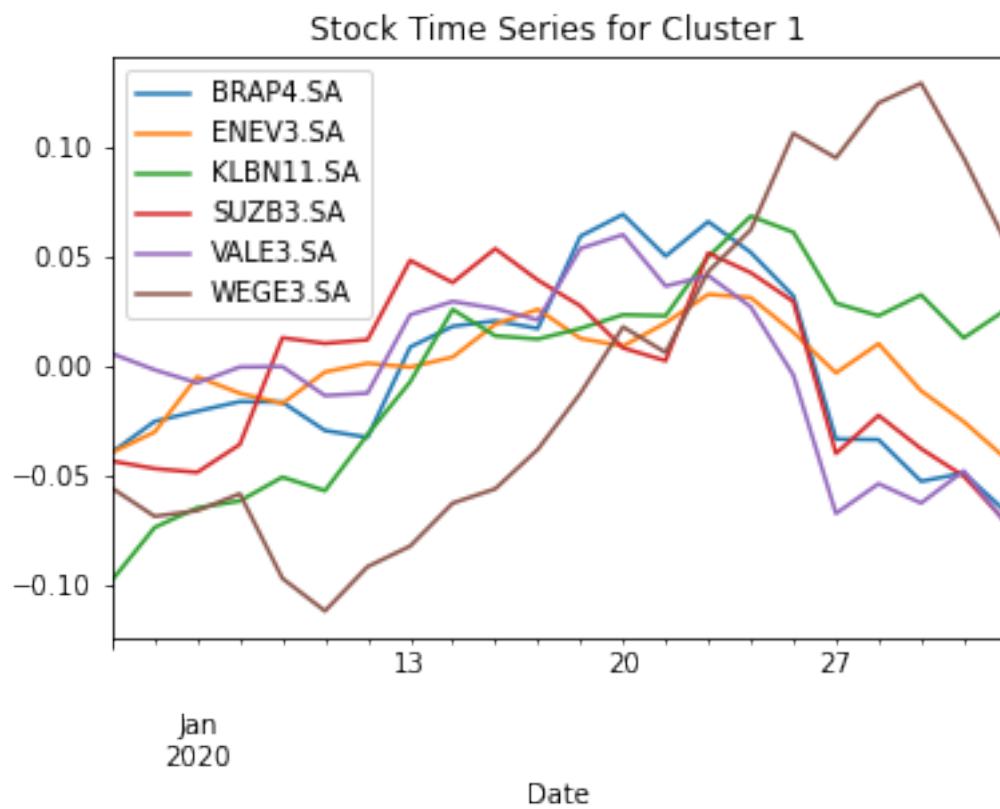
```
Clusters formed: 10
Pairs to evaluate: 1034
```

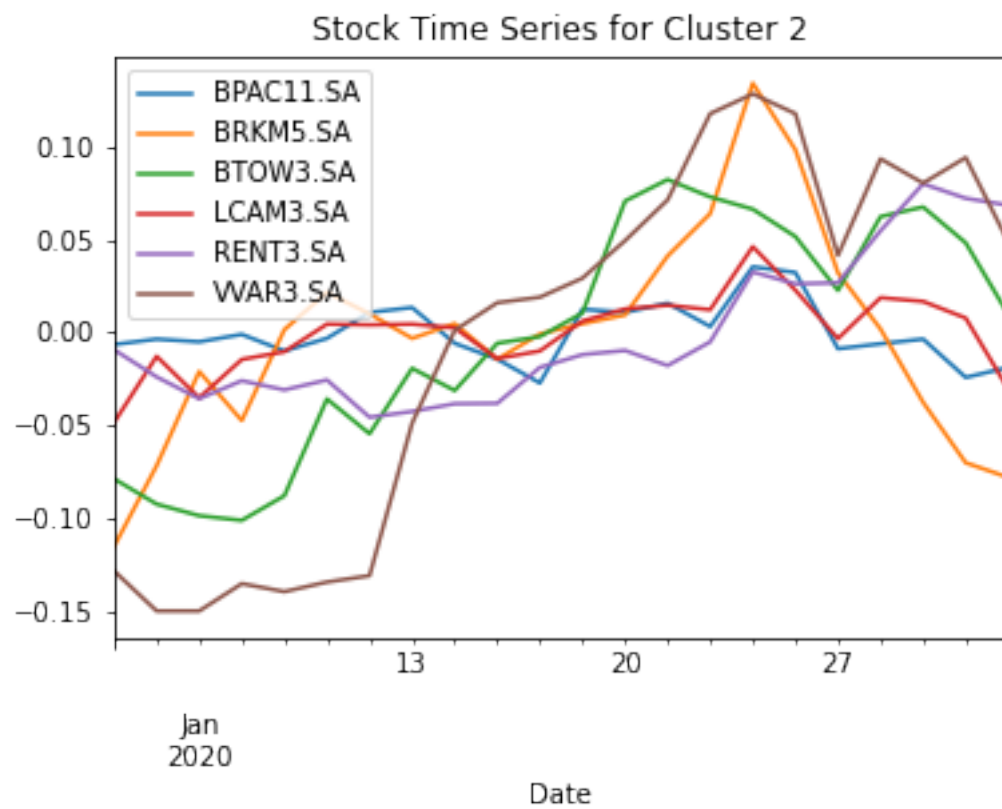
```
[15]: # Vizualizando os 10 clusters

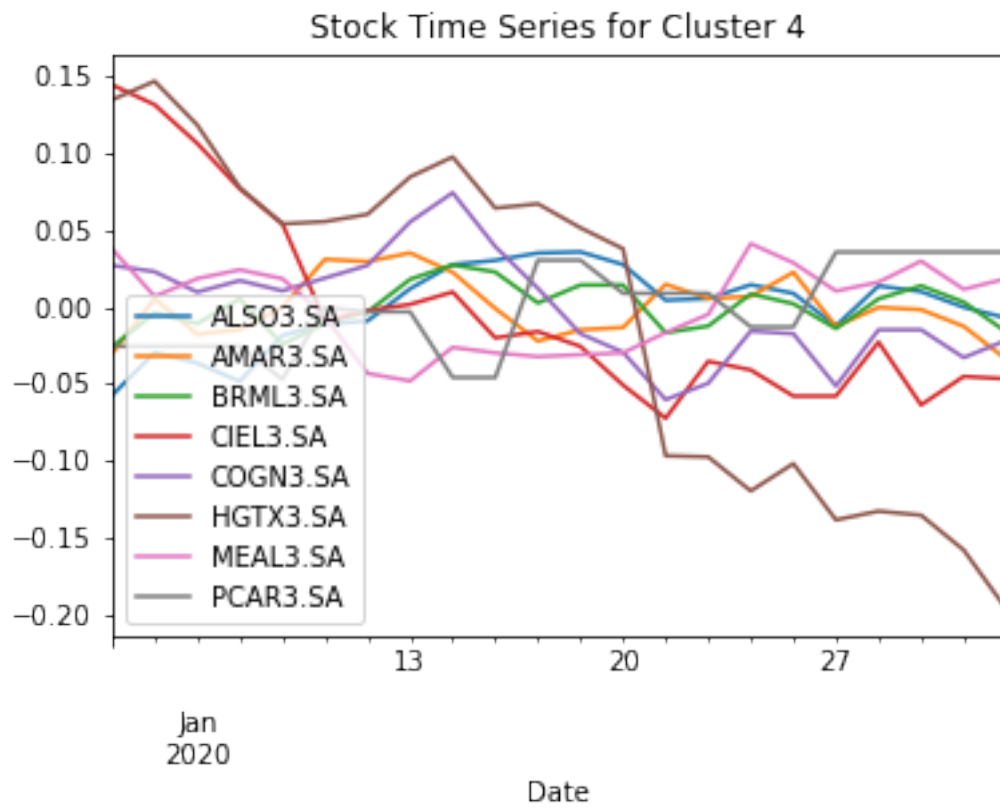
for clust in cluster_vis_list[0:min(len(cluster_vis_list),10)]:
    tickers = list(clustered_series[clustered_series==clust].index)
    means = np.log(dataset.loc["2020-02-02", tickers].mean())
    data = np.log(dataset.loc["2020-02-02", tickers]).sub(means)
    data.plot(title='Stock Time Series for Cluster %d' % clust)
plt.show()
```

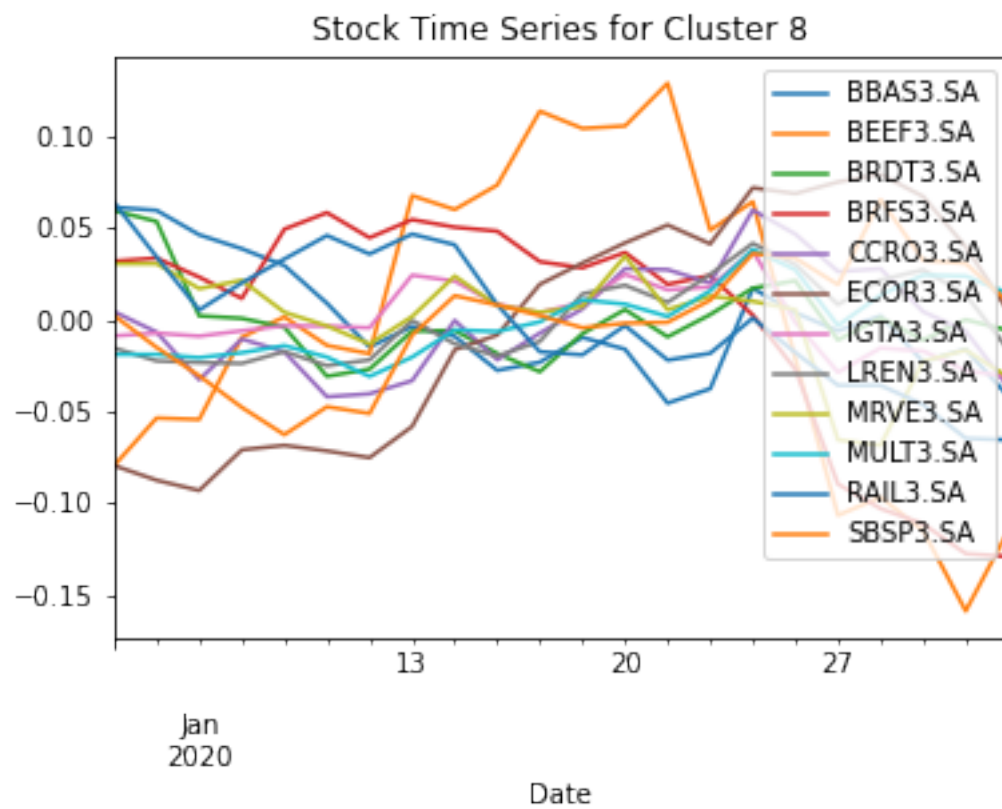


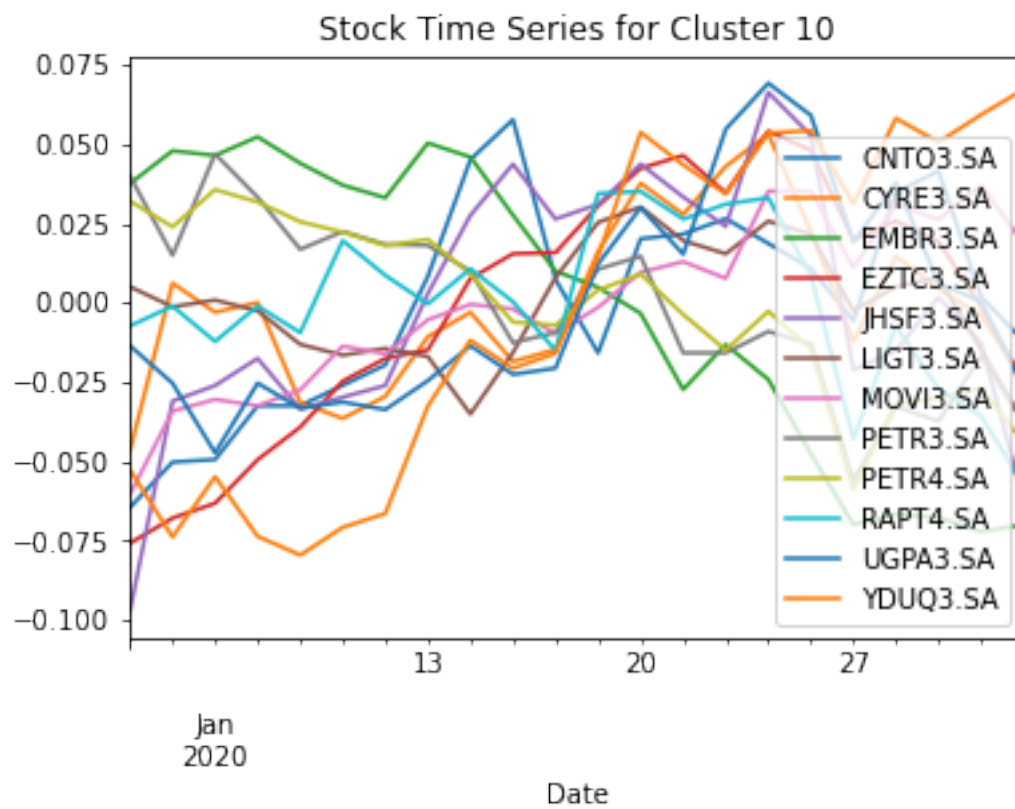




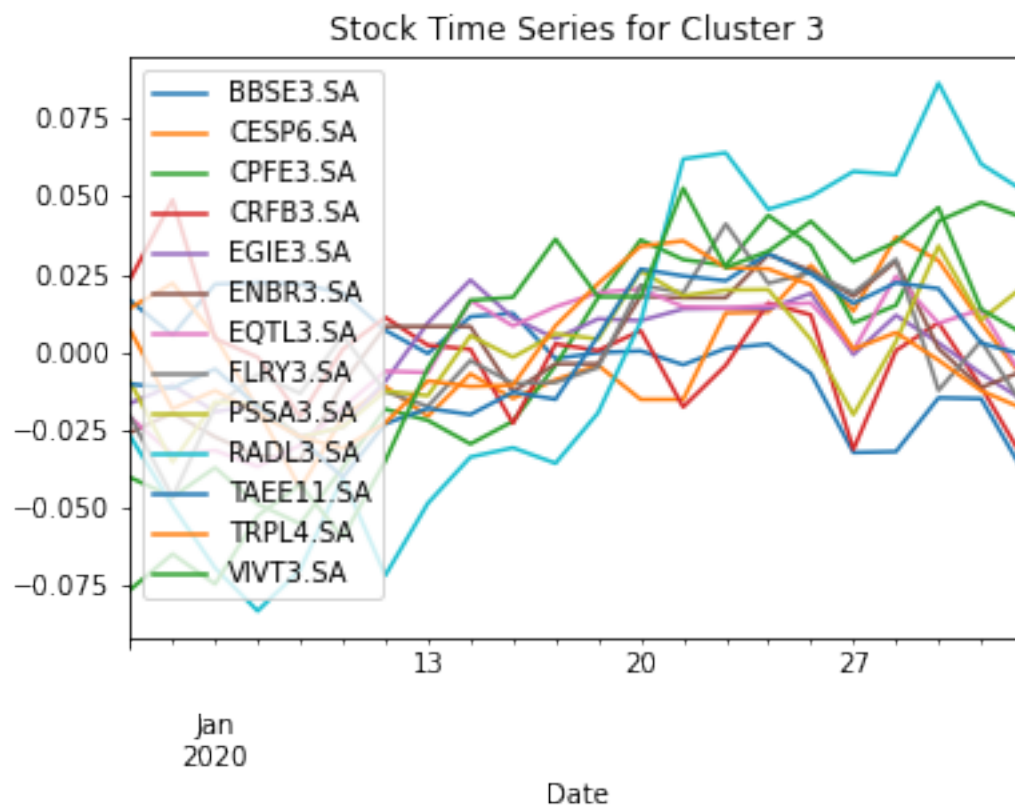


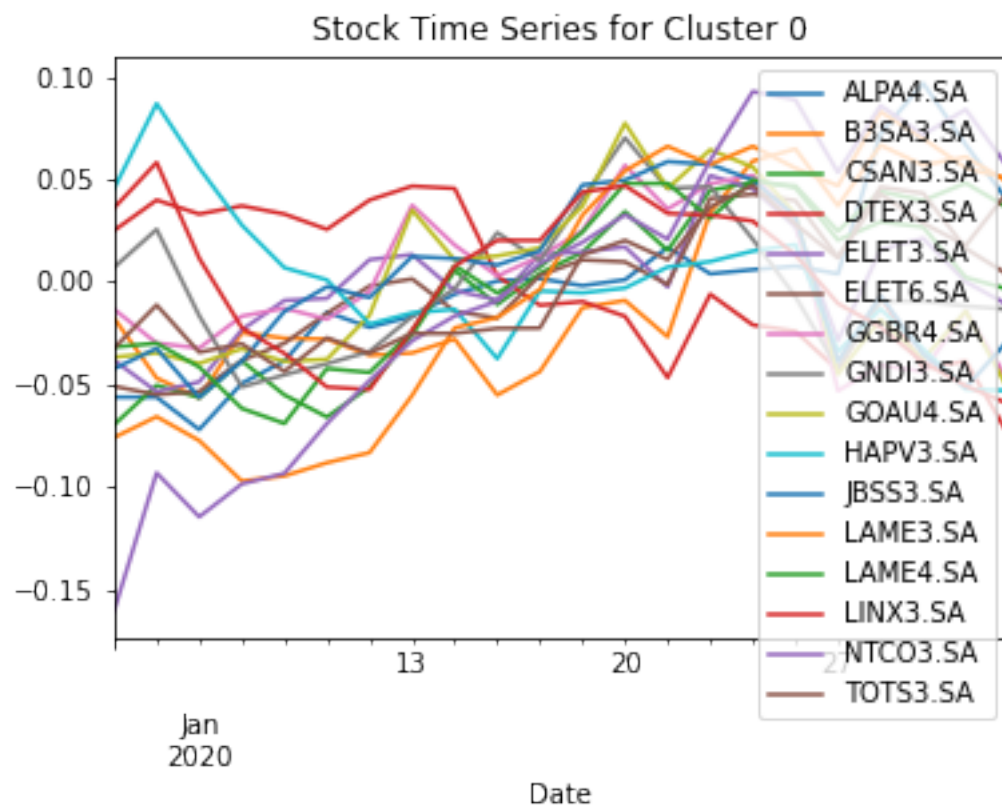


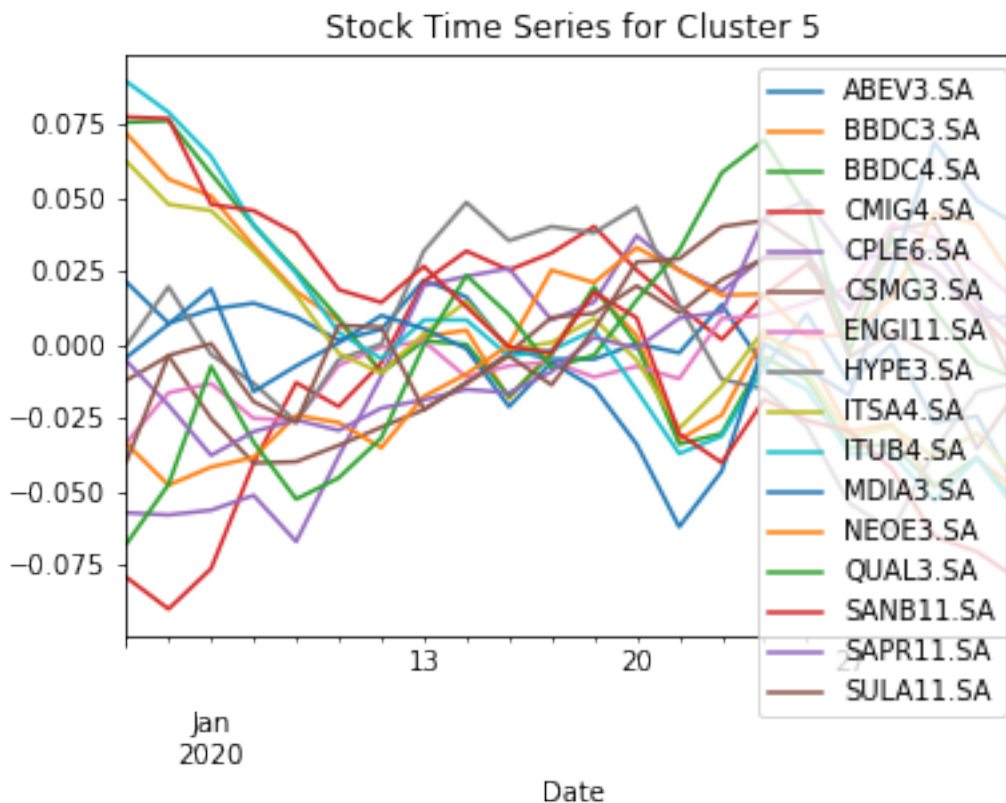












### 3 Seleção de pares

Uma vez que os clusters são criados, várias técnicas estatísticas baseadas em cointegração podem ser aplicadas nos estoques dentro de um cluster para criar os pares. Nesta etapa, examinamos uma lista de títulos dentro de um cluster e testamos a cointegração entre os pares. Primeiro, escrevemos uma função que retorna uma matriz de pontuação de teste de cointegração, uma matriz de valor p e quaisquer pares para os quais o valor p seja menor que 0.05.

```
[16]: def find_cointegrated_pairs(data, significance=0.05):
    # This function is from https://www.quantopian.com/lectures/
    ↪ introduction-to-pairs-trading
    n = data.shape[1]
    score_matrix = np.zeros((n, n))
    pvalue_matrix = np.ones((n, n))
    keys = data.keys()
    pairs = []
    for i in range(1):
        for j in range(i+1, n):
            S1 = data[keys[i]]
            S2 = data[keys[j]]
```

```

        result = coint(S1, S2)
        score = result[0]
        pvalue = result[1]
        score_matrix[i, j] = score
        pvalue_matrix[i, j] = pvalue
        if pvalue < significance:
            pairs.append((keys[i], keys[j]))
    return score_matrix, pvalue_matrix, pairs

```

```

[17]: from statsmodels.tsa.stattools import coint
cluster_dict = {}
for i, which_clust in enumerate(ticker_count_reduced.index):
    tickers = clustered_series[clustered_series == which_clust].index
    score_matrix, pvalue_matrix, pairs = find_cointegrated_pairs(
        dataset[tickers]
    )
    cluster_dict[which_clust] = {}
    cluster_dict[which_clust]['score_matrix'] = score_matrix
    cluster_dict[which_clust]['pvalue_matrix'] = pvalue_matrix
    cluster_dict[which_clust]['pairs'] = pairs

```

```

[18]: pairs = []
for clust in cluster_dict.keys():
    pairs.extend(cluster_dict[clust]['pairs'])

```

```

[19]: print ("Number of pairs found : %d" % len(pairs))
print ("In those pairs, there are %d unique tickers." % len(np.unique(pairs)))

```

Number of pairs found : 20  
In those pairs, there are 26 unique tickers.

```

[20]: pairs

```

```

[20]: [('ABEV3.SA', 'BBDC3.SA'),
      ('ABEV3.SA', 'BBDC4.SA'),
      ('ABEV3.SA', 'CMIG4.SA'),
      ('ABEV3.SA', 'CPLE6.SA'),
      ('ABEV3.SA', 'ENGI11.SA'),
      ('ABEV3.SA', 'HYPE3.SA'),
      ('ABEV3.SA', 'ITSA4.SA'),
      ('ABEV3.SA', 'ITUB4.SA'),
      ('ALPA4.SA', 'GGBR4.SA'),
      ('ALPA4.SA', 'LINX3.SA'),
      ('BBSE3.SA', 'CPFE3.SA'),
      ('BBSE3.SA', 'EGIE3.SA'),
      ('BBSE3.SA', 'ENBR3.SA'),
      ('CNT03.SA', 'YDUQ3.SA'),

```

```
( 'BBAS3.SA', 'ECOR3.SA'),
( 'BBAS3.SA', 'IGTA3.SA'),
( 'BBAS3.SA', 'LREN3.SA'),
( 'BBAS3.SA', 'MULT3.SA'),
( 'ALSO3.SA', 'BRML3.SA'),
( 'ALSO3.SA', 'MEAL3.SA')]
```

## 4 Vizualização das ações em pares

Vamos analisar o gráfico de 4 pares dos 16 formados.

```
[21]: tickers = ['ABEV3.SA', 'ALPA4.SA', 'BBAS3.SA', 'CNT03.SA', 'ALSO3.SA', 'CPFE3.
↪SA', 'EGIE3.SA',
               'ENBR3.SA', 'ENGI11.SA', 'ITSA4.SA', 'ITUB4.SA', 'GGBR4.SA', 'LINX3.
↪SA', 'RAPT4.SA',
               'ECOR3.SA', 'IGTA3.SA', 'LREN3.SA', 'MULT3.SA', 'YDUQ3.SA', 'BRML3.
↪SA', 'MEAL3.SA']

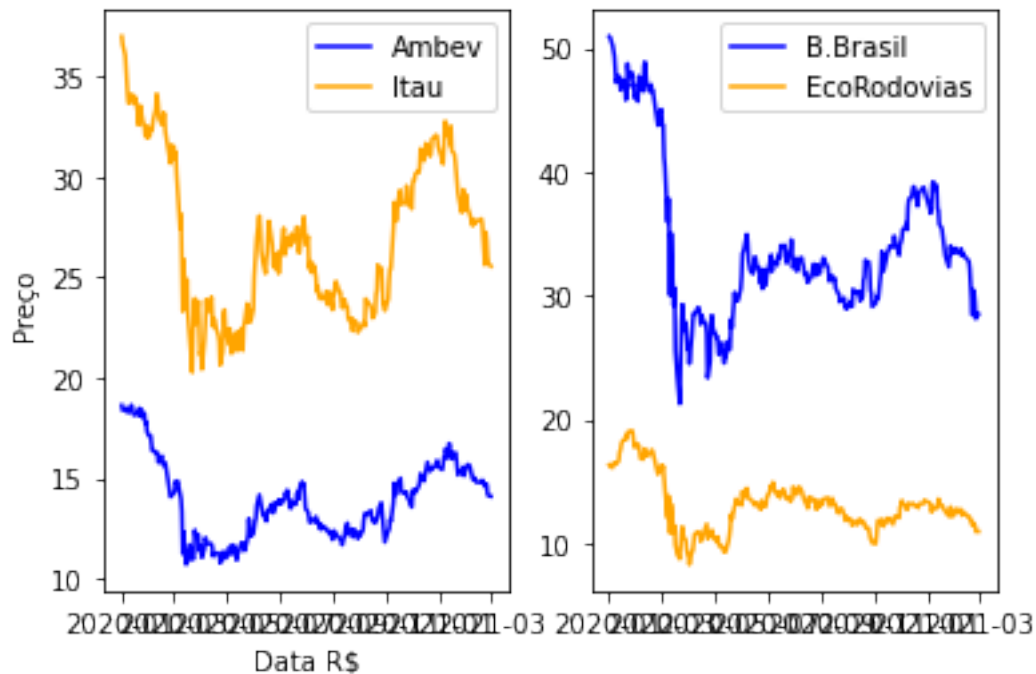
stocks = yf.download(tickers = tickers, start = '2020-01-01')['Adj Close']
stocks = pd.DataFrame(stocks)
```

[\*\*\*\*\*100%\*\*\*\*\*] 21 of 21 completed

```
[22]: plt.subplot(1,2,1)
plt.plot(stocks['ABEV3.SA'], color = 'blue', label = 'Ambev')
plt.plot(stocks['ITUB4.SA'], color = 'orange', label = 'Itau')
plt.legend(loc = 'best')
plt.xlabel('Data R$')
plt.ylabel('Preço')

plt.subplot(1,2,2)
plt.plot(stocks['BBAS3.SA'], color = 'blue', label = 'B.Brasil')
plt.plot(stocks['ECOR3.SA'], color = 'orange', label = 'EcoRodovias')
plt.legend(loc = 'best')
fig.autofmt_xdate()

plt.show()
```



```
[23]: plt.subplot(1,2,1)
plt.plot(stocks['ALSO3.SA'], color = 'blue', label = 'Aliansce')
plt.plot(stocks['MEAL3.SA'], color = 'orange', label = 'Meal')
plt.legend(loc = 'best')
fig.autofmt_xdate()

plt.subplot(1,2,2)
plt.plot(stocks['CNT03.SA'], color = 'blue', label = 'Grupo SBF')
plt.plot(stocks['YDUQ3.SA'], color = 'orange', label = 'YDUQS')
plt.legend(loc = 'best')
fig.autofmt_xdate()
```

