

Aplicação de Análise Estatística Descritiva no Mercado de Criptoativos

July 26, 2022

0.1 Aplicação de Análise Estatística Descritiva no Mercado de Criptoativos

0.2 Importando as Bibliotecas

```
[ ]: import math

import requests

import numpy as np
import pandas as pd
import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt

from scipy import stats
from scipy import optimize
from pandas_datareader import data
```

0.2.1 Construindo uma Base de Dados Financeiros com Ações Do Mercado De Criptoativos

O Mercado Bitcoin, maior corretora de criptomoedas da América Latina, disponibiliza uma [API pública](#) onde é possível consultar o resumo diário de negociações realizadas para um dado criptoativo.

Realizando a seguinte [chamada](#) tem-se os dados de negociação (compra e venda) do criptoativo *Bitcoin* (BTC) para o dia 29/06/2022. Desta maneira, serão realizadas chamadas a API pública com intuito de construir uma Base de Dados com os ativos BITCOIN(BTC), CARDANO (ADA), LITECOIN(LTC), CHILIZ(CHZ), USDC, NANO(LINK) ao longo dos anos de 2015 a 2022, em todos os dias e meses do ano.

```
[ ]: ativos = {"BITCOIN": "BTC", "CARDANO": "ADA", "LITECOIN": "LTC", "CHILIZ": "CHZ", "USDC": "USDC", "NANO": "LINK"}
anos = [2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022]
meses = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
dias = list(range(1, 32))
```

```
[ ]: resumo_ativos = []
url_resumo_diario = "https://www.mercadobitcoin.net/api/{ativo}/day-summary/
↳{ano}/{mes}/{dia}/"

for ativo in ativos.values():
    print(ativo)
    resumo_ativo = []
    for ano in anos:
        print(ano)
        for mes in meses:
            # pular os meses futuros do ano de 2022
            if ano == 2022 and mes > 7:
                continue
            for dia in dias:
                # pular os dias futuros de julho de 2022
                if ano == 2022 and mes == 7 and dia >= 25:
                    continue
                url_dia_formatada = url_resumo_diario.
↳format(ativo=ativo,ano=ano, mes=mes, dia=dia)
                resposta = requests.get(url=url_dia_formatada)
                if resposta.status_code == 200:
                    resumo_ativo.append(resposta.json())
                else:
                    continue
            resumo_ativos.append(resumo_ativo)
```

```
[ ]: resumo_ativos
```

```
[234]: acoes_df = pd.DataFrame()

for ativo, resumo in zip(ativos, resumo_ativos):
    indice = [r["date"] for r in resumo]
    acoes_df[ativo] = pd.DataFrame(resumo)["closing"]

acoes_df["Date"] = [r["date"] for r in resumo_ativos[0]]
acoes_df
```

```
[234]:
```

	BITCOIN	CARDANO	LITECOIN	CHILIZ	USDC	NANO	\
0	880.100050	12.89956	8.21562	0.07000	5.44500	60.37949	
1	921.847150	12.43990	8.39407	0.06800	5.38850	62.54979	
2	810.010000	12.40000	6.50000	0.06540	5.46790	64.27989	
3	848.999990	12.33131	6.02040	0.06510	5.44001	65.48000	
4	788.000000	12.07408	6.37030	0.06439	5.39874	65.57001	
...	
2809	125000.000000	NaN	313.65668	NaN	NaN	NaN	
2810	126708.000000	NaN	311.61799	NaN	NaN	NaN	
2811	126017.000000	NaN	NaN	NaN	NaN	NaN	

2812	124115.769950	NaN	NaN	NaN	NaN	NaN
2813	121148.148952	NaN	NaN	NaN	NaN	NaN

	Date
0	2015-01-01
1	2015-01-02
2	2015-01-03
3	2015-01-04
4	2015-01-05
...	...
2809	2022-07-20
2810	2022-07-21
2811	2022-07-22
2812	2022-07-23
2813	2022-07-24

[2814 rows x 7 columns]

0.2.2 Visualização dos Dados

O gráfico abaixo nos apresenta o histórico de preço do determinado criptoativo ao longo dos anos de 2015 a 2022.

```
[235]: figura = px.line(title = "Histórico do preço das ações")
for ativo in acoes_df.columns[:-1]:
    figura.add_scatter(x=acoes_df["Date"], y=acoes_df[ativo], name=ativo)
figura.show()
```

Taxa de Retorno de Ações

$$\mathbb{E}[R_i] = \log\left(\frac{P_t}{P_{t-1}}\right) \quad (1)$$

```
[236]: dataset = acoes_df.copy()
dataset
```

	BITCOIN	CARDANO	LITECOIN	CHILIZ	USDC	NANO	\
0	880.100050	12.89956	8.21562	0.07000	5.44500	60.37949	
1	921.847150	12.43990	8.39407	0.06800	5.38850	62.54979	
2	810.010000	12.40000	6.50000	0.06540	5.46790	64.27989	
3	848.999990	12.33131	6.02040	0.06510	5.44001	65.48000	
4	788.000000	12.07408	6.37030	0.06439	5.39874	65.57001	
...	
2809	125000.000000	NaN	313.65668	NaN	NaN	NaN	
2810	126708.000000	NaN	311.61799	NaN	NaN	NaN	
2811	126017.000000	NaN	NaN	NaN	NaN	NaN	
2812	124115.769950	NaN	NaN	NaN	NaN	NaN	

2813	121148.148952	NaN	NaN	NaN	NaN	NaN
------	---------------	-----	-----	-----	-----	-----

	Date
0	2015-01-01
1	2015-01-02
2	2015-01-03
3	2015-01-04
4	2015-01-05
...	...
2809	2022-07-20
2810	2022-07-21
2811	2022-07-22
2812	2022-07-23
2813	2022-07-24

[2814 rows x 7 columns]

```
[237]: dataset.drop(labels = ['Date'], axis=1, inplace=True)
dataset
```

	BITCOIN	CARDANO	LITECOIN	CHILIZ	USDC	NANO
0	880.100050	12.89956	8.21562	0.07000	5.44500	60.37949
1	921.847150	12.43990	8.39407	0.06800	5.38850	62.54979
2	810.010000	12.40000	6.50000	0.06540	5.46790	64.27989
3	848.999990	12.33131	6.02040	0.06510	5.44001	65.48000
4	788.000000	12.07408	6.37030	0.06439	5.39874	65.57001
...
2809	125000.000000	NaN	313.65668	NaN	NaN	NaN
2810	126708.000000	NaN	311.61799	NaN	NaN	NaN
2811	126017.000000	NaN	NaN	NaN	NaN	NaN
2812	124115.769950	NaN	NaN	NaN	NaN	NaN
2813	121148.148952	NaN	NaN	NaN	NaN	NaN

[2814 rows x 6 columns]

```
[238]: dataset.shift(1)
```

	BITCOIN	CARDANO	LITECOIN	CHILIZ	USDC	NANO
0	NaN	NaN	NaN	NaN	NaN	NaN
1	880.10005	12.89956	8.21562	0.0700	5.44500	60.37949
2	921.84715	12.43990	8.39407	0.0680	5.38850	62.54979
3	810.01000	12.40000	6.50000	0.0654	5.46790	64.27989
4	848.99999	12.33131	6.02040	0.0651	5.44001	65.48000
...
2809	127065.22928	NaN	311.44263	NaN	NaN	NaN
2810	125000.00000	NaN	313.65668	NaN	NaN	NaN
2811	126708.00000	NaN	311.61799	NaN	NaN	NaN

2812	126017.00000	NaN	NaN	NaN	NaN	NaN
2813	124115.76995	NaN	NaN	NaN	NaN	NaN

[2814 rows x 6 columns]

O cálculo abaixo representa a Taxa de Retorno de Ações que consiste em na diferença do valor de um investimento de um dia para o outro, ou seja, o quanto esse ativo apresentou lucro ou prejuízo.

```
[239]: taxas_retorno = np.log(dataset / dataset.shift(1))
taxas_retorno
```

```
[239]:
```

	BITCOIN	CARDANO	LITECOIN	CHILIZ	USDC	NANO
0	NaN	NaN	NaN	NaN	NaN	NaN
1	0.046344	-0.036284	0.021488	-0.028988	-0.010431	0.035313
2	-0.129333	-0.003213	-0.255723	-0.038985	0.014628	0.027284
3	0.047013	-0.005555	-0.076648	-0.004598	-0.005114	0.018498
4	-0.074561	-0.021081	0.056493	-0.010966	-0.007615	0.001374
...
2809	-0.016387	NaN	0.007084	NaN	NaN	NaN
2810	0.013571	NaN	-0.006521	NaN	NaN	NaN
2811	-0.005468	NaN	NaN	NaN	NaN	NaN
2812	-0.015202	NaN	NaN	NaN	NaN	NaN
2813	-0.024201	NaN	NaN	NaN	NaN	NaN

[2814 rows x 6 columns]

```
[240]: taxas_retorno.describe()
```

```
[240]:
```

	BITCOIN	CARDANO	LITECOIN	CHILIZ	USDC \
count	2813.000000	296.000000	2810.000000	754.000000	804.000000
mean	0.001751	-0.005255	0.001294	0.002804	0.000014
std	0.036786	0.049907	0.056861	0.081998	0.010333
min	-0.338736	-0.273860	-0.416640	-0.507883	-0.058051
25%	-0.012420	-0.035786	-0.023377	-0.030629	-0.005783
50%	0.001301	-0.007014	0.000346	0.000154	0.000000
75%	0.018018	0.023472	0.023323	0.032850	0.005611
max	0.224670	0.131900	0.500463	0.663503	0.040221

	NANO
count	603.000000
mean	-0.000802
std	0.061187
min	-0.362016
25%	-0.037127
50%	0.000878
75%	0.036673
max	0.202124

A média de retorno para cada ativo listado anteriormente. Como esperado o Bitcoin apresentou a maior Taxa de Retorno de Ações.

```
[249]: medias = (taxas_retorno[list(ativos.keys())].sum() /  
↳ len(taxas_retorno[list(ativos.keys())]))*100  
medias
```

```
[249]: BITCOIN      0.175008  
CARDANO      -0.055274  
LITECOIN     0.129202  
CHILIZ       0.075143  
USDC         0.000403  
NANO         -0.017181  
dtype: float64
```

```
[250]: taxas_retorno.mean()*100
```

```
[250]: BITCOIN      0.175071  
CARDANO      -0.525479  
LITECOIN     0.129386  
CHILIZ       0.280442  
USDC         0.001409  
NANO         -0.080178  
dtype: float64
```

```
[252]: vars_acoes = ((taxas_retorno[list(ativos.keys())] - taxas_retorno.mean()) ** 2).  
↳ sum() / (len(taxas_retorno[list(ativos.keys())]) - 1)  
vars_acoes
```

```
[252]: BITCOIN      0.001353  
CARDANO      0.000261  
LITECOIN     0.003229  
CHILIZ       0.001800  
USDC         0.000030  
NANO         0.000801  
dtype: float64
```

```
[253]: taxas_retorno.var()
```

```
[253]: BITCOIN      0.001353  
CARDANO      0.002491  
LITECOIN     0.003233  
CHILIZ       0.006724  
USDC         0.000107  
NANO         0.003744  
dtype: float64
```

```
[254]: taxas_retorno.std()*100
```

```
[254]: BITCOIN      3.678649  
CARDANO      4.990717  
LITECOIN     5.686067  
CHILIZ       8.199754  
USDC         1.033278  
NANO         6.118684  
dtype: float64
```

```
[255]: dataset_date = acoes_df.copy()  
date = dataset_date.filter(["Date"])  
date
```

```
[255]:          Date  
0      2015-01-01  
1      2015-01-02  
2      2015-01-03  
3      2015-01-04  
4      2015-01-05  
...      ...  
2809   2022-07-20  
2810   2022-07-21  
2811   2022-07-22  
2812   2022-07-23  
2813   2022-07-24
```

[2814 rows x 1 columns]

```
[256]: taxas_retorno_date = pd.concat([date, taxas_retorno], axis=1)  
taxas_retorno_date
```

```
[256]:          Date  BITCOIN  CARDANO  LITECOIN  CHILIZ  USDC  NANO  
0      2015-01-01      NaN      NaN      NaN      NaN      NaN      NaN  
1      2015-01-02  0.046344 -0.036284  0.021488 -0.028988 -0.010431  0.035313  
2      2015-01-03 -0.129333 -0.003213 -0.255723 -0.038985  0.014628  0.027284  
3      2015-01-04  0.047013 -0.005555 -0.076648 -0.004598 -0.005114  0.018498  
4      2015-01-05 -0.074561 -0.021081  0.056493 -0.010966 -0.007615  0.001374  
...      ...      ...      ...      ...      ...      ...  
2809   2022-07-20 -0.016387      NaN  0.007084      NaN      NaN      NaN  
2810   2022-07-21  0.013571      NaN -0.006521      NaN      NaN      NaN  
2811   2022-07-22 -0.005468      NaN      NaN      NaN      NaN      NaN  
2812   2022-07-23 -0.015202      NaN      NaN      NaN      NaN      NaN  
2813   2022-07-24 -0.024201      NaN      NaN      NaN      NaN      NaN
```

[2814 rows x 7 columns]

De maneira visual é possível chegar a mesma conclusão dos passos anteriores: Ao longo do tempo o ativo Bitcoin apresentou a maior Taxa de Retorno de Ações.

```
[258]: figura = px.line(title = 'Histórico de retorno das ações')
for i in taxas_retorno_date.columns[1:]:
    figura.add_scatter(x = taxas_retorno_date["Date"] ,y = taxas_retorno_date[i],
name = i)
figura.show()
```

```
[259]: taxas_retorno.cov()
```

```
[259]:
```

	BITCOIN	CARDANO	LITECOIN	CHILIZ	USDC	NANO
BITCOIN	0.001353	-0.000028	0.000073	-0.000092	0.000002	0.000035
CARDANO	-0.000028	0.002491	0.000050	-0.000403	-0.000022	-0.000058
LITECOIN	0.000073	0.000050	0.003233	-0.000050	-0.000011	0.000077
CHILIZ	-0.000092	-0.000403	-0.000050	0.006724	0.000025	-0.000099
USDC	0.000002	-0.000022	-0.000011	0.000025	0.000107	0.000013
NANO	0.000035	-0.000058	0.000077	-0.000099	0.000013	0.003744

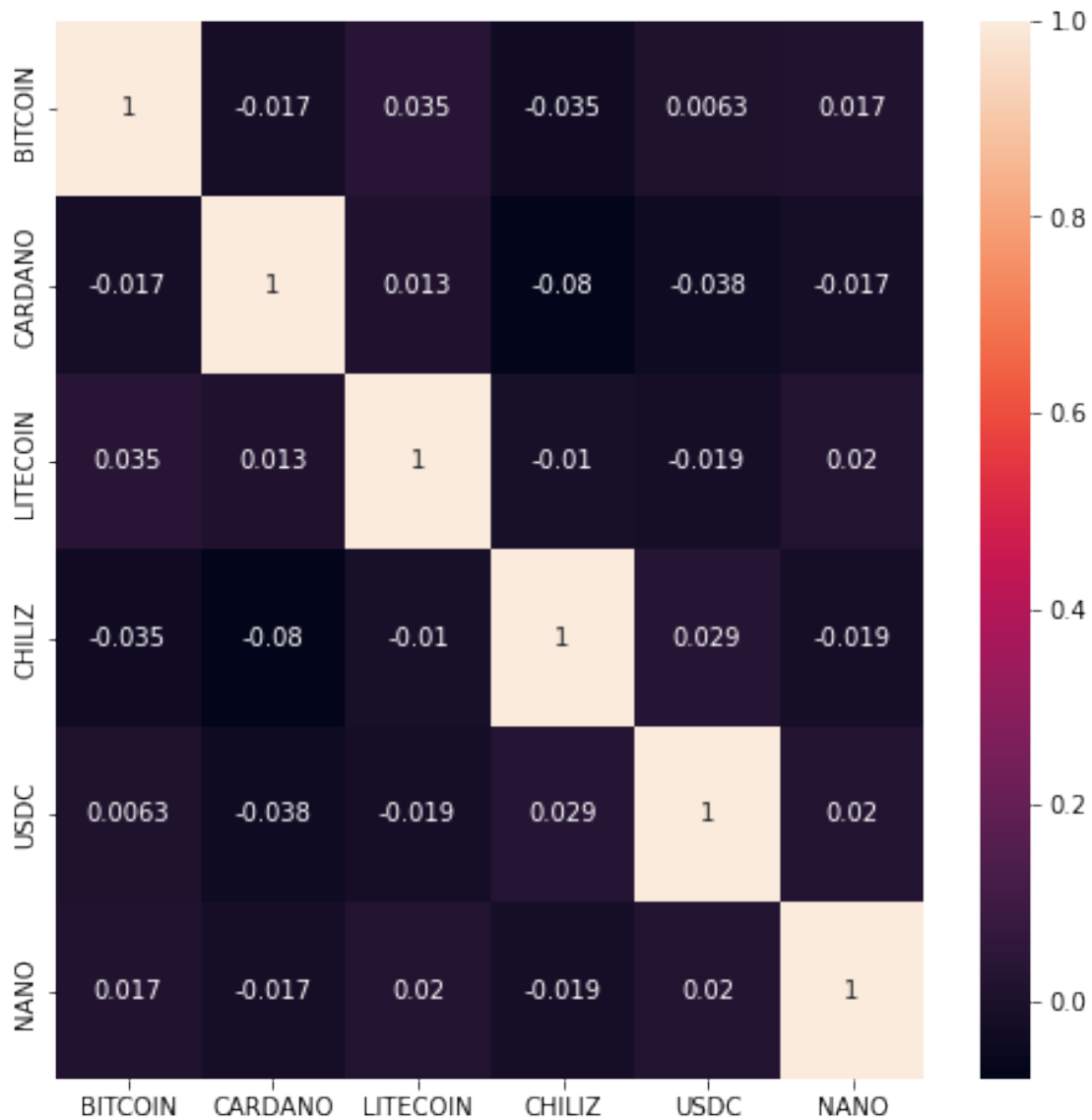
```
[260]: taxas_retorno.corr()
```

```
[260]:
```

	BITCOIN	CARDANO	LITECOIN	CHILIZ	USDC	NANO
BITCOIN	1.000000	-0.016890	0.034717	-0.034754	0.006280	0.017157
CARDANO	-0.016890	1.000000	0.012622	-0.079718	-0.038034	-0.017095
LITECOIN	0.034717	0.012622	1.000000	-0.010169	-0.019006	0.019868
CHILIZ	-0.034754	-0.079718	-0.010169	1.000000	0.028713	-0.018692
USDC	0.006280	-0.038034	-0.019006	0.028713	1.000000	0.019841
NANO	0.017157	-0.017095	0.019868	-0.018692	0.019841	1.000000

O mapa de calor da Taxa de Retorno de Ações, apresenta, graficamente, a relação de correlação entre um ativo e outro.

```
[261]: plt.figure(figsize=(8,8))
sns.heatmap(taxas_retorno.corr(), annot=True);
```

Montando uma Carteira de Ativos

```
[262]: taxas_retorno_date["CARTEIRA"] = (taxas_retorno_date["BITCOIN"] +
↳taxas_retorno_date["CARDANO"] +
taxas_retorno_date["LITECOIN"] +
↳taxas_retorno_date["CHILIZ"] +
taxas_retorno_date["USDC"])/5
taxas_retorno_date
```

```
[262]:
```

	Date	BITCOIN	CARDANO	LITECOIN	CHILIZ	USDC	NANO	\
0	2015-01-01	NaN	NaN	NaN	NaN	NaN	NaN	
1	2015-01-02	0.046344	-0.036284	0.021488	-0.028988	-0.010431	0.035313	

2	2015-01-03	-0.129333	-0.003213	-0.255723	-0.038985	0.014628	0.027284
3	2015-01-04	0.047013	-0.005555	-0.076648	-0.004598	-0.005114	0.018498
4	2015-01-05	-0.074561	-0.021081	0.056493	-0.010966	-0.007615	0.001374
...
2809	2022-07-20	-0.016387	NaN	0.007084	NaN	NaN	NaN
2810	2022-07-21	0.013571	NaN	-0.006521	NaN	NaN	NaN
2811	2022-07-22	-0.005468	NaN	NaN	NaN	NaN	NaN
2812	2022-07-23	-0.015202	NaN	NaN	NaN	NaN	NaN
2813	2022-07-24	-0.024201	NaN	NaN	NaN	NaN	NaN

	CARTEIRA
0	NaN
1	-0.001574
2	-0.082525
3	-0.008980
4	-0.011546
...	...
2809	NaN
2810	NaN
2811	NaN
2812	NaN
2813	NaN

[2814 rows x 8 columns]

```
[263]: taxas_retorno_port = taxas_retorno_date.filter(["Date", "CARTEIRA", "NANO"])
taxas_retorno_port
```

```
[263]:
```

	Date	CARTEIRA	NANO
0	2015-01-01	NaN	NaN
1	2015-01-02	-0.001574	0.035313
2	2015-01-03	-0.082525	0.027284
3	2015-01-04	-0.008980	0.018498
4	2015-01-05	-0.011546	0.001374
...
2809	2022-07-20	NaN	NaN
2810	2022-07-21	NaN	NaN
2811	2022-07-22	NaN	NaN
2812	2022-07-23	NaN	NaN
2813	2022-07-24	NaN	NaN

[2814 rows x 3 columns]

A partir dos ativos BITCOIN(BTC), CARDANO (ADA), LITECOIN(LTC), CHILIZ(CHZ), USDC montou-se uma carteira de lucros. Esta carteira é comparada com o retorno do ativo Nano, que comparativamente mostrou-se também lucrativo, com maiores quedas no valor da Taxa de Retorno de Ações.

```
[264]: figura = px.line(title = 'Comparação de retorno Carteira x Ativo Nano')
for i in taxas_retorno_port.columns[1:]:
    figura.add_scatter(x = taxas_retorno_port["Date"], y = taxas_retorno_port[i],
        name = i)
figura.add_hline(y = taxas_retorno_port['CARTEIRA'].mean(), line_color="green",
    line_dash="dot", )
figura.show()
```

```
[266]: taxas_retorno_port_corr = taxas_retorno_date.filter(["CARTEIRA", "NANO"])
taxas_retorno_port_corr
```

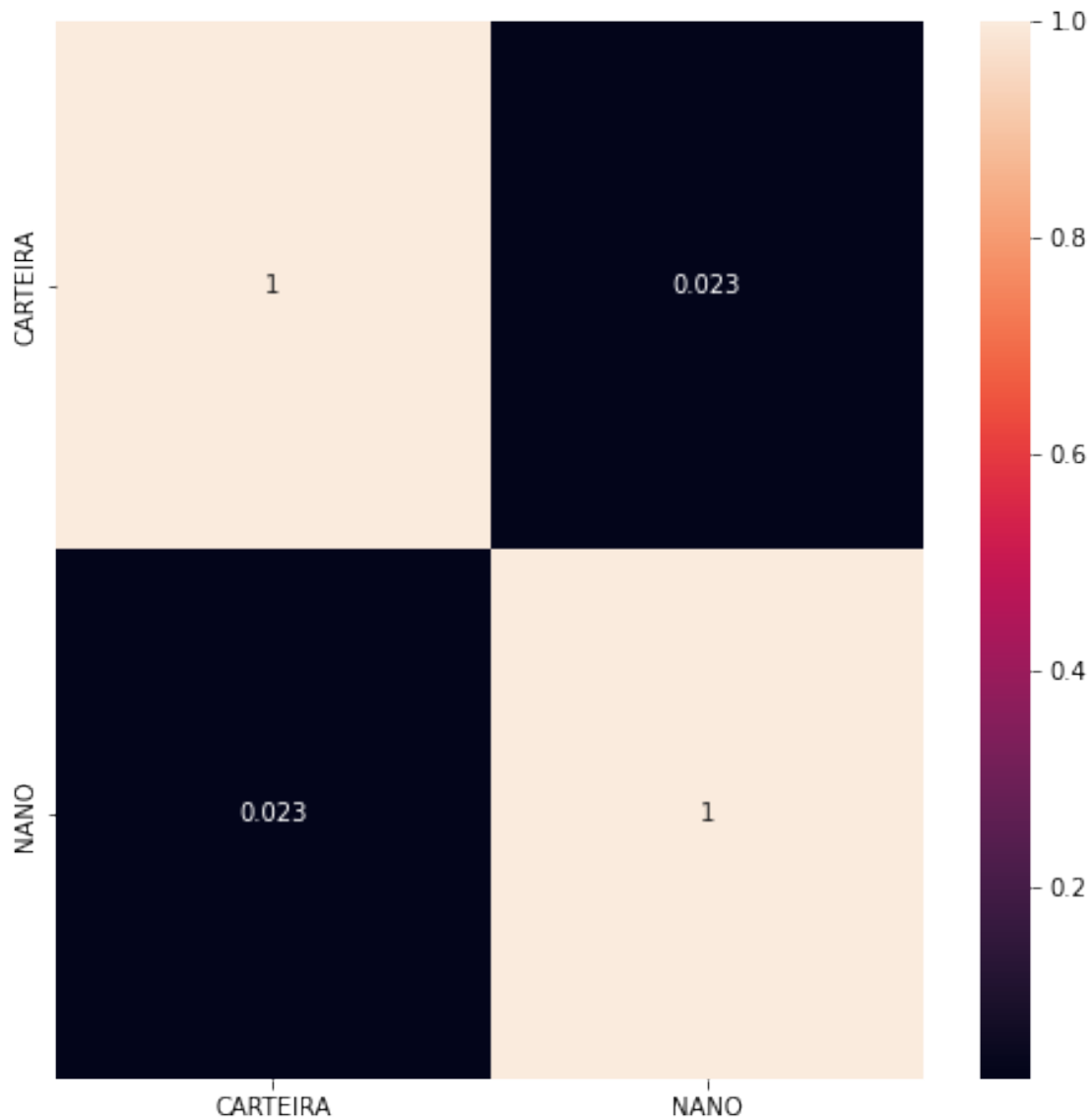
```
[266]:
```

	CARTEIRA	NANO
0	NaN	NaN
1	-0.001574	0.035313
2	-0.082525	0.027284
3	-0.008980	0.018498
4	-0.011546	0.001374
...
2809	NaN	NaN
2810	NaN	NaN
2811	NaN	NaN
2812	NaN	NaN
2813	NaN	NaN

[2814 rows x 2 columns]

Como esperado e demonstrado pelo gráfico acima, os valores de retorno da carteira montada e do ativo Nano são muito próximos uns dos outros. Em especial por se tratar de criptoativos com quedas e altas bastante acentuadas.

```
[267]: plt.figure(figsize=(8,8))
sns.heatmap(taxas_retorno_port_corr.corr(), annot=True);
```



```
[268]: ##### Alocação Aleatória de Ativos - Portfólio Markowitz
```

```
[269]: acoes_port = acoes_df.copy()
acoes_port.drop(labels = ["NANO"], axis=1, inplace=True)
acoes_port
```

```
[269]:
```

	BITCOIN	CARDANO	LITECOIN	CHILIZ	USDC	Date
0	880.100050	12.89956	8.21562	0.07000	5.44500	2015-01-01
1	921.847150	12.43990	8.39407	0.06800	5.38850	2015-01-02
2	810.010000	12.40000	6.50000	0.06540	5.46790	2015-01-03
3	848.999990	12.33131	6.02040	0.06510	5.44001	2015-01-04
4	788.000000	12.07408	6.37030	0.06439	5.39874	2015-01-05

...
2809	125000.000000	NaN	313.65668	NaN	NaN	2022-07-20
2810	126708.000000	NaN	311.61799	NaN	NaN	2022-07-21
2811	126017.000000	NaN	NaN	NaN	NaN	2022-07-22
2812	124115.769950	NaN	NaN	NaN	NaN	2022-07-23
2813	121148.148952	NaN	NaN	NaN	NaN	2022-07-24

[2814 rows x 6 columns]

```
[277]: def alocacao_ativos(dataset, dinheiro_total, seed = 0, melhores_pesos = []):
    dataset = dataset.copy()

    if seed != 0:
        np.random.seed(seed)

    if len(melhores_pesos) > 0:
        pesos = melhores_pesos
    else:
        pesos = np.random.random(len(dataset.columns) - 1)
        pesos = pesos / pesos.sum()

    colunas = dataset.columns[:-1]
    for i in colunas:
        dataset[i] = (dataset[i] / dataset[i][0])

    for i, acao in enumerate(dataset.columns[:-1]):
        dataset[acao] = dataset[acao] * pesos[i] * dinheiro_total

    dataset['soma valor'] = dataset.sum(axis = 1)

    datas = dataset['Date']
    dataset.drop(labels = ['Date'], axis = 1, inplace = True)
    dataset['taxa retorno'] = 0.0

    for i in range(1, len(dataset)):
        dataset['taxa retorno'][i] = np.log(dataset['soma valor'][i] /
        dataset['soma valor'][i - 1]) * 100

    acoes_pesos = pd.DataFrame(data = {'Ações': colunas, 'Pesos': pesos})

    return dataset, datas, acoes_pesos, dataset.loc[len(dataset) - 1]['soma_
    valor']
```

Montando uma carteira com os ativos BITCOIN(BTC), CARDANO (ADA), LITECOIN(LTC), CHILIZ(CHZ), USDC e com um valor inicial de investimento de R\$ 10.000,00 e com valores aleatórios de retorno.

```
[278]: dataset, datas, acoes_pesos, soma_valor = alocacao_ativos(acoes_port, 10000, 10)
```

```
/tmp/ipykernel_22615/3471116991.py:20: FutureWarning:
```

```
Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None')
is deprecated; in a future version this will raise TypeError.  Select only valid
columns before calling the reduction.
```

```
[279]: dataset
```

```
[279]:
```

	BITCOIN	CARDANO	LITECOIN	CHILIZ	USDC \
0	2885.564869	77.634505	2370.522690	2801.328081	1864.949855
1	3022.440177	74.868094	2422.012383	2721.290136	1845.598217
2	2655.762149	74.627961	1875.500265	2617.240807	1872.793262
3	2783.597780	74.214558	1737.117199	2605.235115	1863.240746
4	2583.598441	72.666449	1838.076821	2576.821645	1849.105488
...
2809	409834.778060	NaN	90502.028680	NaN	NaN
2810	415434.760468	NaN	89913.788118	NaN	NaN
2811	413169.193815	NaN	NaN	NaN	NaN
2812	406935.672250	NaN	NaN	NaN	NaN
2813	397205.797905	NaN	NaN	NaN	NaN

	soma valor	taxa retorno
0	10000.000000	0.000000
1	10086.209007	0.858395
2	9095.924443	-10.334260
3	9063.405398	-0.358153
4	8920.268843	-1.591884
...
2809	500336.806740	-1.218201
2810	505348.548585	0.996690
2811	413169.193815	-20.139121
2812	406935.672250	-1.520206
2813	397205.797905	-2.420059

```
[2814 rows x 7 columns]
```

```
[280]: acoes_pesos
```

```
[280]:
```

	Ações	Pesos
0	BITCOIN	0.288556
1	CARDANO	0.007763
2	LITECOIN	0.237052
3	CHILIZ	0.280133
4	USDC	0.186495

```
[281]: datas
```

```
[281]: 0      2015-01-01
      1      2015-01-02
      2      2015-01-03
      3      2015-01-04
      4      2015-01-05
      ...
     2809    2022-07-20
     2810    2022-07-21
     2811    2022-07-22
     2812    2022-07-23
     2813    2022-07-24
      Name: Date, Length: 2814, dtype: object
```

```
[282]: soma_valor
```

```
[282]: 397205.7979047329
```

Com o retorno aleatório dos ativos citados na carteira, houveram momentos de até 60% de lucro no valor inicialmente investido. Assim como em 2017 houve uma queda expressiva do investimento em mais de 80%.

```
[283]: figura = px.line(x = datas, y = dataset['taxa retorno'], title = 'Retorno_
      ↳diário do portfólio',
      labels=dict(x="Data", y="Retorno %"))
      figura.add_hline(y = dataset['taxa retorno'].mean(), line_color="red",
      ↳line_dash="dot", )
      figura.show()
```

Ao longo do tempo a carteira de ativos com retornos aleatórios apresentou alta no ativo Bitcoin, com as maiores variações no ativo USDC.

```
[284]: figura = px.line(title = 'Evolução do patrimônio')
      for i in dataset.drop(columns = ['soma valor', 'taxa retorno']).columns:
          figura.add_scatter(x = datas, y = dataset[i], name = i)
      figura.show()
```

Houve um lucro expressivo de mais de R\$ 1.000.000,00 na carteira montada. O comportamento randômico é quase idêntico ao observado no mercado de criptoativos. Extremamente volátil e com várias altas e baixas.

```
[285]: figura = px.line(x = datas, y = dataset['soma valor'],
      title = 'Evolução do patrimônio da Carteira',
      labels=dict(x="Data", y="Valor R$"))
      figura.add_hline(y = dataset['soma valor'].mean(),
      line_color="green", line_dash="dot", )
      figura.show()
```

Mais estatísticas sobre o portfólio aleatório

```
[286]: # Retorno
dataset.loc[len(dataset) - 1]['soma valor'] / dataset.loc[0]['soma valor'] - 1
```

[286]: 38.720579790473295

```
[287]: # Desvio-Padrão
dataset['taxa retorno'].std()
```

[287]: 4.712250446018992

```
[288]: # Sharpe Ratio
(dataset['taxa retorno'].mean() / dataset['taxa retorno'].std())
```

[288]: 0.027766166684584117

```
[289]: dinheiro_total = 10000
soma_valor - dinheiro_total
```

[289]: 387205.7979047329

0.3 Simulação da Fronteira Eficiente

```
[290]: acoes_port
```

```
[290]:
```

	BITCOIN	CARDANO	LITECOIN	CHILIZ	USDC	Date
0	880.100050	12.89956	8.21562	0.07000	5.44500	2015-01-01
1	921.847150	12.43990	8.39407	0.06800	5.38850	2015-01-02
2	810.010000	12.40000	6.50000	0.06540	5.46790	2015-01-03
3	848.999990	12.33131	6.02040	0.06510	5.44001	2015-01-04
4	788.000000	12.07408	6.37030	0.06439	5.39874	2015-01-05
...
2809	125000.000000	NaN	313.65668	NaN	NaN	2022-07-20
2810	126708.000000	NaN	311.61799	NaN	NaN	2022-07-21
2811	126017.000000	NaN	NaN	NaN	NaN	2022-07-22
2812	124115.769950	NaN	NaN	NaN	NaN	2022-07-23
2813	121148.148952	NaN	NaN	NaN	NaN	2022-07-24

[2814 rows x 6 columns]

```
[291]: log_ret = acoes_port.copy()
log_ret.drop(labels = ["Date"], axis = 1, inplace = True)
log_ret = np.log(log_ret/log_ret.shift(1))
log_ret
```

```
[291]:
```

	BITCOIN	CARDANO	LITECOIN	CHILIZ	USDC
0	NaN	NaN	NaN	NaN	NaN


```

1      0.046344 -0.036284  0.021488 -0.028988 -0.010431
2     -0.129333 -0.003213 -0.255723 -0.038985  0.014628
3      0.047013 -0.005555 -0.076648 -0.004598 -0.005114
4     -0.074561 -0.021081  0.056493 -0.010966 -0.007615
...
2809 -0.016387      NaN  0.007084      NaN      NaN
2810  0.013571      NaN -0.006521      NaN      NaN
2811 -0.005468      NaN      NaN      NaN      NaN
2812 -0.015202      NaN      NaN      NaN      NaN
2813 -0.024201      NaN      NaN      NaN      NaN

```

[2814 rows x 5 columns]

```

[292]: np.random.seed(42)
num_ports = 1000
all_weights = np.zeros((num_ports, len(acoess_port.columns[1:])))
ret_arr = np.zeros(num_ports)
vol_arr = np.zeros(num_ports)
sharpe_arr = np.zeros(num_ports)

for x in range(num_ports):
    # Weights
    weights = np.array(np.random.random(5))
    weights = weights/np.sum(weights)

    # Save weights
    all_weights[x,:] = weights

    # Expected return
    ret_arr[x] = np.sum((log_ret.mean() * weights))

    # Expected volatility
    vol_arr[x] = np.sqrt(np.dot(weights.T, np.dot(log_ret.cov(), weights)))

    # Sharpe Ratio
    sharpe_arr[x] = ret_arr[x]/vol_arr[x]

```

```

[293]: print(f"Max Sharpe Ratio: {sharpe_arr.max()}")
print(f"Local do Max Sharpe Ratio: {sharpe_arr.argmax()}")

```

Max Sharpe Ratio: 0.06260877589164685

Local do Max Sharpe Ratio: 955

```

[294]: # Pesos do Portfólio do Max Sharpe Ratio
print(all_weights[643,:])

```

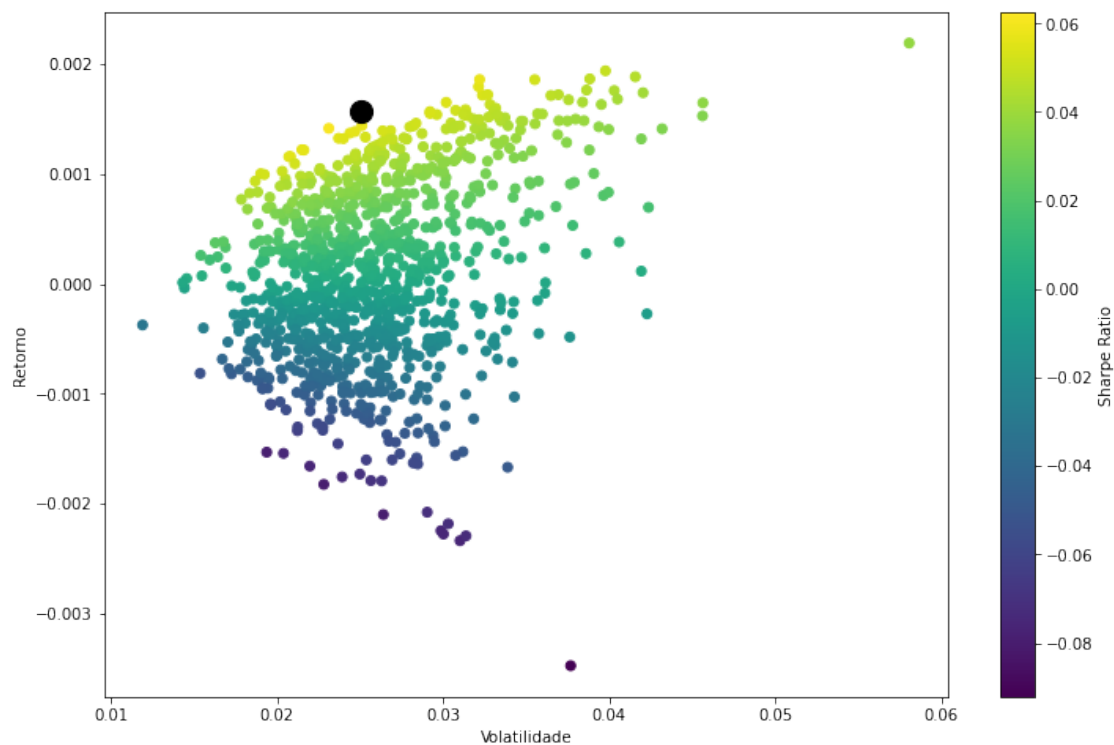
[0.06771511 0.09521168 0.00114756 0.41402801 0.42189764]

```
[295]: # salvando os dados do Max Sharpe Ratio
max_sr_ret = ret_arr[sharpe_arr.argmax()]
max_sr_vol = vol_arr[sharpe_arr.argmax()]
print(max_sr_ret)
print(max_sr_vol)
```

```
0.0015667370679111631
```

```
0.025024240541335906
```

```
[296]: plt.figure(figsize=(12,8))
plt.scatter(vol_arr, ret_arr, c=sharpe_arr, cmap='viridis')
plt.colorbar(label='Sharpe Ratio')
plt.xlabel('Volatilidade')
plt.ylabel('Retorno')
plt.scatter(max_sr_vol, max_sr_ret, c='black', s=200) # black dot
plt.show()
```



Nós podemos ver no gráfico assim o conjunto de portfólios simulados, pois o peso w_i de cada ativo foi simulado e criamos um conjunto de $n = 1000$ carteiras e escolhemos no ponto vermelho a que tem maior **Sharpe Ratio**, que é a razão retorno sobre a volatilidade. Esse dado nos dá uma noção do portfólio ponderado pelo risco.

```
[297]: def get_ret_vol_sr(weights):
    weights = np.array(weights)
    ret = np.sum(log_ret.mean() * weights)
    vol = np.sqrt(np.dot(weights.T, np.dot(log_ret.cov(), weights)))
    sr = ret/vol
    return np.array([ret, vol, sr])

def neg_sharpe(weights):
    # the number 2 is the sharpe ratio index from the get_ret_vol_sr
    return get_ret_vol_sr(weights)[2] * -1

def check_sum(weights):
    #return 0 if sum of the weights is 1
    return np.sum(weights)-1
```

```
[298]: cons = ({'type': 'eq', 'fun': check_sum})
bounds = ((0,1), (0,1), (0,1), (0,1), (0,1))
init_guess = ((0.2),(0.2),(0.2),(0.2),(0.2))
```

```
[299]: op_results = optimize.minimize(neg_sharpe, init_guess, method="SLSQP", bounds=
    ↪ bounds, constraints=cons)
print(op_results)
```

```

    fun: -0.06323003792313457
    jac: array([-9.77870077e-05,  1.72606335e-01, -5.08139841e-04,
7.20449723e-04,
    -1.30431727e-05])
    message: 'Optimization terminated successfully'
    nfev: 79
    nit: 13
    njev: 13
    status: 0
    success: True
    x: array([0.60432927, 0.          , 0.17477836, 0.20332145, 0.01757092])
```

```
[300]: frontier_y = np.linspace(-0.0006, 0.0008, 200)
```

```
[301]: def minimize_volatility(weights):
    return get_ret_vol_sr(weights)[1]
```

```
[302]: frontier_x = []

for possible_return in frontier_y:
    cons = ({'type':'eq', 'fun':check_sum},
            {'type':'eq', 'fun': lambda w: get_ret_vol_sr(w)[0] -
    ↪ possible_return})
```

```

    result = optimize.minimize(minimize_volatility,init_guess,method='SLSQP',
↪ bounds=bounds, constraints=cons)
    frontier_x.append(result['fun'])

```

```

[303]: plt.figure(figsize=(12,8))
plt.scatter(vol_arr, ret_arr, c=sharpe_arr, cmap='viridis')
plt.colorbar(label='Sharpe Ratio')
plt.xlabel('Volatilidade')
plt.ylabel('Retorno')
plt.plot(frontier_x,frontier_y, 'r--', linewidth=3)
plt.scatter(max_sr_vol, max_sr_ret,c='black', s=200)
# plt.savefig('cover.png')
plt.show()

```

