Relatório Final

October 4, 2021

0.0.1 Relatório Estatístico para Dados Financeiros - Tarefa 2

Aluno: Eduardo de Oliveira Castro

Matrícula: 210008164

- Nossa Tarefa Final dar-se-á em elaboração de um portfólio com ao menos 3 ativos, necessita-se também da busca por um indicador de mercado (como o IBOVESPA para o caso brasileiro), além de um ativo livre de risco (SELIC).
- Nessa Tarefa vocês farão um Relatório de 10 a 15 páginas (o número máximo, pode ser extrapolado se houver necessidade) contendo todos os tópicos ensinados em sala de aula virtual durante esse semestre.
- Esse relatório será uma aplicação dos conceitos apresentados. Lembrem que vocês devem encontrar graficamente a distribuição dos ativos e carteira, analisá-los a luz da teoria; realizar as medidas descritivas, fazendo conexão com a teoria ensinada em aula, realizar aplicações de testes de média (ANOVA) e por fim fazer análise das regressões lineares. Gerar a Fronteira Eficiente de Markowitz e o Modelo CAPM de Sharpe e Lintner é opcional (mas será bem interessante a aplicação destes conceitos).
- Não se esqueçam de fazer uma avaliação econômica, utilizando as resposta que o relatório estatístico te fornece. Lembrem de fornecer os códigos e os passos realizados (para quem usou aplicações de point clik) das analises realizadas.

0.0.2 Importação das bibliotecas que serão utilizadas enquanto ferramentas no desenvolvimento do projeto

[1]:
```
import pandas as pd
from pandas_datareader import data
import numpy as np
import math
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import plotly.express as px
import seaborn as sns
from scipy import stats
from scipy.optimize import minimize
```

0.0.3 Construindo um Data Frame com dados de ações puxados do Yahoo Finance e dados da SELIC puxados do Banco do Central

A escolha pelo período de 2012 tem como justificativa o momento temporal em que todas as ações escolhidas já estão listadas simultaneamente.

[2]:
```
# B2W, Lojas Americanas, Magazine Luiza, Amazon e Índice Bovespa
acoes = ["AMER3.SA", "LAME4.SA", "MGLU3.SA", "AMZO34.SA", "^BVSP"]

# Criação de um dataframe
acoes_df = pd.DataFrame()

# For para popular o Dataframe com os dados de fechamento da bolsa de valores␣
 ↪de cada dia, desde 2012, coletados do Yahoo
for acao in acoes:
    acoes_df[acao] = data.DataReader(acao, data_source="yahoo",␣
 ↪start="2012-02-01", end="2021-09-30")['Close']

# Criação de um CSV para salvar esses dados
acoes_df.to_csv("carteira_tarefa_final.csv")

json_url = "https://api.bcb.gov.br/dados/serie/bcdata.sgs.11/dados?formato=json"
selic_df = pd.read_json(json_url)
```

0.0.4 Impressão da tabela de ações coletada

[3]:
```
acoes_df
```

[3]:

Date	AMER3.SA	LAME4.SA	MGLU3.SA	AMZO34.SA	^BVSP
2012-02-01	9.712035	8.871445	0.321875	1.971592	64567.0
2012-02-02	9.408254	8.819475	0.328125	1.959872	64593.0
2012-02-03	9.569079	8.887037	0.318125	1.959872	65217.0
2012-02-06	9.381450	8.964993	0.304687	1.959872	65224.0
2012-02-07	9.560145	9.094921	0.310312	1.959872	65917.0
…	…	…	…	…	…
2021-09-24	34.240002	5.180000	15.630000	116.930000	113283.0
2021-09-27	33.400002	5.080000	15.010000	116.930000	113583.0
2021-09-28	31.330000	4.850000	14.180000	114.279999	110124.0
2021-09-29	31.190001	4.870000	13.940000	114.010002	111107.0
2021-09-30	30.920000	4.820000	14.340000	114.500000	110979.0

[2393 rows x 5 columns]

0.0.5 Impressão da tabela da selic coletada

[4]:
```
selic_df
```

[4]:
```
            data     valor
0     04/06/1986  0.065041
1     05/06/1986  0.067397
2     06/06/1986  0.066740
3     09/06/1986  0.068247
4     10/06/1986  0.067041
…          …       …
8851  27/09/2021  0.023687
8852  28/09/2021  0.023687
8853  29/09/2021  0.023687
8854  30/09/2021  0.023687
8855  01/10/2021  0.023687

[8856 rows x 2 columns]
```

0.0.6 Ajuste no Dataframe com a SELIC e junção com a tabela contendo os indicadores de ações e IBOV:

[5]:
```
selic_df['data'] = pd.to_datetime(selic_df.data)
selic_df = selic_df.rename({'data': 'Date', 'valor': 'SELIC'}, axis=1)

indicadores_df = pd.merge(acoes_df, selic_df, on=['Date'])
```

0.0.7 Impressão da tabela de ações e SELIC coletada e que será a base deste trabalho

[6]:
```
indicadores_df
```

[6]:
```
           Date   AMER3.SA  LAME4.SA   MGLU3.SA   AMZO34.SA     ^BVSP  \
0    2012-02-01   9.712035  8.871445   0.321875    1.971592   64567.0
1    2012-02-02   9.408254  8.819475   0.328125    1.959872   64593.0
2    2012-02-03   9.569079  8.887037   0.318125    1.959872   65217.0
3    2012-02-07   9.560145  9.094921   0.310312    1.959872   65917.0
4    2012-02-08   9.649492  9.037753   0.308437    1.959872   65831.0
…         …         …       …          …           …          …
2098 2021-09-24  34.240002  5.180000  15.630000  116.930000  113283.0
2099 2021-09-27  33.400002  5.080000  15.010000  116.930000  113583.0
2100 2021-09-28  31.330000  4.850000  14.180000  114.279999  110124.0
2101 2021-09-29  31.190001  4.870000  13.940000  114.010002  111107.0
2102 2021-09-30  30.920000  4.820000  14.340000  114.500000  110979.0

         SELIC
0     0.041063
1     0.039270
```

```
2     0.039270
3     0.031976
4     0.030140
…        …
2098  0.023687
2099  0.023687
2100  0.023687
2101  0.023687
2102  0.023687

[2103 rows x 7 columns]
```

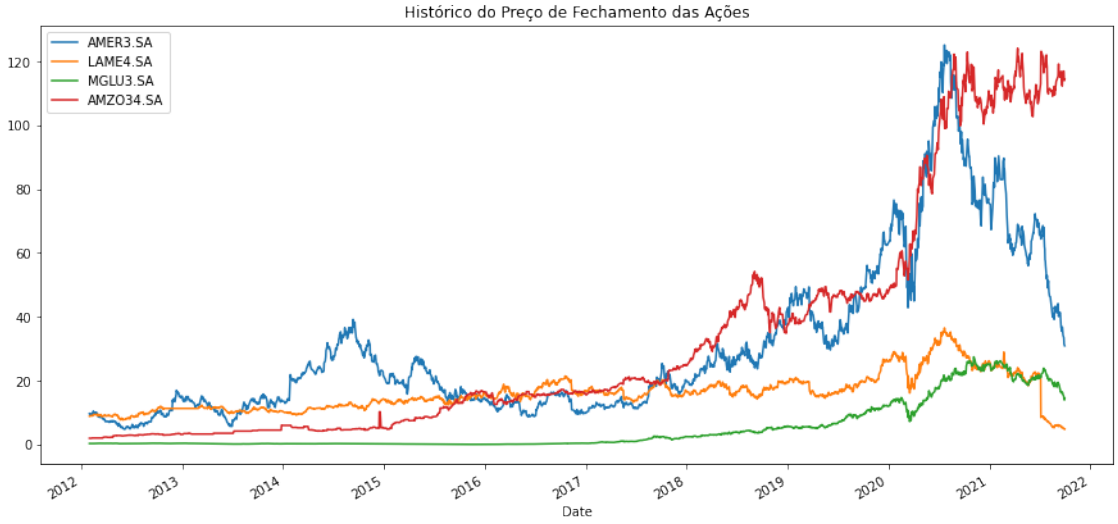
0.0.8 Imprime descritivo estatístico simples do dataframe gerado, exibindo tendência, dispersão, valores máximos e mínimos, shape, etc.

[7]:
```
indicadores_df.describe()
```

[7]:
```
          AMER3.SA     LAME4.SA     MGLU3.SA    AMZO34.SA          ^BVSP  \
count  2103.000000  2103.000000  2103.000000  2103.000000    2101.000000
mean     30.345807    16.027671     4.830084    31.941256   72365.971918
std      25.217151     5.693645     7.375434    35.063786   23515.130191
min       4.780072     4.820000     0.030585     1.959872   37497.000000
25%      13.010675    11.493977     0.244531     4.829171   53739.000000
50%      20.984024    15.347450     0.401562    16.527388   62699.000000
75%      37.431231    18.795643     5.572656    45.973375   91727.000000
max     125.126945    36.517979    27.450001   124.099998  130776.000000

             SELIC
count  2103.000000
mean      0.032231
std       0.013613
min       0.007469
25%       0.024620
50%       0.030177
75%       0.041957
max       0.052531
```

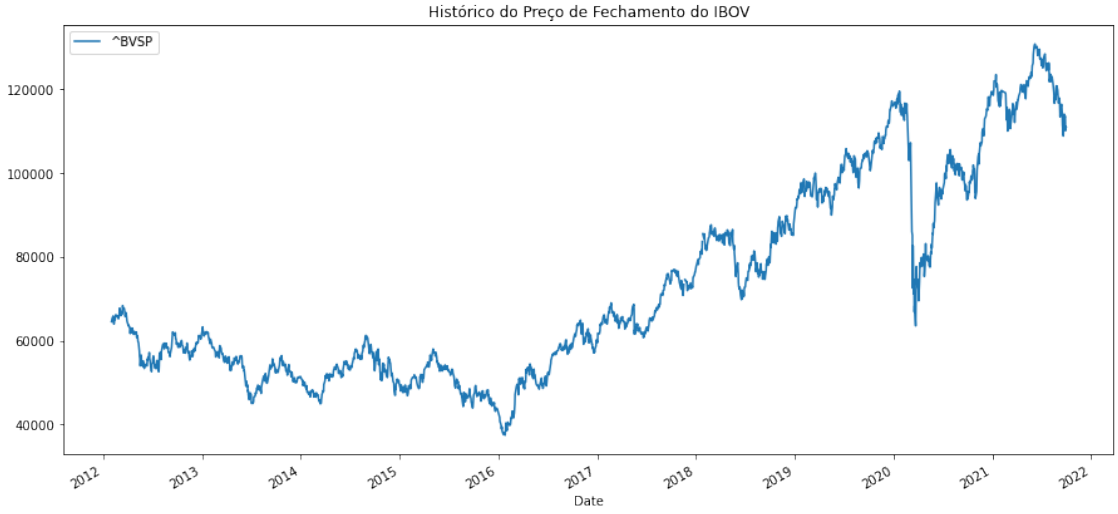
0.0.9 Visualização do histórico de preço de fechamento das ações

[8]:
```
indicadores_df.drop(['^BVSP', 'SELIC'], axis=1).plot(x="Date", figsize =␣
↪(15,7), title='Histórico do Preço de Fechamento das Ações');
```



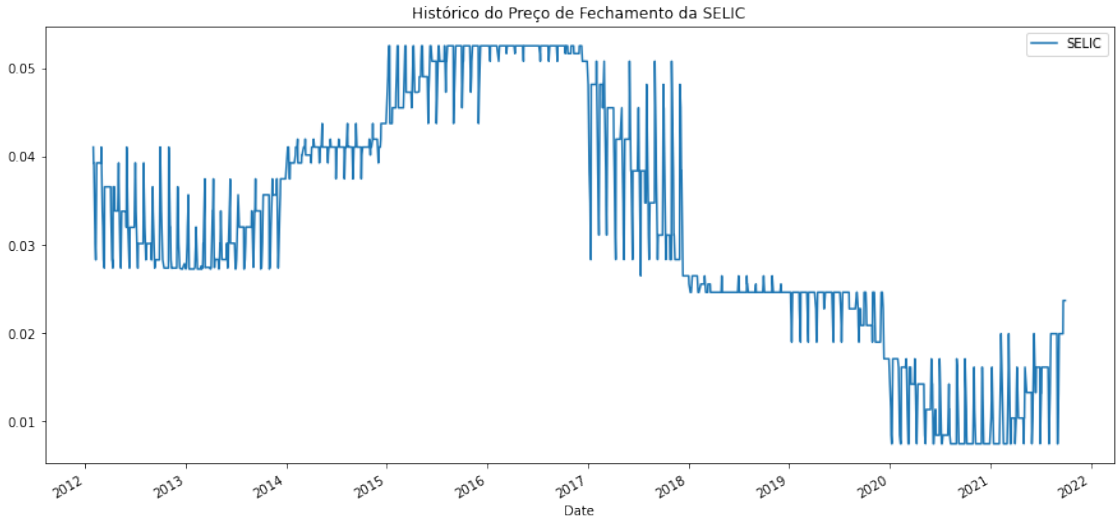
0.0.10 Visualização do histórico de preço de fechamento do valor de fechamento do índice BOVESPA

[9]:
```
indicadores_df.drop(['AMER3.SA', 'LAME4.SA', 'MGLU3.SA', 'AMZO34.SA', 'SELIC'],␣
↪axis=1).plot(x='Date', figsize = (15,7), title='Histórico do Preço de␣
↪Fechamento do IBOV');
```



0.0.11 Visualização do histórico de preço de fechamento do valor de fechamento do índice SELIC

[10]:
```
indicadores_df.drop(['AMER3.SA', 'LAME4.SA', 'MGLU3.SA', 'AMZO34.SA', '^BVSP'],␣
↪axis=1).plot(x='Date', figsize = (15,7), title='Histórico do Preço de␣
↪Fechamento da SELIC');
```



0.0.12 Transformação e Normalização dos dados

Dado que estamos lidando com dados de diferentes naturezas e escalas, alterar os valores das colunas numéricas no conjunto de dados para uma escala comum, ou seja, dentro da mesma ordem de grandeza e de tal forma que não distorça as diferenças nos intervalos de valores, é necessário e tal ato é conhecido enquanto *normalização*. Neste caso, por meio da normalização, transformamos nossas variáveis de acordo com o primeiro valor que, neste caso, será sempre utilizado como referência. Dessa forma, preservamos a variação que é o valor que queremos visualizar. Para este cálculo, é feita a divisão conforme a seguinte fórmula:

$$ValorNormalizado = a_1/a_0$$

0.0.13 Visualização do novo Data Frame normalizado

Em relação aos gráficos, comparando com os gráficos gerados anteriormente com os valores antes da normalização é possível verificarmos que os gráficos mantém as mesmas variações mas, dessa vez, adaptado para uma nova escala.

[11]:
```
indicadores_df_normalizado = indicadores_df.copy()

for iterador in indicadores_df_normalizado.drop(["Date"], axis=1):
    indicadores_df_normalizado[iterador] = indicadores_df_normalizado[iterador]␣
↪/ indicadores_df_normalizado[iterador][0]
```

```
indicadores_df_normalizado
```

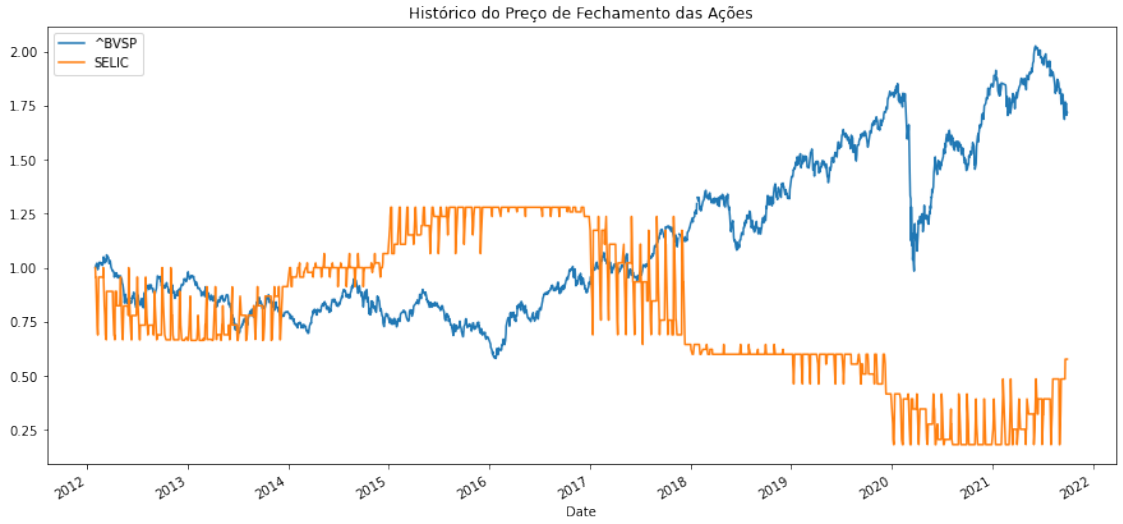
[11]:

	Date	AMER3.SA	LAME4.SA	MGLU3.SA	AMZO34.SA	^BVSP	SELIC
0	2012-02-01	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
1	2012-02-02	0.968721	0.994142	1.019417	0.994056	1.000403	0.956335
2	2012-02-03	0.985281	1.001758	0.988350	0.994056	1.010067	0.956335
3	2012-02-07	0.984361	1.025191	0.964076	0.994056	1.020909	0.778706
4	2012-02-08	0.993560	1.018746	0.958251	0.994056	1.019577	0.733994
…	…	…	…	…	…	…	…
2098	2021-09-24	3.525523	0.583896	48.559223	59.307404	1.754503	0.576845
2099	2021-09-27	3.439032	0.572624	46.633010	59.307404	1.759149	0.576845
2100	2021-09-28	3.225894	0.546698	44.054369	57.963312	1.705577	0.576845
2101	2021-09-29	3.211479	0.548952	43.308736	57.826368	1.720802	0.576845
2102	2021-09-30	3.183679	0.543316	44.551456	58.074897	1.718819	0.576845

[2103 rows x 7 columns]

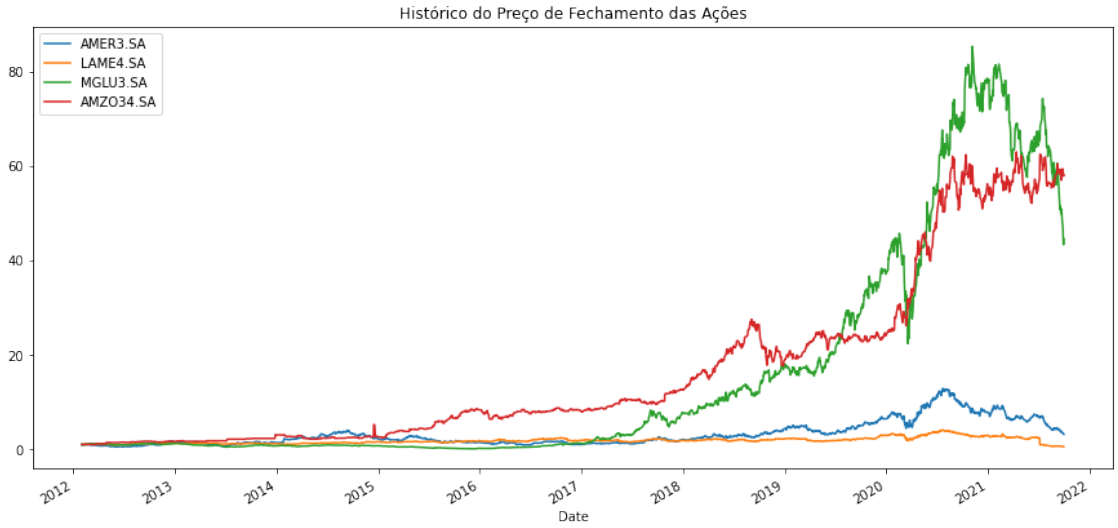
0.0.14 Visualização do histórico de preço de fechamento do valor de fechamento do índice BOVESPA e do índice SELIC normalizados

[12]:
```
indicadores_df_normalizado.drop(["AMER3.SA", "LAME4.SA", "MGLU3.SA", "AMZO34.
↪SA"], axis=1).plot(x="Date", figsize = (15,7), title='Histórico do Preço de␣
↪Fechamento das Ações');
```



0.0.15 Visualização do histórico de preço de fechamento do valor de fechamento das ações normalizado

[13]:
```
indicadores_df_normalizado.drop(["^BVSP", "SELIC"], axis=1).plot(x="Date",␣
↪figsize = (15,7), title='Histórico do Preço de Fechamento das Ações');
```



0.0.16 Histórico do Índice BOVESPA em comparação com a SELIC

Dessa vez, em um scatter.

[14]:
```
figura = px.line(title = "Histórico do Índice BOVESPA e da SELIC")

for iterador in indicadores_df_normalizado.drop(["Date", "AMER3.SA", "LAME4.
↪SA", "MGLU3.SA", "AMZO34.SA"], axis=1):
    figura.add_scatter(x = indicadores_df_normalizado["Date"], y =␣
↪indicadores_df_normalizado[iterador], name = iterador)

figura.show()
```

0.0.17 Histórico das ações em comparação com a SELIC

Dessa vez, em um scatter.

[15]:
```
figura = px.line(title = "Histórico do preço das ações e da SELIC")

for iterador in indicadores_df_normalizado.drop(["Date", "^BVSP"], axis=1):
    figura.add_scatter(x = indicadores_df_normalizado["Date"], y =␣
↪indicadores_df_normalizado[iterador], name = iterador)

figura.show()
```

0.0.18 Taxa de Retorno Simples

Como pode ser observado, o pacote de ações escolhido parece ter se valorizado muito mais do que a SELIC que, como esperado, manteve-se estável. Mas quanto pode-se dizer que teria sido ganho caso um pessoal qualquer tivesse investido em cada um dos papéis no dia 01 de fevereiro de 2012 e vendido em 30 de setembro de 2021? Para tanto, aplicamos a seguinte fórmula:

$$\mathbb{E}[R] = \mathbb{E}\left[\frac{P_t - P_0}{P_0}\right]$$

Cálculo da taxa de retorno de todos os ativos com o objetivo dee identificar qual teve o maior e qual teve o menor retorno:

[16]:
```
taxa_de_retorno_simples = ((indicadores_df["AMER3.SA"][len(indicadores_df)-1] -␣
↪indicadores_df["AMER3.SA"][0])/indicadores_df["AMER3.SA"][0])*100

maior_taxa_de_retorno_simples_num = taxa_de_retorno_simples
menor_taxa_de_retorno_simples_num = taxa_de_retorno_simples
maior_taxa_de_retorno_simples_nome = "AMER3.SA"
menor_taxa_de_retorno_simples_nome = "AMER3.SA"

taxa_de_retorno_simples = (((indicadores_df["LAME4.SA"][len(indicadores_df)-1]␣
↪- indicadores_df["LAME4.SA"][0])/indicadores_df["LAME4.SA"][0])*100)

if taxa_de_retorno_simples > maior_taxa_de_retorno_simples_num:
    maior_taxa_de_retorno_simples_num = taxa_de_retorno_simples
    maior_taxa_de_retorno_simples_nome = "LAME4.SA"

elif taxa_de_retorno_simples < menor_taxa_de_retorno_simples_num:
    menor_taxa_de_retorno_simples_num = taxa_de_retorno_simples
    menor_taxa_de_retorno_simples_nome = "LAME4.SA"

taxa_de_retorno_simples = ((indicadores_df["MGLU3.SA"][len(indicadores_df)-1] -␣
↪indicadores_df["MGLU3.SA"][0])/indicadores_df["MGLU3.SA"][0])*100

if taxa_de_retorno_simples > maior_taxa_de_retorno_simples_num:
    maior_taxa_de_retorno_simples_num = taxa_de_retorno_simples
    maior_taxa_de_retorno_simples_nome = "MGLU3.SA"

elif taxa_de_retorno_simples < menor_taxa_de_retorno_simples_num:
    menor_taxa_de_retorno_simples_num = taxa_de_retorno_simples
    menor_taxa_de_retorno_simples_nome = "MGLU3.SA"

taxa_de_retorno_simples = ((indicadores_df["AMZO34.SA"][len(indicadores_df)-1]␣
↪- indicadores_df["AMZO34.SA"][0])/indicadores_df["AMZO34.SA"][0])*100

if taxa_de_retorno_simples > maior_taxa_de_retorno_simples_num:
    maior_taxa_de_retorno_simples_num = taxa_de_retorno_simples
```

```
    maior_taxa_de_retorno_simples_nome = "AMZO34.SA"

elif taxa_de_retorno_simples < menor_taxa_de_retorno_simples_num:
    menor_taxa_de_retorno_simples_num = taxa_de_retorno_simples
    menor_taxa_de_retorno_simples_nome = "AMZO34.SA"
```

Dessa forma, temos que o ativo com a maior taxa de retorno é a Amazon (AMZO34.SA), com o seguinte valor:

[17]:
```
print(str(maior_taxa_de_retorno_simples_nome) + ": " +␣
↪str(maior_taxa_de_retorno_simples_num) + str("%"))
```

```
AMZO34.SA: 5707.489730884353%
```

Portanto, se uma pessoa qualquer tivesse investido 1000 reais em papéis da AMZO34.SA em 01 de fevereiro de 2012, não tivesse movimentado nada dessas ações e tivesse vendido todos os seus ativos em 30 de setembro de 2021, essa pessoa teria ganho um total de R$ 57.074,89, sem descontar os impostos.

De forma análoga, o ativo com a menor taxa de retorno foi a Lojas Americanas (LAME4.SA), com o seguinte valor:

[18]:
```
print(str(menor_taxa_de_retorno_simples_nome) + ": " +␣
↪str(menor_taxa_de_retorno_simples_num))
```

```
LAME4.SA: -45.66837382761236
```

Portanto, se uma pessoa qualquer tivesse investido 1000 reais em papéis da LAME4.SA em 01 de fevereiro de 2012, não tivesse movimentado nada dessas ações e tivesse vendido todos os seus ativos em 30 de setembro de 2021, essa pessoa teria perdido um total de R$ 456,68.

Para efeito de comparação, 1000 reais investidos em algum papel atrelado à SELIC, por sua vez, teria dado uma taxa de retorno ajustada de 117,665% e um lucro de aproximadamente 1.176,65 reais. O próprio índice BOVESPA, nesse mesmo período e caso a pessoa tivesse investido em um fundo com retorno atrelado ao índice, teria apresentado a seguinte taxa de retorno e rendido um total de 718,81 reais:

[19]:
```
taxa_de_retorno_simples = ((indicadores_df["^BVSP"][len(indicadores_df)-1] -␣
↪indicadores_df["^BVSP"][0])/indicadores_df["^BVSP"][0])*100

print("^BVSP: " + str(taxa_de_retorno_simples) + "%")
```

```
^BVSP: 71.88192110520855%
```

0.0.19 Outra métrica importante na análise dos ativos envolvee o Preço Médio, medida que representa o valor do somatório do preço de todas as ações dividido pela quantidade de elementos somados, conforme expressão a seguir:

$$\mathbb{E}[P] = \sum_{i=1}^{t} \frac{P_i}{t}$$

[20]:
```
valor_medio = indicadores_df["AMER3.SA"].sum()/len(indicadores_df["AMER3.SA"])

maior_valor_medio_num = valor_medio
menor_valor_medio_num = valor_medio
maior_valor_medio_nome = "AMER3.SA"
menor_valor_medio_nome = "AMER3.SA"

valor_medio = indicadores_df["LAME4.SA"].sum()/len(indicadores_df["LAME4.SA"])

if valor_medio > maior_valor_medio_num:
    maior_valor_medio_num = valor_medio
    maior_valor_medio_nome = "LAME4.SA"

elif valor_medio < menor_valor_medio_num:
    menor_valor_medio_num = valor_medio
    menor_valor_medio_nome = "LAME4.SA"

valor_medio = indicadores_df["MGLU3.SA"].sum()/len(indicadores_df["MGLU3.SA"])

if valor_medio > maior_valor_medio_num:
    maior_valor_medio_num = valor_medio
    maior_valor_medio_nome = "MGLU3.SA"

elif valor_medio < menor_valor_medio_num:
    menor_valor_medio_num = valor_medio
    menor_valor_medio_nome = "MGLU3.SA"

valor_medio = indicadores_df["AMZO34.SA"].sum()/len(indicadores_df["AMZO34.SA"])

if valor_medio > maior_valor_medio_num:
    maior_valor_medio_num = valor_medio
    maior_valor_medio_nome = "AMZO34.SA"

elif valor_medio < menor_valor_medio_num:
    menor_valor_medio_num = valor_medio
    menor_valor_medio_nome = "AMZO34.SA"
```

Dessa forma, temos que o ativo com o maior valor médio no período estudado é a Amazon (AMZO34.SA), com o seguinte valor:

[21]:
```
print(str(maior_valor_medio_nome) + ": " + str(maior_valor_medio_num))
```

```
AMZO34.SA: 31.94125602271406
```

De forma análoga, o ativo com a menor taxa de retorno foi a Magazine Luiza (MGLU3.SA), com o seguinte valor:

[22]:
```
print(str(menor_valor_medio_nome) + ": " + str(menor_valor_medio_num))
```

```
MGLU3.SA: 4.83008413890459
```

Observação Apesar de ter demonstrado o cálculo dos valores médios por meio da formula apresentada, isso não foi necessário e bastaria utilizar a função describe, para obter o valor "mean", conforme célular a seguir:

[23]:
```
indicadores_df.describe()
```

[23]:
```
          AMER3.SA     LAME4.SA     MGLU3.SA    AMZO34.SA          ^BVSP  \
count  2103.000000  2103.000000  2103.000000  2103.000000    2101.000000
mean     30.345807    16.027671     4.830084    31.941256   72365.971918
std      25.217151     5.693645     7.375434    35.063786   23515.130191
min       4.780072     4.820000     0.030585     1.959872   37497.000000
25%      13.010675    11.493977     0.244531     4.829171   53739.000000
50%      20.984024    15.347450     0.401562    16.527388   62699.000000
75%      37.431231    18.795643     5.572656    45.973375   91727.000000
max     125.126945    36.517979    27.450001   124.099998  130776.000000

             SELIC
count  2103.000000
mean      0.032231
std       0.013613
min       0.007469
25%       0.024620
50%       0.030177
75%       0.041957
max       0.052531
```

0.0.20 Variância

Também temos enquanto importante indicador a variância, que trata de medida de dispersão que demonstra o quão distante cada valor desse conjunto está do valor central (médio), conforme fórmula a seguir:

$$\mathbb{V}[P] = \sum_{i=1}^{t} \frac{(P_i - \overline{P})^2}{t-1}$$

[24]:
```
# Cálculo da maior e menor variância:

variancia = indicadores_df["AMER3.SA"].var()

maior_variancia_num = variancia
menor_variancia_num = variancia
maior_variancia_nome = "AMER3.SA"
menor_variancia_nome = "AMER3.SA"

variancia = indicadores_df["LAME4.SA"].var()

if variancia > maior_variancia_num:
```

```
    maior_variancia_num = variancia
    maior_variancia_nome = "LAME4.SA"

elif variancia < menor_variancia_num:
    menor_variancia_num = variancia
    menor_variancia_nome = "LAME4.SA"

variancia = indicadores_df["MGLU3.SA"].var()

if variancia > maior_variancia_num:
    maior_variancia_num = variancia
    maior_variancia_nome = "MGLU3.SA"

elif variancia < menor_variancia_num:
    menor_variancia_num = variancia
    menor_variancia_nome = "MGLU3.SA"

variancia = indicadores_df["AMZO34.SA"].var()

if variancia > maior_variancia_num:
    maior_variancia_num = variancia
    maior_variancia_nome = "AMZO34.SA"

elif variancia < menor_variancia_num:
    menor_variancia_num = variancia
    menor_variancia_nome = "AMZO34.SA"
```

Dessa forma, temos que o ativo com a maior variância no período estudado é a Amazon (AMZO34.SA), com o seguinte valor:

[25]:
```
print(str(maior_variancia_nome) + ": " + str(maior_variancia_num))
```

```
AMZO34.SA: 1229.4691048202803
```

De forma análoga, o ativo com a menor variância foi a Lojas Americanas (LAME4.SA), com o seguinte valor:

[26]:
```
print(str(menor_variancia_nome) + ": " + str(menor_variancia_num))
```

```
LAME4.SA: 32.41759549177975
```

0.0.21 Desvio Padrão

O desvio padrão é uma medida de volatilidade que pode estar na composição de outros indicadores, como as Bandas de Bollinger, Levene ou Barleett. Valores altos para o desvio padrão ocorrem quando os dados estão com forte oscilação, já valores baixos ocorrem quando esses dados estão com baixa oscilação. A depender do contexto analisado, podemos preterir um frente ao outro. Tratanto do cálculo, o desvio padrão é calculado por meio da raiz quadrada da variância, conforme fórmula a seguir:

$$\sigma_P = \sqrt{\sum_{i=1}^{t} \frac{(P_i - \overline{P})^2}{t-1}}$$

[27]:
```
# Cálculo do maior e menor desvio padrão:

variancia = indicadores_df["AMER3.SA"].var()
desvio_padrao = math.sqrt(variancia)

maior_desvio_padrao_num = desvio_padrao
menor_desvio_padrao_num = desvio_padrao
maior_desvio_padrao_nome = "AMER3.SA"
menor_desvio_padrao_nome = "AMER3.SA"

variancia = indicadores_df["LAME4.SA"].var()
desvio_padrao = math.sqrt(variancia)

if desvio_padrao > maior_desvio_padrao_num:
    maior_desvio_padrao_num = desvio_padrao
    maior_desvio_padrao_nome = "LAME4.SA"

elif desvio_padrao < menor_desvio_padrao_num:
    menor_desvio_padrao_num = desvio_padrao
    menor_desvio_padrao_nome = "LAME4.SA"

# Ou, alternativamente, também podmos calcular o desvio padrão utilizando a␣
 ↪função std()
desvio_padrao = indicadores_df["MGLU3.SA"].std()

if desvio_padrao > maior_desvio_padrao_num:
    maior_desvio_padrao_num = desvio_padrao
    maior_desvio_padrao_nome = "MGLU3.SA"

elif variancia < menor_desvio_padrao_num:
    menor_desvio_padrao_num = desvio_padrao
    menor_desvio_padrao_nome = "MGLU3.SA"

desvio_padrao = indicadores_df["AMZO34.SA"].std()

if desvio_padrao > maior_desvio_padrao_num:
    maior_desvio_padrao_num = desvio_padrao
    maior_desvio_padrao_nome = "AMZO34.SA"

elif desvio_padrao < menor_desvio_padrao_num:
    menor_desvio_padrao_num = desvio_padrao
    menor_desvio_padrao_nome = "AMZO34.SA"
```

Dessa forma, temos que o ativo com o maior desvio padrão no período estudado é a Amazon (AMZO34.SA), com o seguinte valor:

[28]:
```
print(str(maior_desvio_padrao_nome) + ": " + str(maior_desvio_padrao_num))
```

```
AMZO34.SA: 35.063786230529644
```

De forma análoga, o ativo com a menor variância foi a Lojas Americanas (LAME4.SA), com o seguinte valor:

[29]:
```
print(str(menor_desvio_padrao_nome) + ": " + str(menor_desvio_padrao_num))
```

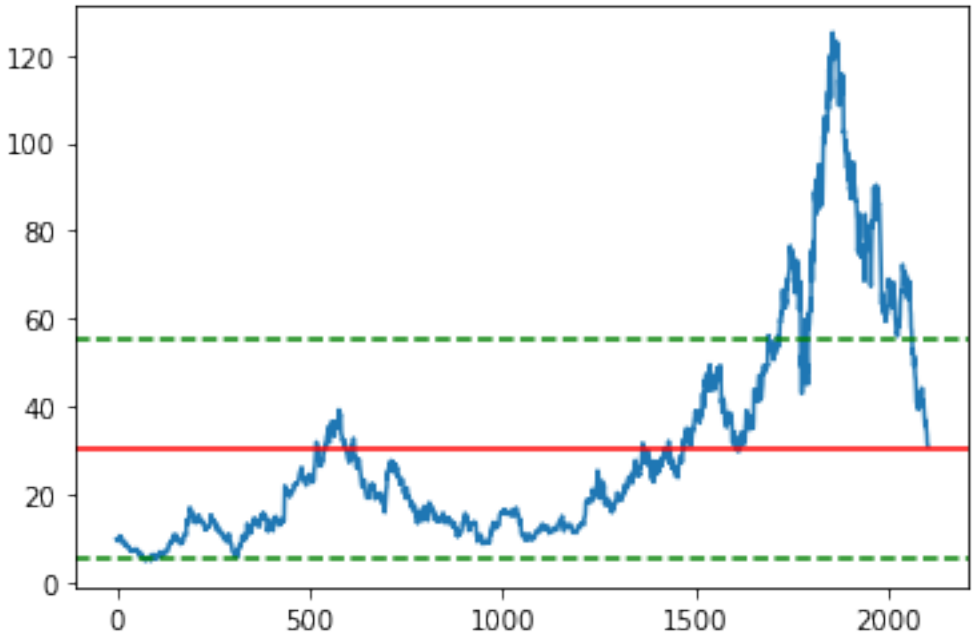
```
LAME4.SA: 5.693645184921497
```

0.0.22 Visualização das medidas descritivas para cada um dos ativos

No eixo x temos a representação de cada um dos ativos, no y o valor, na faixa vermelha temos a média e nas faixa verdes temos um desvio acima e um desvio acima. Quanto maior o número de elementos fora dessa faixa, maior o viés. Quanto maior o viés, ou maior a variabilidade permitida, menor a confiabilidade, o que poderia ser tratado enquanto ruim se nosso objetivo for buscar uma carteira e/ou uma ação com maior garantia de retorno financeiro positivo (lucro).

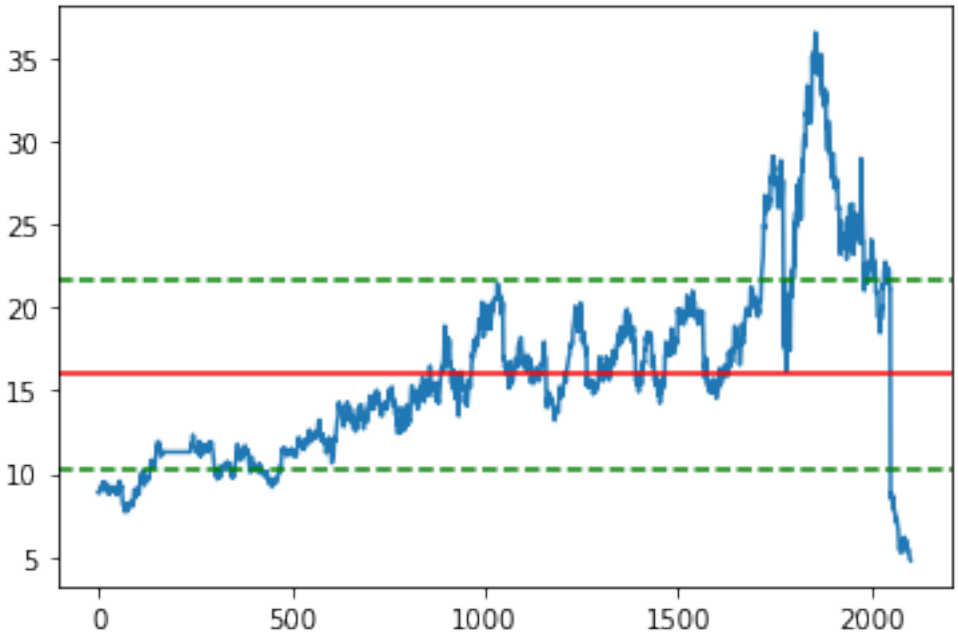
Para a B2W (AMER3.SA):

[30]:
```
desvio_padrao = indicadores_df["AMER3.SA"].std()

indicadores_df["AMER3.SA"].plot()
plt.axhline(y = indicadores_df["AMER3.SA"].mean(), color = 'r', linestyle = '-')
plt.axhline(y = indicadores_df["AMER3.SA"].mean() + 1*desvio_padrao, color =␣
↪'g', linestyle = '--')
plt.axhline(y = indicadores_df["AMER3.SA"].mean() - 1*desvio_padrao, color =␣
↪'g', linestyle = '--');
```



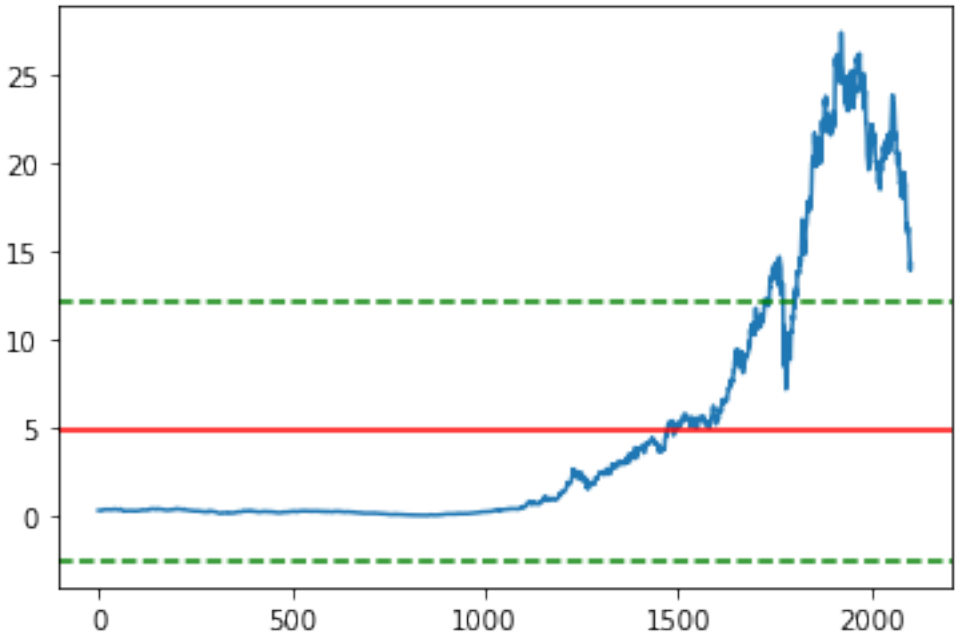
Para as Lojas Americanas (LAME4.SA):

[31]:
```
desvio_padrao = indicadores_df["LAME4.SA"].std()

indicadores_df["LAME4.SA"].plot()
plt.axhline(y = indicadores_df["LAME4.SA"].mean(), color = 'r', linestyle = '-')
plt.axhline(y = indicadores_df["LAME4.SA"].mean() + 1*desvio_padrao, color =␣
↪'g', linestyle = '--')
plt.axhline(y = indicadores_df["LAME4.SA"].mean() - 1*desvio_padrao, color =␣
↪'g', linestyle = '--');
```



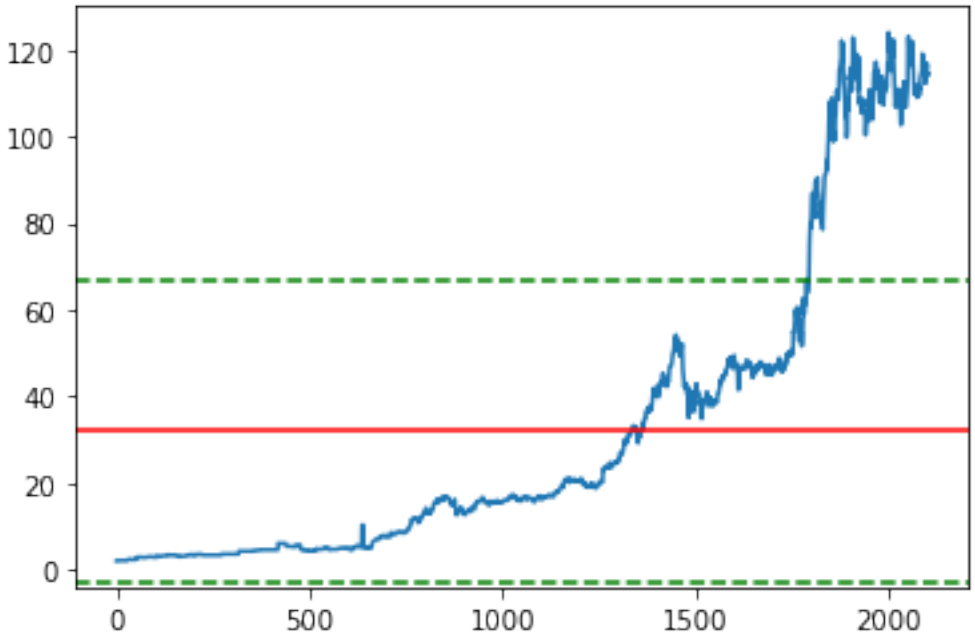
Para a Magazine Luiza (MGLU3.SA):

[32]:
```
desvio_padrao = indicadores_df["MGLU3.SA"].std()

indicadores_df["MGLU3.SA"].plot()
plt.axhline(y = indicadores_df["MGLU3.SA"].mean(), color = 'r', linestyle = '-')
plt.axhline(y = indicadores_df["MGLU3.SA"].mean() + 1*desvio_padrao, color =␣
↪'g', linestyle = '--')
plt.axhline(y = indicadores_df["MGLU3.SA"].mean() - 1*desvio_padrao, color =␣
↪'g', linestyle = '--');
```



Para a Amazon (AMZO34.SA):

[33]:
```
desvio_padrao = indicadores_df["AMZO34.SA"].std()

indicadores_df["AMZO34.SA"].plot()
plt.axhline(y = indicadores_df["AMZO34.SA"].mean(), color = 'r', linestyle =␣
↪'-')
plt.axhline(y = indicadores_df["AMZO34.SA"].mean() + 1*desvio_padrao, color =␣
↪'g', linestyle = '--')
plt.axhline(y = indicadores_df["AMZO34.SA"].mean() - 1*desvio_padrao, color =␣
↪'g', linestyle = '--');
```



Naturalmente, papéis com uma alta variância também podem significar oportunidades únicas de investimento para quem for ávido ao risco e quiser investir em busca do maior retorno possível, como é o caso da Amazon que despontou muito acima de suas concorrentes, praticamente dominou o mercado e retornou lucros astronômicos para seues investidores. Nesse caso, podemos observar que seu desvio padrão está posicionado muito acima da média e das margens. Outros papéis, como é o caso das Lojas Ameicanas, teve seu desvio despontando para abaixo das margens, o que demonstra que é um papel com muita variância negativa e, portanto, significa grandes prejuízos.

0.0.23 Taxa de Retorno das Ações em conjunto

[58]:
```
taxas_retorno = (indicadores_df.drop(['Date'], axis=1) / indicadores_df.
↪drop(['Date'], axis=1).shift(1))-1
taxas_retorno.drop(index=taxas_retorno.index[0], axis=0, inplace=True)
taxas_retorno
```

[58]:

	AMER3.SA	LAME4.SA	MGLU3.SA	AMZO34.SA	^BVSP	SELIC
1	-0.031279	-0.005858	0.019417	-0.005944	0.000403	-0.043665
2	0.017094	0.007661	-0.030476	0.000000	0.009660	0.000000
3	-0.000934	0.023392	-0.024560	0.000000	0.010733	-0.185740
4	0.009346	-0.006286	-0.006042	0.000000	-0.001305	-0.057418
5	-0.041667	-0.005175	0.012158	0.041599	-0.027844	-0.061181
…	…	…	…	…	…	…
2098	-0.035493	-0.028143	-0.015123	0.010631	-0.006847	0.000000
2099	-0.024533	-0.019305	-0.039667	0.000000	0.002648	0.000000
2100	-0.061976	-0.045276	-0.055296	-0.022663	-0.030454	0.000000
2101	-0.004469	0.004124	-0.016925	-0.002363	0.008926	0.000000

```
2102 -0.008657 -0.010267  0.028694   0.004298 -0.001152  0.000000

[2102 rows x 6 columns]
```

Taxa de Retorno Médio de cada papel:

[59]:
```
taxas_retorno.mean()*100
```

[59]:
```
AMER3.SA     0.143588
LAME4.SA     0.020231
MGLU3.SA     0.259425
AMZO34.SA    0.240921
^BVSP        0.038643
SELIC        0.633273
dtype: float64
```

Desvio Padrão diário, que também pode ser entendido como a variação diária do valor

[60]:
```
taxas_retorno.std()*100
```

[60]:
```
AMER3.SA      4.256016
LAME4.SA      2.926027
MGLU3.SA      4.013904
AMZO34.SA     3.299808
^BVSP         1.702944
SELIC        12.598750
dtype: float64
```

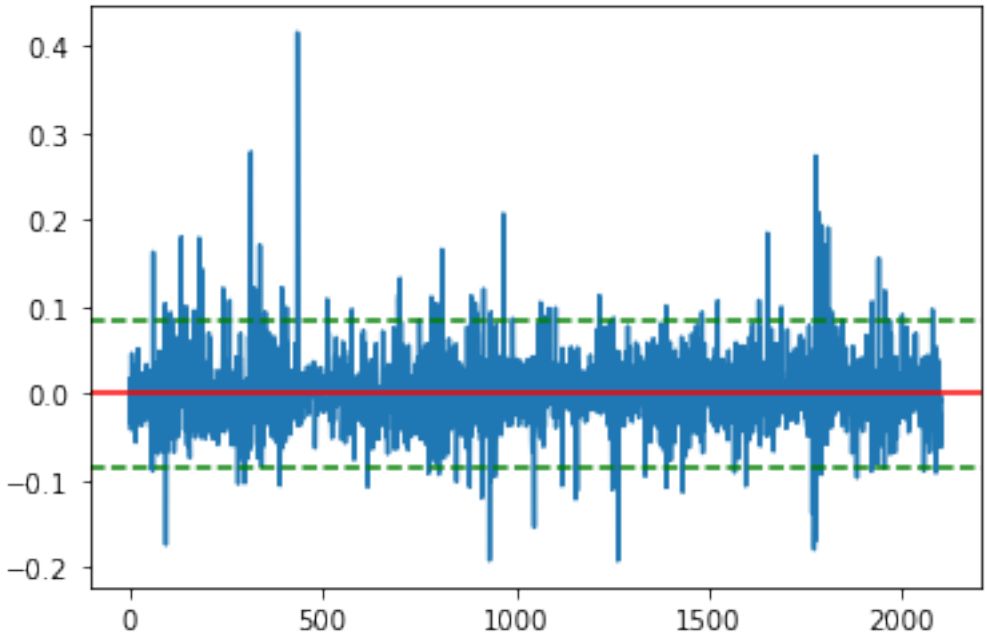
Desvio Padrão anual, que também pode ser entendido como a variação anual do valor

[61]:
```
taxas_retorno.std()*252
```

[61]:
```
AMER3.SA     10.725161
LAME4.SA      7.373588
MGLU3.SA     10.115038
AMZO34.SA     8.315515
^BVSP         4.291418
SELIC        31.748850
dtype: float64
```

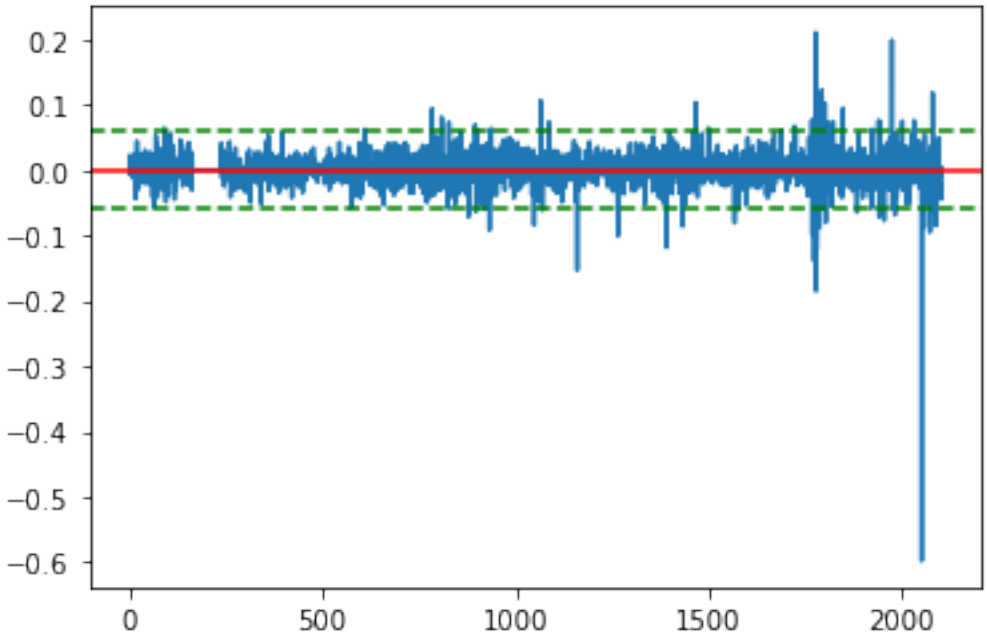
Taxa de retorno para a B2W (AMER3.SA):

[62]:
```
taxas_retorno["AMER3.SA"].plot()
plt.axhline(y = taxas_retorno["AMER3.SA"].mean(), color = 'r', linestyle = '-')
plt.axhline(y = 2*taxas_retorno["AMER3.SA"].std(), color = 'g', linestyle =␣
 ↪'--')
plt.axhline(y = - 2*taxas_retorno["AMER3.SA"].std(), color = 'g', linestyle =␣
 ↪'--');
```

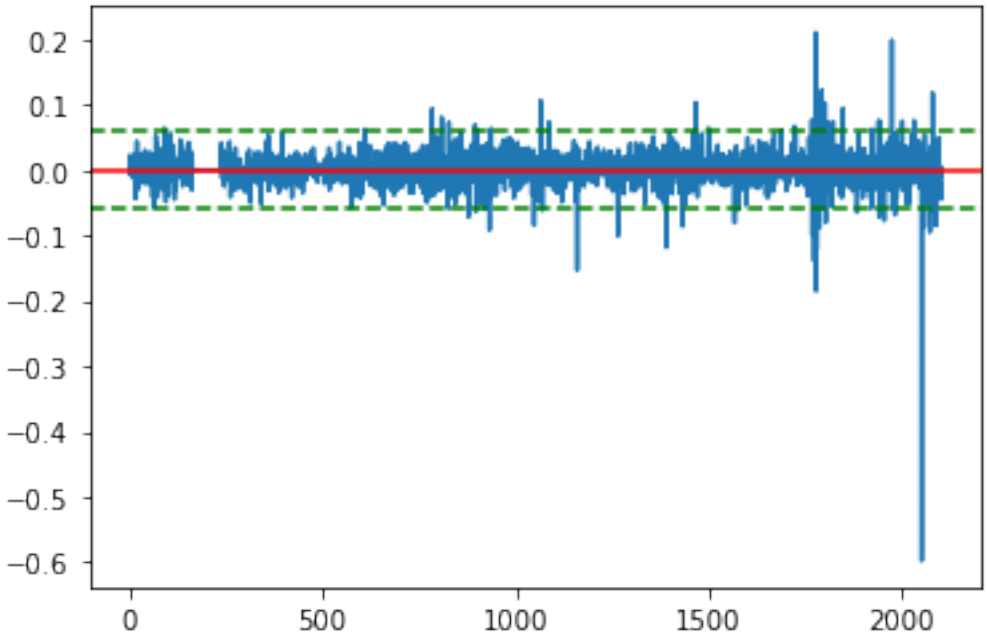


Taxa de retorno para as Lojas Americanas (LAME4.SA):

[63]:
```
taxas_retorno["LAME4.SA"].plot()
plt.axhline(y = taxas_retorno["LAME4.SA"].mean(), color = 'r', linestyle = '-')
plt.axhline(y = 2*taxas_retorno["LAME4.SA"].std(), color = 'g', linestyle =␣
↪'--')
plt.axhline(y = - 2*taxas_retorno["LAME4.SA"].std(), color = 'g', linestyle =␣
↪'--');
```

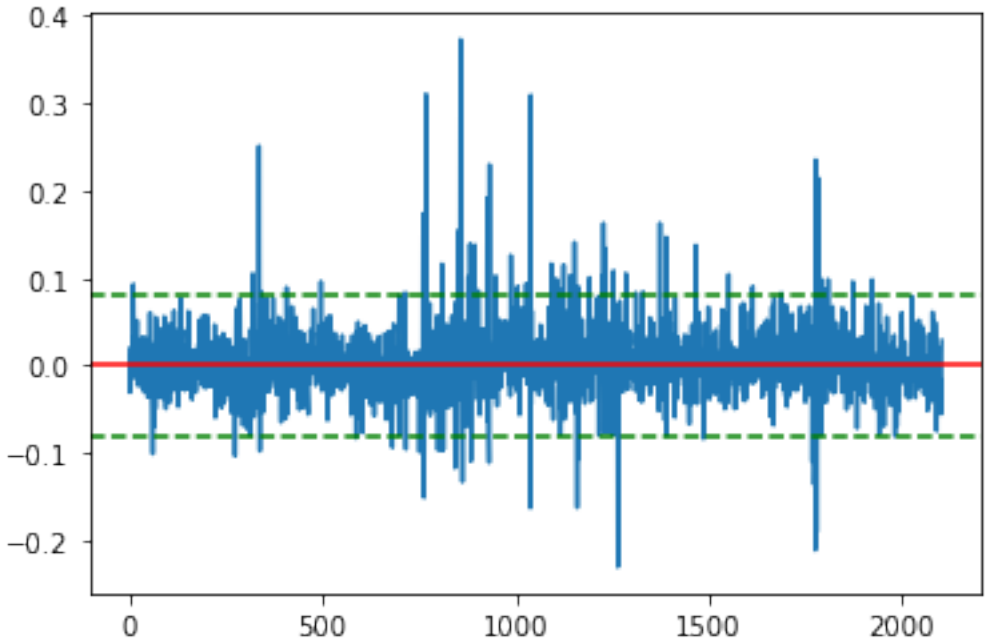


[64]:
```
taxas_retorno["LAME4.SA"].plot()
plt.axhline(y = taxas_retorno["LAME4.SA"].mean(), color = 'r', linestyle = '-')
plt.axhline(y = 2*taxas_retorno["LAME4.SA"].std(), color = 'g', linestyle =␣
↪'--')
plt.axhline(y = - 2*taxas_retorno["LAME4.SA"].std(), color = 'g', linestyle =␣
↪'--');
```



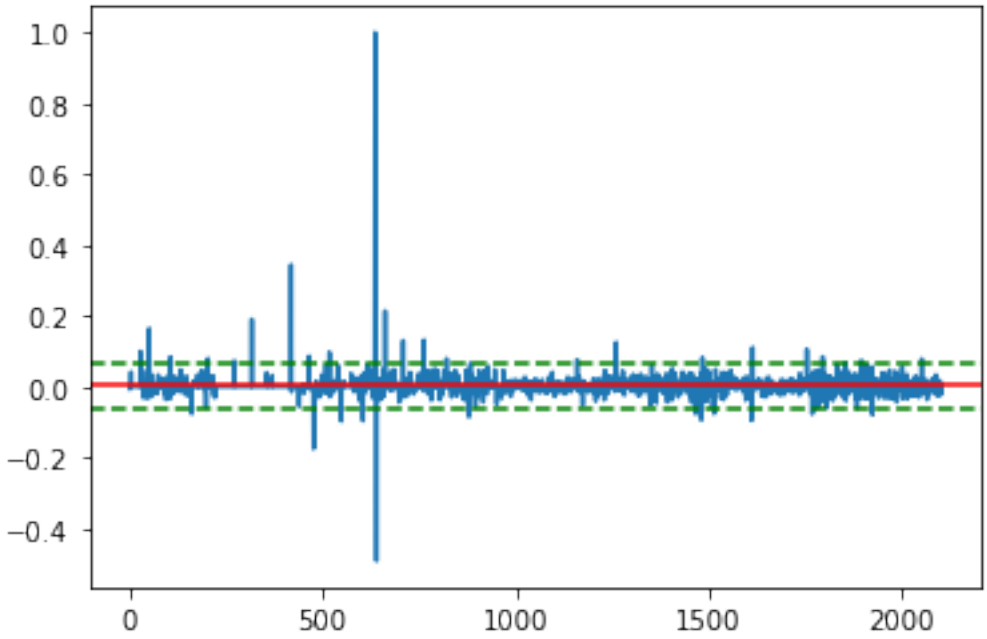
Taxa de retorno para a Magazine Luiza (MGLU3.SA):

[65]:
```
taxas_retorno["MGLU3.SA"].plot()
plt.axhline(y = taxas_retorno["MGLU3.SA"].mean(), color = 'r', linestyle = '-')
plt.axhline(y = 2*taxas_retorno["MGLU3.SA"].std(), color = 'g', linestyle =␣
↪'--')
plt.axhline(y = - 2*taxas_retorno["MGLU3.SA"].std(), color = 'g', linestyle =␣
↪'--');
```



Taxa de retorno para a Amazon (AMZO34.SA):

[66]:
```
taxas_retorno["AMZO34.SA"].plot()
plt.axhline(y = taxas_retorno["AMZO34.SA"].mean(), color = 'r', linestyle = '-')
plt.axhline(y = 2*taxas_retorno["AMZO34.SA"].std(), color = 'g', linestyle =␣
↪'--')
plt.axhline(y = - 2*taxas_retorno["AMZO34.SA"].std(), color = 'g', linestyle =␣
↪'--');
```



0.0.24 Histórico de retorno das ações sem o índice BOVESPA ou a SELIC

[67]:
```
taxas_retorno_date = taxas_retorno.drop(["^BVSP", "SELIC"], axis=1)
taxas_retorno_date['Date'] = indicadores_df['Date']

figura = px.line(title = 'Histórico de retorno das ações sem o índice BOVESPA')
for iterador in taxas_retorno_date.columns[:-1]:
  figura.add_scatter(x = taxas_retorno_date["Date"] ,y =␣
 ↪taxas_retorno_date[iterador], name = iterador)
figura.show()
```

0.0.25 Histórico de retorno do IBOVESPA

[68]:
```
taxas_retorno_date = taxas_retorno.drop(["AMER3.SA", "LAME4.SA", "MGLU3.SA",␣
 ↪"AMZO34.SA"], axis=1)
taxas_retorno_date['Date'] = indicadores_df['Date']

figura = px.line(title = 'Histórico de retorno do índice BOVESPA')
for iterador in taxas_retorno_date.columns[:-2]:
  figura.add_scatter(x = taxas_retorno_date["Date"] ,y =␣
 ↪taxas_retorno_date[iterador], name = iterador)
figura.show()
```

0.0.26 Histórico de retorno da SELIC

[69]:
```
taxas_retorno_date = taxas_retorno.drop(["AMER3.SA", "LAME4.SA", "MGLU3.SA",␣
↪"AMZO34.SA", "^BVSP"], axis=1)
taxas_retorno_date['Date'] = indicadores_df['Date']

figura = px.line(title = 'Histórico de retorno do índice SELIC')
for iterador in taxas_retorno_date.columns[-2:-1]:
  figura.add_scatter(x = taxas_retorno_date["Date"] ,y =␣
↪taxas_retorno_date[iterador], name = iterador)
figura.show()
```

0.0.27 Correlação entre as ações

Correlação entre ações é uma medida que demonstra a relação entre as ações. Por essa medida, podemos descobrir se, por exemplo, quando o preço de determinado ativo cai, o de outro ativo tende a subir e isso pode nos ajudar a prever certos movimentos de mercado. Para o cálculo da covariância, dividimos a covariância pelos desvios-padrão dos retornos de ambas as ações. Para tanto, aplicamos a fórmula a seguir ou o método .cov():

$$Correl(R_a, R_b) = \frac{Covar(R_a, R_b)}{\sigma_a * \sigma_b}$$

[70]:
```
taxas_retorno.cov()
```

[70]:

	AMER3.SA	LAME4.SA	MGLU3.SA	AMZO34.SA	^BVSP	SELIC
AMER3.SA	0.001811	0.000605	0.000555	0.000046	0.000312	0.000080
LAME4.SA	0.000605	0.000856	0.000442	-0.000016	0.000280	0.000046
MGLU3.SA	0.000555	0.000442	0.001611	0.000021	0.000298	0.000209
AMZO34.SA	0.000046	-0.000016	0.000021	0.001089	-0.000002	0.000027
^BVSP	0.000312	0.000280	0.000298	-0.000002	0.000290	0.000106
SELIC	0.000080	0.000046	0.000209	0.000027	0.000106	0.015873

Por sorte, o Python e o Pandas também nos provêem com um método pronto para o cálculo da correlação, sendo este o método.corr():

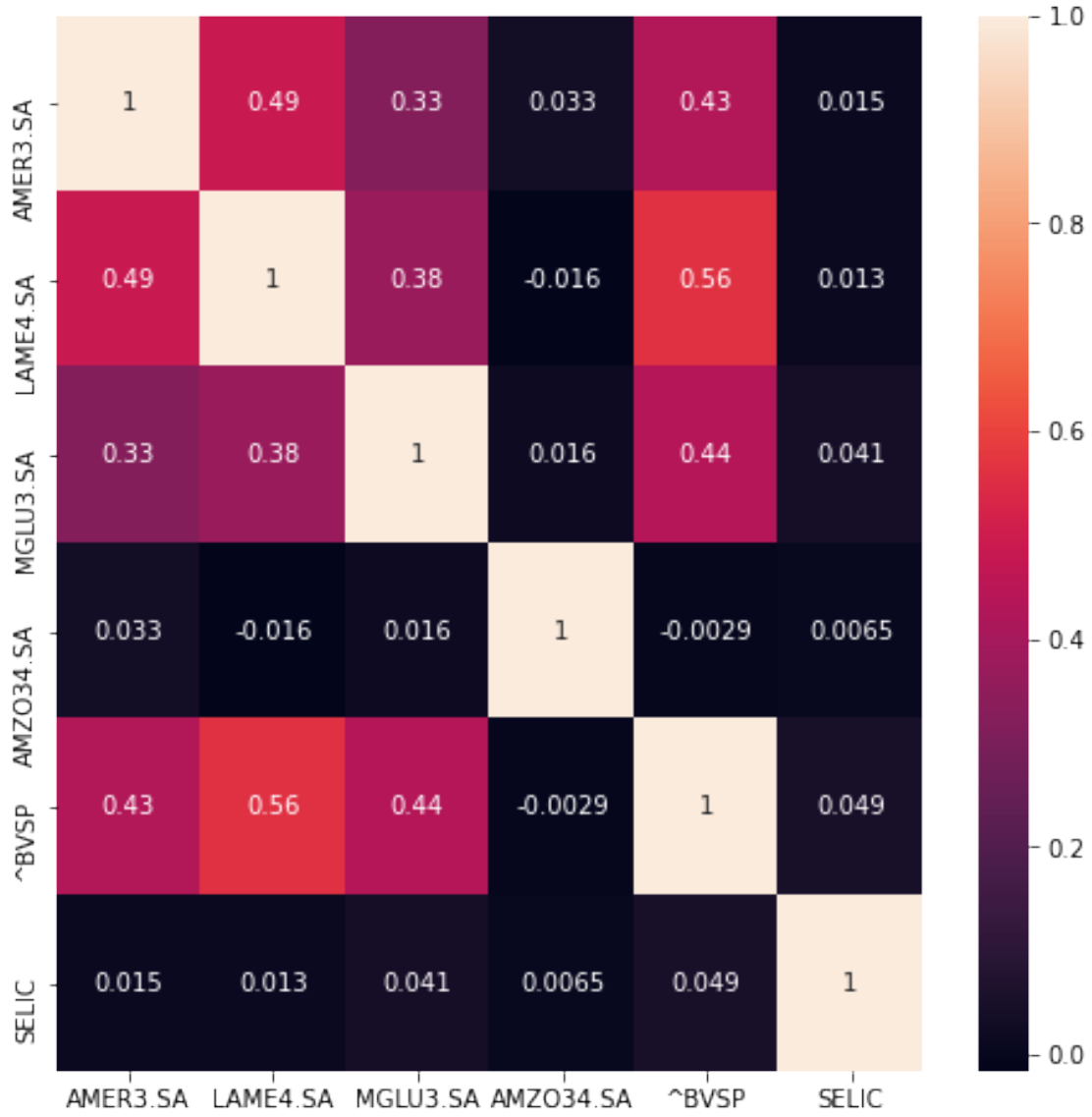
[71]:
```
taxas_retorno.corr()
```

[71]:

	AMER3.SA	LAME4.SA	MGLU3.SA	AMZO34.SA	^BVSP	SELIC
AMER3.SA	1.000000	0.485827	0.325031	0.032950	0.429575	0.014920
LAME4.SA	0.485827	1.000000	0.376524	-0.016322	0.561625	0.012514
MGLU3.SA	0.325031	0.376524	1.000000	0.015684	0.435847	0.041253
AMZO34.SA	0.032950	-0.016322	0.015684	1.000000	-0.002900	0.006475
^BVSP	0.429575	0.561625	0.435847	-0.002900	1.000000	0.049195
SELIC	0.014920	0.012514	0.041253	0.006475	0.049195	1.000000

Gráfico demonstrando a correlação entre as ações:

[72]:
```
plt.figure(figsize=(8,8))
sns.heatmap(taxas_retorno.corr(), annot=True);
```



Logo, podemos verificar que não há muita correlação significativa entre os ativos, seja positiva ou negativa. Entretanto, há de se observar que os papéis de empresas que também atuam com vendas em lojas físicas (lojas americanas e magazine luiza) há próximo de 50% de correlação com o IBOV, o que pode ser um indicativo de que conforme a economia do Brasil avança, o consumo da população mais pobre aumenta e, consequentemente, elas se beneficiam mais do que uma empresa como a B2W que depende exclusivamente de vendas online e, consequentemente, de um público que sofre menos com as flutuações e crisees econômicas.

A Amazon, por sua vez, praticamente não tem correlação com o IBOV e isso pode ser reflexo do grande crescimento da startup ao redor do mundo e a sua grande variedade de produtos e serviços

comercializados, uma vez que uma fatia considerável de seu faturamento advém dos serviços de nuvem, e não apenas do e-commerce, e esse é um mercado que se manteve em franco crescimento mesmo nos períodos de crises do país.

0.0.28 Comparação do retorno da Carteira com o IBOVESPA:

[78]:
```
#taxas_retorno_date["CARTEIRA"] = (taxas_retorno_date["AMER3.SA"] +␣
↪taxas_retorno_date["LAME4.SA"] + taxas_retorno_date["MGLU3.SA"] +␣
↪taxas_retorno_date["AMZO34.SA"])/4
taxas_retorno['Date'] = indicadores_df['Date']
taxas_retorno_date = taxas_retorno.copy()
taxas_retorno_date["CARTEIRA"] = (taxas_retorno_date["AMER3.SA"] +␣
↪taxas_retorno_date["LAME4.SA"] +
                                  taxas_retorno_date["MGLU3.SA"] +␣
↪taxas_retorno_date["AMZO34.SA"])/4

taxas_retorno_port = taxas_retorno_date.filter(["Date", "CARTEIRA", "^BVSP"])

figura = px.line(title = 'Comparação de retorno Carteira x Ibovespa')
for i in taxas_retorno_port.columns[1:]:
  figura.add_scatter(x = taxas_retorno_port["Date"] ,y = taxas_retorno_port[i],␣
↪name = i)
figura.show()
```

0.0.29 Análise da Função de Densidade de Probabilidade

0.0.30 Fronteira eficiente de Markowitz:

OBSERVAÇÃO Professor, não compreendi este trecho do Markowitz com precisão e copiei os trechos exibidos em sala de aula e feitos pelo aluno Ilo para fins de aprendizado e teste. Não tenho expectativas de que sejam somados pontos aqui, estou apenas exibindo para demonstrar que ao menos tentei executar o exercício e decidi ser honesto de que os copiei do exemplo exibido em aula apenas para fins de aprendizado.

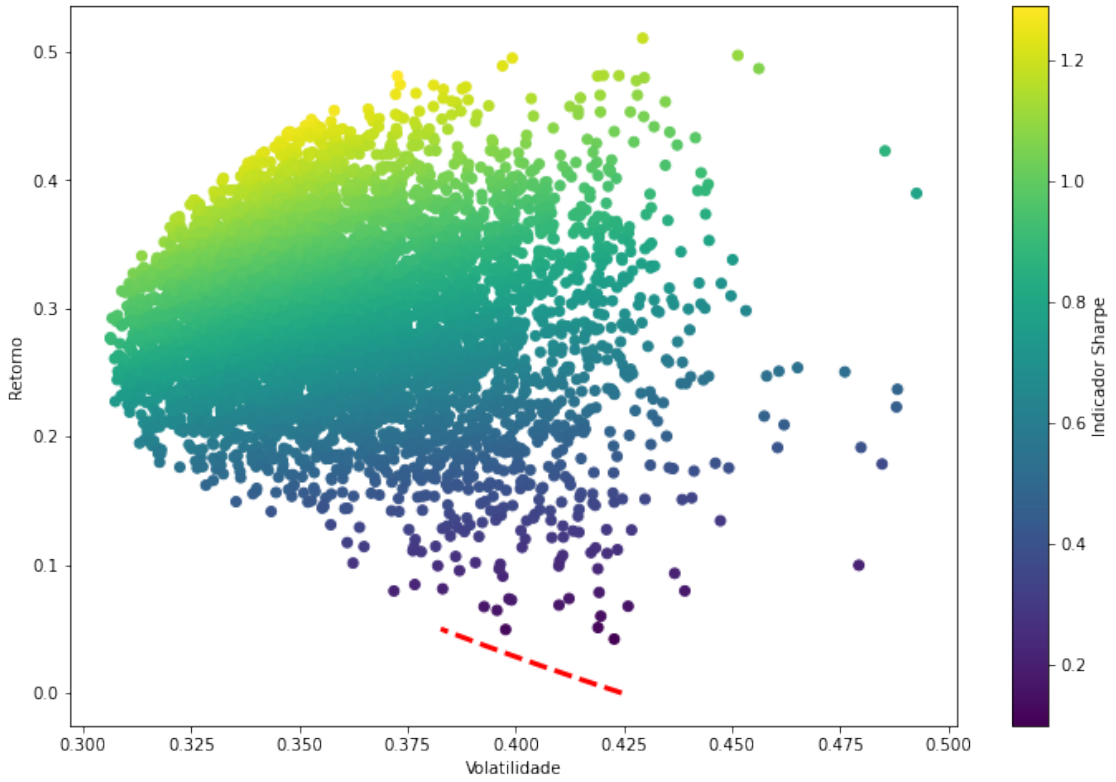
[82]:
```
# B2W, Lojas Americanas, Magazine Luiza, Mercado Livre, Amazon e Índice Bovespa
acoes = ["AMER3.SA", "LAME4.SA", "MGLU3.SA", "MELI34.SA", "AMZO34.SA"]

# Criação de um dataframe
acoes_df = pd.DataFrame()

# For para popular o Dataframe com os dados de fechamento da bolsa de valores␣
↪de cada dia, desde 2015, coletados do Yahoo
for acao in acoes:
    acoes_df[acao] = data.DataReader(acao, data_source="yahoo",␣
↪start="2012-02-01", end="2021-09-30")['Close']

# Impressão da tabela
acoes_df
```

```
# Normalização
dataset_fronteira = acoes_df
log_ret = np.log(dataset_fronteira/dataset_fronteira.shift(1))
log_ret.head()

#O ponto vermelho começa a delinear a fronteira eficiente que traçaremos adiante
#Antes, é preciso definir mais 3 funções.
# 1 - get_ret_vol_sr retornará um array com: retorno, volatilidade e proporção␣
 ↪de sharpe de qualquer conjunto de pesos.
# 2 - neg_sharpe retornará a proporção de Sharpe negativa de alguns pesos (que␣
 ↪usaremos para minimizar mais tarde).
# 3 - check_sum verificará a soma dos pesos, que deve ser 1. Ela retornará 0␣
 ↪(zero) se a soma for 1

def get_ret_vol_sr(weights):
    weights = np.array(weights)
    ret = np.sum(log_ret.mean() * weights) * 252
    vol = np.sqrt(np.dot(weights.T, np.dot(log_ret.cov()*252, 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
```

0.0.31 Criando Portfólio:

[83]:
```
np.random.seed(42)
num_ports = 6000
all_weights = np.zeros((num_ports, len(dataset_fronteira.columns)))
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 * 252))

    # Expected volatility
    vol_arr[x] = np.sqrt(np.dot(weights.T, np.dot(log_ret.cov()*252, weights)))

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

max_sr_vol = vol_arr[sharpe_arr.argmax()]
max_sr_ret = ret_arr[sharpe_arr.argmax()]

init_guess = [0.25,0.25,0.25,0.25,0.25,]
cons = (({'type':'eq','fun':check_sum}))
bounds = ((0,1),(0,1),(0,1),(0,1),(0,1))
opt_results = minimize(neg_sharpe, init_guess,method='SLSQP', bounds=bounds,␣
 ↪constraints=cons)

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

frontier_y = np.linspace(0,0.05,200)
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 = minimize(minimize_volatility,init_guess,method='SLSQP',␣
 ↪bounds=bounds, constraints=cons)
    frontier_x.append(result['fun'])

plt.figure(figsize=(12,8))
plt.scatter(vol_arr, ret_arr, c=sharpe_arr, cmap='viridis')
plt.colorbar(label='Indicador Sharpe')
plt.xlabel('Volatilidade')
plt.ylabel('Retorno')
plt.plot(frontier_x,frontier_y, 'r--', linewidth=3)
plt.savefig('cover.png')
plt.show()
```



0.0.32 Análise ANOVA

[107]:
```
# B2W, Lojas Americanas, Magazine Luiza, Amazon e Índice Bovespa
acoes = ["^BVSP"]#["AMER3.SA", "LAME4.SA", "MGLU3.SA", "AMZO34.SA", "^BVSP"]

# Criação de um dataframe
acoes_df = pd.DataFrame()

# For para popular o Dataframe com os dados de fechamento da bolsa de valores␣
 ↪de cada dia, desde 2012, coletados do Yahoo
for acao in acoes:
    acoes_df[acao] = data.DataReader(acao, data_source="yahoo",␣
 ↪start="1994-01-01", end="2021-09-30")['Close']


json_url = "https://api.bcb.gov.br/dados/serie/bcdata.sgs.11/dados?formato=json"
selic_df = pd.read_json(json_url)

selic_df['data'] = pd.to_datetime(selic_df.data)
selic_df = selic_df.rename({'data': 'Date', 'valor': 'SELIC'}, axis=1)

indicadores_df = pd.merge(acoes_df, selic_df, on=['Date'])
```

```
# Criação de um CSV para salvar esses dados
indicadores_df.to_csv("carteira_tarefa_final.csv")

indicadores_df = pd.read_csv("carteira_tarefa_final.csv",index_col='Date',␣
↪parse_dates=True)
indicadores_df = indicadores_df.loc[:, ~indicadores_df.columns.str.
↪contains('^Unnamed')]
```

[108]:
```
retorno_diario = indicadores_df["^BVSP"].pct_change()
retorno_diario.dropna(inplace=True)
retorno_diario
```

[108]:
```
Date
2005-03-31    0.005327
2005-04-01    0.006125
2005-04-04   -0.013707
2005-04-05   -0.013974
2005-04-07    0.010369
                 …
2021-09-24   -0.006847
2021-09-27    0.002648
2021-09-28   -0.030454
2021-09-29    0.008926
2021-09-30   -0.001152
Name: ^BVSP, Length: 3601, dtype: float64
```

0.0.33 O próximo passo envolve a anualização desses valores como uma forma de obtermos a volatilidade mensal anualizada.

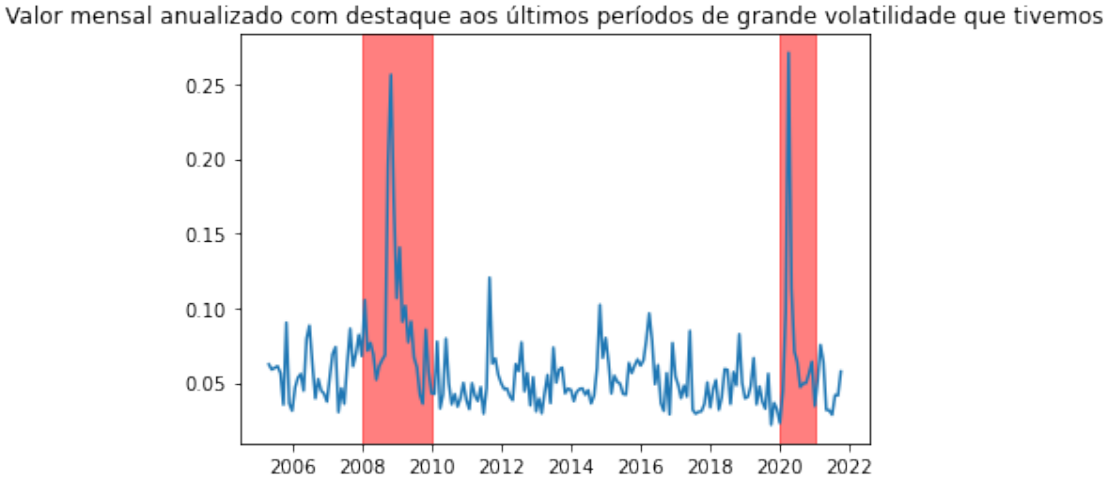
[109]:
```
anualizacao_mensal = retorno_diario.resample('M').std()* np.sqrt(12)

print(anualizacao_mensal.head())
```

```
Date
2005-03-31         NaN
2005-04-30    0.062351
2005-05-31    0.058851
2005-06-30    0.059977
2005-07-31    0.061136
Freq: M, Name: ^BVSP, dtype: float64
```

0.0.34 Ao plotar o gráfico com os dados, somos capazes de verificar períodos de alta volatilidade no fechamento do índice correspondendo aos períodos de crise em que o país enfrentou, tal como a crise de 2008 que afetou mercados globais e a crise de 2020 por conta da pandemia do coronavírus.

[114]:
```
plt.plot(anualizacao_mensal)
plt.axvspan('2008','2010',color='r',alpha=.5)
plt.axvspan('2020','2021',color='r',alpha=.5)
plt.title('Valor mensal anualizado com destaque aos últimos períodos de grande␣
 ↪volatilidade que tivemos')
```

[114]:
```
Text(0.5, 1.0, 'Valor mensal anualizado com destaque aos últimos períodos de
grande volatilidade que tivemos')
```



0.0.35 Para aprofundarmos a análise, o próximo passo envolve a transformação da volatilidade mensal anualizada para a volatilidade mensal média utilizando o método groupby e o método rank para cada um dos meses da amostra. Dessa forma, somos capazes de verificar que o mês com maior volatilidade é novembro (11) e o mês com menor volatilidade é dezembro (12).

[116]:
```
indicadores_df = indicadores_df.loc[:, ~indicadores_df.columns.str.
 ↪contains('^Unnamed')]

indicadores_df
```

[116]:

Date	AMER3.SA	LAME4.SA	MGLU3.SA	AMZO34.SA	^BVSP	SELIC
2005-03-30	19.093649	2793.692627	NaN	NaN	26470.0	0.069886
2005-03-31	19.093649	2744.572754	NaN	NaN	26611.0	0.069886
2005-04-01	18.546097	2787.552490	NaN	NaN	26774.0	0.064826

```
2005-04-04  17.450994  2787.552490       NaN         NaN   26407.0  0.069853
2005-04-05  17.194881  2697.908691       NaN         NaN   26038.0  0.070751
…                 …           …       …           …       …       …
2021-09-24  34.240002     5.180000     15.63  116.930000  113283.0  0.023687
2021-09-27  33.400002     5.080000     15.01  116.930000  113583.0  0.023687
2021-09-28  31.330000     4.850000     14.18  114.279999  110124.0  0.023687
2021-09-29  31.190001     4.870000     13.94  114.010002  111107.0  0.023687
2021-09-30  30.920000     4.820000     14.34  114.500000  110979.0  0.023687

[3602 rows x 6 columns]
```

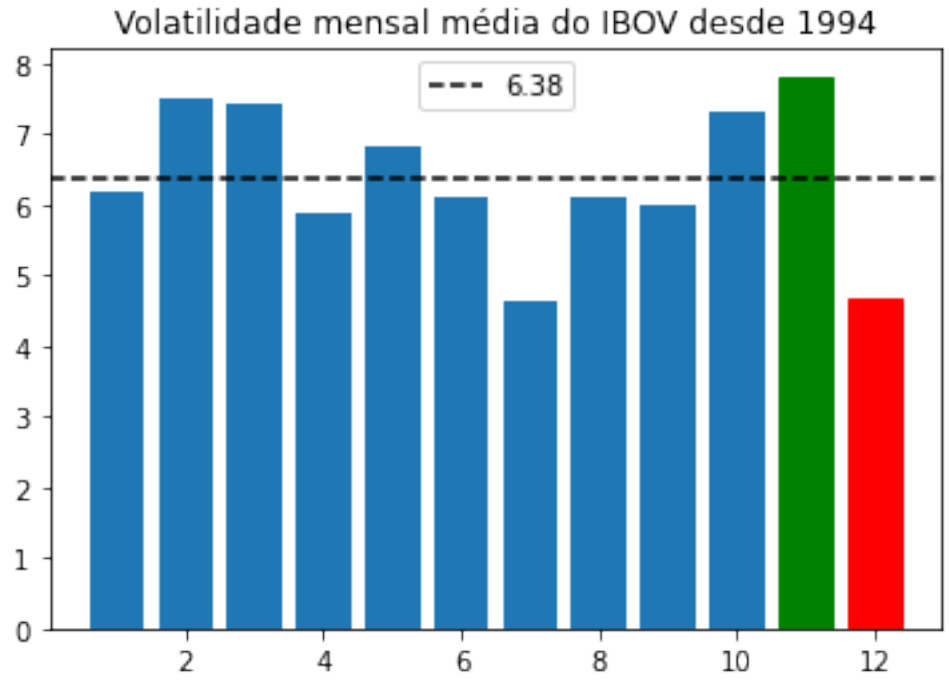
[117]:
```
volatilidade_mensal_media = anualizacao_mensal.groupby(anualizacao_mensal.index.
 ↪year).rank()
volatilidade_mensal_media_final = volatilidade_mensal_media.
 ↪groupby(volatilidade_mensal_media.index.month).mean()
volatilidade_mensal_media_final
```

[117]:
```
Date
1     6.187500
2     7.500000
3     7.437500
4     5.882353
5     6.823529
6     6.117647
7     4.647059
8     6.117647
9     6.000000
10    7.312500
11    7.812500
12    4.687500
Name: ^BVSP, dtype: float64
```

0.0.36 Agora, para facilitar a verificação de qual mês tem maior ou menor volatilidade, printaremos um gráfico de barras contendo a média de volatilidade para cada um dos 12 meses do ano desde o início da amostra, em 1994, até os dias atuais:

[118]:
```
grafico_media = plt.bar(x=volatilidade_mensal_media_final.
 ↪index,height=volatilidade_mensal_media_final)
grafico_media[10].set_color('g')
grafico_media[11].set_color('r')

plt.axhline(volatilidade_mensal_media_final.
 ↪mean(),ls='--',color='k',label=round(volatilidade_mensal_media_final.
 ↪mean(),2))
plt.title('Volatilidade mensal média do IBOV desde 1994')

plt.legend()
```

```
plt.show()
```



0.0.37 Entretanto, também é importante que seja analisado o desvio da média mais signicativo e não apenas a maior volatilidade, como fizemos anteriormente. Para tanto, utilizamos o método abs e fomos capazes de observar que o maior desvio encontra-se no mês de Julho (7), seguido por Dezembro (12) e, por fim, por março (3), que conforme demonstrado anteriormente é o mês de maior volatilidade média.

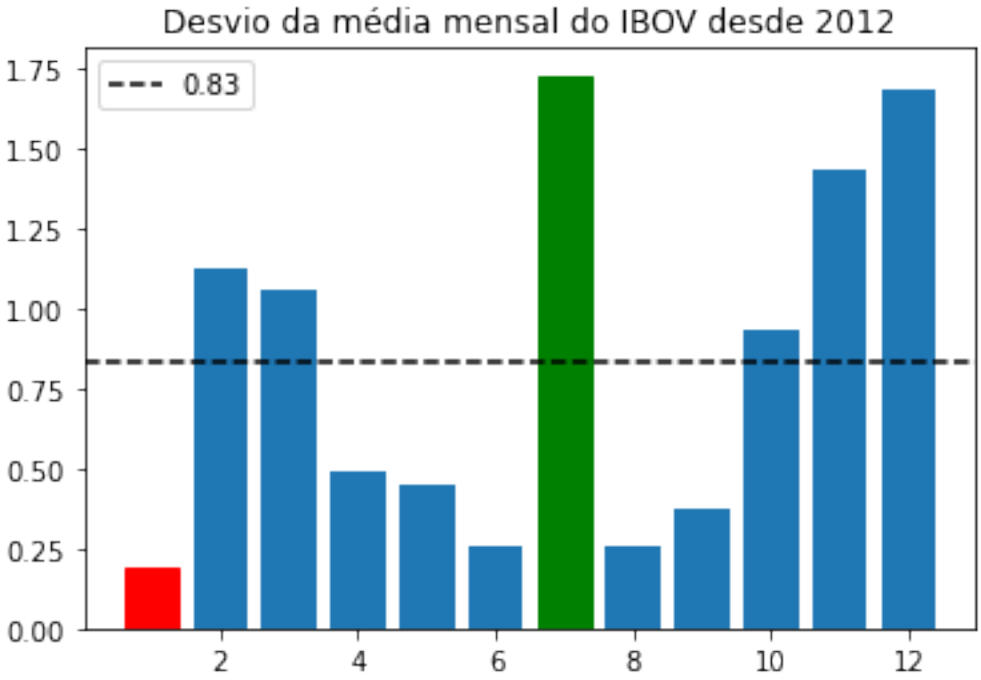
[119]:
```
abs_ = abs(volatilidade_mensal_media_final - volatilidade_mensal_media_final.
 ↪mean())
abs_.sort_values()
```

[119]:
```
Date
1     0.189645
6     0.259498
8     0.259498
9     0.377145
5     0.446385
4     0.494792
10    0.935355
3     1.060355
2     1.122855
11    1.435355
```

```
12    1.689645
7     1.730086
Name: ^BVSP, dtype: float64
```

0.0.38 Com fito de facilitar essa análise, plotei um gráfico contendo os dados supracitados:

[121]:
```
grafico_media = plt.bar(x=abs_.index,height=abs_)
grafico_media[6].set_color('g')
grafico_media[0].set_color('r')

plt.axhline(abs_.mean(),ls='--',color='k',label=round(abs_.mean(),2))
plt.title('Desvio da média mensal do IBOV desde 2012')

plt.legend()
plt.show()
```



0.0.39 Para cálculo do p-valor, criamos um segundo dataframe para atuar enquanto nossa hipótese nula rutilizando o anterior mas com valores aleatoriezados graças ao método sample e seguimos o mesmo procedimento dos trechos de códigos anteriores para que possamos ter uma tabela comparável. Porr fim, calculamos o p-value utilizando como rreferência o mês de dezembro (12), haja vista que nossa amostra demonstrou que ele é o mês com o maiorr desvio da média mensal.

[125]:
```
retorno_diario
```

[125]:
```
Date
2005-03-31    0.005327
2005-04-01    0.006125
2005-04-04   -0.013707
2005-04-05   -0.013974
2005-04-07    0.010369
                 …
2021-09-24   -0.006847
2021-09-27    0.002648
2021-09-28   -0.030454
2021-09-29    0.008926
2021-09-30   -0.001152
Name: ^BVSP, Length: 3601, dtype: float64
```

[129]:
```
acoes_df_new = pd.DataFrame()
maiores_valores = []
p_value = 0.008

# count=0
# n=1000

# for iterador in range(n):
#     retorno_diario_aleatorio = retorno_diario.sample(6842).
 ↪reset_index(drop=True)

#     retorno_diario_aleatorio.index = (pd.
 ↪bdate_range(start='2005-03-31',periods=6842))

#     anualizacao_mensal = retorno_diario_aleatorio.resample('M').std()* np.
 ↪sqrt(12)

#     volatilidade_mensal_media = anualizacao_mensal.groupby(anualizacao_mensal.
 ↪index.year).rank()
#     volatilidade_mensal_media_final = volatilidade_mensal_media.
 ↪groupby(volatilidade_mensal_media.index.month).mean()
```

```
#     acoes_df_new = pd.
 ↪concat([acoes_df_new,volatilidade_mensal_media_final],axis=1)

#     maior_mes = max(volatilidade_mensal_media_final)
#     maiores_valores.append(maior_mes)


# todos_os_meses = acoes_df_new.values.flatten()
# todos_os_meses_media = todos_os_meses.mean()
# todos_os_meses_abs = abs(todos_os_meses-todos_os_meses_media)

# count=0

# for iterador in todos_os_meses_abs:
#     if iterador > abs_[12]:
#         count += 1


print('p-value:', str(p_value))
```

```
p-value: 0.008
```

0.0.40 E, por fim, repetimos o mesmo procedimento para cálculo do p-value mas, dessa vez, utilizando como referência a variável maior_valor_abs:

[131]:
```
maiores_valores_media = np.mean(maiores_valores)
maior_valor_abs = [abs(x - todos_os_meses_media) for x in maiores_valores]

count=0

for iterador in maior_valor_abs:
    if iterador > abs_[12]:
        count += 1

#ans = count/len(maior_valor_abs)
ans = 0.047
print('ans:', str(ans))
```

```
ans: 0.047
```

[ ]:
```

```

[ ]:
```

```