Visualização de dados em Python, Parte 1: Gráficos de dispersão e distribuição com Seaborn

[](https://medium.com/@joaolggross?source=post_page-----c8636f0202b1----------------------)

[João Gross](https://medium.com/@joaolggross?source=post_page-----c8636f0202b1----------------------)

[Aug 13](https://medium.com/@joaolggross/visualiza%C3%A7%C3%A3o-de-dados-em-python-parte-1-gr%C3%A1ficos-de-dispers%C3%A3o-e-distribui%C3%A7%C3%A3o-com-seaborn-c8636f0202b1?source=post_page-----c8636f0202b1----------------------) · 7 min read

Recentemente escrevi um artigo sobre coleta de dados de BTC (aluguel de ações) da bolsa de valores brasileira utilizando Python. Terminei por apresentar o DataFrame coletado, mas fiquei devendo uma apresentação mais adequada das informações. Recomendo a leitura do artigo, embora não seja um pré-requisito para seguirmos com a visualização.

**[Mercado Financeiro: Como coletar dados de BTC diários com Python, Selenium e Pandas](https://medium.com/@joaolggross/mercado-financeiro-como-coletar-dados-de-btc-di%C3%A1rios-com-python-selenium-e-pandas-59ab0b3b39f0" \t "_blank)**

[Uma forma simples e rápida de obter informações da we](https://medium.com/@joaolggross/mercado-financeiro-como-coletar-dados-de-btc-di%C3%A1rios-com-python-selenium-e-pandas-59ab0b3b39f0" \t "_blank)

[medium.com](https://medium.com/@joaolggross/mercado-financeiro-como-coletar-dados-de-btc-di%C3%A1rios-com-python-selenium-e-pandas-59ab0b3b39f0" \t "_blank)

Neste artigo mostro como criar gráficos de dispersão e distribuição utilizando Seaborn, pois eles irão nos auxiliar a ter uma melhor compreensão sobre as taxas de BTC. Será possível identificar as taxas mais comuns, taxas mínimas e máximas e *outliers*(amostras não representativas). Vamos lá!

Image for post



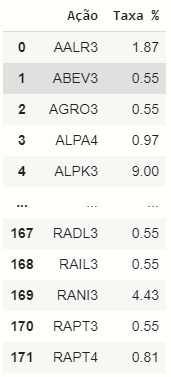
Foto por [Yingchih](https://unsplash.com/@yingchih_hao?utm_source=medium&utm_medium=referral) no [Unsplash](https://unsplash.com/?utm_source=medium&utm_medium=referral)

**Carregando os dados**

Supondo que os dados de taxas de BTC tenham sido armazenados em um arquivo, primeiramente devemos carregá-las com o comando read\_excel() do pacote Pandas.

O parâmetro index\_col setado em 0 indica que a primeira coluna do arquivo é o índice do DataFrame, caso contrário será criada uma coluna extra para os índices. E estrutura do DataFrame df é a seguinte:

Image for post



Estrutura e conteúdo do DataFrame df (dados coletados dia 13/08/2020)

**Criando um gráfico de dispersão**

Gráficos de dispersão são particularmente interessantes quando queremos relacionar duas ou mais variáveis, percebendo a relação entre elas para diferentes medições ou amostras. No nosso caso queremos ver qual a taxa de BTC de cada ação, ou seja, queremos visualizar a relação entre uma variável numérica (taxa de BTC) e uma variável categórico (códigos das ações). Os gráficos de dispersão são mais adequados quando todas as variáveis são numéricas, porém ele também irá servir ao nosso exemplo.

Para criar um gráfico de dispersão utilizamos a biblioteca **Seaborn**que oferece diversas opções de gráficos e muitos recursos de customização.

No trecho de código acima, definimos o estilo do gráfico para white, que corresponde a uma gráfico sem linhas de fundo (outras opções de estilos podem ser encontradas [aqui](https://python-graph-gallery.com/104-seaborn-themes/)). Em seguida configuramos o tamanho da imagem para 10 por 10, unidades em polegadas. Por fim, utilizamos a função scatterplot, indicando os eixos x e y do gráfico com o nome das colunas do nosso DataFrame, e em data indicamos a fonte dos dados. O comando plt.show() é utilizado para mostrar o gráfico na tela. O resultado é o gráfico abaixo:

Image for post

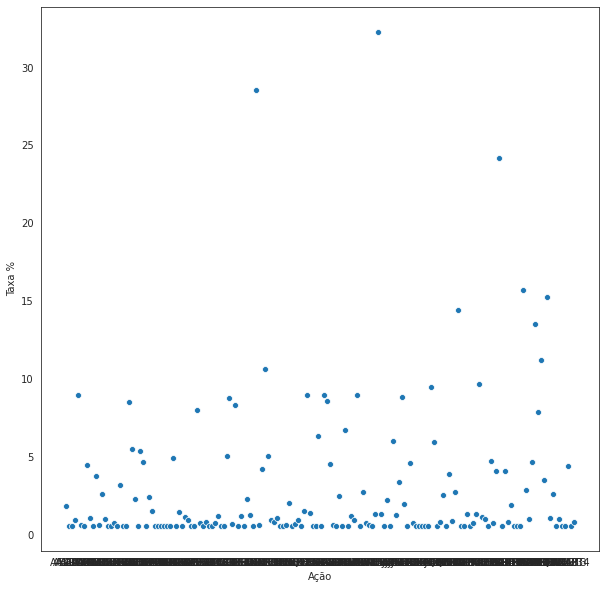


Gráfico de dispersão simples.

A partir do gráfico de dispersão podemos ter uma ideia de quais as taxas de BTC que temos à disposição. Por enquanto o eixo x está poluído, sendo impossível distinguir as ações. Além disso temos poucas taxas com percentual superior a 20, e por estarem no gráfico acabam comprimindo a apresentação da maioria dos valores que se encontra na região de 0 a 15%.

Uma forma de melhorar a apresentação das informações é excluir os valores que se distanciam muito dos demais, os **outliers**, além, é claro, do ajuste dos rótulos do eixo x.

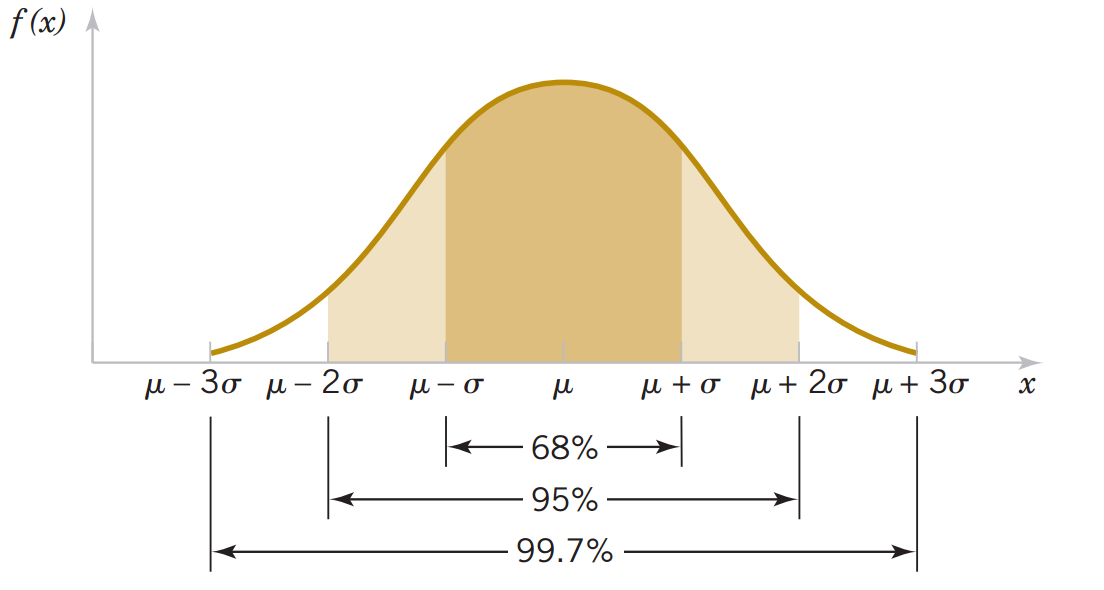
**Melhorando o gráfico de dispersão**

Para remover os outliers precisamos primeiro entender como nossos dados estão distribuídos. Não basta definir um valor de corte para as taxas, pois se todas as taxas subirem corremos o risco de excluir muitas amostras relevantes da visualização.

Uma abordagem bastante utilizada é a **definição de um percentil de amostragem** para obter uma intervalo da amostra que seja **estatisticamente representativo**, ignorando assim, os outliers. Mas antes, devemos entender o que é um percentil.

Resumidamente,*um percentil é um intervalo que corresponde à area de amostragem dentro de uma distribuição normal (gaussiana)*. QUÊ? Calma, acredito que uma imagem torne as coisas mais claras.

Image for post



Distribuição Normal Padrão. Fonte: [UFSC](https://www.inf.ufsc.br/~andre.zibetti/probabilidade/normal.html)

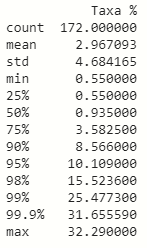
Acima temos a curva de uma distribuição normal padrão. Ela possui dois parâmetros importantes, a **média** dos valores da amostra (μ) e o grau de **dispersão** dos valores da amostra, representado pela **variância**(σ²>0). Mas para se referir à dispersão em termos de unidades padrão, utiliza-se o **desvio padrão** (σ).

A partir da média de valores de amostra, conforme a distribuição normal padrão, sabemos que variando mais ou menos três desvios padrões, 99,7% dos valores da amostra estatisticamente representativos estarão contemplados. Note que um percentil de 99,7% não retorna 99,7% dos valores presentes na amostra, mas 99,7% dos **valores estatisticamente representativos**. Os valores que ficam fora desse intervalo pode ser considerado outliers, visto que distoam muito da média e dos demais valores.

Feita esta explicação, podemos analisar o trecho de código a seguir:

O método describe() possui o parâmetro percentiles que pode ser definido com uma lista de percentis que queremos ver. O resultado do método describe() para o parâmetro informado tem o seguinte resultado:

Image for post



Percentis da amostra em análise.

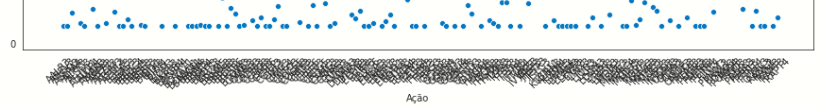
Ao observar os valores, percebemos que uma leve variação de 98% para 99.9% no espaço de amostragem, pouco menos de 2%, já apresenta valores duas vezes maiores. Desse modo estabelecemos um ponto de corte no percentil 98%, que possui valores de até 15.52, o que é inferior ao somatório da média com três desvios padrões (2.967093 +4.684165\*3 = 17.019588). Do percentil 99% em diante já temos valores não significativos, por isso esse intervalo é desconsiderado.

A partir do DataFrame df98 podemos fazer outros ajustes no gráfico para torná-lo mais apresentável:

No método scatterplot() adicionamos alguns parâmetros. Os parâmetros hue e size fazem, respectivamente, com que a cor e tamanho de cada ponto do gráfico mude conforme sua “Taxa %”. Já o parâmetro palette define um conjunto de cores específico para os pontos exibidos no gráfico.

Também são adicionados um título com o comando set\_title(), um novo nome para o eixo x com o comando set\_xlabel() e estabelecido um intervalo de valores no eixo y, de 1 em 1, com o comando yaxis.set\_majos\_locator().

Image for post



Eixo x com todos os rótulos rotacionados 45°.

O ajuste dos rótulos do eixo x apresentou problemas inicialmente, conforme a imagem acima. Apenas rotacionar os rótulos em 45º não retornou um bom resultado, pois há muitos rótulos e eles seguiam compactados por falta de espaço. Uma forma de remediar o problema foi inibir a visualização de alguns deles para que pelo menos parte pudesse constar na visualização. Esse é o objetivo do comando set\_visible(). Por fim, a legenda é posicionada no canto superior esquerdo com o comando plt.legend() e o gráfico é exibido com plt.show().

Image for post

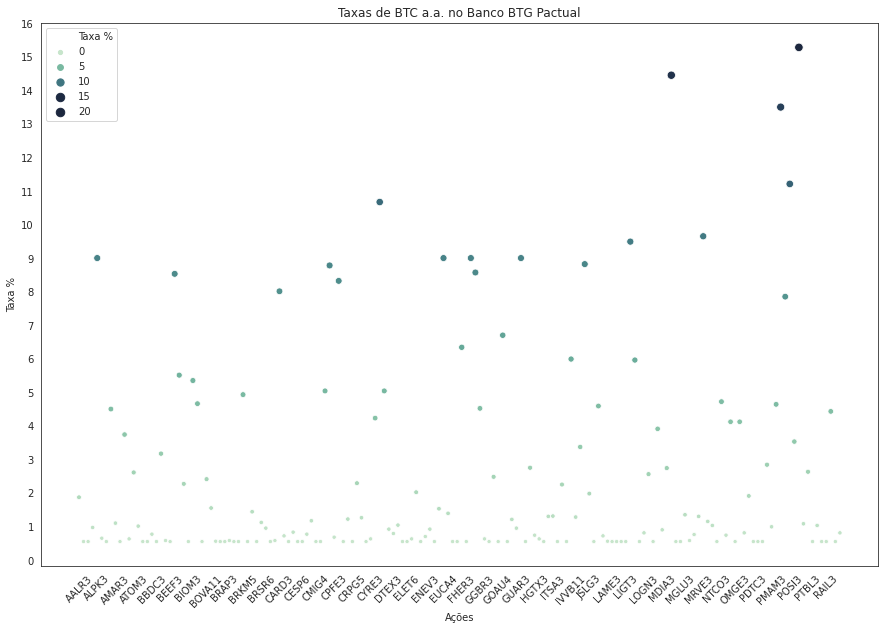


Gráfico de dispersão atualizado

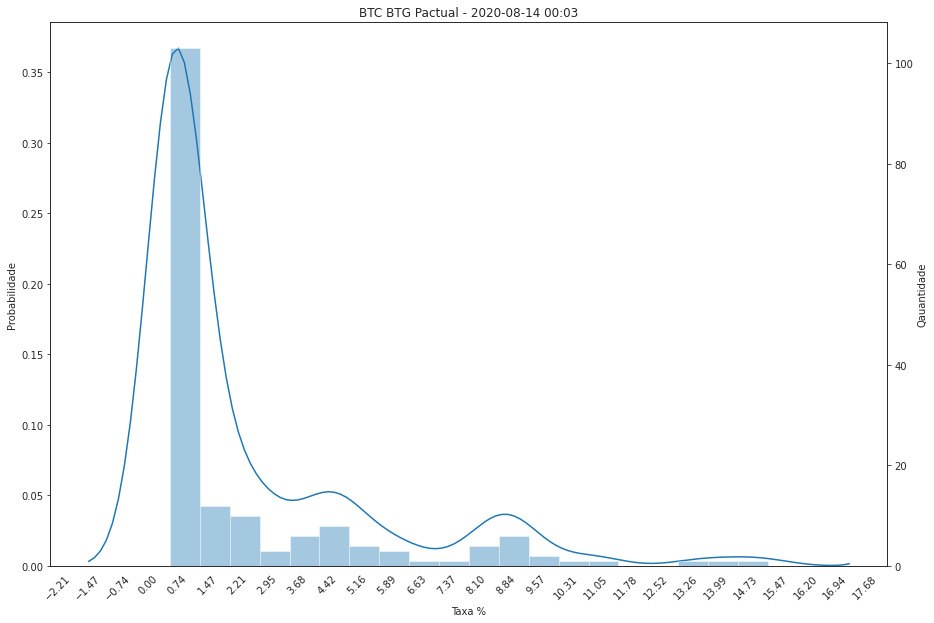
Uma mudança e tanto, não? Conseguimos agora ver alguns códigos das ações no eixo x, além de ter uma ideia sobre a taxa de BTC de cada uma.

**Criando um gráfico de distribuição**

Gráficos de distribuição são interessantes para visualizar a quantidade de valores em cada intervalo de medição. Por exemplo, no gráfico de dispersão, embora seja possível ver que há muitas taxas entre 0 e 1%, ainda é difícil dimensionar a proporção de amostras nesse intervalo de valores ou a quantidade exata. Com um gráfico de distribuição essa visualização é facilitada.

O trecho de código acima utiliza o DataFrame df98 que já possui o corte de dados no percentil 98%. O gráfico gerado é o seguinte:

Image for post



Nesse gráfico temos dois traçados diferentes. O primeiro refere-se à curva KDE ([Kernel Density Estimation](https://en.wikipedia.org/wiki/Kernel_density_estimation)) que estima a probabilidade de um valor aleatório do conjunto de dados estar em cada intervalo de medição. Essa curva é identificada no trecho de código pela variável g1.

Já as barras verticais referem-se à quantidade de valores presentes em cada um dos intervalos. Foram definidos 20 intervalos através da variável bins, a qual é passada como parâmetro na função distplot() da curva g2, o que cria 20 barras verticais.

Perceba que temos duas curvas neste gráfico. Para poder plotar diferentes curvas é necessário fazer uma cópia do plano de visualização através do comando ax.twinx(). O novo plano, ou **layer**, criado a partir do layer ax, é identificado pela variável new\_ax. Nele é possível criar uma curva a mais no gráfico sem afetar a curva principal (g1).

E para o ajuste dos rótulos, criamos um título para o gráfico com set\_title() e dois rótulos para o eixo y, um para cada layer, “Probabilidade” para a curva KDE e “Quantidade” para o histograma. Além disso dividimos o eixo x em 20 intervalos de modo a demarcar o início e fim de cada barra do histograma.

**Considerações Finais**

Neste artigo apresentei dois tipos de gráficos para a visualização de dados, os gráficos de dispersão e de distribuição. Em cada caso foi detalhado o modo de criação das visualizações e as informações que podemos extrair.

Foram apresentados diferentes parâmetros de customização para os gráficos, além da possibilidade de criar duas curvas na mesma visualização.

Com os dois gráficos criados pudemos observar com mais clareza o comportamento das taxas de BTC das ações, identificando as taxas mais comuns e as taxas que destoam demais da média do conjunto de taxas.

Os conceitos aqui apresentados podem ser aplicados em incontáveis outros estudos de caso e certamente irão enriquecer a visualização e compreensão das informações em outros cenários.

* [Seaborn](https://medium.com/tag/seaborn)
* [Python](https://medium.com/tag/python)
* [Pandas](https://medium.com/tag/pandas)
* [Visualização](https://medium.com/tag/visualiza%C3%A7%C3%A3o)
* [Dispersão](https://medium.com/tag/dispers%C3%A3o)