Visualização de dados em Python, Parte 2: Gráficos Treemap Simples e Hierárquico com Ploty

[](https://medium.com/@joaolggross?source=post_page-----383c434ef44----------------------)

[João Gross](https://medium.com/@joaolggross?source=post_page-----383c434ef44----------------------)

[Aug 20](https://medium.com/@joaolggross/visualiza%C3%A7%C3%A3o-de-dados-em-python-parte-2-gr%C3%A1ficos-treemap-simples-e-hier%C3%A1rquico-com-ploty-383c434ef44?source=post_page-----383c434ef44----------------------) · 7 min read

Este artigo é uma continuação da série sobre visualização de dados em Python. O primeiro artigo abordou formas de visualizar dados em gráficos de dispersão e distribuição, além de customizações para deixar as visualizações mais bonitas e organizadas.

**[Visualização de dados em Python, Parte 1: Gráficos de dispersão e distribuição com Seaborn](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" \t "_blank)**

[Recentemente escrevi um artigo sobre coleta de dados de BTC (aluguel de ações) da bolsa de valores brasileira…](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" \t "_blank)

[medium.com](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" \t "_blank)

Neste artigo você vai aprender a fazer gráficos do tipo Treemap simples e hierárquico. É mais uma forma que auxilia a visualização, pois permite o agrupamento de informações a partir de um determinado valor ou intervalo de valores.

Image for post



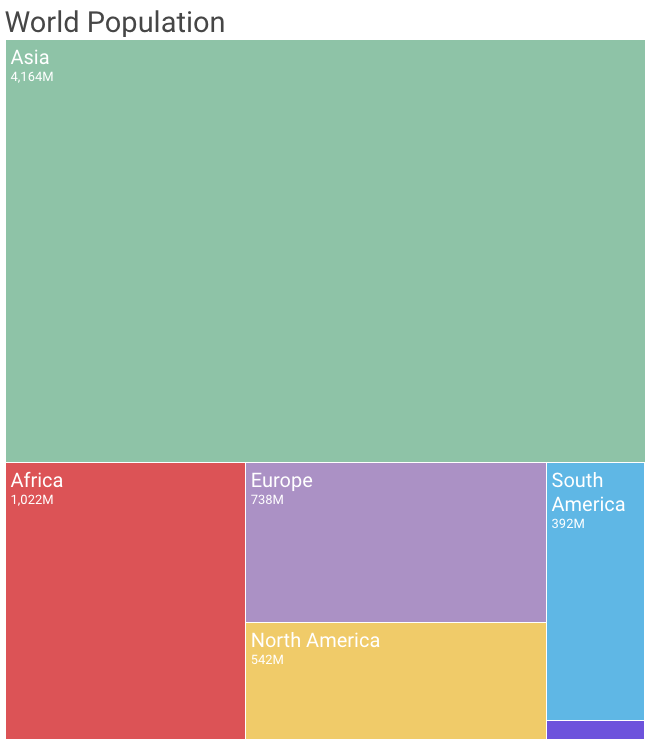
Foto por [Todd Quackenbush](https://unsplash.com/@toddquackenbush?utm_source=medium&utm_medium=referral) no [Unsplash](https://unsplash.com/?utm_source=medium&utm_medium=referral)

**Conhecendo os Treemaps**

Treemaps são visualizações de dados em agrupamentos retangulares. Cada retângulo de informação possui um tamanho proporcional ao todo, logo, quanto maior o retângulo, maior será valor da variável analisada. Assim, ao comparar dois retângulos podemos imediatamente saber qual deles possui um valor maior a partir do tamanho de seu retângulo.

As informações podem estar dispostas em planos sem hierarquia, no qual todas as informações estão no mesmo nível, ou também em planos hierárquicos, no qual as informações são separadas por grupos com base em classificação por diferentes chaves.

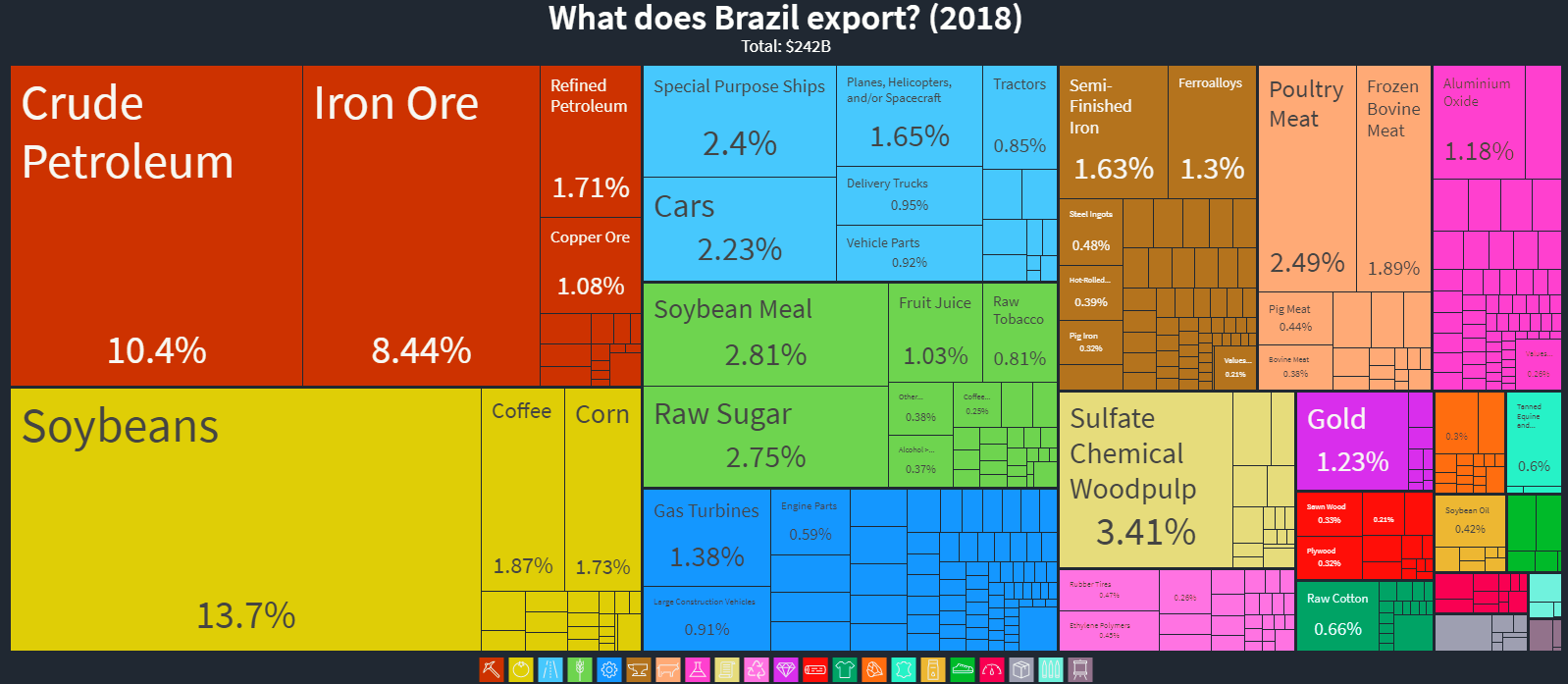
Image for post



Treemap simples. População de cada continente em milhões. Fonte: Infogram.

Na imagem acima é apresentado um Treemap simples, sem divisão hierárquica. Todos os dados estão divididos por apenas uma variável, que neste caso é o tamanho da população.

Image for post



Treemap hierárquico. Produtos exportados pelo Brasil em 2018. Fonte: OEC (the Observatory of the Economic Complexity)

Nesta outra imagem são apresentados todos os produtos exportados pelo Brasil em 2018 para outros países. O valor em % representa o peso daquele determinado produto nas exportações totais do ano. Já as cores representam classes de produtos distintos. Por exemplo, os retângulos em laranja no lado esquerdo do gráfico representam *commodities*, enquanto os retângulos na cor verde abacate, também no lado esquerdo do gráfico, representam grãos. As divisões em retângulos consideram não só o produto como também seu tipo, havendo, portanto, duas variáveis para classificação. Nesse contexto temos uma divisão hierárquica, visto haver uma divisão dentro de outra.

**A Biblioteca Ploty**

Ploty é uma biblioteca gráfica disponível para Python que permite a criação de gráficos interativos de alta qualidade. É possível criar gráficos de linha, de distribuição, de área, de barras, histogramas, mapas de calor, treemaps, dentro outros. **Ploty.py** é *gratuito e open source* e você pode visualizar o código fonte e contribuir no [GitHub](https://github.com/plotly/plotly.py) [1].

Para instalar a biblioteca Ploty basta acessar o CMD do Windows ou terminal do Linux e executar o comando:

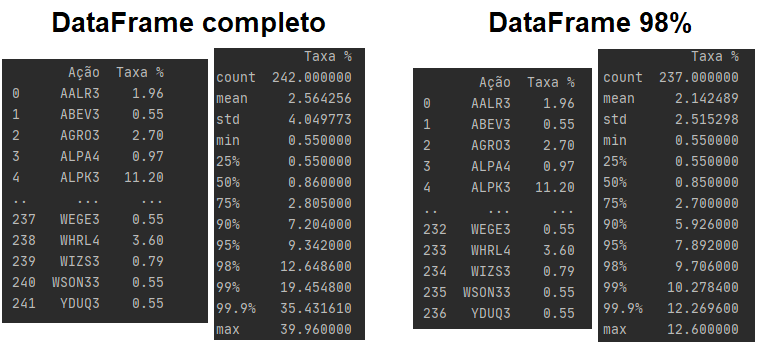
> pip3 install plotly

Caso não consiga executar o comando pelo CMD do Windows, recomendo a leitura do artigo no qual explico [como instalar o Python e o Pip no Windows 10](https://www.linkedin.com/pulse/como-instalar-o-python-3-windows-10-jo%25C3%25A3o-gross/).

**Criando um Treemap Simples**

Na [Parte 1](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) dessa série coletamos os dados de BTC (Aluguel de Ações) no site do Banco BTG Pactual e realizamos um tratamento para excluir os *outliers*(valores coletados que destoaram muito da média do conjunto de valores)*.*Os dados coletados no dia 20/08/2020 apresentam os valores abaixo:

Image for post



DataFrame coletado vs DataFrame tratado.

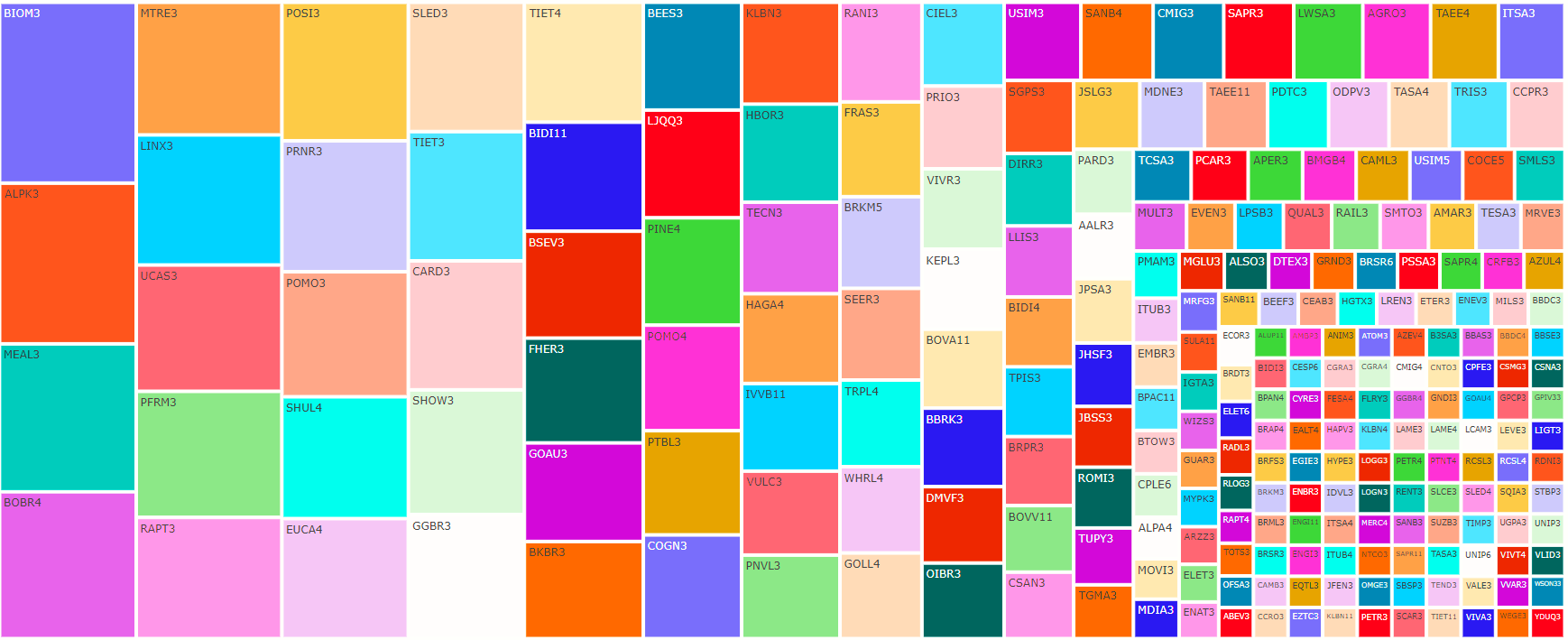
No DataFrame tratado ficamos 237 amostras, enquanto no DataFrame original havia 242, porém o intervalo de valores ficou mais estreito e homogêneo. No DataFrame tratado o intervalo de valores é de 0,5% a 12,6%, enquanto no DataFrame original o intervalo é de 0,5% a 39,9%. Esse corte é importante para suavizar as diferenças entre os retângulos do Treemap.

Com esses dados disponíveis podemos preparar o código para imprimir o primeiro Treemap em tela. Acompanhe o trecho de código a seguir:

Primeiramente importamos a biblioteca ploty com import ploty.express as px. Realizamos o tratamento dos dados, o que gera o DataFrame df98. Depois, criamos o Treemap com px.treemap(). Nos parâmetros da função são utilizados os seguintes parâmetros:

* df98 indica a fonte de dados;
* Em path é especificada a hierarquia de agrupamento por código da ação (coluna “Ação”);
* Em values são especificados os valores utilizados para realizar o cálculo do tamanho de cada retângulo. O dimensionamento dos retângulos é feito a partir a coluna “Taxa %”.

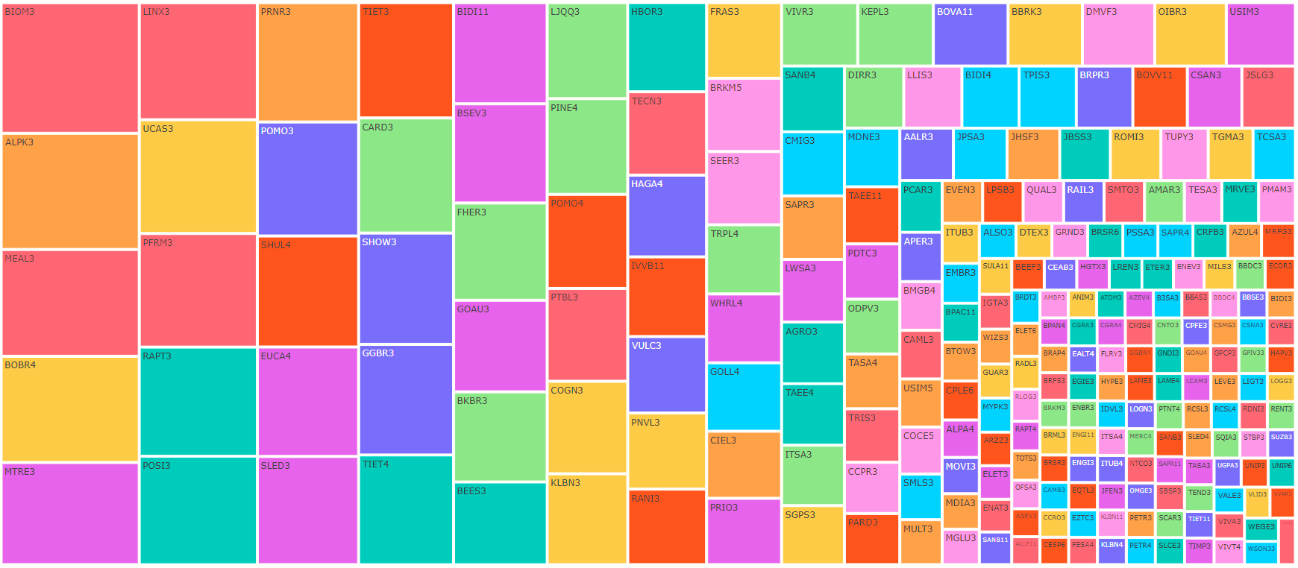
Image for post



Treemap simples. Dados de BTC tratados.

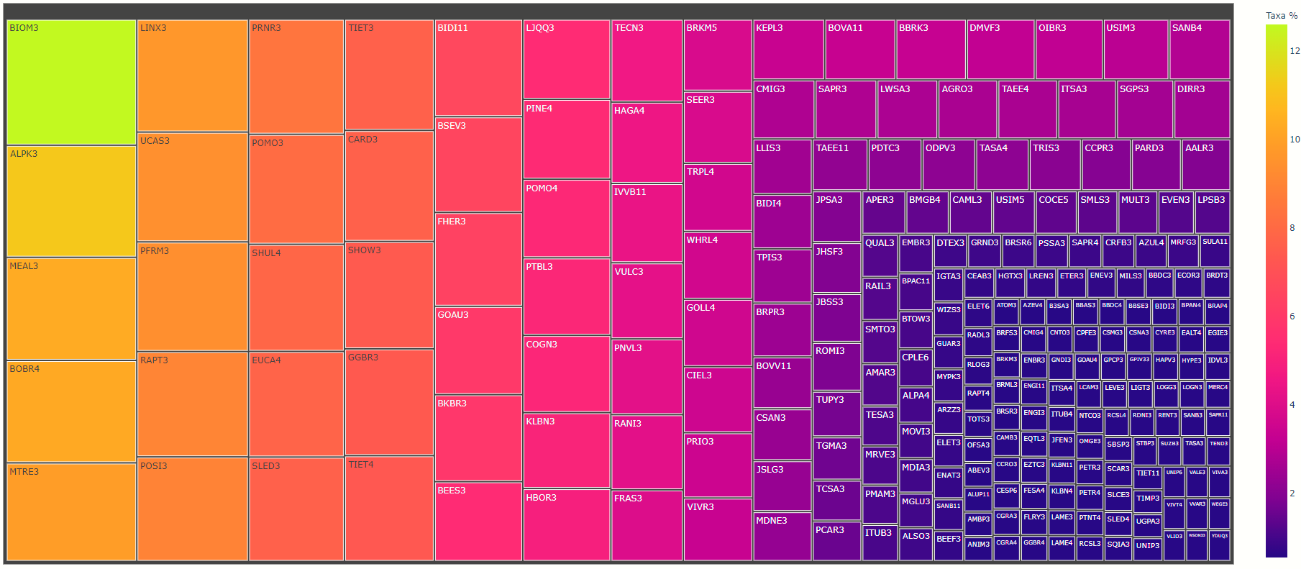
O resultado do treemap pode ser observado na imagem acima. Também é possível alterar as cores especificando o parâmetro color. Atribuindo “Ação” ou “Taxa %” ao parâmetro color temos diferentes resultados:

Image for post



Treemap simples. Dados de BTC tratados. Parâmetro color configurado para “Ação”.

Image for post



Treemap simples. Dados de BTC tratados. Parâmetro color configurado para “Taxa %”.

Ao atribuir uma cor ao Treemap podemos ter comportamentos diferentes de acordo com o tipo de dados que é definido. Ao atribuir "Ação" ao parâmetro color estamos definindo um conjunto de valores não-numéricos, ou categóricos, pois cada ação tem um código específico em formato de texto. Nesse caso as **cores escolhidas são discretas** e atribuídas a cada retângulo conforme a lista de cores do Ploty. Já ao atribuir "Taxa %" estamos passando valores numéricos, e o Ploty define **cores contínuas** aos retângulos.

**Criando um Treemap Hierárquico**

Como nossos dados não possuem nenhum tipo de hierarquia, foi criada uma nova coluna no DataFrame df98, separando as taxas de BTC em 20 grupos. O trecho de código abaixo inclui a criação dos grupos, definição dos rótulos dos grupos e, por fim, a criação do Treemap.

Para criar os grupos, coletamos o valor mínimo e máximo do conjunto e divimos a diferença entre eles em 20 intervalos iguais, o que chamamos de bin\_interval. O primeiro intervalo vai de 0% a 1 bin\_interval. O segundo intervalo em 1 bin\_interval e termina em 2 bin\_interval. O terceiro começa 2 bin\_interval e termina em 3 bin\_interval. O processo segue até o vigésimo grupo.

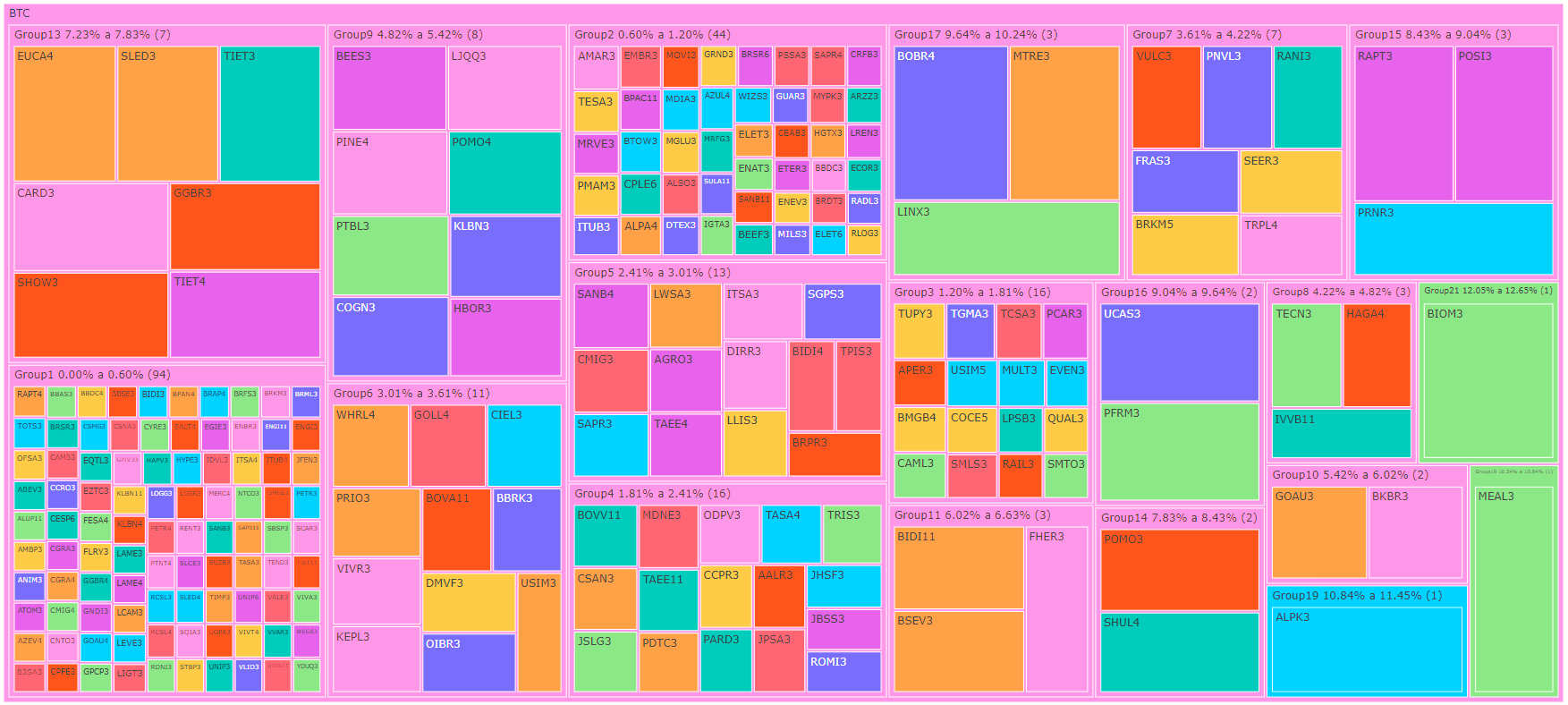
Na sequência é criada uma nova coluna no DataFrame, chamada “Group”. Cada linha do DataFrame recebe um rótulo de grupo específico na coluna “Group” a partir da sua Taxa %. Utilizamos o comando pd.cut() para que essa atribuição de rótulos ocorra. O comando pd.cut() pega as taxas df98["Taxa %"], separa os dados nos intervalos que especificamos em bins=ranges e atribui os rótulos para cada intervalo de acordo com labels=group\_names.

Após atribuir os rótulos no formato “Group1 0.00% a 0.60%” a cada linha do DataFrame, é realizado um [groupby](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.groupby.html)("Group").agg("count") para contar a quantidade de taxas em cada intervalo, representado por cada um dos rótulos dos grupos. Essa quantidade é utilizada para atualizar os rótulos dos grupos, indicando no próprio rótulo a quantidade de elementos daquele grupo. Assim, os rótulos ficam com o formato:

"Group<número do grupo> <taxa inicial intervalo>% a <taxa final intervalo>% (quantidade elementos intervalo)"Ex.: "Group1 0.00% a 0.60% (94)"

Por fim, é criada uma nova coluna “BTC” no DataFrame, para ser a raiz da hierarquia. Todos os elementos possuem essa coluna configurada com o mesmo valor. O Treemap é configurado com a hierarquia “BTC”->“Group”->“Ação”, os retângulos são dimensionados pelo valor da taxa % e coloridos de acordo com o código da ação.

Image for post



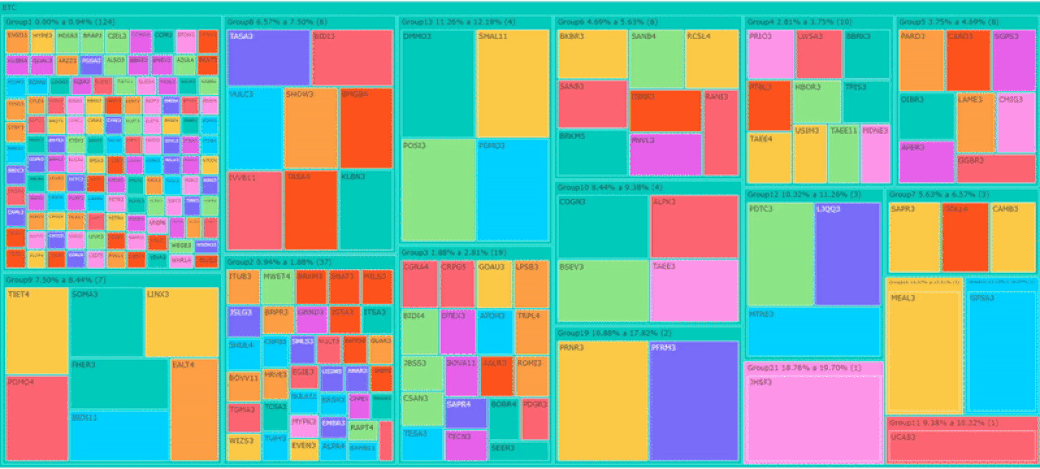
Treemap hierárquico. Dados tratados de BTC.

Acima podemos observar em detalhes o gráfico Treemap criado. Podemos observar que para cada grupo um retângulo foi criado e dentro de cada grupo retângulos menores indicam cada ação. Os rótulos de cada grupo possuem a quantidade de elementos daquele grupo, conforme a explicação sobre a atualização dos rótulos utilizando groupby(). Com essa divisão em grupos é possível ver quais ações possuem mais afinidade por taxa de aluguel de ações e quais as quantidades de ações em cada grupo.

**Interação com gráficos criados com Ploty**

Gráficos criados com Ploty possuem diferentes customizações de interação. Isso daria por si um artigo completo e não será abordado aqui. Entretando, abaixo você pode ter uma ideia sobre de que modo é possível interagir com as células do gráfico. Basta passar o mouse sobre cada retângulo para ter acesso a mais informações aquele elemento.

Image for post



Interação fornecida pelo Ploty para Treemaps.

**Considerações Finais**

Neste artigo você aprendeu a definição de dois tipos de Treemaps, simples e hierárquico. Conheceu um pouco sobre a biblioteca Ploty e descobriu como é possível criar gráficos Treemap com ela em Python. Os trechos de código apresentados aqui estão disponível no meu [GitHub](https://github.com/jlggross/articles).

**Referências**

1. Ploty — Python Open Source Graphing Library: <https://plotly.com/python/>