**CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA**
**Faculdade de Tecnologia de Jundiaí – "Deputado Ary Fossen"**
**Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas**

Jônatas Domingues de Almeida Chizzolini

# ESTUDO COMPARATIVO DE BIBLIOTECAS PYTHON PARA VISUALIZAÇÃO DE DADOS: MATPLOTLIB E SEABORN

**Jundiaí**
**2023**

**Jônatas Domingues de Almeida Chizzolini**

# ESTUDO COMPARATIVO DE BIBLIOTECAS PYTHON PARA VISUALIZAÇÃO DE DADOS: MATPLOTLIB E SEABORN

Trabalho de Graduação apresentado à Faculdade de Tecnologia de Jundiaí - “Deputado Ary Fossen” como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, sob a orientação do Professor Me. Peter Jandl Junior.

**Jundiaí**
**2023**

Dedico este trabalho
à minha família,
minha esposa,
e ao meu filho,
Benjamim.

# AGRADECIMENTOS

Agradeço em primeiro lugar ao Deus Pai, Criador dos céus e da Terra; agradeço-o pela minha vida e por me conceder a graça de poder trabalhar e estudar, especialmente pela possibilidade de concluir o meu curso de graduação de Tecnologia em Análise e Desenvolvimento de Sistemas na Faculdade de Tecnologia de Jundiaí – “Deputado Ary Fossen”.

Agradeço a minha família de origem, meus pais (Gilberto e Vera) e meu irmão (Samuel), pelo suporte e apoio inconteste, pelos conselhos e encorajamento os quais têm me concedido durante toda a minha trajetória. Agradeço-os por desde cedo terem me indicado o caminho a seguir e terem me instruído adequadamente de modo que eu viesse a trilhá-lo e dele não desviar.

Agradeço à minha esposa (Giseli) pelo seu amor, cuidado e suporte que sempre reserva a mim e aos projetos os quais persigo nesta jornada. Seu companheirismo e conselhos têm sido fundamentais para que eu tenha sucesso nos meus objetivos.

Devo destacar que se hoje eu entrego esse TCC, e tenho a condição de colar grau em Análise e Desenvolvimento de Sistemas, isso se deve em grande medida pela insistência e perseverança por parte da minha esposa e da minha família no sentido de me incentivar para a conclusão deste curso.

Agradeço a FATEC de Jundiaí, e ao corpo docente da área de Tecnologia da Informação, pelo acolhimento e ensino prestado com qualidade e excelência. Agradeço, pois, no período em que estive na faculdade, todo o suporte necessário para a conclusão do curso foi devidamente prestado, e toda a expectativa que eu tinha com relação a instituição foi superada.

Finalmente, gostaria de reservar um agradecimento especial ao coordenador do curso e meu orientador, o Professor Mestre Peter Jandl Junior, por todo o suporte e apoio que tem dedicado a mim desde antes mesmo do meu ingresso na FATEC. O Professor Peter é uma pessoa extremamente solícita e que sempre tem uma palavra positiva no sentido de fazer com que seus alunos acreditem em si mesmos. Destaco um caráter de extrema humanidade do Professor, o qual já não é mais tão comum no contexto atual em que vivemos.

Obrigado.

Educação é uma descoberta progressiva de nossa própria ignorância.

Voltaire

CHIZZOLINI, Jônatas Domingues de Almeida Chizzolini. **Estudo comparativo de bibliotecas Python para visualização de dados: Matplotlib e Seaborn**. 69 p. Trabalho de Conclusão de Curso de Tecnólogo em Análise e Desenvolvimento de Sistemas. Faculdade de Tecnologia de Jundiaí - “Deputado Ary Fossen”. Centro Estadual de Educação Tecnológica Paula Souza. Jundiaí. 2023.

## RESUMO

Este trabalho busca discorrer sobre a importância da visualização de dados no contexto atual em que uma enorme quantidade de dados é gerada diariamente. Nesse sentido a ideia é explorar os gráficos como um importante conjunto de ferramentas que nos dão a capacidade de ressignificar os dados armazenados nos mais diversos tipos de repositórios corporativos e governamentais. Assim, este texto explora duas bibliotecas da linguagem de programação Python, as quais tem por função precípua a criação de gráficos a partir de grandes conjuntos de dados, a saber: Matplotlib, e Seaborn. Ao analisá-las a ideia foi fazer um estudo comparativo entre as principais vantagens e desvantagens de cada uma, além de traçar um paralelo do uso destas com o uso do Microsoft Excel, ferramenta *benchmark* de mercado em função da sua grande disseminação e do seu grande tempo de uso nos mais variados setores da economia global. Ao término desta pesquisa é possível ter uma boa ideia do funcionamento dessas bibliotecas, sua relevância, bem como os pontos fortes e fracos de cada uma delas.

**Palavras-chave:** Visualização de dados. Gráficos. Python. Matplotlib. Seaborn.

CHIZZOLINI, Jônatas Domingues de Almeida. **Comparative study of Python's libraries for data visualization: Matplotlib and Seaborn.** 69 p. End-of-course paper in Technologist Degree in Computer System Analysis and Development. Faculdade de Tecnologia de Jundiaí - “Deputado Ary Fossen”. Centro Estadual de Educação Tecnológica Paula Souza. Jundiaí. 2023.

## ABSTRACT

This work seeks to discuss the importance of data visualization in the current context in which a huge amount of data is generated daily. In this sense, the idea is to explore graphics as an important set of tools that give us the ability to reframe data stored in the most diverse types of corporate and government repositories. This text explores two libraries of the Python programming language, which have as their primary function the creation of graphs from large datasets, which are: Matplotlib, and Seaborn. When analyzing them, the idea was to carry out a comparative study between the main advantages and disadvantages of each one, in addition to drawing a parallel between their use and the use of Microsoft Excel, a market benchmark tool due to its wide dissemination and its great usage time in the most varied sectors of the global economy. At the end of this research, it is possible to have a good idea of how these libraries work, their relevance, as well as the strengths and weaknesses of each one of them.

**Keywords:** Data visualization. Charts. Python. Matplotlib. Seaborn.

# LISTA DE FIGURAS

# SUMÁRIO

# 1 INTRODUÇÃO

A ideia deste trabalho é tratar sobre a visualização de dados, sua função e relevância, bem como traçar breve estudo comparativo entre algumas ferramentas de código aberto (em linguagem Python) para a criação de gráficos, a saber: Matplotlib, e Seaborn.

No contexto atual em que os dados gerados e armazenados crescem exponencialmente, é preciso que haja a capacidade de esses serem interpretados e que a partir deles se gerem informação e conhecimento.

O Big Data é o conjunto de ferramentas e tecnologias responsável pela coleta, armazenamento, processamento e disponibilização dessa grande quantidade de dados, e ao mesmo tempo a Ciência de dados atua na ponta da interpretação e criação de significado para esses fragmentos de informação. É importante notar que a Ciência de dados também é composta por áreas menores de conhecimento que juntas contribuem no sentido de alcançar o objetivo de dar ressignificado aos dados.

Nesse sentido, um desses componentes é a Visualização de dados que auxilia na compreensão de grades conjuntos de dados, os quais não apresentam um sentido completo e aparente desde o princípio. A prática de criar gráficos e ilustrações a partir de dados organizados e armazenados em tabelas pode trazer à tona tendências que antes não estavam claras, demonstrar relações das quais não se tinham conhecimento inicialmente. E a partir daí é possível evoluir nas análises e na construção de conhecimento advindo de grandes massas de dados brutos.

Os estudiosos de dados dispõem de algumas ferramentas para esta tarefa, como por exemplo o Microsoft Excel e o Matlab, sendo que cada uma delas possui seus benefícios e qualidades, mas como qualquer outra ferramenta também possuem limitações, além de, no caso dessas duas citadas, ambas serem de licença proprietária incorrendo em custos para o seu uso.

Nesse cenário é que aparecem outras opções de instrumento para trabalho com visualização de dados tais como Matplotlib, Seaborn e outros. Essas citadas sendo bibliotecas para execução em ambiente de programação da linguagem Python. Elas possuem uma ampla variedade de gráficos os quais podem ser criados a partir de estruturas de dados Python, além de permitirem várias possibilidades de edição e estilização das figuras, tudo com o objetivo de que sejam criadas figuras agradáveis

e com recursos que facilitem e viabilizem uma melhor compressão dos dados apresentados.

É fundamental saber que essas não são as únicas bibliotecas que disponibilizam tais recursos, bem como existem outras linguagens de programação com forte aplicação em Ciência e análise de dados, tais como o R, por exemplo. Essa ampla gama de tecnologia disponível demonstra o caráter fundamental do tema.

Assim, o presente estudo tem como pretensão fazer uma breve apresentação do tema da visualização de dados, discorrendo sobre seu significado, aplicações, além de conceitos e práticas recomendadas para a elaboração dos gráficos e figuras. Tem também como objetivo a elaboração de uma comparação inicial entre alguns gráficos gerados pelas bibliotecas Python mencionadas, Matplotlib e Seaborn, bem como pelo Excel, discorrendo sobre facilidade de uso e possibilidades geradas pelas ferramentas abordadas.

O presente texto sobre a Visualização de dados e sobre as ferramentas e técnicas utilizadas para desenvolvê-la é importante à medida que contribui para a disseminação desse conhecimento, e propaga sobre a existência dessas bibliotecas repletas de recursos e capazes de elaborar gráficos com alta qualidade, sob um “custo” adequado de esforço criativo.

# 2 VISUALIZAÇÃO DE DADOS E SUA RELEVÂNCIA

O aumento considerável dos dados gerados pela humanidade, e o seu armazenamento só fazem sentido no caso desses fragmentos de informação puderem ser interpretadas tornando-se assim informação e conhecimento os quais possam gerar real valor para a humanidade. Caso não seja dessa maneira, o custo dispendido para a montagem e manutenção de tantos servidores para armazenamento de dados passa a ser algo sem sentido.

Uma forma dessa geração de conhecimento ter vez é viabilizada pela Ciência de dados, área de conhecimento relativamente nova e que vem crescendo em popularidade. Trata-se de um ramo que vem sendo muito reconhecido no mercado de trabalho, muito em função dessa sua capacidade de viabilizar a criação de conhecimento através de tantos dados brutos existentes e disponíveis.

Com o uso adequado desses dados é possível a criação de novos produtos e a abertura de novos mercados pelas grandes corporações; é viabilizada a elaboração e execução de políticas públicas com maior respaldo científico e embasamento, capazes de ajudar mais substancialmente a um maior número de pessoas; possibilita-se a descoberta e a invenção de novas tecnologias e saberes nos mais variados ramos da ciência que podem ajudar e contribuir com a evolução da humanidade.

Dentro das Ciências de dados, e emprestando arcabouço teórico científico de outras disciplinais, tais como a Estatística, por exemplo, existe um conjunto de técnicas e saberes conhecido como Análise Exploratória de Dados (AED), ou na sua nomenclatura em inglês, *Exploratory Data Analisys* (EDA). Essa subárea da Ciência de dados tem direta participação no processo de transformação de dados brutos em informação e conhecimento.

Aliado a AED existe também um outro subconjunto da Ciência de dados que é a Visualização de dados, cuja função peremptória é a de representar gráfica e visualmente as informações que outrora eram disponibilizadas majoritariamente de maneira tabular. Esse saber não é novo, e dispões de muitas ferramentas com um elevado grau de desenvolvimento que estão no mercado já há muito tempo, podendo-se destacar por exemplo o Microsoft Excel e o Matlab.

Esses dois *softwares* são de licença proprietária e possuem um alto custo para sua utilização em contextos profissionais e corporativos, ou até mesmo para uso

doméstico. Assim, ao longo do tempo ferramentas sob licença de software livre foram desenvolvidas e disponibilizadas para uso da comunidade acadêmica e público em geral.

Neste contexto, verifica-se que as técnicas de Visualização de dados têm potencial de se desenvolver nas mais variadas áreas: no meio científico, educacional, dentre as grandes corporações, e até mesmo nos editorais de meios de informação (jornais escritos, jornalismo televisionado).

A maneira de se representar informações visualmente facilita muito a compreensão por parte dos seres humanos e esse é um dos motivos da crescente utilização de gráficos na produção e divulgação de conhecimento. Conforme os autores Alexandre e Tavares,

> A visualização explora principalmente o sentido humano que possui maior aptidão para captação de informação temporal: a visão. Além de ser o primeiro componente do sistema sensorial, a visão é o sentido adquirido mais rapidamente pelo cérebro e possui ainda capacidade de paralelismo... (ALEXANDRE; TAVARES, 2007, p. 4)

## 2.1 Evolução da Visualização de informação ao longo da História

Essa prática de comunicação por meio de figuras e desenhos é antiga e amplamente difundida na humanidade. Desde os primórdios da História da espécie humana observa-se a existência de pinturas rupestres. As imagens comunicam mais rapidamente ao nosso cérebro, e têm o potencial de alcançar mais pessoas conferindo significado até mesmo àqueles que não são capazes de ler e escrever. Como ilustração segue uma imagem que se encontra no Sítio Arqueológico de Lascaux situado na França. Em geral, as pinturas encontradas neste Sítio datam de algo em torno de 16000 anos atrás. Do estudo dessas sabe-se que tinham por objetivo a comunicação e a disseminação de informação no seio das sociedades as quais as produziam.

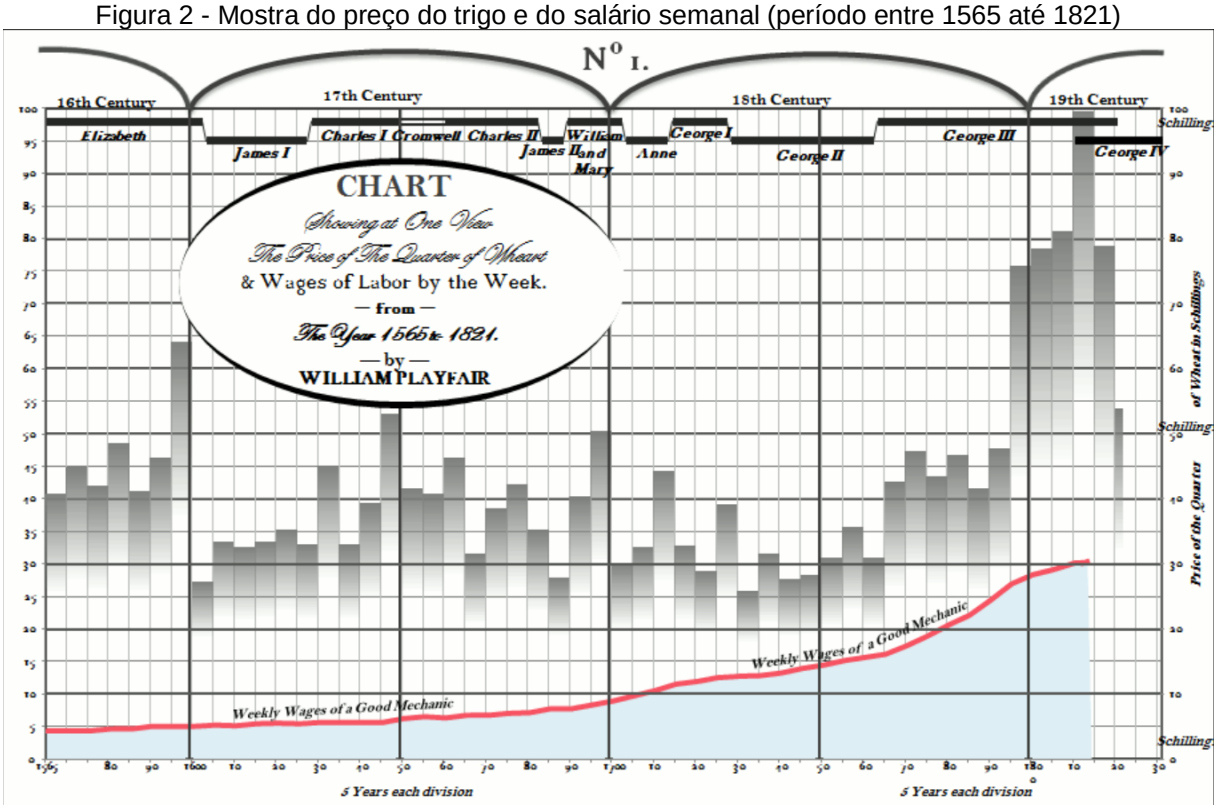
Figura 1 - Imagem encontrada no Sítio arqueológico de Lascaux (França)

Fonte: https://archeologie.culture.gouv.fr/lascaux/en/mediatheque?page=%2C3

Em tempos mais próximos da contemporaneidade, é possível encontrar alguns exemplos de visualização de dados e informações mais próximos do que costumamos ver atualmente. No século XIX existem alguns exemplos importantes os quais serão apresentados agora.

O economista escocês, William Playfair incluiu nas suas publicações gráficos de linhas, de barras, e de pizza. Por esse motivo, Playfair é tido como o pai dos gráficos estatísticos. Todos esses são ainda amplamente utilizados quando se quer demonstrar métricas de tendência, de variações e evolução ao longo de um período. Abaixo segue ilustração de um gráfico elaborado por Playfair no ano de 1821 em que ele exibe e compara a evolução de duas variáveis num período que abarca desde 1565 até o presente ano da elaboração do gráfico, o ano de 1821.

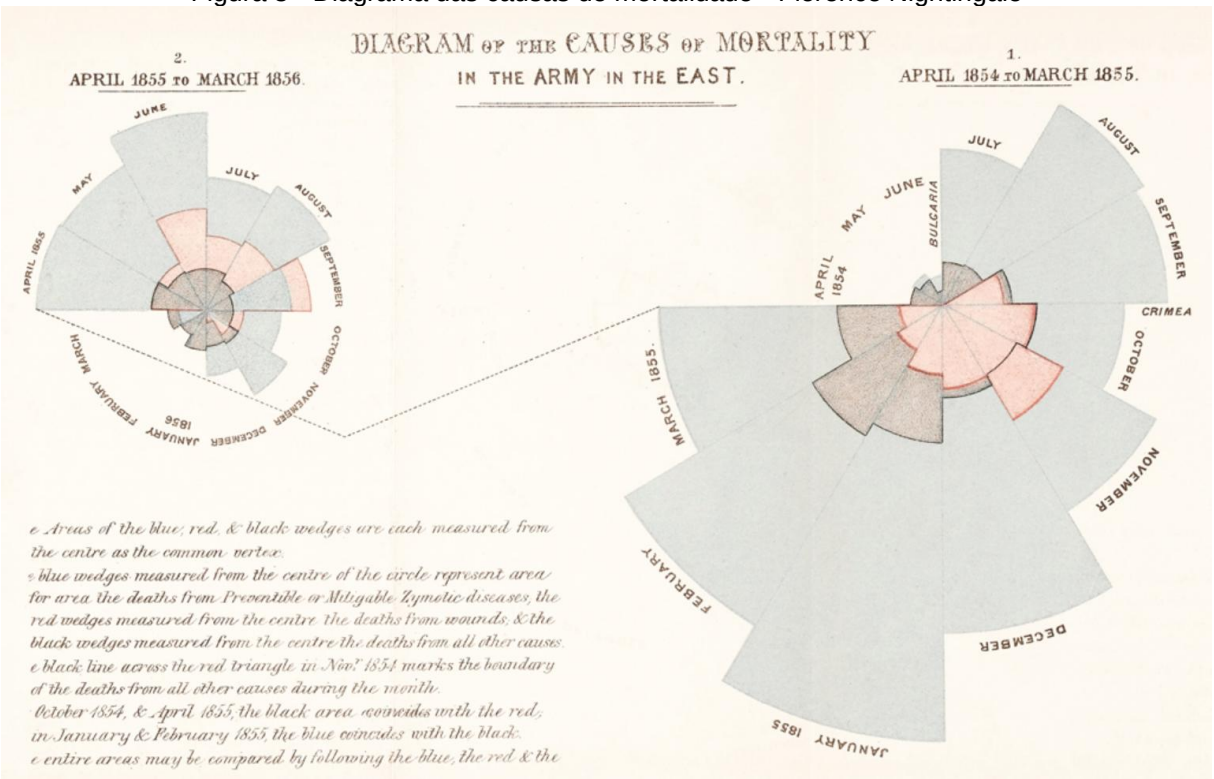
Figura 2 - Mostra do preço do trigo e do salário semanal (período entre 1565 até 1821)




Fonte: https://i0.wp.com/www.excelcharts.com/blog/wp-content/uploads/2011/12/william-playfair-wheat-excel.png

No ano de 1858, Florence Nightingale, tida como a precursora da enfermagem moderna, elaborou o relatório *Notes on matters affecting the health, efficiency, and hospital administration of the British Army* (Nightingale, 1858), cujo objetivo era de trazer informação sobre as condições de saúde durante o período da Guerra da Crimeia (1853-1856) opondo em lados distintos o Império Russo de um lado, e o Império Otomano, Grã-Bretanha e a França do outro. Nessa publicação ela incluiu um gráfico que se tornou notável como as Rosas de Florence no qual ela expõe as causas de mortalidade em um diagrama contendo dois gráficos que comparam a evolução das tendências em dois momentos distintos do conflito. Abaixo segue representação do seu trabalho original.

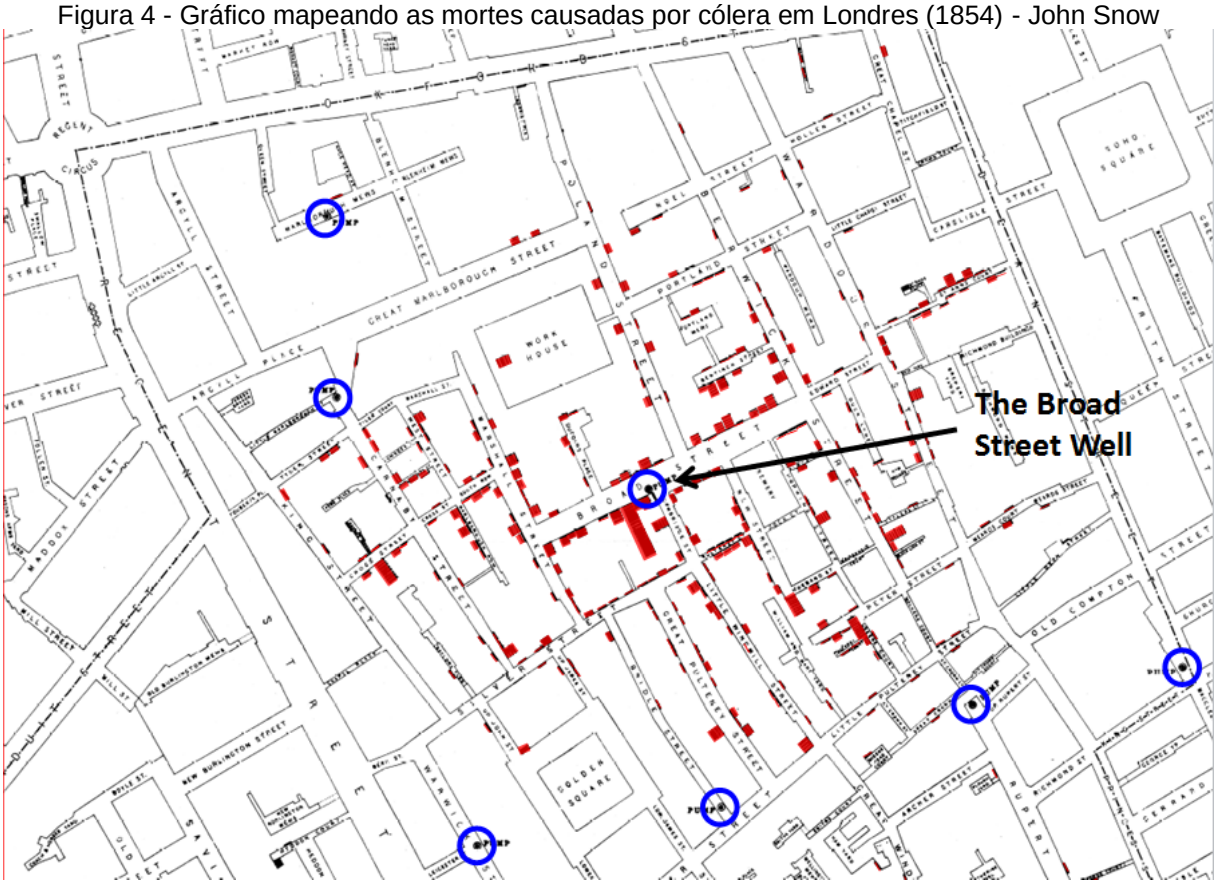
Figura 3 - Diagrama das causas de mortalidade - Florence Nightingale




Fonte: https://www.sciencenews.org/wp-content/uploads/2020/05/050920_scivi_inline1.jpg

Por volta do mesmo período, no ano de 1854, o epidemiologista Dr. John Snow, elaborou um gráfico a fim de compreender as causas e circunstâncias que pudessem explicar uma mortandade que estava ocorrendo por cólera na cidade de Londres naquele período. Nesse gráfico, ele pontua num plano cartesiano os locais e as quantidades de mortes por localidade fazendo com que alguns pontos (em formato de barras) seja maior do que outros. Assim ele relacionou as casas onde tinham mortes registradas com os pontos de fornecimento de água afim de tentar encontrar uma relação de causa-efeito, e eventualmente encontrar possíveis pontos de contaminação. Abaixo segue imagem do gráfico criado em meados do século XIX, conforme mencionado anteriormente.

Figura 4 - Gráfico mapeando as mortes causadas por cólera em Londres (1854) - John Snow




Fonte: https://scienceblogs.com/files/significantfigures/files/2013/03/Cholerafull.png

É importante destacar que a figura acima é ilustrativa da original, tendo em vista que nela há elementos os quais não estão presentes na versão de 1854 elaborada pelo Dr. Snow, como se vê: círculos na cor azul que evidenciam no mapa os locais onde têm bombas de água, e a cor avermelhada dando destaque as barras originais que quantificam as mortes por localidade.

## 2.2 Relevância das imagens

Conforme visto na seção anterior, a prática de apresentar informações de forma gráfica remonta aos primórdios da História da humanidade e vem evoluindo desde então.

A mente humana tem mais facilidade em interpretar e absorver informações quando estas encontram-se exibidas de maneira visual. Assim a existência de grandes conjuntos de dados dispostos de maneira tabular pode não ter o seu pleno

potencial explorado pelos seus leitores, ao passo que se desse conjunto de dados forem elaborados bons gráficos, as informações tendem a ser mais bem aproveitadas.

A representação de informações por intermédio de imagens tem ainda como uma de suas funções a disseminação de conhecimento entre um maior número de pessoas, tendo em vista que quando bem escolhidos e bem elaborados, os gráficos são de fácil entendimento por um grupo de leigos que não necessariamente possuem conhecimentos de estatística ou ciência de dados.

A visualização de dados tem se desenvolvido bastante nas últimas décadas, se constituindo como um ramo de conhecimento e como disciplina acadêmica em diversas universidades ao redor do globo.

Numa época de expansão de ferramentas de *Big data*, a visualização de dados ganha cada vez mais importância pois por meio dela é que os dados podem ganhar significação e contar uma história que faça mais sentido. Transformação de dados em informação e posteriormente em conhecimento.

## 2.3 Revisão bibliográfica do tema

Com o objetivo de elucidar ainda mais os pontos apresentados até aqui neste capítulo, bem como de conferir consistência e rigor científico a este texto, apresenta-se nesta seção algumas passagens da pesquisa bibliográfica realizada com o intuito de embasar a continuidade desta monografia.

Assim, de acordo com Tukey (1977), conforme citado por Waskom (2021, p. 1), “a visualização de dados é uma parte indispensável do processo científico. Visualizações efetivas permitirão ao cientista tanto a compreender os seus próprios dados quanto a comunicar as suas percepções aos outros.”

Quanto a importância que a área de visualização de dados vem ganhando no meio científico e no mercado de trabalho nos últimos anos, tal informação apoia-se na passagem de Irizarry (2019, p. 115), “A crescente disponibilidade de conjuntos de dados informativo e de ferramentas computacionais vem levando a um aumento na confiança nas visualizações de dados em muitas indústrias, academia e governo.”

Ainda no que diz respeito a relevância do estudo da Visualização de dados, de acordo com Silva (2019, p.220),

> A visualização de dados crescerá em importância no curto prazo. Se trata de um termo geral que descreve qualquer esforço para ajudar as pessoas a entender a importância dos dados, colocando-os em um contexto visual. Padrões, tendências e correlações que podem passar despercebidas em dados baseados em texto podem ser mais facilmente expostos e reconhecidos com o software de visualização de dados (SILVA, 2019, p. 220)

Acerca do *Big data* e do desafio de gerar informação e conhecimento a partir desses repositórios de dados (conhecidos como *data lakes* no contexto do *Big data*), de acordo com Mclnerny et al (2014), conforme citado por Silva (2019, p.220),

> Apesar de todos os benefícios que o *big data* oferece, o processamento de dados dessa magnitude não é necessário para todas as empresas e só crescerá à medida que tivermos novas aplicações em nossas mãos para usá-las. Agora, o desafio não se baseia em gerar e capturar mais dados, mas baseia-se na descoberta de novas maneiras de condensar, interpretar e tomar decisões sobre esses dados. A chave está na visualização de dados, que é um dos métodos mais novos e mais conhecidos para reduzir ou ilustrar dados de maneira simplificada e visual (SILVA, 2019, p. 220)

No que diz respeito ao avanço técnico e teórico da área de visualizações de dados, há uma ilustração disso no artigo “Princípios de efetiva visualização de dados” de autoria do professor Midway (2020, p. 2) no qual ele menciona sobre a obra “A gramática dos gráficos” do autor Wilkinson (2013). Na ocasião ele explica a intenção da obra e da ideia de Gramática de dados de Wilkinson como sendo uma subdivisão dos gráficos nas suas partes constituintes (seus dados, escalas, coordenadas, geometrias), tal qual a gramática convencional subdivide uma sentença em substantivos, verbos, e outros elementos da escrita. Além disso ele afirma que esse referencial teórico foi utilizado na implementação da reconhecida biblioteca Ggplot2 para criação de gráficos em R., a qual é apresentada em profundidade na obra do autor Rafael Irizarry (2019) a qual é citada e referenciada neste trabalho.

Quanto aos conceitos abordados anteriormente de Análise Exploratória de Dados (AED), e quanto a intersecção que existe entre essa e o campo da Visualização de dados, esta relação fica patente quando o autor Irizarry (2019, p. 117) afirma que “a visualização de dados é a ferramenta mais forte do que chamamos de Análise exploratória de dados”; na mesma passagem, Irizarry para demonstrar a relevância das imagens, cita textualmente o autor Tukey[1], o qual é tido por ele como sendo o pai

[1] https://en.wikipedia.org/wiki/John_Tukey (mesma referência encontrada em Irizarry).

da AED, “o maior valor de uma imagem é quando ela nos força a enxergar o que nós nunca esperávamos ver”.

Destarte, tendo como referência as informações apresentadas é factível constatar a relevância do tema das visualizações de dados, das imagens em si, além de atestar o quão rico teoricamente este campo científico pode se tornar. Vale notar que esse extenso cabedal acadêmico em torno do tema viabiliza o seu amadurecimento e desenvolvimento como um consolidado ramo dentro do conjunto das Ciências de dados, além de trazer benefícios práticos para a evolução do conhecimento e dos seres humanos como sociedade e civilização.

# 3 BIBLIOTECAS PYTHON PARA CRIAÇÃO DE GRÁFICOS

O Python é uma linguagem de programação que foi desenvolvida pelo programador holandês Van Rossum, o qual lançou a versão Python 1.0 no ano de 1991.

O Python possui diversas características as quais serão abordadas aqui neste texto, no entanto um das que parece ser a principal e mais marcante é o fato dela ser pautada por grande simplicidade e possuir uma sintaxe bastante direta, parecendo-se algumas vezes até mesmo com um pseudocódigo (CHALLENGER-PÉREZ; DÍAZ-RICARDO; BECERRA-GARCÍA, 2014).

Segundo (SRINATH, 2017, p. 354), “sua sintaxe permite aos programadores expressar conceitos em menos linhas de código do que seria possível em linguagens como o C”. Abaixo um comparativo entre C e Python para o tradicional código “Olá Mundo” (CHALLENGER-PÉREZ; DÍAZ-RICARDO; BECERRA-GARCÍA, 2014).

Figura 5 - Código "Olá Mundo" em C

```
#include <iostream>
using namespace std;
int main()
{
cout << “Hello World” <<
endl;
return 0;
}
```

Fonte: CHALLENGER-PÉREZ; DÍAZ-RICARDO; BECERRA-GARCÍA, 2014

Figura 6 - Código "Olá Mundo" em Python

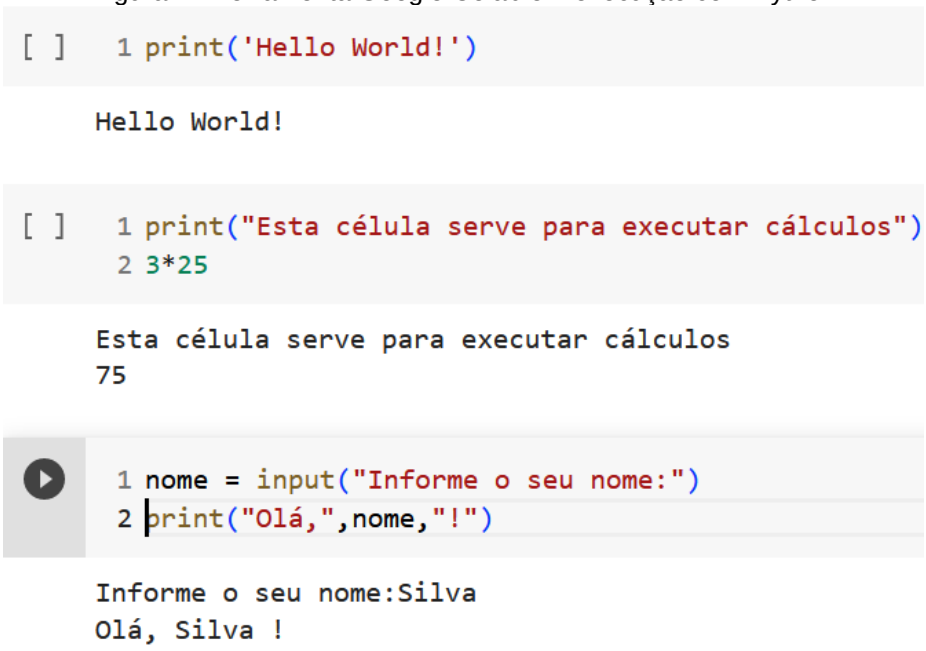
```
print “Hello World”
```

Fonte: CHALLENGER-PÉREZ; DÍAZ-RICARDO; BECERRA-GARCÍA, 2014

Um dos objetivos dessa linguagem é o de conferir facilidade de uso e aprendizado simplificado para os seus usuários. Trata-se de uma linguagem de aprendizagem rápida, e que pela sua popularidade dispõe de incontáveis recursos gratuitos ao redor da internet que viabilizam o estudo desta tecnologia, muitos desses estruturados em formatos de cursos e treinamentos, além de e-books, bem como cursos e formações pagos com o objetivo de formar pessoas capazes de criar aplicações e trabalhar com o Python usando-o para Ciência de dados ou alguma outra aplicação de cunho científico.

O Python pode ser utilizado como ferramenta de desenvolvimento ao ser instalado na máquina do programador, mas pode também ser usado com o apoio de ferramentas online tais como o Jupyter Notebook, ou o Google Colab. Por intermédio delas é possível um uso pleno da linguagem sem que nenhum recurso tenha que ser instalado na máquina do utilizador. Além disso, essas ferramentas, por serem online, permitem que o código esteja facilmente disponível em qualquer terminal o qual o programador esteja utilizando, bastando para isso que o computador disponha de acesso à internet. Ademais, os projetos podem ser compartilhados e com isso podem ser escritos de maneira colaborativa, fato este que serve para explicar o nome da ferramenta do Google. Abaixo uma visualização da ferramenta Google Colab com o Python em uso.

Figura 7 - Ferramenta Google Colab em execução com Python




Fonte: Autoria própria

Linguagem *open source*, recebendo contribuições de uma extensa comunidade ao longo do globo terrestre, podendo ter alterações mesmo até no código fonte da própria linguagem. As aplicações desenvolvidas em Python podem ser distribuídas livremente, bem como podem ser comercializadas devido ao seu modelo de licenciamento.

Linguagem interpretada, mas sem muita perda de desempenho pois grande parte a sua programação original foi desenvolvida em C o que garante uma performance considerável na execução das aplicações desenvolvidas em Python.

O Python é uma linguagem multiplataforma, podendo ser utilizada nos sistemas operacionais mais populares disponíveis atualmente, a saber: Windows, Mac OS e Linux. É também uma linguagem multipropósitos, podendo ser utilizada para o desenvolvimento de variados tipos de aplicações, desde jogos, sistemas web, e com um uso científico bastante disseminado, servindo em análise e ciência de dados. Além disso, trata-se também de ser uma linguagem multiparadigma, haja vista que pode ter desenvolvimento orientado a objetos, imperativa ou funcional.

A existência de inúmeras bibliotecas estende o potencial da linguagem para uma ampla gama de desenvolvimento diferente. Dentre elas pode-se citar Numpy, Pandas, Scikit-learn, Scipy, Matplotlib, Seaborn, entre outras. Cada biblioteca enriquece a linguagem com funcionalidades específicas para uma determinada área de conhecimento, como por exemplo o Scikit-learn, confere ao Python a capacidade de viabilizar projetos de inteligência artificial e aprendizagem de máquina; enquanto o Numpy introduz aprimoramentos na capacidade de computação e processamento numérico do Python, ainda com o benefício de ter sio implementado em C concedendo mais performance aos cálculos e manipulações de *arrays* e *datasets*. As duas últimas bibliotecas citadas, Matplotlib e Seaborn, são objeto de estudo do presente trabalho.

O Numpy e o Matplotlib juntos fazem com que o Python tenha condições de ser funcional e entregar as soluções equivalentes e com a mesma performance e qualidade tal qual o celebrado software pago, Matlab.

Em virtude de todas essas características expostas, o Python é uma linguagem bastante disseminada e muito popular atualmente. É usada tanto por programadores com formação em cursos específicos de Tecnologia da Informação, mas também por outros profissionais e cientistas de áreas como Estatística, Matemática e Engenharias. Segundo levantamento feito por Srinath (2017), esta linguagem figura como sendo a

5ª maior comunidade no site StackOverflow, e como sendo a 4ª linguagem mais usada no Github.

Na sequência serão abordadas com mais detalhes as bibliotecas necessárias para a elaboração de visualizações de dados com Python, bem como algumas bibliotecas que são tidas como requisitos *sine qua non* para uso delas.

## 3.1 Matplotlib

O Matplotlib é uma biblioteca para criação de gráficos criado e disponibilizado em 2007 pelo pesquisador e programador John D. Hunter, e bastante disseminada na comunidade científica desde então. No ano de 2007 Hunter publicou um artigo intitulado “Matplotlib: a 2D Graphics Environment” no qual discorreu sobre o conceito da biblioteca, e quais eram as suas expectativas quando resolveu desenvolvê-la.

Vale mencionar também que em os autores Barret et al (2005, p. 91) definiram o Matplotlib, ainda em fase de desenvolvimento, como sendo, “... um pacote portátil de plotagem e geração de imagem 2D focado primariamente na visualização de dados científicos, de engenharia, e financeiros”. Nesse mesmo texto os autores destacam sobre o caráter de facilidade de uso como sendo algo de relevância no projeto de criação da ferramenta.

É importante destacar que o sucesso do Matplotlib é tão grande que ela serviu e ainda serve de base para a concepção e desenvolvimento de outras bibliotecas de geração de visualização de dados, tais como a próprio Seaborn a qual será tratada na sequência deste trabalho.

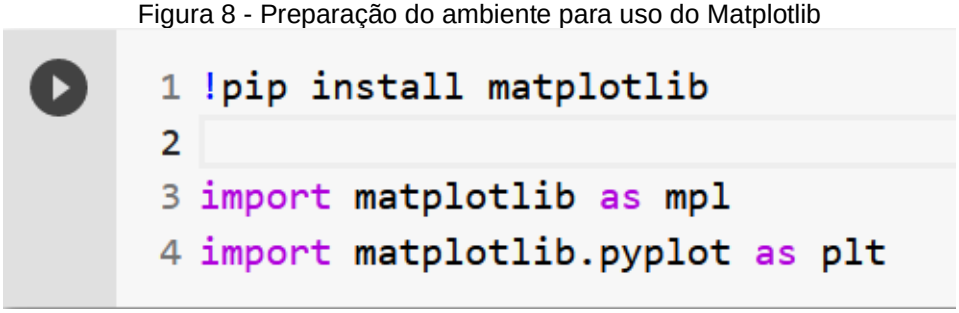
Segundo Hunter (2007), o Matplotlib é uma biblioteca Python de criação de gráficos 2D que pode ser usada para desenvolvimento de aplicações, e geração de gráficos com qualidade e publicação para múltiplos sistemas operacionais. De fato, por ser desenvolvida para Python, ela consegue atingir esse objetivo tendo em vista que o Python é uma linguagem multiplataforma como visto previamente.

Ao desenvolver o Matplotlib, Hunter fez uso do Matlab como referência de *benchmark*, tendo em vista que o desenvolvedor trabalhava em projetos científicos nos quais utilizava o Matlab como instrumento de trabalho. Nessas ocasiões Hunter identificou alguns pontos de melhoria os quais este poderia ter, e assim optou por fazer uma engenharia reversa em cima dele e criar uma ferramenta alternativa que

tivesse o mesmo ponto de partida em termos de qualidade e proporcionasse uma experiência de uso similar beneficiando assim a ampla comunidade de usuários do Matlab.

O Matplotlib é uma biblioteca de código e licença abertos, sendo facilmente acessível a qualquer usuário bastando que sejam instalados e importados os elementos para o seu imediato uso a partir dessas ações. Confira abaixo figura ilustrativa da facilidade de configuração do ambiente para a execução do Matplotlib.

Figura 8 - Preparação do ambiente para uso do Matplotlib




Fonte: Autoria própria

## 3.2 Seaborn

Segundo Waskom (2021), a biblioteca Seaborn é um projeto que oferece uma interface ao Matplotlib o que permite uma rápida exploração e criação de visualização de dados.

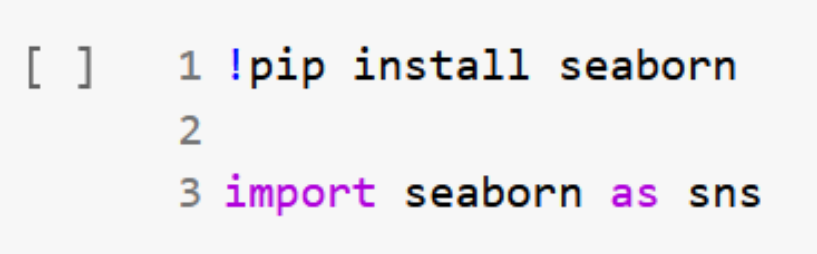
Esta ferramenta foi desenvolvida após o Matplotlib e guarda algumas diferenças conceituais com relação a esta. Aquela enfatiza suas visualizações em formas mais específicas atinentes ao campo da estatística, e para formas diversas ou de maior complexidade acaba por usar os algoritmos do Matplotlib para a sua criação. Assim, conforme (Waskom, 2021, p. 3), "o Seaborn não tem objetivo de encapsular ou substituir completamente o Matplotlib".

Um destaque que justifica o uso do Seaborn é a facilidade com que questões sobre os dados são convertidas em gráficos capazes de explicá-las e respondê-las, isso por conta das funções de caráter declarativo constantes nesta biblioteca e sua orientação a conjunto de dados.

Para o uso desta ferramenta, a exemplo do Matplotlib, é necessário a instalação dos elementos necessários. Trata-se de um procedimento rápido e que envolve poucos segundos. Importante notar que na documentação, menciona-se a existência

de algumas dependências obrigatórias, a saber: Pandas, Numpy e Matplotlib; no entanto a instalação desses requisitos é tão simples quanto a instalação do Seaborn em si, seguindo a mesma simplicidade que marca o uso do Python como um todo. Segue ilustração da configuração do ambiente para uso do Seaborn.

Figura 9 - Preparação do ambiente para uso do Seaborn




Fonte: Autoria própria

## 3.3 Bibliotecas de apoio a construção de gráficos

Para o aprimoramento e a viabilização do uso em pleno potencial de ferramentas como as já mencionadas previamente, é necessário que se lance mão de outras bibliotecas que conferem ao Python mais flexibilidade, funcionalidades e aumento no desempenho no que diz respeito à cálculos, processamento e manipulação de dados.

Assim, segundo HARRIS et al (2020), o Numpy é importante pois viabiliza uma sintaxe aprimorada para programação e manipulação de vetores e matrizes com maior eficiência e desempenho. Essa ferramenta serve de apoio a outras mais específicas com atuação em campos diversos, tais como: Economia, Finanças, Ciência de dados, Química etc. Além disso ela aprimora o poder de processamento do Python, ao usar uma rica API em C.

Enquanto isso, o Pandas auxilia na manipulação de dados, sendo que este permite com que o Python se torne praticamente um software de criação e edição de planilhas. O Pandas enriquece o Python com novas estruturas de dados, e permite a execução de comandos que conferem mais praticidade e poder na manipulação de grandes *dataframes*. Por meio dele é possível criar colunas, filtrar e melhorar a qualidade dos dados, excluir dados, incluir dados, gerar rapidamente dados estatísticos que resumem os dados contidos no *dataframe* em questão.

# 4 APRESENTAÇÃO DOS GRÁFICOS ESCOLHIDOS

A tarefa de escolha de gráficos em si não é algo simples ou trivial e depende de uma série de circunstâncias que acabam por influenciar na decisão do cientista, ou do produtor de conteúdo em questão, qualquer que seja a pessoa que esteja elaborando o gráfico.

As condições que direcionam essa escolha, podem ser, mas não se resumem a: origem dos dados e tipo das variáveis a ser estudadas, se descritivas ou quantitativas (discretas ou contínuas); número de variáveis as quais serão estudadas; público ao qual a visualização irá se comunicar; local onde o gráfico será publicado (se mídia impressa, ou se online, e neste caso se haverá possibilidade de o gráfico ter elementos de interatividade ou não), dentre outras. Dessa maneira não se pode escolher um gráfico indiscriminadamente para a plotagem dos dados.

Segundo Mackinlay (1988), conforme citado por Alexandre; Tavares (2007):

> a escolha da representação adequada para um determinado conjunto de dados, deve ser baseada em critérios de expressividade (*expressiveness*) e eficácia (*effectiveness*). O critério de expressividade diz respeito às representações gráficas que traduzem exatamente a informação com interesse para o utilizador. O critério de eficácia está relacionado com a facilidade de compreender as representações e as informações que elas expressam.

De acordo com Macedo et al (2020), “cada tipo de gráfico atende a propósitos específicos e é aplicável apenas a determinadas categorias de dados”, e os autores vão além quando afirmam que “devido à existência de abordagens e técnicas de visualização mais adequadas a um propósito ou a outro, existe a importância de compreender as diferentes visualizações de dados, seus propósitos e aplicações.”

Com o intuito de realizar a comparação entre as bibliotecas previamente abordadas foram escolhidos três gráficos bastante populares para a observação do comportamento prático das ferramentas. Os gráficos apresentados neste trabalho são: gráfico de barras verticais (colunas), gráfico de linha, e gráfico de pizza, o qual em inglês é conhecido como *pie chart* (gráfico de torta, em tradução livre).

A escolha destes gráficos justifica-se por serem gráficos amplamente utilizados e bastante difundidos nos mais variados contextos, não estando presentes apenas em meios científicos. Além disso, escolheu-se essas figuras pois são de fácil aplicação podendo ser pertinentes a muitos conjuntos de dados disponíveis na internet.

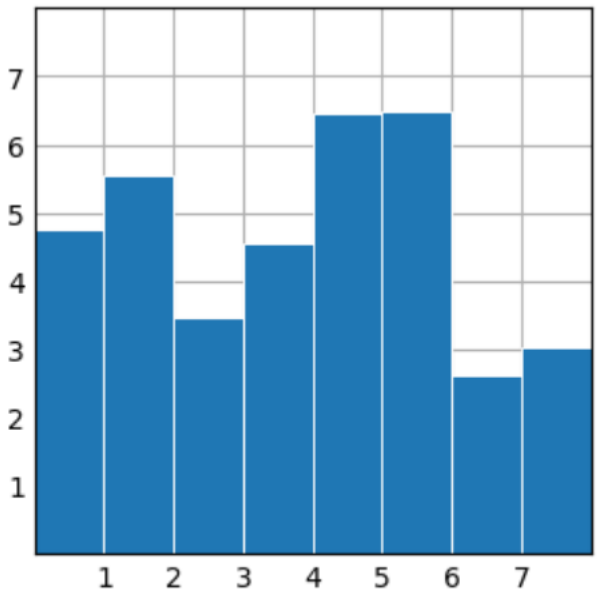
No que diz respeito às bases de dados as quais foram utilizadas nesta pesquisa, estas são: Pydataset, e Gapminder. Mais detalhes delas serão fornecidos oportunamente no início do capítulo ao se explicar a metodologia empregada para a geração dos gráficos com as duas bibliotecas: Matplotlib e Seaborn.

## 4.1 Gráfico de barras verticais (colunas)

O Gráfico de barras verticais, também conhecido como de colunas, tem a capacidade de mostrar a comparação de variáveis descritivas (categóricas) com variáveis numéricas (discretas). Esse gráfico exibe o número de ocorrências para cada uma das categorias que está sendo estudada a partir do conjunto de dados.

Assim, para a elaboração deste gráfico é necessário que se tenha ao menos duas variáveis distintas, uma numérica e que possa ser quantificada, e a outra categórica para a qual seja possível demonstrar a quantidade das suas ocorrências. Abaixo, segue exemplo de gráfico de barras verticais.

Figura 10 - Exemplo de gráfico de barras verticais



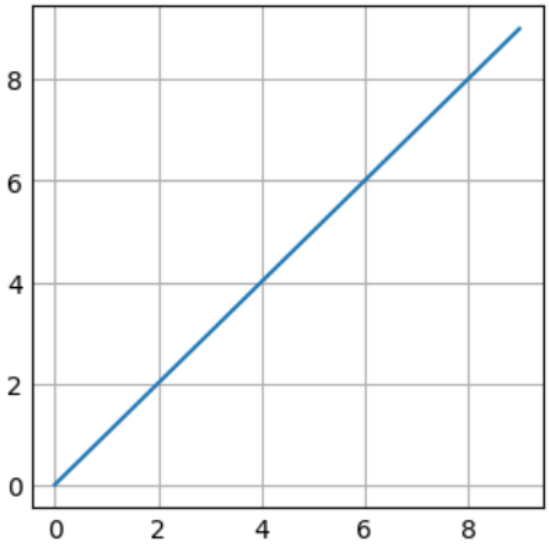

Fonte: Autoria própria

## 4.2 Gráfico de linhas

O gráfico de linhas é simples, porém muito eficiente para mostrar a evolução de variáveis quantitativas ao longo de um determinado período. Ele pode demonstrar a evolução de tendências para ocasiões em que o gráfico apresenta uma única variável, mas pode explicitar tendência e relações quando mais de uma série de variáveis são exibidas. Um benefício deste gráfico é a facilidade com que normalmente é compreendido, muito em função da sua simplicidade.

Para a criação deste gráfico são necessários ao menos duas séries de dados, uma quantitativa e a outra temporal a qual será plotada no eixo x, a fim de que seja possível observar a variação da variável numérica. Abaixo, segue exemplo de gráfico de linha:

Figura 11 - Exemplo de gráfico de linhas




Fonte: Autoria própria

## 4.3 Gráfico de pizza

Esse gráfico tem como finalidade a demonstração da quantidade de ocorrências de determinadas categorias no conjunto de dados estudado. Sua função é bastante semelhante ao do gráfico de colunas verticais e pode ser usado alternativamente a este em alguns casos. Nas palavras de Pereira (2015), essas visualizações

> ajudam a comunicar proporções e percentagens entre categorias, dividem um círculo em segmentos proporcionais. Cada fatia representa uma proporção e o círculo completo representa a soma total de dados (PEREIRA, 2015, p. 62).

Para a sua elaboração é necessário que se tenha no mínimo duas variáveis distintas: uma com dados categóricos/descritivos, e outra numérica, capaz de quantificar os dados categóricos.

É importante mencionar que o uso desse tipo de gráfico não é pacificado, isso porque para que se tenha uma ideia de qual categoria é mais representativa que a outra dentro de um determinado conjunto, é preciso que se avalie o tamanho da área de cada fatia da pizza do gráfico; e sabe-se que a percepção humana para avaliação do tamanho de área desse tipo de formato estando adjacentes não é algo natural e evoluído quanto a percepção do tamanho de barras colocadas lado a lado. Levando isso em consideração recomenda-se algumas boas práticas para o seu uso: limitar a plotagem para um número que não ultrapasse de 7 a 8 categorias; e exibir uma informação numérica, em porcentagem para cada uma das fatias com a finalidade de auxiliar na interpretação da figura[2]. Abaixo segue exemplo do gráfico de pizza.

Figura 12 - Exemplo de gráfico de pizza


Fonte: Autoria própria

[2] De acordo com IRIZARRY (2019, p. 201), “Se por alguma razão você precisar fazer um gráfico de pizza, coloque legenda em cada fatia da pizza com a sua respectiva porcentagem para que os observadores não tenham que inferi-los a partir dos ângulos ou da área.”

# 5 GRÁFICOS COM MATPLOTLIB

Neste capítulo serão mostrados os gráficos apresentados no capítulo anterior, todos elaborados com o uso do Matplotlib, e com o apoio das bibliotecas Numpy e Pandas.

Para a criação destes gráficos foram utilizados os conjuntos de dados (ou *datasets*, em inglês) “Gapminder”, disponível no endereço https://pypi.org/project/gapminder/, o qual consiste em uma versão para Python do pacote Gapminder para linguagem R criada pela engenharia de software Jennifer Bryan[3]. Além do mencionado anteriormente, usou-se também o conjunto de dados “Cars93”, o qual é descrito como possuindo dados de carros disponíveis para venda nos EUA em 1993, e faz parte do pacote Py*dataset*, disponível no endereço https://py*dataset*.readthedocs.io/en/latest/index.html. Esse pacote dispõe atualmente de mais de 750 conjuntos de dados prontos para uso e treinamento de técnicas utilizadas em análises e ciências de dados com Python. No apêndice C há uma demonstração visual das tabelas de dados utilizadas.

Vale dizer que esses mesmos *datasets* foram utilizados para desenvolvimento dos gráficos em Seaborn e Excel, os quais serão exibidos no “capítulo 6” e no “apêndice A” respectivamente. Uma vez feita essa breve contextualização de ordem metodológica, parte-se para a apresentação dos gráficos feitos com o Matplotlib.

## 5.1 Gráfico de barras verticais (colunas)

De acordo com o que foi descrito na seção 4.1, este gráfico tem como função a quantificação das variáveis categóricas estudadas. Assim, usando-se do *dataset* “Gapminder”, procedeu-se com a comparação da evolução da expectativa de vida entre os anos de 1952 até 2007, no Brasil e no Canadá.

Para a criação deste gráfico, tendo já o conjunto de dados carregado no ambiente, foi necessário acessar as linhas e colunas específicas para a obtenção dos dados os

[3] O conjunto de dados Gapminder para R, trata-se de um excerto do conjunto Gapminder original, possui caráter essencialmente educacional, e está disponível no endereço https://github.com/jennybc/gapminder/.

quais seriam plotados no gráfico, e passá-los como argumentos da função “.bar()” (responsável pela criação do gráfico de barras), conforme consta da figura a seguir:

Figura 13 - Acesso aos dados e uso da função ".bar()" do Matplotlib

```
17 ax.bar(br1, br_life, label = 'Brazil', color = 'green', width = barWidth)
18 ax.bar(br2, df.iloc[240:252].lifeExp, label = 'Canada', color = 'red', width = barWidth)
```

Fonte: Autoria própria

Para que o gráfico fosse criado com uma barra ao lado da outra, permitindo uma comparação mais clara sobre a situação num país e no outro, foi preciso também fornecer elementos ao Matplotlib para que ele pudesse dispô-las tal como se tinha a intenção, e isso segue evidenciado na próxima figura. Vale ressaltar que estes comandos só foram necessários pois tinha-se o objetivo de ter as duas colunas lado a lado na mesma plotagem, caso contrário não seria preciso.

Figura 14 - Definindo as posições das barras na plotagem do Matplotlib

```
13 # Set position of bar on X axis
14 br1 = np.arange(len(br_year))
15 br2 = [x + barWidth for x in br1]
```

Fonte: Autoria própria

Na sequência, ocupou-se de fazer algumas personalizações com a finalidade de agregar mais informações à figura e torná-la ainda mais inteligível, definindo claramente os pontos no Eixo X, nomeando-se os dois eixos, e nomeando-se o gráfico. Seguem detalhes na próxima imagem.

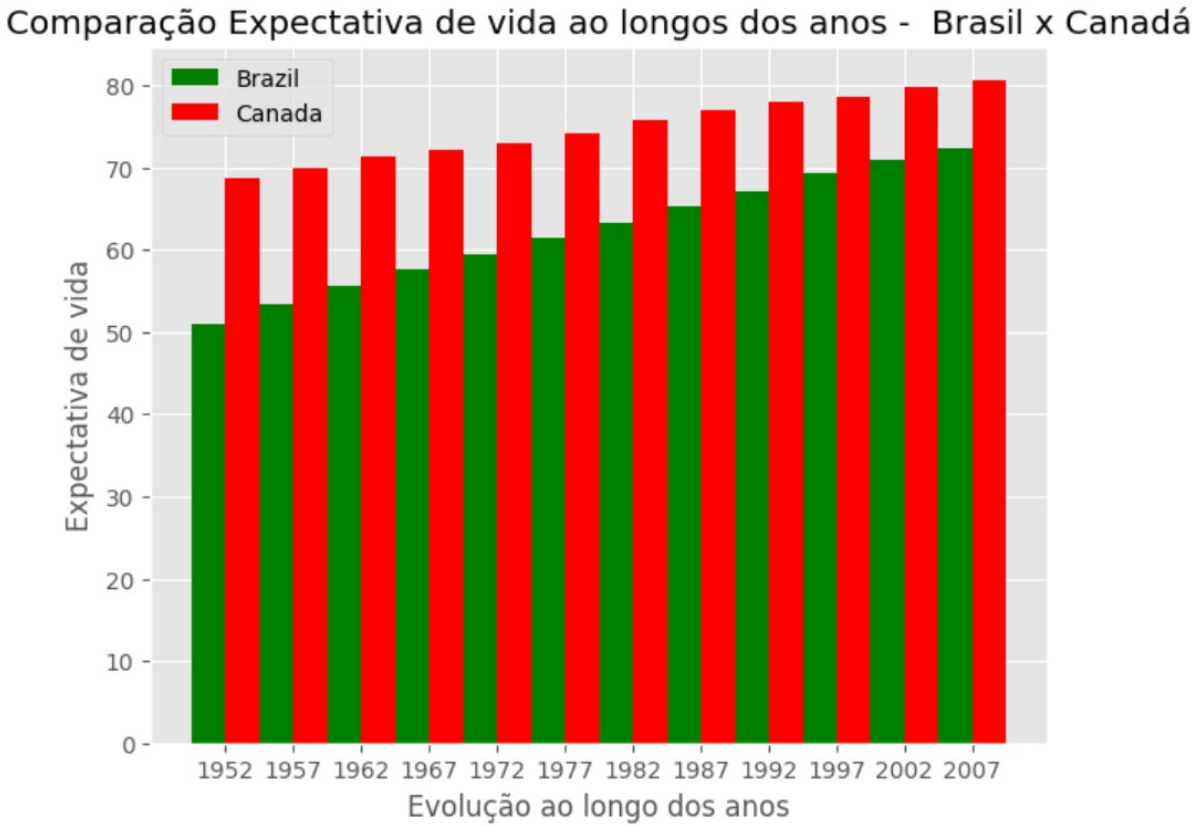
Figura 15 - Personalização do gráfico (agregando valor e melhorando a interpretação)

```
19 ax.set_xlabel('Evolução ao longo dos anos')
20 ax.set_ylabel('Expectativa de vida')
21 ax.set_title("Comparação Expectativa de vida ao longos dos anos -  Brasil x Canadá")
22 ax.set_xticks(colunas1 + barWidth/2, labels=['1952', '1957', '1962', '1967', '1972',
```

Fonte: Autoria própria

A seguir, depois de escritos as linhas de código necessárias para a construção do gráfico, ao pedir para o Python interpretar os comandos redigidos, em poucos segundos é possível visualizar o gráfico pronto e da forma como queríamos vê-lo. Segue ilustração da criação.

Figura 16 - Gráfico de barras verticais gerado pelo Matplotlib




Fonte: Autoria própria

Destarte, neste gráfico, conforme era esperado temos uma visualização comparativa de quanto era a expectativa de cada país (Brasil em verde, e Canadá em vermelho) no ano de 1952 (início dos registros deste *dataset*), e evoluindo a cada 5 anos até chegar o ano de 2007.

Tendo esta intenção de comparar ponto a ponto, dois ou mais grupos distintos (no caso em tela: Brasil e Canadá), a opção por colocar todas as barras no mesmo gráfico é bastante pertinente pois a aposição lado a lado das barras faz com que fique visualmente bem fácil e rápida a distinção entre os grupos comparados.

## 5.2 Gráfico de linhas

Na seção 4.2 foi feita a apresentação formal deste gráfico demonstrando o que é preciso para gerá-lo com sucesso de maneira que este empreste sentido e conhecimento aos dados estudados.

Assim, usou-se o *dataset* “Gapminder” para compreender a tendência de evolução para o indicador econômico de Produto Interno Bruto (PIB) para os países Brasil e Canadá, também entre os anos de 1952 até 2007 (a exemplo do gráfico anterior).

A exemplo do que ocorreu no gráfico anterior é preciso ter os dados carregados, e depois acessar as informações necessárias. Isso foi feito usando-se a função de acesso e localização de dados do Pandas, “.iloc()”. Segue ilustração.

Figura 17 - Acesso aos dados usando a função ".iloc()" do Pandas

```
3 #Dados
4 br_year = df.iloc[168:180].year
5 br_gdp = df.iloc[168:180].gdpPercap
```

Fonte: Autoria própria

De posse dos dados, para criar o gráfico foi preciso apenas usar a função “.plot()” do Matplotlib passando as variáveis “br_year” como argumento para o eixo x, e “br_gdp” como argumento para o eixo y. Na sequência, seguem ilustrações mostrando as linhas de código usadas para a criação do gráfico (Figura 18), e exibindo as linhas de código responsáveis pela personalização do gráfico (Figura 19).

Figura 18 - Criação de gráfico de linha no Matplotlib

```
7 # Gráfico
8 plt.style.use('ggplot')
9 fig, ax = plt.subplots(figsize=(5, 4.5))
10 ax.plot(br_year, br_gdp, linewidth=2.0, label = 'Brasil', color ='green')
```

Fonte: Autoria própria

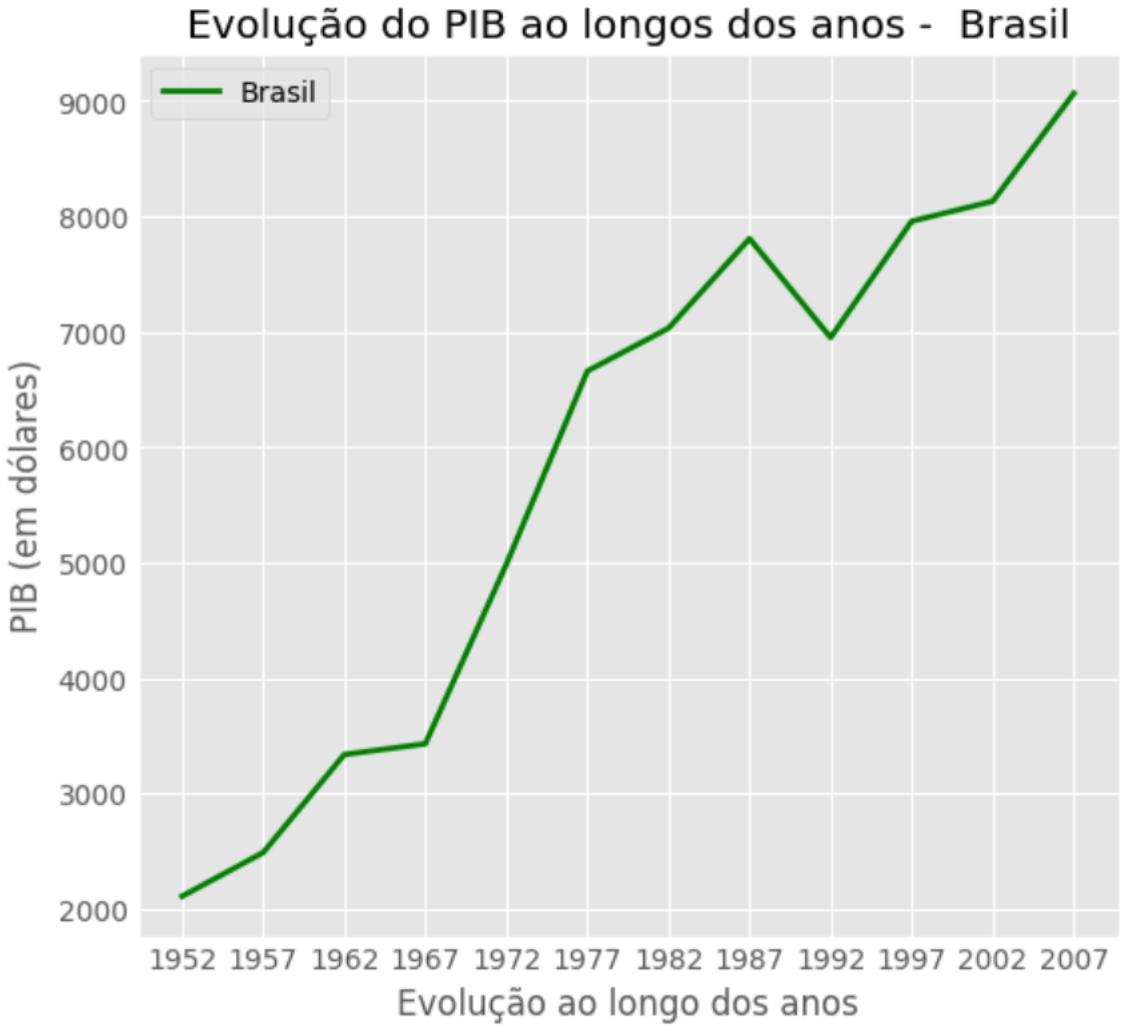
Figura 19 - Personalização do gráfico de linhas

```
12 #Personalização
13 ax.set_xlabel('Evolução ao longo dos anos')
14 ax.set_ylabel('PIB (em dólares)')
15 ax.set_title("Evolução do PIB ao longos dos anos -  Brasil")
16 ax.set_xticks(br_year)
17 ax.legend()
```

Fonte: Autoria própria

Conforme visto nas figuras anteriores e de acordo com o que foi explicado nos últimos parágrafos, o gráfico de linhas é simples para ser feito pelo Matplotlib, sendo necessário apenar ter os dados para passar para os eixos x e y da função “.plot()”; e o resultado pode ser visto na próxima figura.

Figura 20 - Gráfico de linhas gerado pelo Matplotlib



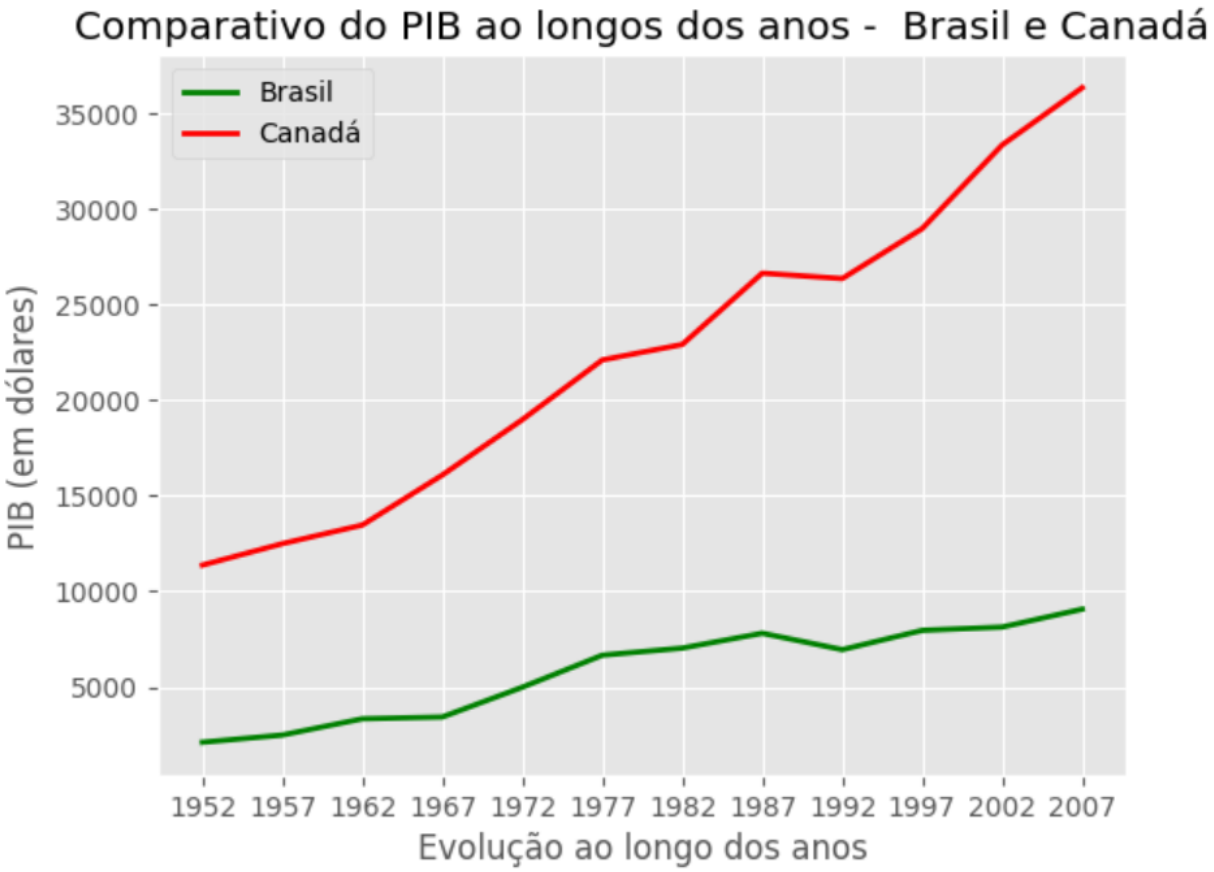

Fonte: Autoria própria

O gráfico de linhas liga os pontos em que se cruzam os valores dos eixos x e y. Na ocasião em que o eixo x contém uma série temporal, é possível distinguir uma linha de tendência sobre o comportamento dos valores do eixo y, neste caso o PIB. Como dito previamente, é um gráfico que traz uma rápida compreensão para o leitor.

Vale ainda acrescentar que o gráfico de linhas pode exibir a linha de uma única série de dados, bem como de mais de uma série. Nessa circunstância, é possível mostrar uma comparação da tendência (por meio do formato e angulação das linhas),

bem como uma avalição numérica ponto a ponto referente a todos os conjuntos de dados plotados no gráfico, veja exemplo abaixo.

Figura 21 - Gráfico de linhas com plotagem de dois conjuntos de dados simultaneamente




Fonte: Autoria própria

Neste gráfico o formato da curva do PIB brasileiro está bem diferente do gráfico anterior, e isso decorre da mudança de escala no eixo y. Quando os valores canadenses são acrescentados na imagem, o eixo das ordenadas passa a ter valores representados num intervalo de cinco mil dólares, sendo que no gráfico em que tinha apenas as informações do Brasil, os valores eram marcados de mil em mil dólares.

## 5.3 Gráfico de pizza

Para a criação do gráfico de pizza utilizando-se o Matplotlib, usou-se o *dataset* “Cars93”, e o objetivo era visualizar a quantia de cada tipo de carro que havia disponível no mercado de veículos estadunidense no ano de 1993.

A fim de conseguir ter sucesso nessa visualização, o primeiro passo foi realizar a contagem da quantidade de ocorrência de cada linha do conjunto de dados, de acordo com a coluna “Tipo”. Para fazer isso, a função Pandas usada foi a “.value_counts()”.

Tendo essa contagem feita e atribuída a uma variável, usa-se a função “.pie()” do Matplotlib, passando-se a variável com os valores como parâmetro para a função. Além desse, passa-se outros parâmetros sendo o próximo contendo os rótulos para plotagem em cada uma das fatias do gráfico, além de outros que visam a personalização do gráfico.

Essa figura tem uma elaboração bastante simplificada no Matplotlib e pode ser gerado com menos de 10 linhas de código, conforme se vê na próxima figura.

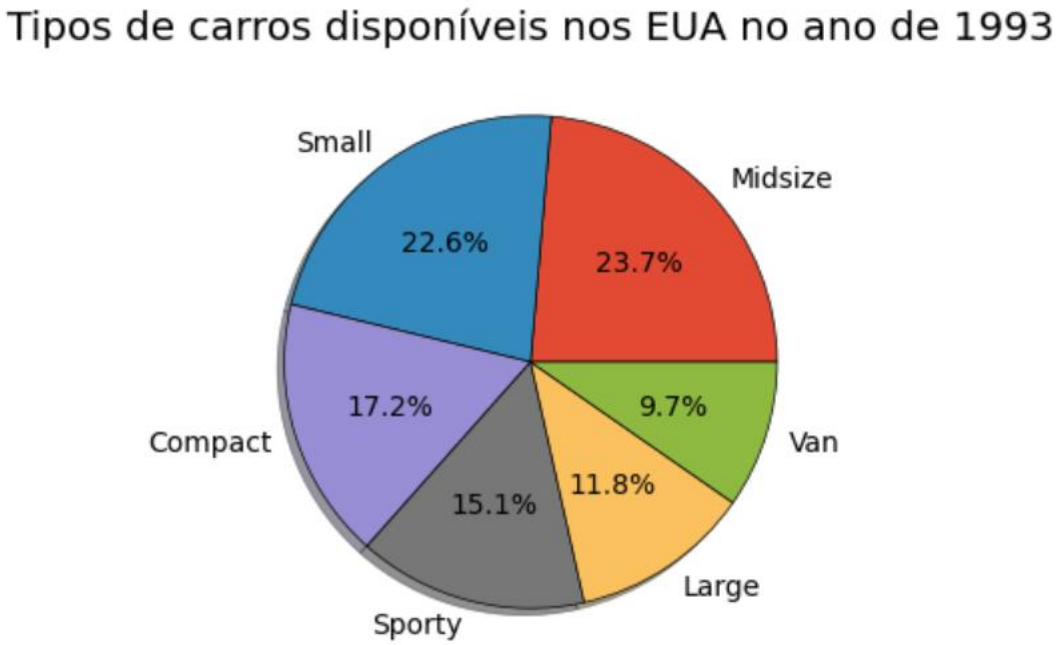
Figura 22 - Código para criação do gráfico de pizza no Matplotlib

```
3 #Dados
4 carType_amount = df6.value_counts('Type')
5
6 #Gráfico e personalização
7 fig, ax = plt.subplots(figsize = (4,4))
8 plt.style.use('ggplot')
9 labels = 'Midsize', 'Small', 'Compact', 'Sporty', 'Large', 'Van'
10 ax.pie(carType_amount, labels=labels, autopct='%1.1f%%', shadow=True,
11         wedgeprops={'edgecolor': 'black'})
12 ax.set_title("Tipos de carros disponíveis nos EUA no ano de 1993")
13
14 plt.show()
```

Fonte: Autoria própria

O resultado pode ser observado na imagem seguinte depois de uma execução que não demorou nem 10 de segundos para a criação da visualização.

Figura 23 - Gráfico de pizza no Matplotlib




Fonte: Autoria própria

Vale dizer que para a elaboração dessa figura, atentou-se para as recomendações de boa prática para a plotagem desse tipo de gráfico; com isso incorporou-se ao gráfico uma informação numérica em formato de porcentagem referente ao quantitativo de cada uma das categorias, bem como foram plotadas informações referentes a apenas 6 tipos distintos, facilitando assim a interpretação do gráfico.

## 5.4 Mais possibilidades em Matplotlib

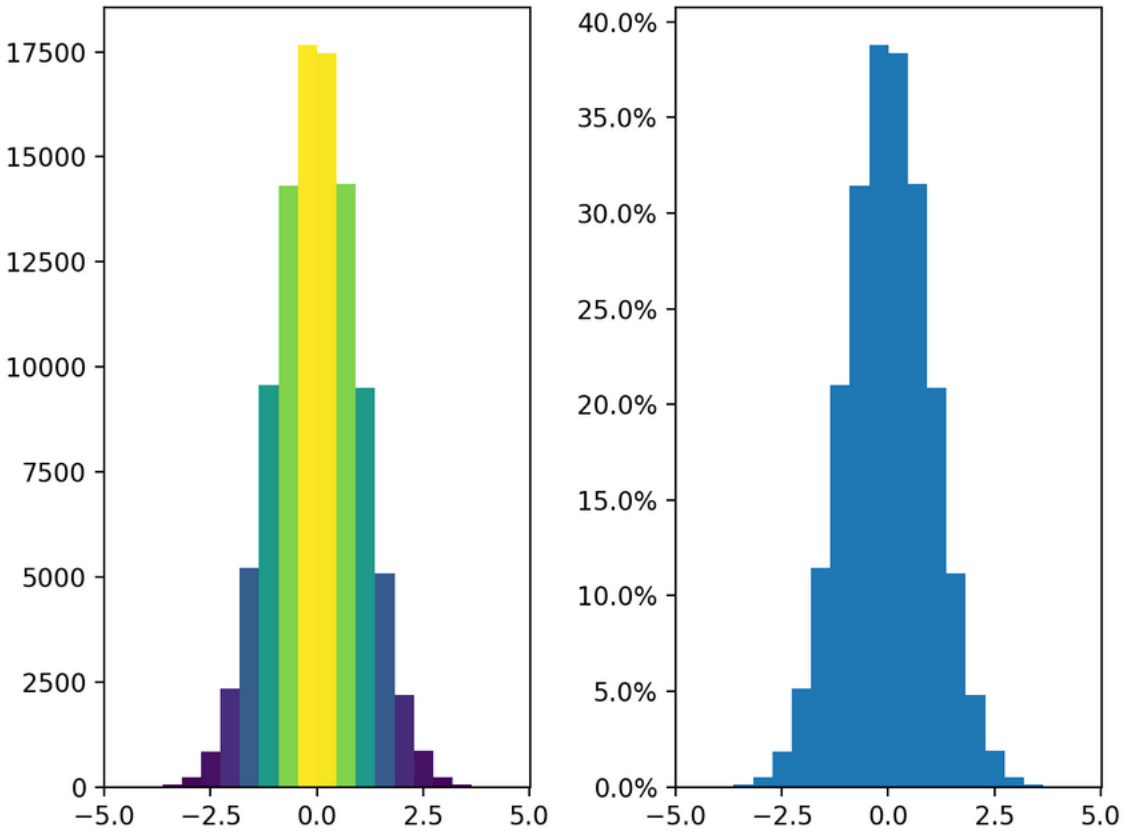
Com o intuito de apresentar outros recursos existentes no Matplotlib, nesta seção serão exibidos alguns outros gráficos os quais podem ser criados com essa ferramenta, a saber: Histogramas, e Gráficos de Pilha. Mesmo com a exibição desses outros gráficos, este trabalho está ainda muito longe de conseguir demonstrar a totalidade do potencial desta biblioteca, que é uma das pioneiras em Visualizações de dados com Python.

Para a apresentação desses gráficos, os exemplos e detalhes de implementação serão extraídos da documentação da ferramenta disponível no seu sítio oficial. Vale

ainda mencionar que o Matplotlib possui centenas[4] de funções capazes de criar diferentes gráficos e que permitem opções de variações e personalizações nas imagens e modelos gerados.

O primeiro gráfico que se exibe aqui é o Histograma que tem por principal função demonstrar a distribuição de ocorrências de um determinado fato ou característica numa determinada população ou ao longo do tempo. Para a sua criação no Matplotlib usa-se a função ".hist()" passando-se os argumentos necessários (dados para plotagem, e propriedades para desenho da figura tal qual o autor deseje). Nesse gráfico é possível aplicar diferentes estilos e cores dentro das barras desenhadas, bastando para isso alterar os comandos responsáveis pelo controle dessas características. Abaixo, seguem imagens apresentando o gráfico em, e detalhes da implementação, respectivamente.

Figura 24 - Histograma no Matplotlib




Fonte: https://matplotlib.org/stable/gallery/statistics/hist.html#histograms

[4] Modelos disponíveis em https://matplotlib.org/stable/gallery/index.html.

Figura 25 - Implementação de Histograma no Matplotlib

```
fig, axs = plt.subplots(1, 2, tight_layout=True)

# N is the count in each bin, bins is the lower-limit of the bin
N, bins, patches = axs[0].hist(dist1, bins=n_bins)

# We'll color code by height, but you could use any scalar
fracs = N / N.max()

# we need to normalize the data to 0..1 for the full range of the colormap
norm = colors.Normalize(fracs.min(), fracs.max())

# Now, we'll loop through our objects and set the color of each accordingly
for thisfrac, thispatch in zip(fracs, patches):
    color = plt.cm.viridis(norm(thisfrac))
    thispatch.set_facecolor(color)

# We can also normalize our inputs by the total number of counts
axs[1].hist(dist1, bins=n_bins, density=True)

# Now we format the y-axis to display percentage
axs[1].yaxis.set_major_formatter(PercentFormatter(xmax=1))
```

Fonte: https://matplotlib.org/stable/gallery/statistics/hist.html#histograms

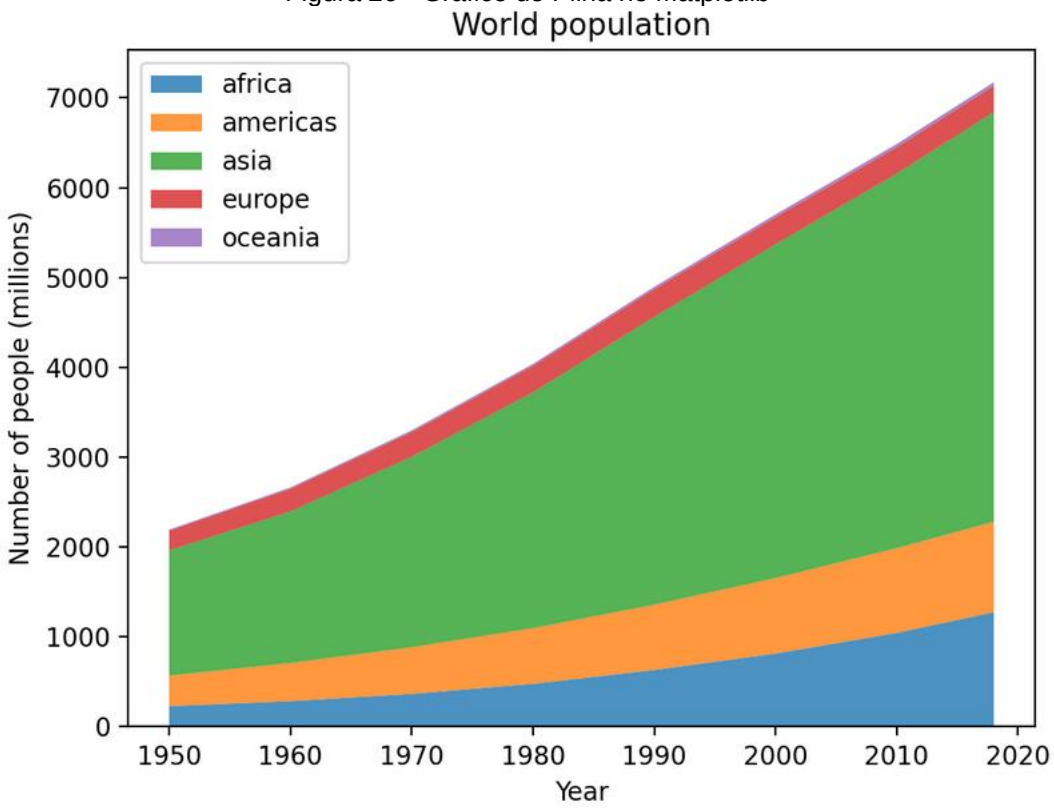
Nesse gráfico as cores foram aplicadas de acordo com a variação apresentada no eixo Y para cada ocorrência distinta.

O Gráfico de Pilha é o próximo que se apresenta aqui como ilustração da riqueza de possibilidades existente no Matplotlib. Este gráfico é relevante pois demonstra numa mesma imagem duas informações distintas: valores individualizados de mais de uma série de dados, bem como o acumulado delas. Este objetivo é alcançado ao se desenhar linhas sobrepostas umas às outras (empilhadas) e usando-se cores diferentes para identificar corretamente cada uma das séries trabalhadas.

O Matplotlib cria essa figura através da função “.stackplot()” passando-se como parâmetros uma série temporal para o eixo X, e, no eixo Y, as respectivas ocorrências para cada um das séries de dados contempladas em função com os anos plotados.

Abaixo seguem duas figuras: a primeira com o gráfico em si, e a segunda com os comandos utilizados para elaboração da visualização. O exemplo em tela demonstra o crescimento populacional de cada um dos continentes num intervalo de tempo que se inicia em 1950, indo até 2018.

Figura 26 - Gráfico de Pilha no Matplotlib




Fonte: https://matplotlib.org/stable/gallery/lines_bars_and_markers/stackplot_demo.html#stackplots

Figura 27 - Implementação de Gráfico de Pilha no Matplotlib

```
import numpy as np
import matplotlib.pyplot as plt

# data from United Nations World Population Prospects (Revision 2019)
# https://population.un.org/wpp/, license: CC BY 3.0 IGO
year = [1950, 1960, 1970, 1980, 1990, 2000, 2010, 2018]
population_by_continent = {
    'africa': [228, 284, 365, 477, 631, 814, 1044, 1275],
    'americas': [340, 425, 519, 619, 727, 840, 943, 1006],
    'asia': [1394, 1686, 2120, 2625, 3202, 3714, 4169, 4560],
    'europe': [220, 253, 276, 295, 310, 303, 294, 293],
    'oceania': [12, 15, 19, 22, 26, 31, 36, 39],
}

fig, ax = plt.subplots()
ax.stackplot(year, population_by_continent.values(),
             labels=population_by_continent.keys(), alpha=0.8)
ax.legend(loc='upper left')
ax.set_title('World population')
ax.set_xlabel('Year')
ax.set_ylabel('Number of people (millions)')

plt.show()
```

Fonte: https://matplotlib.org/stable/gallery/lines_bars_and_markers/stackplot_demo.html#stackplots

Novamente, é importante frisar que a apresentação destes gráficos constantes da presente seção não é suficiente para esgotar o imenso potencial da biblioteca em questão, para tanto recomenda-se novamente a leitura da sua documentação a fim de que o leitor possa apreender todos os recursos disponíveis para emprego nos seus projetos e análises.

# 6 GRÁFICOS COM SEABORN

Neste capítulo, a exemplo do que foi feito no anterior, serão apresentadas as mesmas categorias de gráficos discutidas e tratadas ao longo deste trabalho. Para a criação destes, usou-se os *datasets* descritos no início do capítulo precedente, bem como os mesmos princípios e parâmetros que vêm norteando a presente pesquisa.

Assim, procede-se com a imediata demonstração dos resultados.

## 6.1 Gráfico de barras verticais (colunas)

Para a criação do gráfico de linhas usando o Seaborn foi necessário acessar os dados que seriam plotados no gráfico e armazená-los numa variável. Tendo feito isso, procedeu-se com a criação do gráfico, o que foi feito com apenas uma única linha de código (linha 8 da figura abaixo). Além disso foi feita a personalização dando-se nomes aos eixos x, e y, acrescentando-se um título à visualização, alterando a cor de fundo do gráfico etc. A função do Seaborn necessária para a criação desta imagem é a ".barplot()", e é para esta que são passados os dados, e as referências para a montagem dos eixos da figura.

É importante salientar que para a elaboração desse gráfico usou-se conjuntamente o Seaborn e o Matplotlib. Seguem detalhes abaixo.

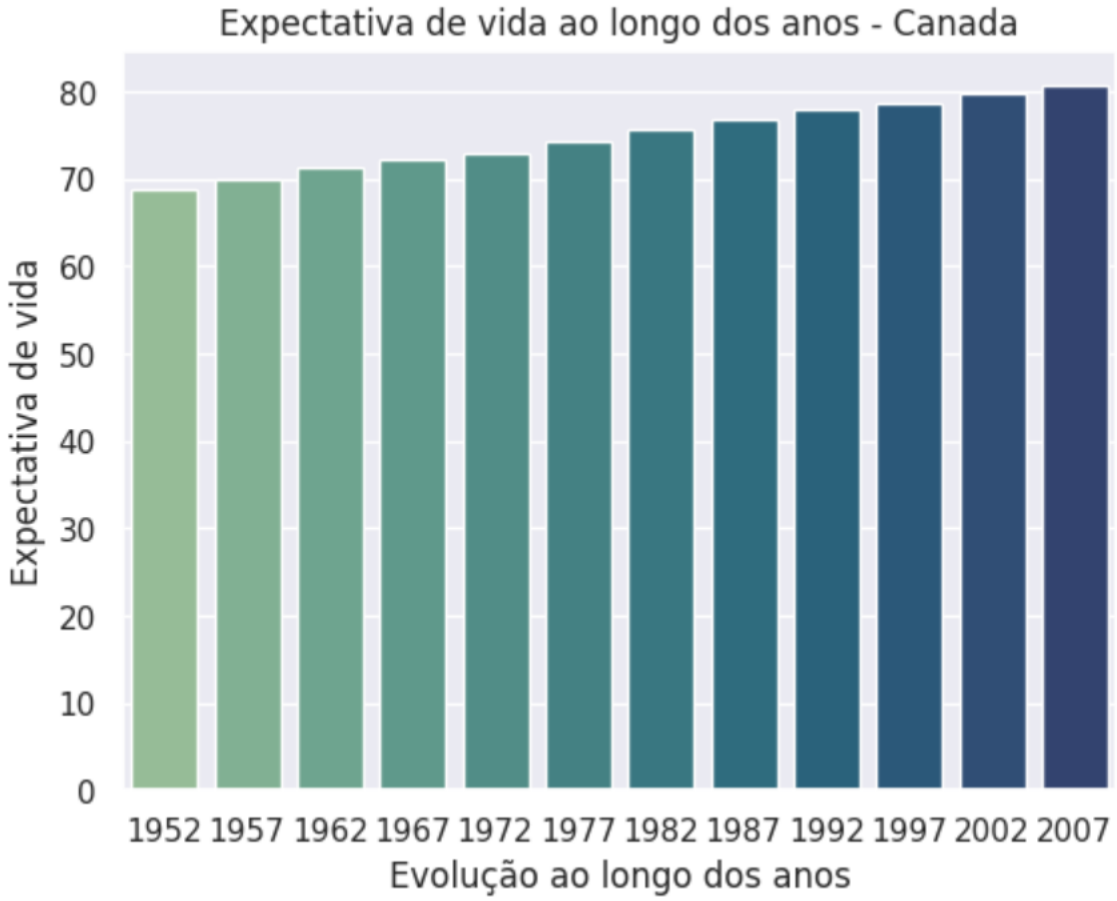
Figura 28 - Criação de gráfico de barras verticais no Seaborn

```
3 #Dados
4 canada = df.iloc[240:252]
5
6 #Personalização e gráfico
7 sns.set_theme(style="darkgrid")
8 ax = sns.barplot(data=canada, x='year', y='lifeExp', palette='crest')
9 ax.set_title('Expectativa de vida ao longo dos anos - Canada')
10 ax.set_xlabel('Evolução ao longo dos anos')
11 ax.set_ylabel('Expectativa de vida')
12
13 plt.show()
```

Fonte: Autoria própria

O resultado das instruções acima resulta no seguinte gráfico:

Figura 29 - Gráfico de barras verticais (Seaborn)




Fonte: Autoria própria

Esse gráfico nos permite visualizar os dados de expectativa de vida de um único país sem que seja possível compará-lo a outro. Assim, é possível compreender a linha de tendência de crescimento ou diminuição dos dados estudados, bem como ter a visualização da quantidade ponto a ponto (no eixo das abscissas) da variável observada.

Não obstante, a exemplo do que foi feito no gráfico demonstrado na seção 5.1, é possível criar alternativas de plotagem com o intuito de permitir a comparação das informações entre dois ou mais conjuntos de dados.

Destarte, com o Seaborn escolheu-se usar a função “.FacetGrid()” criando-se uma visualização nos moldes de uma matriz, de modo que um gráfico pudesse ser colocado ao lado do outro. Com isso, foi posicionado um gráfico com as informações de expectativa de vida no Brasil ao lado do corresponde com dados canadenses (o comando determinante para viabilizar essa configuração encontra-se na linha 10 da

ilustração). A próxima imagem mostra o detalhe das linhas escritas para a construção do gráfico de barras posicionados um ao lado do outro.

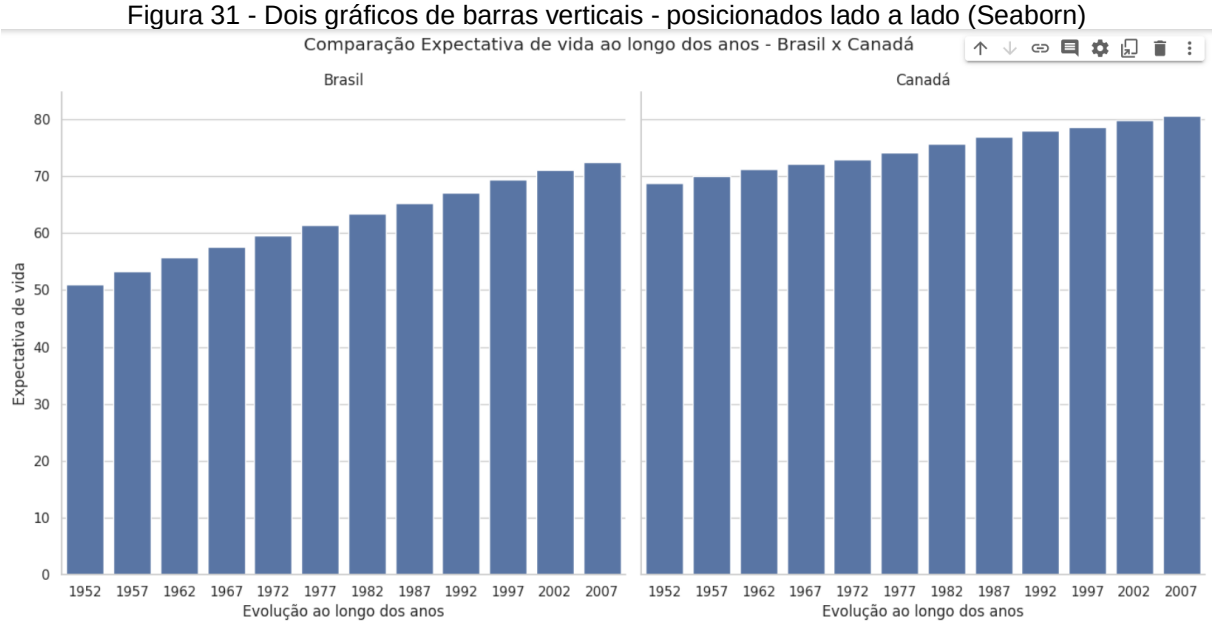
Figura 30 - Detalhe das instruções para criação de visualização em estilo painel (lado a lado) no Seaborn

```
10 g = sns.FacetGrid(both_countries, col="country", height=7)
11 g.fig.suptitle('Comparação Expectativa de vida ao longo dos anos - Brasil x Canadá')
12 axes = g.axes.flatten()
13 axes[0].set_title("Brasil")
14 axes[1].set_title("Canadá")
15 g.map(sns.barplot, "year", "lifeExp")
```

Fonte: Autoria própria

Na próxima imagem apresenta-se o resultado no qual aparece a capacidade de comparação que uma plotagem como essa viabiliza. No entanto, vale mencionar que esta visão permite verificar as diferenças nas tendências de evolução de dados entre um país e outro, mas não concede a mesma capacidade de comparação ponto a ponto tal qual evidenciou-se na figura 16 (quando os dois gráficos estavam sobrepostos numa única imagem).

Figura 31 - Dois gráficos de barras verticais - posicionados lado a lado (Seaborn)




Fonte: Autoria própria

## 6.2 Gráfico de linhas

O gráfico de linhas elaborado com auxílio do Seaborn foi gerado a partir do conjunto de dados “Gapminder”, sob as mesmas condições do que foi colocado quando utilizado o Matplotlib.

Para a criação desse gráfico sem que se coloque nome nos eixos x e y, não é necessário usar o Matplotlib, mas apenas funções do Seaborn. Isso será exibido na figura 27 (linhas de código), e figura 28 (resultado do gráfico). No entanto, nas figuras 29 e 30 serão demonstrados respectivamente o gráfico com os eixos devidamente nomeados, e o trecho do código que responsável por isso.

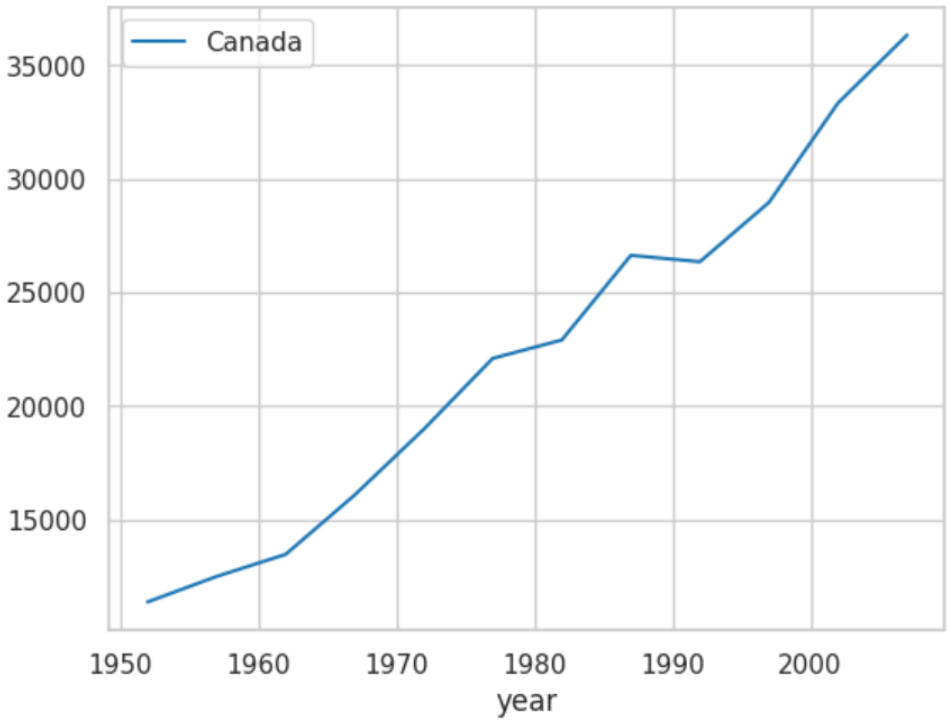
Figura 32 - Instruções para criação do gráfico de linhas no Seaborn

```
3 # Dados
4 br_year = df.iloc[168:180].year
5 br_gdp = df.iloc[168:180].gdpPercap
6 ca_gdp = df.iloc[240:252].gdpPercap
7 #Transformação de dataframes Pandas para arrays Numpy (para uso por parte do Seaborn)
8 br_gdp_sns = br_gdp.to_numpy()
9 ca_gdp_sns = ca_gdp.to_numpy()
10
11 sns.set_theme(style="whitegrid")
12 data_ca = pd.DataFrame(ca_gdp_sns, br_year, columns=["Canada"])
13 sns.lineplot(data=data_ca, palette="tab10", linewidth=1.5)
```

Fonte: Autoria própria

Vale destacar que para a criação desse gráfico foi preciso preparar e elaborar um pouco mais os dados, até mesmo transformando um objeto Pandas num outro Numpy (estando esse trecho comentado no código). A visualização em si foi gerada na linha 13 ao se usar a função “.lineplot()”, e o resultado encontra-se na próxima imagem.

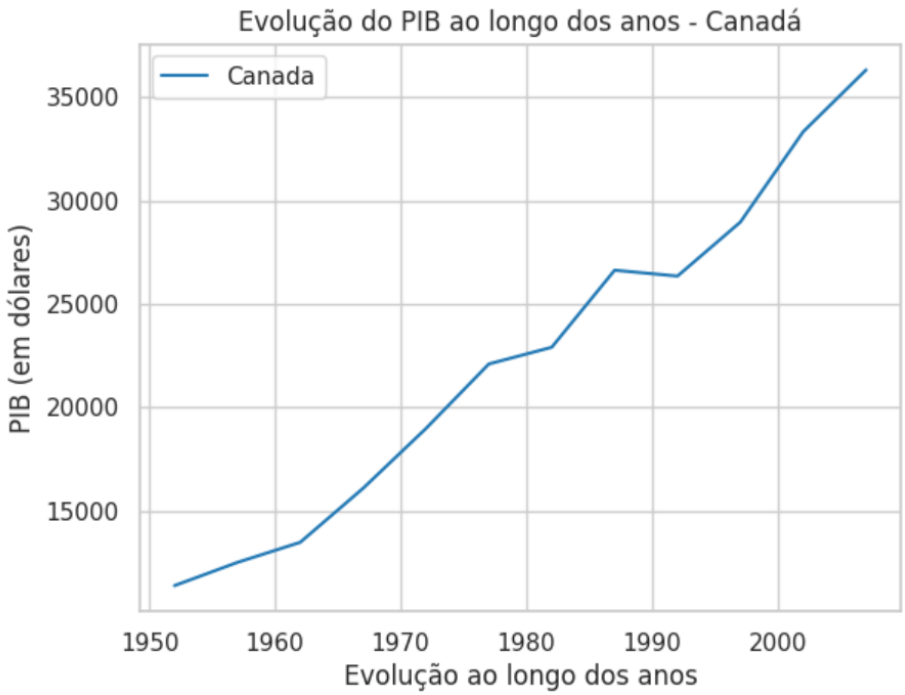
Figura 33 - Gráfico de linhas (Seaborn)




Fonte: Autoria própria

Na sequência mostra-se o resultado da versão na qual se usou também o Matplotlib para inclusão de nomes nos eixos e título no gráfico.

Figura 34 - Gráfico de linhas (Seaborn) com elementos do Matplotlib




Fonte: Autoria própria

Para a inclusão desses elementos textuais da figura, foi usado o Matplotlib da maneira como se segue.

Figura 35 - Código do Matplotlib para personalização do gráfico

```
13 ax = sns.lineplot(data=data_ca, palette="tab10", linewidth=1.5)
14 ax.set_title('Evolução do PIB ao longo dos anos - Canadá')
15 ax.set_xlabel('Evolução ao longo dos anos')
16 ax.set_ylabel('PIB (em dólares)')
17 plt.show()
```

Fonte: Autoria própria

As informações exibidas no gráfico em linhas gerado pelo Seaborn não diferem em nada do que nos gráficos criados com o uso do Matplotlib ou qualquer outra ferramenta de criação de visualizações. Dessa forma, a maneira de visualizar e compreender as informações exibidas aqui são semelhantes ao que foi comentado na seção 5.2.

## 6.3 Gráfico de pizza

Conforme foi abordado na seção 5.3, o gráfico de pizza não é de uso pacificado, e em geral este é marcado por algumas ressalvas quanto ao seu uso. Decorrência disso ou não, fato é que o Seaborn não possui uma função específica para a criação desse modelo de representação visual[5].

Assim, diferente dos gráficos apresentados neste capítulo até então, nos quais a principal função para criação do gráfico pertencia diretamente a biblioteca Seaborn, não é possível exibir aqui um gráfico de pizza usando-se uma função específica do Seaborn.

Sob essas circunstâncias, quando se quer criar esse tipo de gráfico deve ser usado o Matplotlib, e na sequência usar elementos de personalização do Seaborn dando a ele uma aparência mais similar aos gráficos criados por ele.

---

[5] Informação disponível em https://www.statology.org/seaborn-pie-chart/, e verificada na documentação do Seaborn quando constata-se a inexistência de qualquer menção ao termo “pie chart” (gráfico de torta em tradução livre e literal; maneira como gráfico de pizza é denominado em inglês).

Na sequência, exibe-se os comandos usados para a geração da figura, com destaque para a linha 10 na qual consta a personalização feita com Seaborn (o código encontra-se comentado no local para facilitar a referenciação)

Figura 36 - Criação de gráfico usando-se recursos estilísticos do Seaborn

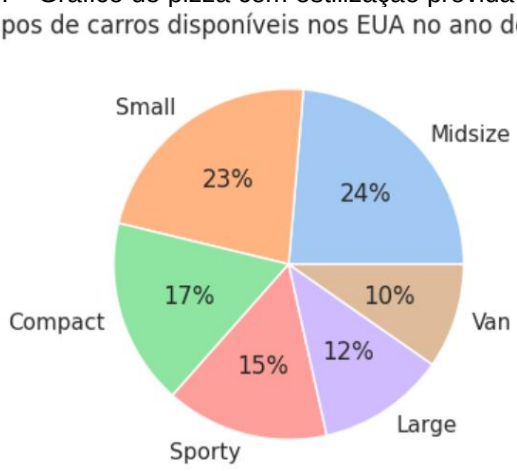
```
3 #Dados
4 carType_amount = df6.value_counts('Type')
5
6 #Gráfico e personalização
7 fig, ax = plt.subplots(figsize = (4,4))
8 labels = 'Midsize', 'Small', 'Compact', 'Sporty', 'Large', 'Van'
9 #Personalização de cores com o Seaborn
10 colors = sns.color_palette('pastel')[0:6]
11 ax.pie(carType_amount, labels=labels, colors = colors, autopct='%.0f%%')
12 ax.set_title("Tipos de carros disponíveis nos EUA no ano de 1993")
13
14 plt.show()
```

Fonte: Autoria própria

Conforme pode ser observado, o código e a maneira de criar o gráfico é a mesma que se encontra na figura 22, por se tratar de um gráfico do Matplotlib na sua essência, ficando a diferença na personalização trazida pelo Seaborn com a inclusão das instruções contidas na linha 10.

A próxima imagem mostra o resultado obtido após o uso de um único recurso de estilização da biblioteca do Seaborn. Sugere-se a comparação dessa figura com a de número 23, a qual se encontra na seção 5.3.

Figura 37 - Gráfico de pizza com estilização provida pelo Seaborn




Fonte: Autoria própria

## 6.4 Mais possibilidades em Seaborn

Nesta seção serão incluídos alguns outros gráficos bastante populares na literatura e que podem ser criados com o uso do Seaborn, são eles: Gráficos de Dispersão, e o de Caixa (*Boxplot*). Vale a mesma nota já dita anteriormente, com a adição destes gráficos feitos com Seaborn não se espera demonstrar tudo que é possível ser feito com o uso dessa biblioteca, sabendo-se que ela também tem várias funções específicas para criação de gráficos, suas variações, e personalizações.

Os gráficos aqui incluídos e seus comandos de implementação foram extraídos da documentação encontrada no sítio oficial, podendo ser acessada pelo link https://seaborn.pydata.org/examples/index.html, bem como pelo próximo link o qual contém diversos tutoriais https://seaborn.pydata.org/tutorial.html.

O primeiro gráfico desta parte da pesquisa trata-se do Gráfico de dispersão que tem o objetivo de plotar duas variáveis distintas nos eixos X e Y do plano cartesiano a fim de verificar se há relação ou não entre as variáveis estudadas. Esses gráficos podem exibir uma relação positiva (quando as duas variáveis crescem juntas), uma relação negativa (uma crescendo ao passo que a outra diminui), e uma relação nula (quando não se vê um padrão entre as séries avaliadas). Para a sua criação no Seaborn usa-se a função “.scatterplot()”.

A próxima imagem exibe as poucas linhas (três no total) de código usadas para a sua elaboração, sendo que a primeira está carregando os dados de um *dataset* já existente no Seaborn para fins de instrução, a segunda exibe de forma tabular um pouco dos dados carregados, e a terceira e última linha é a única realmente necessária para a criação do gráfico. O seu resultado visual é visto na figura subsequente, de número 39.

Figura 38 - Implementação do Gráfico de Dispersão no Seaborn

```
tips = sns.load_dataset("tips")
tips.head()
```

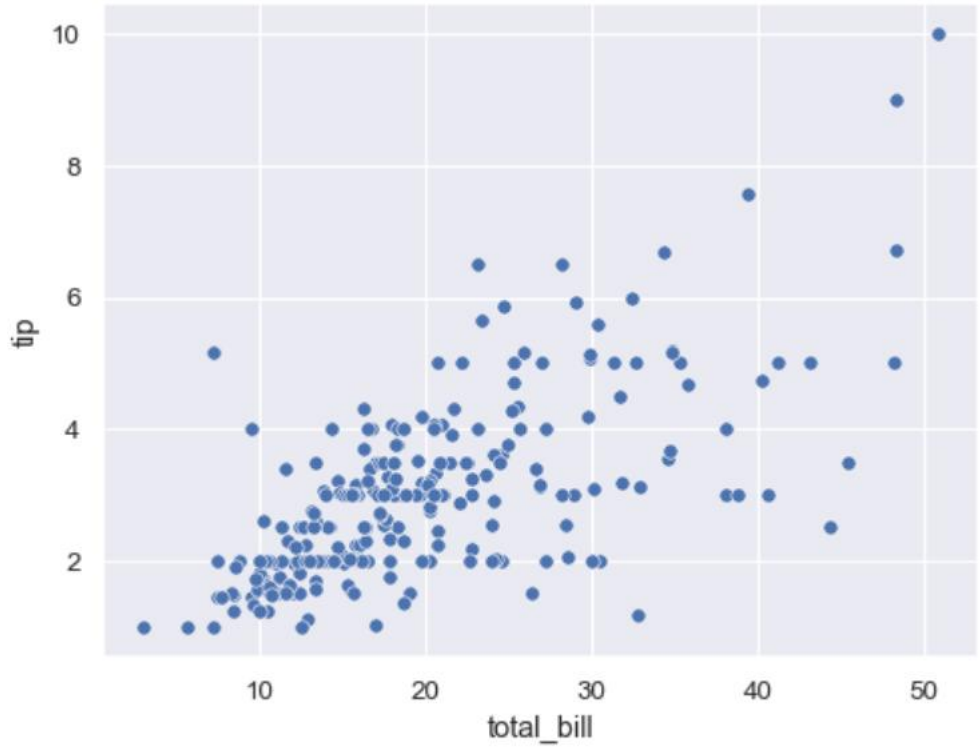
| | total_bill | tip | sex | smoker | day | time | size |
|---|---|---|---|---|---|---|---|
| **0** | 16.99 | 1.01 | Female | No | Sun | Dinner | 2 |
| **1** | 10.34 | 1.66 | Male | No | Sun | Dinner | 3 |
| **2** | 21.01 | 3.50 | Male | No | Sun | Dinner | 3 |
| **3** | 23.68 | 3.31 | Male | No | Sun | Dinner | 2 |
| **4** | 24.59 | 3.61 | Female | No | Sun | Dinner | 4 |

Passing long-form data and assigning `x` and `y` will draw a scatter plot between two variables:

```
sns.scatterplot(data=tips, x="total_bill", y="tip")
```

Fonte: https://seaborn.pydata.org/generated/seaborn.scatterplot.html

Figura 39 - Gráfico de dispersão no Seaborn



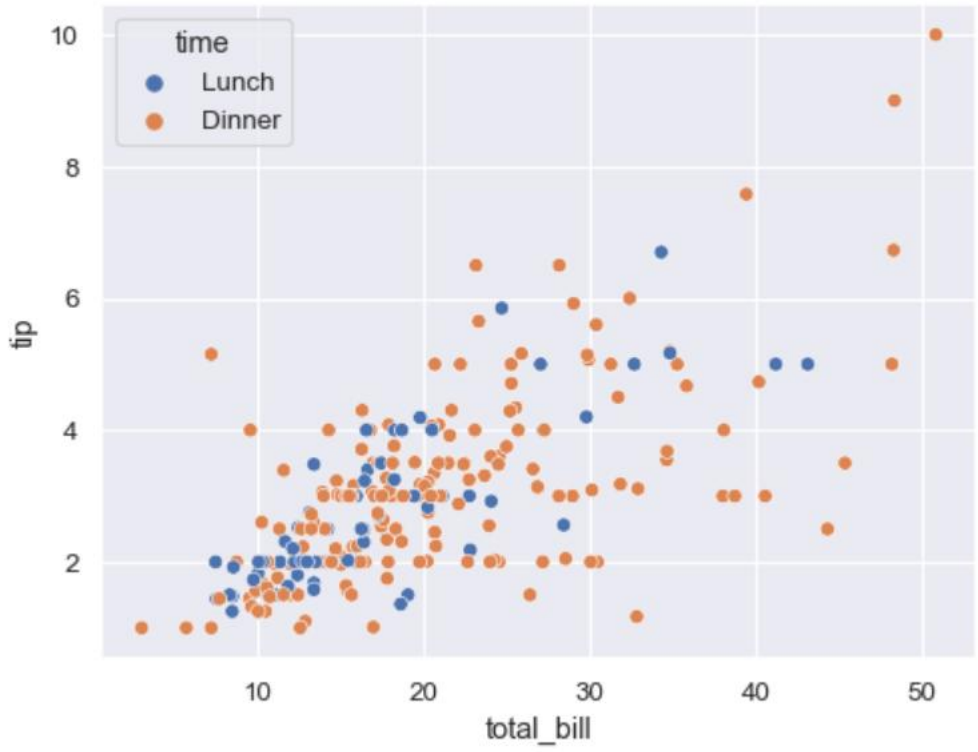

Fonte: https://seaborn.pydata.org/generated/seaborn.scatterplot.html

O gráfico de dispersão ele tem uma ampla gama de possibilidades de personalizações que permitem assim demonstrar mais variáveis do que as duas iniciais (as quais foram posicionadas nos eixos X e Y), fazendo assim com que o gráfico adquira ainda mais potencial informativo, gerando mais valor e conhecimento ao seu leitor. Algumas possibilidades para tanto são: inclusão de cores aos pontos, ou então representar os pontos com formatos e figuras geométricas, sendo assim possível diferenciar várias categorias constantes do conjunto de dados estudados.

Para ilustrar isso, nessa ocasião será exibida uma única imagem que contempla a implementação de cores, diferenciando o momento do dia em que as gorjetas são dadas, e logo abaixo o resultado atingido. Vale destacar que a única diferença na implementação foi a inclusão do parâmetro "*hue*" à função retromencionada (".scatterplot()"), e a indicação de que esse parâmetro receberia a coluna "*time*" do *dataset* carregado, a saber "*Tips*" (gorjetas, em tradução livre).

Figura 40 - Gráfico de dispersão com personalização de cores para inclusão de terceira variável de estudo

```
sns.scatterplot(data=tips, x="total_bill", y="tip", hue="time")
```




Fonte: https://seaborn.pydata.org/generated/seaborn.scatterplot.html

O outro gráfico a ser apresentado aqui é o de caixa, também conhecido pelo seu nome em inglês, *Boxplot*. É um gráfico essencialmente de estatística e que demonstra medidas de posição (primeiro, segundo e terceiro quartis) de uma amostragem ou população, além de exibir as ocorrências de valor mais baixo e de valor mais alto que aparecem no conjunto de dados, e ainda eventuais pontos que estejam fora do intervalo, esses conhecidos como *outliers*.

No Seaborn, a função usada para a sua criação é a ".boxplot()", e ela é demonstrada na próxima imagem. É mister mencionar que a instrução para criação é dada em apenas uma linha de código, a segunda na ocasião da ilustração em tela. A primeira está carregando o conjunto de dados "Titanic" (que contém informações de idade, gênero, classe de embarque no navio, e se sobreviveu ou não ao desastre) na variável "df".

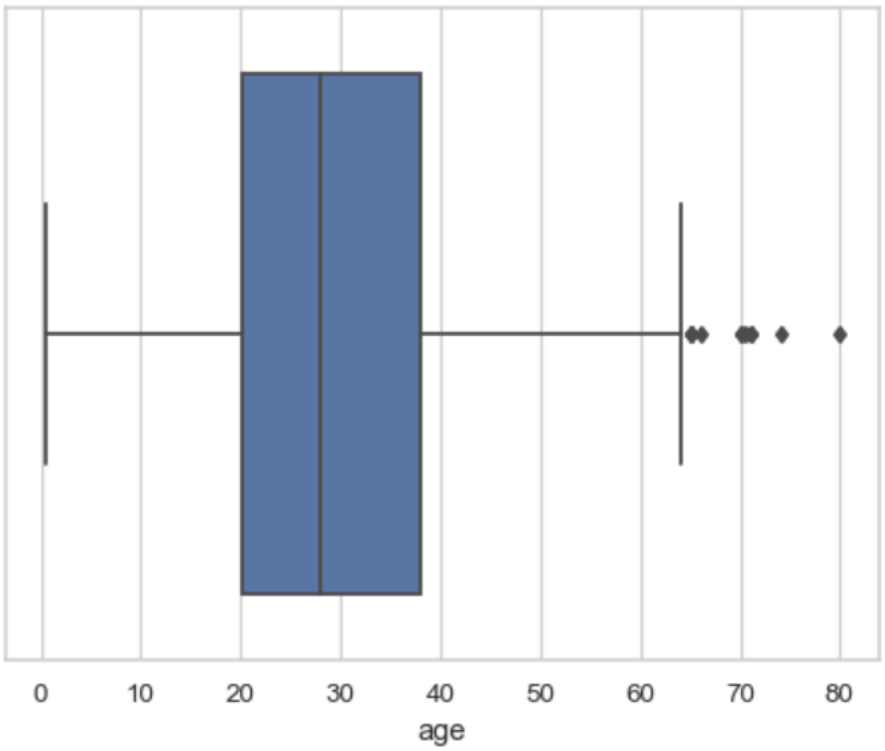
Figura 41 - Implementação do Gráfico de caixa no Seaborn

```
df = sns.load_dataset("titanic")
sns.boxplot(x=df["age"])
```

Fonte: https://seaborn.pydata.org/generated/seaborn.boxplot.html

A próxima figura exibe o gráfico de ocorrências das idades dos passageiros a bordo do Titanic, sendo que essa informação é demonstrada horizontalmente, em relação ao eixo X.

Figura 42 - Gráfico de caixa no Seaborn

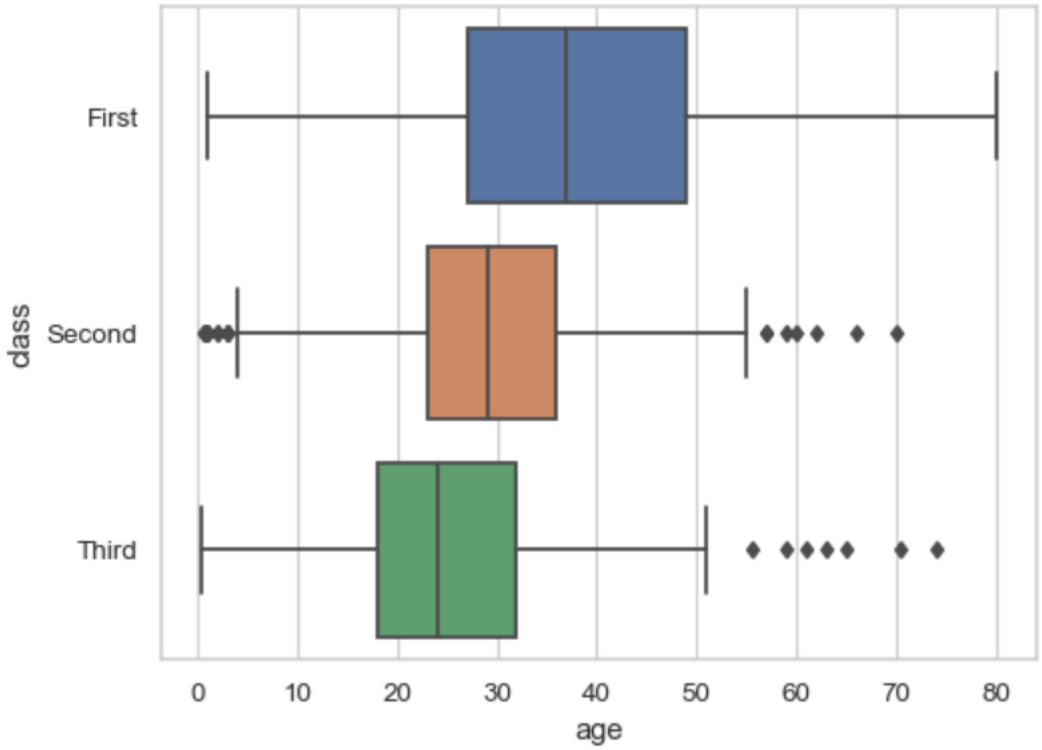



Fonte: https://seaborn.pydata.org/generated/seaborn.boxplot.html

A exemplo do que foi comentado previamente quando se apresentou o gráfico de dispersão, o de caixa também pode ser personalizado, e ter mais variáveis incluídas na mesma figura. Nesta ocasião, a título de exemplificação mostrar-se-á a inclusão da variável classe de embarque a ser exibida no eixo Y. Desta maneira é possível comparar as idades dos passageiros que estavam nas três classes do navio. Para alcançar esse objetivo foi necessário incluir o eixo Y como parâmetro e atribuir a ele a coluna “*class*” do *dataframe* “Titanic”. Vide abaixo implementação feita em apenas uma linha de código, e o respectivo resultado (tudo na mesma ilustração).

Figura 43 - Gráfico de caixa no Seaborn - com inclusão de variável adicional para ser plotada em referência ao Eixo Y

```
sns.boxplot(data=df, x="age", y="class")
```




Fonte: https://seaborn.pydata.org/generated/seaborn.boxplot.html

Nesta seção tendo incluído esses dois gráficos (Dispersão e Caixa) e algumas alternativas para enriquecimento deles, o objetivo foi apenas de ilustrar de maneira rápida que o Seaborn possui outras inúmeras figuras disponíveis para criação com funções específicas e essas podem ser encontradas na sua documentação (a qual recomenda-se a consulta).

Os gráficos gerados a partir do Seaborn em geral são dotados de bastante elegância e de fácil compreensão por parte do leitor, auxiliando assim o cientista na comunicação dos resultados das suas análises aos seus interlocutores.

# 7 ANÁLISE DOS RESULTADOS

Diante do uso do Python e das duas bibliotecas para visualização de dados apresentadas ao longo desta pesquisa observa-se um conjunto de ferramentas muito poderoso para a construção de gráficos, sendo que cada uma delas tem um enfoque e potencial diferente da outra.

Vale mencionar que todas as essas bibliotecas dispõem de uma documentação bem elaborada e disponível nos seus respectivos sítios oficiais. Fato esse que viabiliza uma rápida curva de aprendizagem por parte dos seus usuários. Há inúmeros exemplos, descrições de uso, e explicações que visam capacitar os utilizadores para atenderem as suas necessidades quando usando essas bibliotecas.

As duas bibliotecas possuem uma extensa lista de gráfico disponíveis para plotagem de dados, sendo que neste rol encontram-se desde os tipos mais básicos, como outros capazes de atender os mais diversos e complexos problemas de ordem científica auxiliando nos mais diversos campos: estatística, engenharia, saúde, meteorologia etc.

Aqui apresentou-se gráficos comuns e bastante difundidos tais como: barras verticais, linhas e pizza; os quais são facilmente construídos e manipuláveis tanto usando-se o Matplotlib, como usando-se o Seaborn.

Quanto ao gráfico de barras verticais, as duas ferramentas têm funcionamento equivalente e permitem uma ampla variedade de plotagens distintas. É possível criar o gráfico com a exibição de apenas um grupo de dados (conforme Figura 25), ou lado a lado como em um painel (Figura 27), ou com dois gráficos um estando sobreposto ao outro permitindo que barras de séries distintas de dados fiquem adjacentes (Figura 16) viabilizando uma capacidade de comparação ao observador da imagem. É importante destacar que foi usado o Matplotlib para a realização de algumas personalizações dos gráficos gerados com Seaborn de uma maneira mais simples.

Igualmente o gráfico de linhas pode ser criado pelas duas bibliotecas, nas suas mais variadas nuances: com apenas uma série de dados plotada (uma única linha na imagem), com duas ou mais linhas, cores ou tracejados distintos, posicionando-se plotagens de conjuntos de dados distintos lado a lado montando-se um painel que possibilite a comparação e análise de diferentes evoluções de padrões de tendências.

Com relação ao gráfico de pizza, nesse sim é possível constatar a existência de uma diferença significativa entre o Matplotlib e o Seaborn, tendo em conta que o Seaborn não possui função nativa para criação deste modelo de gráfico, enquanto o Matplotlib sim possui uma função dedicada, a ".pie()". Assim, o gráfico de pizza é criado apenas pelo Matplotlib. Porém como ficou demonstrado na figura 33, é viável criar uma visualização estilo pizza e caracterizá-lo com recursos do Seaborn alternado a sua identidade visual.

Vale ainda dizer, como pode ser visto no Apêndice A, que o Excel é capaz de criar os mesmos gráficos apresentados até o momento, além de outros que também são criados pelas ferramentas estudadas. No entanto, a lista de gráficos criados pelo Excel tem limitações quando comparada às bibliotecas Python específicas para criação de gráficos. Além disso, a criação de gráficos pelo Excel não é tão simples ou intuitiva da forma como pode se imaginar por este ser um software de interface gráfica; para que um gráfico seja criado tal qual o usuário necessite, os dados precisam estar dispostos da maneira correta e muito bem tratados, caso contrário o gráfico não será exibido adequadamente.

Destarte, sob algumas circunstâncias, por vezes os gráficos criados com o Matplotlib podem ter uma aparência mais simples e lhe faltar um pouco de apelo visual, mas isso é compensado satisfatoriamente ao se observar o potencial do seu algoritmo de criação e as possibilidades de criação que a biblioteca oferece ao usuário. Além disso vale ressaltar que outro ponto que contribui grandemente a favor dele é o tempo que ele já está disponível para uso (desde sua criação em 2007), e o fato dele servir de ponto de partida para a construção de várias outras bibliotecas para visualização de dados com Python.

Por sua vez, o Seaborn possui criações com gráficos mais elaborados e com apelo visual mais moderno, deixa de ter alguns gráficos que estão presentes no Matplotlib, e em geral possui um foco mais voltado para gráficos estatísticos, tal qual está descrito na página inicial do seu sítio oficial, "Seaborn é uma biblioteca Python para visualização dados baseada em Matplotlib. Ela provê uma interface de alto nível para o desenho de gráficos estatísticos informativos e atrativos."

No que diz respeito a aspectos de facilidade de uso entre uma e outra comparativamente, ao menos no que diz respeito aos gráficos aqui apresentados, não fica notório nenhum diferencial que se justifique mencionar no sentido de que uma é mais fácil de criar do que a outra. Evidente que há sim diferenças entre as duas

ferramentas, mas o grau de “dificuldade” para o uso delas é similar, no caso considerando-se bem baixo, ou seja, as duas são bem fáceis de usar.

Assim, não há que se falar em uma sendo melhor do que a outra, mas sim que são complementares e podem ser mais adequadas para um ou outro uso a depender do resultado pretendido pelo cientista ou desenvolvedor. Ademais, tal qual foi demonstrado no item 6.3, as duas podem ser usadas conjuntamente quando necessário para obtenção de melhores resultados ou quando alguma limitação advier do uso de uma delas individualmente.

# 8 CONCLUSÕES

De acordo com o desenvolvimento da pesquisa e com o término desta, é possível avaliar que os resultados esperados inicialmente foram, sim, atingidos. As bibliotecas Matplotlib e Seaborn em conjunto com as bibliotecas de manipulação de dados Pandas e Numpy se mostraram eficazes na construção de gráficos claros, expressivos e de fácil compreensão.

Para a elaboração dos gráficos havia duas opções incialmente, a saber: a primeira seria a de configurar a máquina com a instalação do Python localmente, bem como as bibliotecas e pacotes necessários para a criação das visualizações; e havia a segunda alternativa que seria a de fazer uso de algum ambiente online para desenvolver com Python, sendo o Jupyter Notebook e o Google Colab duas das mais conhecidas ferramentas que viabilizam esse desenvolvimento online (assunto abordado no capítulo 3). Nesta ocasião optou-se pelo uso do Google Colab, e o resultado foi bastante satisfatório pois sua usabilidade é bastante direta, bastando ter uma conta do Google para poder alocar uma máquina virtual dentro do Colab, sendo possível o acesso deste ambiente a partir de qualquer máquina conectada na internet e autenticada na conta do Google.

Assim, com o encerramento deste trabalho conclui-se que o Python e suas bibliotecas de visualização de dados, neste caso representadas pelo Matplotlib e Seaborn, é uma linguagem com grande capacidade para apoio do cientista de dados em trabalhos que vão desde um nível mais básico até aqueles que possuem um elevado grau de complexidade. O escopo do Python na tarefa de auxiliar no trabalho com dados é bastante abrangente, abarcando as atividades de acesso a dados, manipulação, limpeza e enriquecimento destes, atingindo até as etapas de criação de visualizações com qualidade de publicação para compartilhamento de informações e conhecimento.

Além disso, foi possível atestar o caráter fundamental que as imagens e gráficos gerados a partir de grandes conjuntos de dados possui atualmente no contexto dos frameworks de big data. Estas visualizações têm a função de auxiliar na interpretação, e de revelar sentidos e tendências que antes estavam ininteligíveis no emaranhado inicial de dados brutos armazenados nos *data lakes*.

Uma das intenções iniciais deste trabalho foi conferir aos seus leitores um primeiro contato com o tema das visualizações de dados, além de apresentar o ecossistema Python disponível para trabalho nesta área.

Espera-se que a partir deste texto, o leitor possa prosseguir com estudos mais pormenorizados acerca dos assuntos aqui abordados, explorando novas categorias de gráficos, ou até mesmo que possa evoluir no estudo de outras bibliotecas que desenvolvem a mesma função de criação de gráficos como por exemplo o Plotly, o Ggplot (baseado na implementação Ggplot2 da linguagem R, e influenciado pelos conceitos de Gramática de dados), e o Bokeh. Vale dizer que essa lista é apenas exemplificativa e não pretende ter caráter extensivo ou definitivo, o que torna evidente o potencial de pesquisas que existe uma vez adentrado nesta seara de visualização de dados com Python.

# REFERÊNCIAS

ALEXANDRE, Dulclerci Sternadt; TAVARES, J. M. R. S. Factores da percepção visual humana na visualização de dados. In: CMNE 2007-Congresso de Métodos Numéricos em Engenharia, XXVIII CILAMCE-Congresso Ibero Latino-Americano sobre Métodos Computacionais em Engenharia, Porto, PT. 2007.

BARRETT, Paul et al. matplotlib--A Portable Python Plotting Package. In: Astronomical data analysis software and systems XIV. 2005. p. 91.

CHALLENGER-PÉREZ, Ivet; DÍAZ-RICARDO, Yanet; BECERRA-GARCÍA, Roberto Antonio. El lenguaje de programación Python. Ciencias Holguín [en linea]. 2014, XX(2), 1-13[fecha de Consulta 27 de Abril de 2023]. ISSN: . Disponível em: https://www.redalyc.org/articulo.oa?id=181531232001

HARRIS, C.R., MILLMAN, K.J., VAN DER WALT, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020)

HUNTER, J. D. Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, 9(3), 90–95, 2007.

IRIZARRY, Rafael A. Introduction to data science: Data analysis and prediction algorithms with R. CRC Press, 2019.

MACEDO, Daiane et al. Uma ferramenta para recomendação de visualização de dados governamentais abertos. In: Anais do VIII Workshop de Computação Aplicada em Governo Eletrônico. SBC, 2020. p. 96-107.

MIDWAY, Stephen R. Principles of effective data visualization. Patterns, v. 1, n. 9, p. 100141, 2020.

PEREIRA, Flávia Patricia Alves. Big data e data analysis: visualização de informação. 2015. Tese de Doutorado.

SILVA, Fabiano Couto Corrêa da. Visualização de dados: passado, presente e futuro. LIINC em revista. Rio de Janeiro, RJ. Vol. 15, n. 2 (nov. 2019), p. 205-223, 2019.

SRINATH, K. R. Python–the fastest growing programming language. International Research Journal of Engineering and Technology, v. 4, n. 12, p. 354-357, 2017.

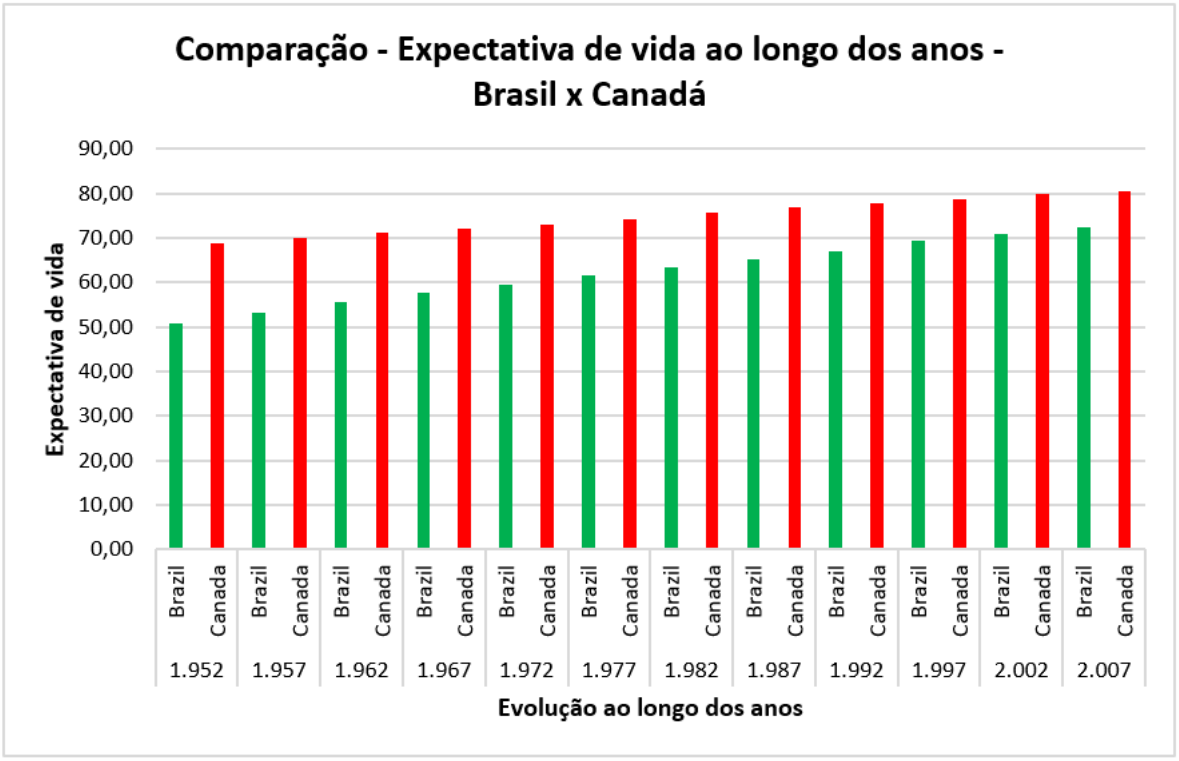
WASKOM, Michael L. Seaborn: statistical data visualization. Journal of Open Source Software, v. 6, n. 60, p. 3021, 2021.

# APÊNDICE A – GRÁFICOS GERADOS EM EXCEL

À guisa de comparação e com a finalidade de enriquecer este trabalho são trazidos aqui os mesmos gráficos gerados pelo Matplotlib e Seaborn. É de suma importância destacar que o Excel tem a capacidade de gerar os mesmos gráficos, apesar da limitação de não dispor de alguns gráficos os quais são contemplados pelas ferramentas retromencionadas. Da mesma forma, a exemplo do que já foi abordado, é necessário deixar os dados todos ajustados da maneira correta e compatível para que o Excel possa criar as imagens tais quais se espera.

A próxima imagem mostra o gráfico de barras verticais elaborado pelo Excel, com os mesmos dados e da versão gerada pelo Matplotlib (Seção 5.1)

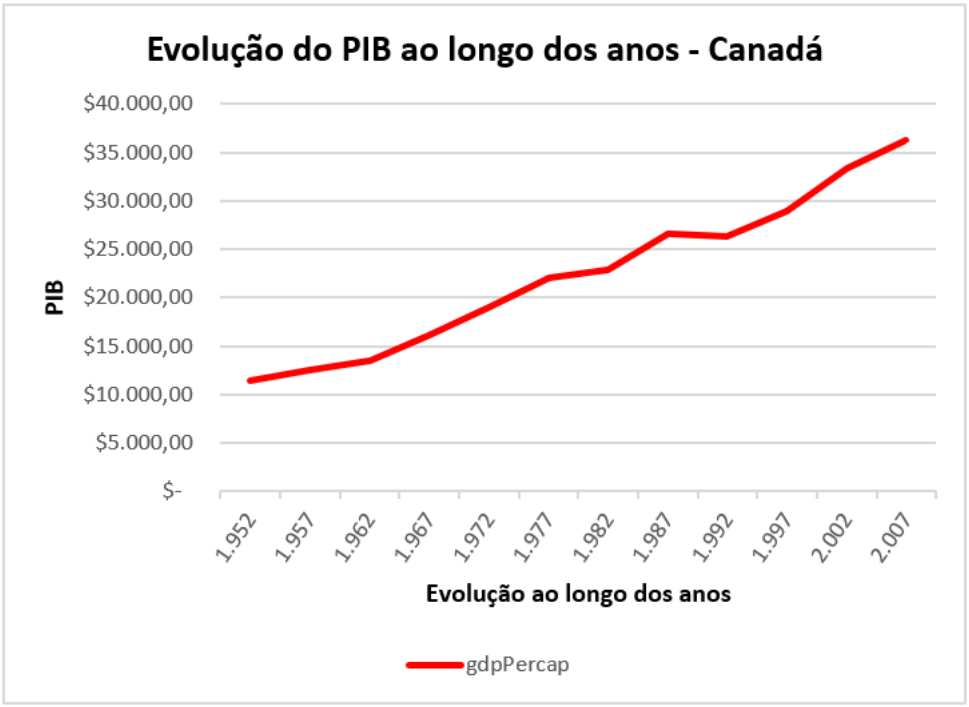
Figura 44 - Gráfico de barras verticais (Excel)




Fonte: Autoria própria

A próxima ilustração demonstra o gráfico de linhas nos moldes do gerado pelo Matplotlib com as informações do PIB canadense.

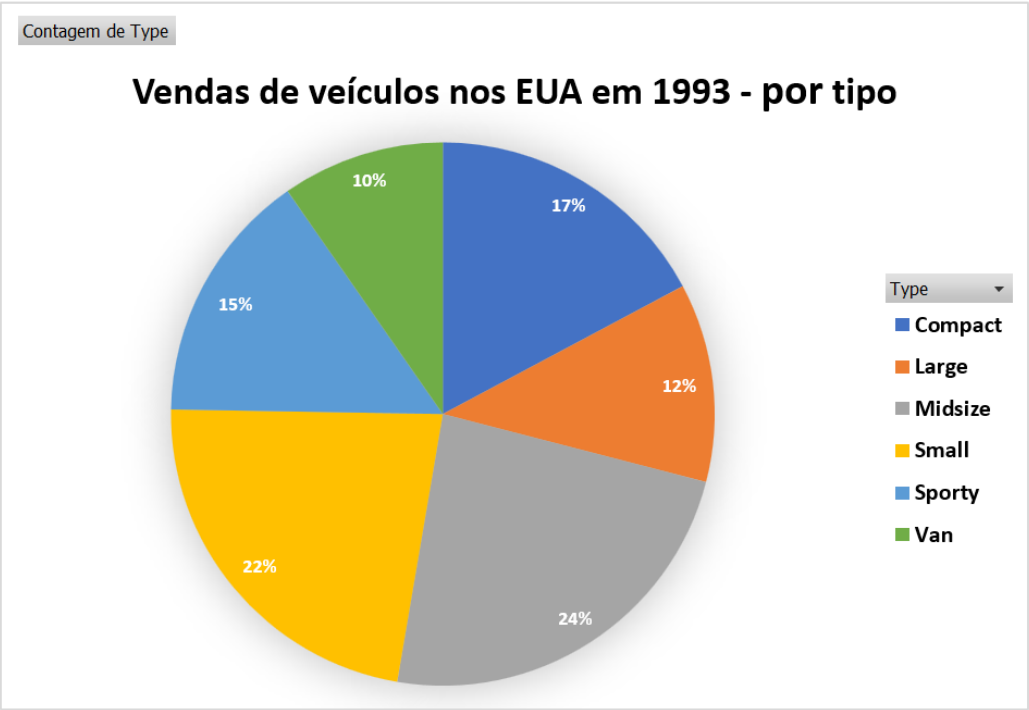
Figura 45 - Gráfico de linhas (Excel)




Fonte: Autoria própria

Na continuidade, a próxima ilustração exibe o gráfico de pizza gerado pelo Excel fazendo uso do *dataset* “Cars93”.

Figura 46 - Gráfico de pizza (Excel)




Fonte: Autoria própria

# APÊNDICE B – ORGANIZAÇÃO DO GOOGLE COLAB

Neste apêndice apresenta-se como foi organizado o Google Colab viabilizando-o para a execução das bibliotecas necessárias para a criação das visualizações de dados com Python.

É importante esclarecer que este ambiente também é conhecido como “caderno”, muito em função da outra ferramenta de grande popularidade, Jupyter Notebook. Ainda em caráter de explicação prévia, vale mencionar que ele é estruturado em células, as quais podem ser usadas para programação e nas quais já é apresentado o resultado do código escrito; ou então podem ser usadas como áreas para texto corrido.

Assim, após esse rápido preâmbulo, apresenta-se a estrutura básica para a elaboração desta pesquisa. Na primeira célula foi feita a instalação das bibliotecas e pacotes (com os *datasets)* que seriam utilizados, veja figura a seguir.

Figura 47 - Célula com comandos para instalação das bibliotecas e pacotes no Google Colab

```
#Célula para instalações das bibliotecas necessárias

#Instalando NumPy
#Site: https://numpy.org/
!pip install numpy

#Instalando Pandas
#Site: https://pandas.pydata.org/
!pip install pandas

#Instalando Matplotlib
#Site: https://matplotlib.org/
!pip install matplotlib

#Instalando Seaborn
#Site: https://seaborn.pydata.org/
!pip install seaborn

#Instalando Dataset Gapminder
#Site: https://pypi.org/project/gapminder/
!pip install gapminder

#Instalando Pydataset
#Site: https://pydataset.readthedocs.io/en/latest/index.html
!pip install pydataset
```

Fonte: Autoria própria

A segunda célula do caderno foi usada para fazer a importação dos elementos os quais deveriam estar alocados para uso durante a interpretação dos comandos pelo Python a fim de que suas funções funcionassem conforme o esperado. Convém destacar que uma vez executada cada uma dessas células, não era mais necessário repetir qualquer instalação ou importação dessas. Segue imagem demonstrando as importações.

Figura 48 - Importações elementos no Google Colab

```
#Importando NumPy
import numpy as np
print('O Numpy foi importado como np e sua versão é: ' + np.__version__)
print('\n')

#Importando Pandas
import pandas as pd
print('O Pandas foi importado como pd e sua versão é: ' + pd.__version__)
print('\n')

#Importando Matplotlib
import matplotlib as mpl
import matplotlib.pyplot as plt
print('O Matplotlib foi importado como mpl e sua versão é: ' + mpl.__version__)
print('\n')

#Importando Seaborn
import seaborn as sns
import random
import warnings
warnings.filterwarnings('ignore')
print('O Seaborn foi importado como sns e sua versão é: ' + sns.__version__)
print('\n')

#Importando Dataset Gapminder
from gapminder import gapminder

#Importando Pydataset
from pydataset import data
```

Fonte: Autoria própria

Na ilustração anterior observa-se a presença de instruções do tipo “print()”, que foram colocadas como uma forte de atestar que os elementos haviam sido importados com sucesso, bem como de verificar qual a versão estava em uso, sendo assim possível pesquisar com precisão a ajuda necessária na documentação quando preciso. O resultado dessa célula encontra-se abaixo.

Figura 49 - Resultado dos *imports* exibidos na célula do Google Colab

```
O Numpy foi importado como np e sua versão é: 1.22.4

O Pandas foi importado como pd e sua versão é: 1.5.3

O Matplotlib foi importado como mpl e sua versão é: 3.7.1

O Seaborn foi importado como sns e sua versão é: 0.12.2
```

Fonte: Autoria própria

# APÊNDICE C – DEMONSTRAÇÃO DOS DATASETS UTILIZADOS

Por fim, com o objetivo de ilustrar ainda mais este trabalho e trazer elementos palpáveis ao leitor deste que porventura não poderá replicar os testes realizados aqui, traz-se as linhas iniciais de cada um dos *datasets* utilizados na execução deste trabalho.

Esta primeira imagem contém as informações contidas no conjunto de dados Gapminder, o qual foi tratado previamente no início do capítulo 5 – “Gráficos com Matplotlib”

Figura 50 - Conteúdo do *dataset* "Gapminder"

```
#Visualizando o conteúdo do Dataset Gapminder

gapminder.head()
```

| | country | continent | year | lifeExp | pop | gdpPercap |
|---|---|---|---|---|---|---|
| 0 | Afghanistan | Asia | 1952 | 28.801 | 8425333 | 779.445314 |
| 1 | Afghanistan | Asia | 1957 | 30.332 | 9240934 | 820.853030 |
| 2 | Afghanistan | Asia | 1962 | 31.997 | 10267083 | 853.100710 |
| 3 | Afghanistan | Asia | 1967 | 34.020 | 11537966 | 836.197138 |
| 4 | Afghanistan | Asia | 1972 | 36.088 | 13079460 | 739.981106 |

Fonte: Autoria própria

A próxima ilustração contém as informações contidas no conjunto de dados “Cars93”, o qual também foi devidamente abordado no início do capítulo 5 – “Gráficos com Matplotlib”.

Vale destacar que nas duas ocasiões para limitar a exibição das 5 linhas iniciais das tabelas foi usada a função “.head()” a qual aplica esse “filtro” nos dados.

Figura 51 - Conteúdo do conjunto de dados "Cars93"

```
#Visualizando o conteúdos do dataset Cars93 (Pydataset)

data('Cars93').head()
```

| | Manufacturer | Model | Type | Min.Price | Price | Max.Price | MPG.city | MPG.highway | AirBags | DriveTrain | ... | Pas |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | Acura | Integra | Small | 12.9 | 15.9 | 18.8 | 25 | 31 | None | Front | ... | |
| **2** | Acura | Legend | Midsize | 29.2 | 33.9 | 38.7 | 18 | 25 | Driver & Passenger | Front | ... | |
| **3** | Audi | 90 | Compact | 25.9 | 29.1 | 32.3 | 20 | 26 | Driver only | Front | ... | |
| **4** | Audi | 100 | Midsize | 30.8 | 37.7 | 44.6 | 19 | 26 | Driver & Passenger | Front | ... | |
| **5** | BMW | 535i | Midsize | 23.7 | 30.0 | 36.2 | 22 | 30 | Driver only | Rear | ... | |

5 rows × 27 columns

Fonte: Autoria própria