

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/336778766>

# Introdução à Análise Exploratória de Dados com Python

Conference Paper · October 2019

CITATIONS

0

READS

10,171

4 authors, including:



**Gesiel Rios Lopes**

University of São Paulo

10 PUBLICATIONS 2 CITATIONS

[SEE PROFILE](#)



**Alexandre C. B. Delbem**

University of São Paulo

217 PUBLICATIONS 1,945 CITATIONS

[SEE PROFILE](#)



**Claudio Toledo**

University of São Paulo

99 PUBLICATIONS 735 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Multidisciplinary Tinnitus Rehabilitation (MTR) Project [View project](#)



Protein structure prediction [View project](#)

## Capítulo

# 8

## Introdução à Análise Exploratória de Dados com Python

Gesiel Rios Lopes, Alessandro Wilk Silva Almeida,  
Alexandre C. B. Delbem, Cláudio Fabiano Motta Toledo

### *Resumo*

*O avanço da tecnologia, observados nos últimos anos, aliado com a popularização da Internet e com o aumento na quantidade e complexidade dos serviços oferecidos por ela, contribuíram de forma significativa à geração massiva de dados. A manipulação e a análise de forma inteligente desse volume de dados têm se tornado um dos grandes desafios computacionais da atualidade. Para este fim, a Análise Exploratória de Dados (AED) é bem adequada, uma vez que é bem conhecida em estatística e ciências. A abordagem operacional da análise de dados visava melhorar a compreensão e a acessibilidade dos resultados. Sem esquecer a solidez dos modelos estatísticos e a formulação de hipóteses, que está intrinsecamente ligada ao conceito de “análise” em seu significado científico, o foco é transferido para a “exploração”. A AED se relaciona com o processo de revelar informações ocultas e desconhecidas dos dados de tal forma que o analista obtém uma representação imediata, direta e fácil de entender. Gráficos visuais são um elemento obrigatório desta abordagem, devido à capacidade intrínseca do cérebro humano de obter uma interpretação mais direta e confiável de similaridades, diferenças, tendências, clusters e correlações através de uma imagem, ao invés de uma série de números. Neste minicurso, será apresentado conceitos sobre como utilizar a linguagem de programação Python para explorar um conjunto de dados, o que é essencial para obter uma boa compreensão e possíveis problemas de um conjunto de dados, além de auxiliar na geração de hipóteses que podem ser extraídas a partir da análise de um conjunto de dados.*

### **8.1. Introdução à análise exploratória de dados**

A Análise Exploratória de Dados (AED) teve seu início com J.W. Tukey [Tukey 1977] e visa aumentar o conhecimento do pesquisador sobre uma população a partir de uma amostra. Dessa forma, podemos descrever a AED como um conjunto de métodos adequados para a coleta, a exploração e descrição e interpretação de conjuntos de dados numéricos.

Estes métodos permitem a exploração dos dados com intuito de identificar padrões de interesse e a representação dos dados caracterizados por estes padrões.

Embora as técnicas da AED sejam simples, geralmente são métodos robustos (válidos para uma grande gama de situações e modelos) e resistentes (insensíveis aos erros grosseiros ou dados estranhos).

Não é exagero reafirmar a citação de TUKEY [Tukey 1977]:

*“A AED é trabalho de detetive, procurando pistas e evidências; análise confirmatória de dados é trabalho judicial ou quase-judicial, que analisa, avalia e julga as provas e as evidências.”*

## 8.2. Por que usar python para análise exploratória de dados ?

Python é uma linguagem de programação muito utilizada em atividades de análise de dados e ao contrário de linguagens como R, Python é uma linguagem de programação de propósito geral, ou seja, não é exclusiva para atividades de análise de dados. Esta característica tem contribuído de forma significativa para aumentar o seu uso dentro das empresas, uma vez que muitas equipes de desenvolvimento de *software* já possuem um certo conhecimento com Python, o que facilita a implantação em produção de modelos desenvolvidos na mesma plataforma.

No desenvolvimento de aplicações científicas é recomendável o utilizar a distribuição Anaconda<sup>1</sup>, uma vez que ela é de código aberto e é a maneira mais fácil de executar aplicações científicas desenvolvidas em Python no Linux, Windows e Mac OS X. Dentro da distribuição Anaconda você também encontrará o Jupyter Notebook, uma interface muito interessante para criar seus modelos e compartilhar com quem quiser.

Uma alternativa similar ao Jupyter Notebook e não requer configuração para ser usada é uma ferramenta desenvolvida pelo Google chamada Colaboratory<sup>2</sup>. Para usar este ambiente, é necessário apenas ter uma conta gmail, pois todo notebook ficará armazenado no Drive.

O código do seu notebook é executado em uma máquina virtual dedicada à sua conta. As máquinas virtuais são recicladas após um determinado tempo ocioso, ou caso a janela seja fechada. Para restaurar seu notebook, talvez seja necessário refazer o *upload* do arquivo .csv e executar as opções “Runtime” e “Restart and run all”.

## 8.3. Fundamentos de python para análise exploratória de dados

As características mais relevantes do Python são a sua capacidade de integração com outras linguagens e seu sistema de bibliotecas bem maduro. As bibliotecas apresentadas a seguir são fortemente analíticas e oferecerão uma completa caixa de ferramentas de ciência de dados composta de funções altamente otimizadas para trabalhar, configuração ideal de memória, pronta para realizar operações de *script* com desempenho ideal [Boschetti and Massaron 2015].

Parcialmente inspirados por ferramentas semelhantes presentes nos ambientes R

---

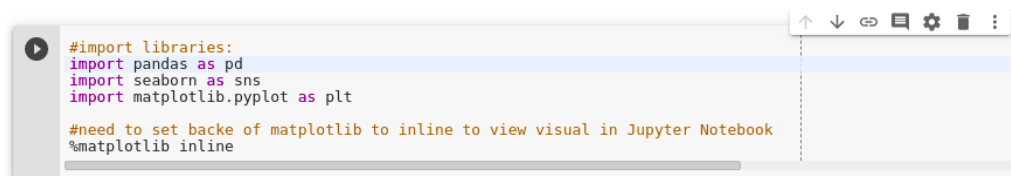
<sup>1</sup>Disponível em: <https://www.anaconda.com/distribution/>

<sup>2</sup>Disponível em: <https://colab.research.google.com/>

e *MATLAB*, serão exploradas juntamente com alguns comandos Python, de forma a permitir que se lide com dados com eficiência e depois explore, transforme, experimente e aprenda com os mesmos sem precisar escrever muito código ou reinventar a roda.

- **Pandas:** Pandas<sup>3</sup> é uma biblioteca licenciada com código aberto que oferece estruturas de dados de alto desempenho e de fácil utilização voltado a análise de dados para a linguagem de programação Python [Coelho 2017].
- **Seaborn:** Seaborn<sup>4</sup> é uma biblioteca de visualização de dados Python baseada no *matplotlib*. Ela fornece uma interface de alto nível para desenhar gráficos estatísticos atraentes e informativos.
- **Matplotlib:** O Matplotlib<sup>5</sup> é uma biblioteca de plotagem 2D do Python que produz números de qualidade de publicação em vários formatos de cópia impressa e ambientes interativos entre plataformas. O Matplotlib pode ser usado em *scripts* Python, nos *shell* Python e IPython, *notebooks* Jupyter e em servidores *web*.

A Figura 8.1 é apresentado a declaração das bibliotecas que serão necessárias para AED.



```
#import libraries:
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

#need to set backe of matplotlib to inline to view visual in Jupyter Notebook
%matplotlib inline
```

**Figura 8.1. Bibliotecas para AED.**

Para se trabalhar com Pandas e o Seaborn, é necessário estar familiarizado com duas estruturas de dados que são: *Séries* e *Dataframes*, descritos resumidamente a seguir.

### 8.3.1. Séries

A estrutura *Série*, é um array unidimensional, semelhante a uma lista em Python, no entanto criado sobre o *numpy*. Além da velocidade de processamento, a principal característica que o difere de uma lista comum é que seus índices podem ser mutáveis.

A Figura 8.2 apresenta a criação e manipulação de uma série. A primeira linha define a série *my\_serie* com 5 elementos, a segunda linha apresenta a série *my\_serie*, a terceira linha apresenta o valor armazenado no índice 2 e a quarta linha altera os índices da série pelas letras A,B,C,D,E e logo em seguida a série é impressa novamente com os novos índices.

<sup>3</sup>Disponível em <https://pandas.pydata.org/>

<sup>4</sup>Disponível em <https://seaborn.pydata.org/>

<sup>5</sup>Disponível em <https://matplotlib.org/>



```
my_series = pd.Series([10,20,30,40,50])
print(my_series)
print(my_series[2])
my_series.index = ['A', 'B', 'C', 'D', 'E']
print(my_series)
```

0 10  
1 20  
2 30  
3 40  
4 50  
dtype: int64

A 10  
B 20  
C 30  
D 40  
E 50  
dtype: int64

Figura 8.2. Criação e manipulação de uma série.

### 8.3.2. Dataframes

Dataframe é uma estrutura de dados tabular bidimensional e mutável em tamanho, potencialmente heterogênea, com eixos rotulados (linhas e colunas). Para se criar um Dataframe a partir das estruturas de dados nativas em Python, pode-se passar um dicionário de listas para seu construtor. Usando-se o parâmetro *columns*, define no construtor como as colunas serão ordenadas. Por padrão, o construtor do Dataframe irá ordenar as colunas em ordem alfabética.



```
# initialise data of lists.
data = {'Name':['Tom', 'nick', 'krish', 'jack'],
        'Age':[20, 21, 19, 18]}

# Create DataFrame
df = pd.DataFrame(data)

# Print the output.
df
```

	Name	Age
0	Tom	20
1	nick	21
2	krish	19
3	jack	18

Figura 8.3. Criação um dataframe.

A Figura 8.3 ilustra a criação e apresentação de um Dataframe contendo informações referentes à idade de quatro pessoas fictícias. A primeira coluna da Figura 8.3 apresenta os índices padrões.

Em geral, é muito mais comum a fazer a leitura de uma base de dados a partir de uma base externa, o Pandas oferece alguns métodos com essa finalidade, dentre os quais se destaca a função *read\_csv* para leituras de arquivos CSV e o *pydataset* que fornece acesso instantâneo a muitos conjuntos de dados diretamente do Python, a versão atual conta com 757 datasets prontos para serem importados e utilizados juntamente com o Pandas. A Figura 8.4 ilustra a criação de um dataframe a partir de um arquivo CSV. Na primeira linha é possível observar a utilização a função *read\_csv* e na segunda linha temos

o uso da função *head* com fornece os cinco primeiros registros de um Dataframe.



```
data = pd.read_csv("nba.csv")
data.head()
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0

**Figura 8.4. Criação um dataframe a partir de um arquivo CSV.**

Séries e Dataframes formam o núcleo dos modelos de dados do Pandas em Python. Os conjuntos de dados são lidos primeiro nos Dataframes e, em seguida, várias operações (por exemplo, *groupby*, agregações, etc.) podem ser aplicadas muito facilmente às suas colunas. A Tabela 8.1 apresenta um pequeno resumo de alguns comandos úteis para manipulação de informações em um Dataframe.

**Tabela 8.1. Comandos mais comuns em um Dataframe.**

Função	Retorno
<i>df.shape()</i>	Quantidade de linhas e colunas do Dataframe
<i>df.head()</i>	Primeiros 5 registros do Dataframe
<i>df.tail()</i>	Últimos 5 registros do Dataframe
<i>df.columns()</i>	Colunas presentes no Dataframe
<i>df.count()</i>	Contagem de dados não-nulos
<i>df.sum()</i>	Soma dos valores de um Dataframe
<i>df.min()</i>	Menor valor de um Dataframe
<i>df.max()</i>	Maior valor de um Dataframe
<i>df.describe()</i>	Resumo estatístico do Dataframe

Para demonstrar o processo exploratório de análise de dados, além de fornecer um exemplo para programadores Python que desejam praticar o trabalho com dados, será utilizado dados da Pesquisa Nacional por Amostra de Domicílio (PNAD) <sup>6</sup>, valores de ações disponibilizados pela MetaTrader <sup>7</sup> e os *datasets* disponibilizados pela função *load\_dataset()* do *seaborn*, cuja descrição dos *datasets* pode ser encontrada em: <https://github.com/mwaskom/seaborn-data>.

Explorar dados por meio de visualizações bem construídas e estatísticas descritivas é uma ótima maneira de se familiarizar com os dados com os quais você está trabalhando e formular hipóteses com base em suas observações.

## 8.4. Medidas de tendência central

As medidas de tendência central ou “de posição” procuram caracterizar a tendência do conjunto (valor “típico”) a um valor numérico que “represente” o conjunto. Esse valor

<sup>6</sup>Disponível em: <https://www.ibge.gov.br/>

<sup>7</sup>Disponível em: <https://www.metatrader5.com/>

pode ser calculado levando em conta todos os valores do conjunto ou apenas alguns valores ordenados [Tukey 1977, Stevenson and De Farias 1981].

#### 8.4.1. Média $\bar{x}$

A média aritmética simples é a soma dos valores observados dividida pelo número desses valores. A média de uma amostra é representada pelo símbolo  $\bar{x}$  (leia-se “*x barra*”), definida pela Equação 1

$$\bar{x} = \frac{x_1 + x_2 + \dots + x_n}{n} = \frac{\sum_{i=1}^n x_i}{n} \quad (1)$$

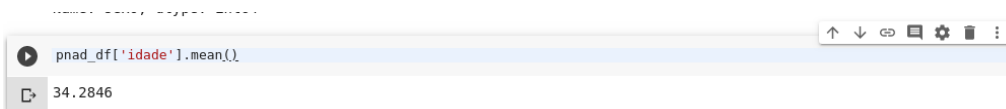
A Figura 8.5 apresenta uma das formas de se calcular a média de um conjunto de dados. A função do pandas para cálculo da média de um conjunto de valores é apresentado na Figura 8.6



```
media_idade = pnad_df['idade'].sum() / len(pnad_df['idade'])
print(media_idade)
```

34.2846

**Figura 8.5.** Uma forma de calcular a média de um conjunto de dados.



```
pnad_df['idade'].mean()
```

34.2846

**Figura 8.6.** Cálculo da média de um conjunto de dados através da função `mean()` do pandas.

#### 8.4.2. Mediana $M_e$

A mediana é o valor médio ou a média aritmética de dois valores centrais de um conjunto de números ordenados (crescente ou decrescente). Para calculá-la, primeiramente temos de reorganizar os dados em ordem crescente (ou decrescente) e, em seguida, escolher o valor central. Se o número de dados for ímpar, então este valor central é único; se for par, fazemos a média dos dois valores centrais. O pandas disponibiliza a função `median()` para determinar a mediana de um conjunto de dados, como pode ser observado da Figura 8.7.



```
pnad_df['rendimento'].median()
```

1200.0

**Figura 8.7.** Cálculo da mediana de um conjunto de dados através da função `median()` do pandas.

### 8.4.3. Moda $M_o$

A moda é o valor que ocorre com maior frequência em um dado conjunto de dados. Se todos os valores aparecem um número igual de vezes (em geral, uma vez cada), dizemos que o conjunto de dados não têm moda, ou seja, a moda pode, muitas vezes, não existir. Uma outra particularidade importante da moda é que ela é a única. O pandas disponibiliza a função `mode()` para determinar a moda de um conjunto de dados, como pode ser observado da Figura 8.8.



**Figura 8.8.** Cálculo da moda de um conjunto de dados através da função `mode()` do pandas.

### 8.4.4. Outras médias

Algumas vezes é interessante calcular médias de forma diferente para dados com características especiais.

#### 8.4.4.1. Média geométrica $G$

As médias geométricas são bastante empregadas para observações positivas referentes a crescimentos exponenciais (como taxas de avanço de doenças, números de habitantes de regiões em colonização, crescimento de produtividade, e etc...) [Stevenson and De Farias 1981]. A média geométrica ( $G$ ) é a raiz  $n$ -ésima do produto dos  $n$  valores de um conjunto de dados  $(x_1, x_2, \dots, x_n)$ , conforme definido na Equação 2.

$$G = \sqrt[n]{x_1 \times x_2 \times \dots \times x_n} = \sqrt[n]{\prod_{i=1}^n x_i} \quad (2)$$

#### 8.4.4.2. Média harmônica $H$

Para fenômenos que dependem fortemente do menor dos dados, em geral, utiliza-se médias harmônicas calculadas como o inverso da média dos inversos de um conjunto de dados  $(x_1, x_2, \dots, x_n)$ , definida conforme a Equação 3.

$$H = \frac{1}{\frac{1}{x_1} + \frac{1}{x_2} + \dots + \frac{1}{x_n}} = \frac{n}{\sum_{i=1}^n \frac{1}{x_i}} \quad (3)$$

Para o cálculo das médias geométricas e harmônicas utilizaremos as funções `gmean()` e `hmean()`, respectivamente, do módulo `scipy.stats` da biblioteca *SciPy* [Jones et al. 2001]. O módulo `scipy.stats` contém um grande número de distribuições de probabilidade, bem como uma crescente biblioteca de funções estatísticas.



A Figura 8.9 apresenta o cálculo das médias geométrica e harmônicas.



```
from scipy import stats

var = {'t1': [2, 6, 15, 59], 't2': [2000, 20, 250, 1500]}
other_means_df = pd.DataFrame(var)
|
other_means_df.head()

geometric_mean = stats.gmean(other_means_df['t1'], axis=0)
print(geometric_mean)

harmonic_mean = stats.hmean(other_means_df['t2'], axis=0)
print(harmonic_mean)
```

10.151521276336176  
72.50755287009062

**Figura 8.9.** Cálculo das médias geométricas e harmônicas através do módulo *scipy.stats* da biblioteca *SciPy*.

## 8.5. Medidas de dispersão ou variabilidade

As medidas de dispersão ou variabilidade indicam se os valores de conjunto de dados estão relativamente próximos uns dos outros, ou separados. Todas elas têm na média o ponto de referência [Tukey 1977, Stevenson and De Farias 1981].

As medidas de dispersão oferecem as condições necessárias para analisar até que ponto os valores de um conjunto de dados oscilam para mais ou para menos em relação a uma medida de posição fixada, em geral a média [Stevenson and De Farias 1981].

### 8.5.1. Amplitude

A amplitude ou intervalo total ( $I_t$ ) de um conjunto de dados  $(x_1, x_2, \dots, x_n)$  é a diferença entre o maior valor e o menor valor, conforme definido pela Equação 4

$$I_t = x_{\max} - x_{\min}, \quad (4)$$

onde:

$x_{\max}$  = valor máximo do conjunto de dados, isto é,  $\max\{x_1, x_2, \dots, x_n\}$ ;

$x_{\min}$  = valor mínimo do conjunto de dados, isto é,  $\min\{x_1, x_2, \dots, x_n\}$ .

A Figura 8.10 apresenta um exemplo do cálculo da amplitude.



```
amplitude = pnad_df['rendimento'].max() - pnad_df['rendimento'].min()
print(amplitude)
```

99985.0

**Figura 8.10.** Cálculo da amplitude.

### 8.5.2. Variância $S^2$

A variância de uma amostra de valores (dados não agrupados),  $x_1, x_2, \dots, x_n$ , é definida como sendo a média dos quadrados dos desvios das medidas em relação à sua média  $\bar{x}$ . A variância da amostrada é dada pela Equação 5.

$$S^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n} \quad (5)$$

A Figura 8.11 apresenta o cálculo da variância.



```
pnad_df['rendimento'].var()
```

```
7722933.596037565
```

**Figura 8.11. Cálculo da variância.**

### 8.5.3. Desvio padrão

o desvio padrão é definido como a raiz quadrada da média aritmética dos quadrados dos desvios em relação à média. É a mais importante medida de variabilidade. O desvio padrão é dado pela Equação 6.

$$S = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n}} \quad (6)$$

A Figura 8.12 apresenta o cálculo do desvio padrão.

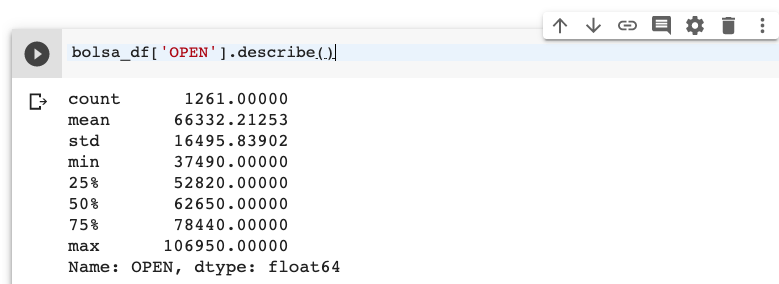


```
pnad_df['rendimento'].std()
```

```
2779.0166599064432
```

**Figura 8.12. Cálculo do desvio padrão.**

O *pandas* disponibiliza a função *describe()* que gera as estatísticas descritivas que resumem a tendência central, a dispersão e a forma da distribuição de um conjunto de dados, excluindo os valores de NaN. Analisa séries numéricas e de objetos, bem como conjuntos de colunas DataFrame de tipos de dados mistos. A Figura 8.13 é apresentado o resultado da função *describe()*.



```
bolsa_df['OPEN'].describe()
```

```
count      1261.000000
mean       66332.21253
std        16495.83902
min         37490.00000
25%         52820.00000
50%         62650.00000
75%         78440.00000
max        106950.00000
Name: OPEN, dtype: float64
```

**Figura 8.13. Resultado da função *describe()* disponibilizada pelo *pandas*.**

A Tabela 8.2 destaca as principais funções para estatísticas básicas disponíveis no Pandas.

**Tabela 8.2. Funções estatísticas básicas do Pandas.**

Função	Retorno
<i>mean()</i>	Média
<i>median()</i>	Mediana
<i>mode()</i>	Moda
<i>var()</i>	Variância
<i>std()</i>	Desvio padrão
<i>describe()</i>	Resumo Estatístico

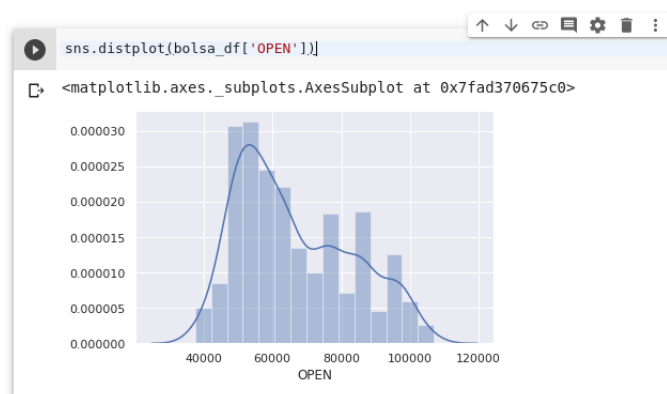
## 8.6. Visualização gráfica de dados

A apresentação dos dados estatísticos através de tabelas ou medidas de centralidade e variabilidade nem sempre proporciona um entendimento adequado dos dados. Assim, com a finalidade de melhorar esse processo, muitos recorrem ao uso dos gráficos. Para isso, é necessário saber o que se pretende mostrar, como elaborar o gráfico e qual o tipo de gráfico mais apropriado para cada tema abordado. Dependendo da mídia em que o gráfico será levado a público, é importante considerar aspectos técnicos da sua elaboração como: o uso de cores e texturas, a espessura das linhas e as fontes dos textos na composição das legendas [Tukey 1977, Stevenson and De Farias 1981]. Nesta seção serão apresentados os gráficos mais utilizados dentro da análise de dados.

### 8.6.1. Histogramas

O histograma representa a distribuição dos dados, formando compartimentos ao longo do intervalo dos dados, desenhados por barras para mostrar a frequência de observações dos dados em cada compartimento [Vigni et al. 2013, Stevenson and De Farias 1981].

O *seaborn* disponibiliza a função *distplot()* para “plotar” histogramas. Essa função combina a função *hist* do *matplotlib* com as funções *kdeplot()* e *rugplot()* do próprio *seaborn* [Haslwanter 2016]. A Figura 8.14 apresenta um exemplo de um histograma a partir da função *distplot()* do *seaborn*.



**Figura 8.14. Histograma.**

### 8.6.2. Gráficos de barra e linhas

No gráfico de barras, cada categoria é representada por uma barra de comprimento proporcional à sua frequência, conforme identificação no eixo horizontal.

O *seaborn* disponibiliza a função `catplot()` para “plotar” gráficos de barra. Esta função fornece acesso a várias funções no nível de eixos que mostram o relacionamento entre uma série numérica de uma ou mais variáveis categóricas, usando uma das várias representações visuais. O parâmetro `kind` seleciona a função de nível de eixo subjacente a ser usada [Haslwanter 2016]. As Figuras 8.15(a) e 8.15(b) apresentam um exemplo de um gráfico de barra a partir da função `catplot()` do *seaborn*.

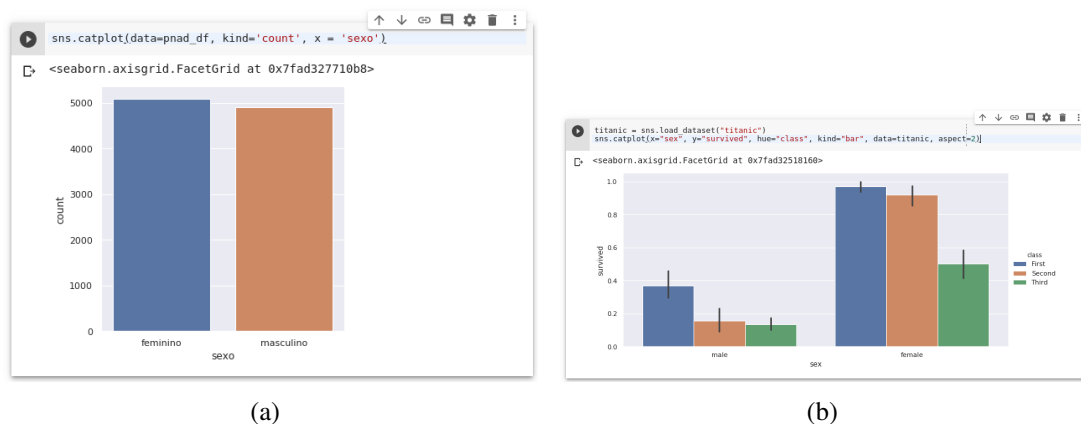


Figura 8.15. Gráficos de barra.

Para “plotar” gráficos de linha, o *seaborn* disponibiliza a função `relplot()`. Essa função fornece acesso a várias funções diferentes no nível de eixos que mostram o relacionamento entre duas variáveis com mapeamentos semânticos de subconjuntos. O parâmetro `kind` seleciona a função de nível de eixo subjacente a ser usada [Haslwanter 2016].

A Figura 8.16 apresenta um exemplo de um gráfico de linha a partir da função `relplot()` do *seaborn*.



### 8.6.3. Gráficos de setores

Este gráfico é constituído com base em um círculo, e é empregado sempre que desejamos ressaltar a participação do dado em relação ao total. O total é representado pelo círculo, que fica dividido em tantos setores quantas são as partes. Os setores são tais que suas áreas são proporcionais aos dados. Cada setor é obtido por meio de uma regra de três simples e direta, lembrando que o total corresponde a  $360^\circ$  [Jelihovschi 2014, Stevenson and De Farias 1981].

Existem algumas recomendações devem ser seguidas ao se elaborar gráficos de setores. Os valores devem ser apresentados em ordem decrescente a partir da parte superior do gráfico e no sentido horário. Ao lado de cada setor, podem-se colocar os percentuais e os nomes de cada parcela. Não é recomendado a utilização deste gráfico usado quando há muitas parcelas e nem quando existem muitas parcelas com valores muito semelhantes, sob pena de se perder uma das suas principais funções: a da comparação [Haslwanter 2016, Stevenson and De Farias 1981].

Para “plotar” gráficos de setores é necessário a utilização da função `matplotlib.pyplot.pie` do `matplotlib` [Haslwanter 2016]. As Figuras 8.17 e 8.18 apresentam exemplos de gráficos de setores (pizza) a partir da função `pie()` do `matplotlib`.

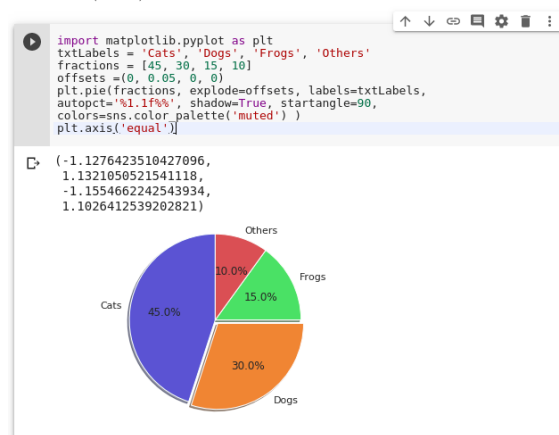


Figura 8.17. Gráfico de setores (pizza).

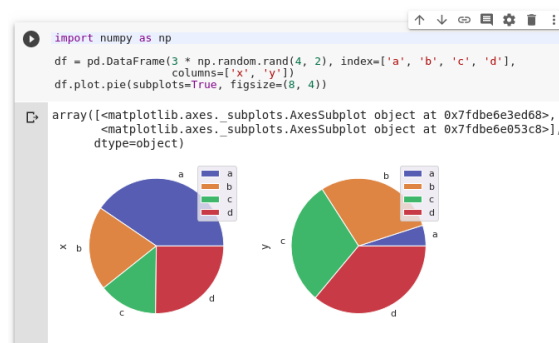


Figura 8.18. Gráfico de setores (pizza).

#### 8.6.4. Boxplots

Um *boxplot* (diagrama de caixa) é um gráfico apresentado em formato de caixa, em que a aresta inferior da caixa representa o primeiro quartil ( $Q_1$ ), a aresta superior representa o terceiro quartil ( $Q_3$ ) e um traço interno à caixa representa a mediana ( $Q_2$ ) de uma amostra.

Vale ressaltar que o *boxplot* não é paramétrico, ou seja, apresenta a variação das amostras de uma população estatística sem fazer qualquer suposição da distribuição estatística subjacente [Haslwanter 2016, Stevenson and De Farias 1981, Spiegel and Stephens 2000].

Para “plotar” *boxplot*, o *seaborn* disponibiliza a função *boxplot()*. Os dados de entrada para a função *boxplot()* podem ser transmitidos em vários formatos, incluindo:

- Vetores de dados representados como listas, matrizes *numpy* ou objetos da série *pandas* passados diretamente para os parâmetros *x*, *y* e/ou *matizes*;
- Um *DataFrame* de “formato longo”; nesse caso, as variáveis *x*, *y* e *matizes* determinarão como os dados são plotados;
- Um *DataFrame* de formato amplo, de modo que cada coluna numérica seja plotada; ou
- Uma matriz ou lista de vetores.

Na maioria dos casos, é possível usar objetos *numpy* ou *Python*, no entanto, objetos do *pandas* são preferíveis pois os nomes associados serão usados para definir os eixos. Além disso, é possível usar tipos categóricos para as variáveis de agrupamento para controlar a ordem dos elementos da plotagem.

As Figuras 8.19(a) e 8.19(b) apresentam exemplos de *boxplots* a partir da função *boxplot()* do *seaborn*.

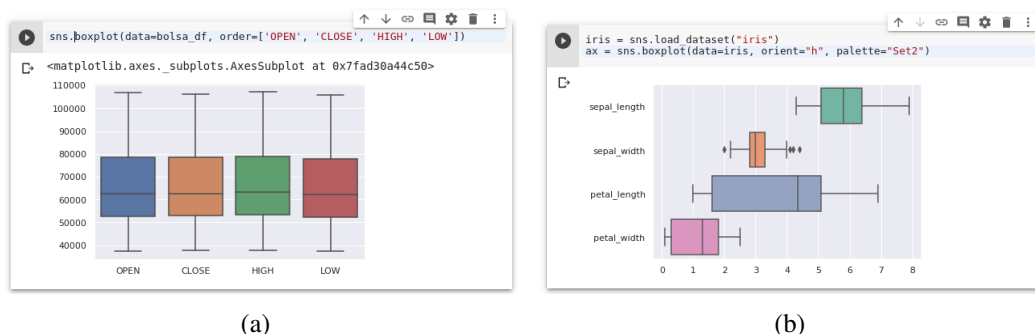


Figura 8.19. *Box plot*. (a) orientação vertical e (b) orientação horizontal.

#### 8.6.5. Gráfico de dispersão

Gráficos de Dispersão são utilizados para pontuar dados em um eixo vertical e horizontal com a intenção de exibir quanto uma variável é afetada por outra.

Para “plotar” gráficos de dispersão, o *seaborn* disponibiliza a função *catplot()*, representação padrão dos dados na função *catplot()*. O *seaborn* disponibiliza dois gráficos de dispersão diferentes e eles adotam abordagens diferentes para resolver o principal desafio na representação de dados categóricos através de gráficos de dispersão, ou seja, todos os pontos pertencentes a uma categoria caem na mesma posição ao longo do eixo correspondente à variável categórica [Haslwanter 2016]

A Figura 8.20 apresenta um exemplo de um gráfico de dispersão a partir da função *catplot()* do *seaborn*.

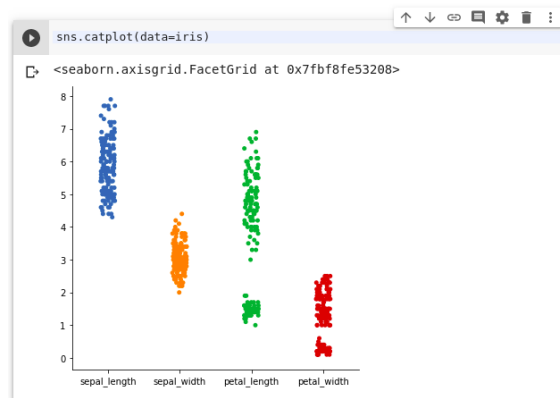


Figura 8.20. Gráfico de dispersão.

A Figura 8.21 apresenta uma abordagem que ajusta os pontos ao longo do eixo categórico usando um algoritmo que os impede de se sobreporem. Essa forma pode dar uma melhor representação da distribuição das observações, embora funcione bem apenas para conjuntos de dados relativamente pequenos. Às vezes, esse tipo de gráfico é chamado de “*beeswarm*” e é desenhado pelo *seaborn* através da função *swarmplot()*, que é ativado pela configuração de *kind = "swarm"* do *catplot()*.

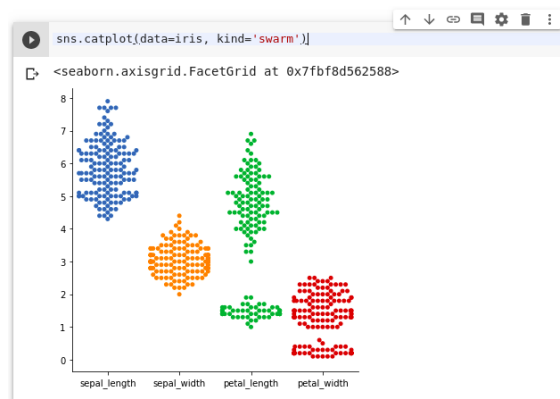


Figura 8.21. Gráfico de dispersão *beeswarm*.

## 8.7. Correlação

O termo correlação representa, sob o ponto de vista da estatística, uma medida de associação entre duas ou mais variáveis. Por definição, se forem considerados numa população,

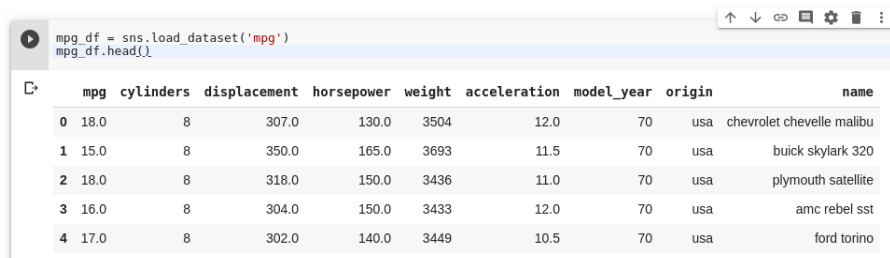
os pares de valores de duas variáveis  $(x_i, y_i)$ , a correlação pode ser definida pela Equação 7 [Spiegel and Stephens 2000, Stevenson and De Farias 1981].

$$\rho = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\left[ \sum_{i=1}^n (x_i - \bar{x})^2 \right] \left[ \sum_{i=1}^n (y_i - \bar{y})^2 \right]}} \quad (7)$$

O valor da correção, conhecido como coeficiente de correlação, assume valores no intervalo de  $-1$  a  $1$ , de acordo com o grau de associação entre as variáveis em questão. Este grau de associação pode ser interpretado da seguinte forma:

- 0,9 a 1 positivo ou negativo indica uma correlação muito forte.
- 0,7 a 0,9 positivo ou negativo indica uma correlação forte.
- 0,5 a 0,7 positivo ou negativo indica uma correlação moderada.
- 0,3 a 0,5 positivo ou negativo indica uma correlação fraca.
- 0 a 0,3 positivo ou negativo indica uma correlação desprezível.

O *pandas* disponibiliza a função *corr()* para calcular a correlação entre duas colunas. Para exemplificar o uso da função *corr()* para calcular a correlação entre duas colunas, será utilizado o conjunto de dados “Auto-Mpg Data” (mpg), conforme pode ser observado na Figura 8.22.



```
mpg_df = sns.load_dataset('mpg')
mpg_df.head()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	name
0	18.0	8	307.0	130.0	3504	12.0	70	usa	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693	11.5	70	usa	buick skylark 320
2	18.0	8	318.0	150.0	3436	11.0	70	usa	plymouth satellite
3	16.0	8	304.0	150.0	3433	12.0	70	usa	amc rebel sst
4	17.0	8	302.0	140.0	3449	10.5	70	usa	ford torino

**Figura 8.22. Obtenção e cinco primeiros valores do dataset Auto-Mpg Data.**

A ideia é verificar qual característica do veículo está mais associada ao seu consumo de combustível (mpg: *miles per gallon* - milhas(1,6 km) por galão( 3,78 litros)). A Figura 8.23 é possível verificar a correlação entre o peso do veículo e seu desempenho mpg do dataset mpg.

É possível também verificar todas as correlações de forma simultânea, como observado na Figura 8.24.



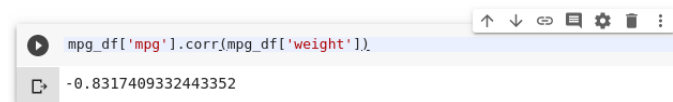


Figura 8.23. Correlação entre o peso do veículo e seu desempenho mpg do *dataset* mpg.

```
mpg_df.corr()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year
mpg	1.000000	-0.775396	-0.804203	-0.778427	-0.831741	0.420289	0.579267
cylinders	-0.775396	1.000000	0.950721	0.842983	0.896017	-0.505419	-0.348746
displacement	-0.804203	0.950721	1.000000	0.897257	0.932824	-0.543684	-0.370164
horsepower	-0.778427	0.842983	0.897257	1.000000	0.864538	-0.689196	-0.416361
weight	-0.831741	0.896017	0.932824	0.864538	1.000000	-0.417457	-0.306564
acceleration	0.420289	-0.505419	-0.543684	-0.689196	-0.417457	1.000000	0.288137
model_year	0.579267	-0.348746	-0.370164	-0.416361	-0.306564	0.288137	1.000000

Figura 8.24. Correlação entre todas as variáveis do *dataset* mpg.

## 8.8. Considerações finais

Neste capítulo foram apresentados conceitos gerais sobre a Análise Exploratória de Dados (AED) através da linguagem de programação Python e a biblioteca Pandas, Matplotlib e Seaborn, demonstrando o quanto é simples e rápida a sua utilização na AED.

Foi apresentado uma breve descrição como se pode calcular medidas de tendência central e de dispersão, além de mostrar a apresentação dos dados estatísticos graficamente. Muitos outros dados e inferências poderiam ter sido abordados e outros gráficos também poderiam ser abordados, porém, como é apenas uma introdução esses outros pontos ficarão para trabalhos futuros.

## Referências

- [Boschetti and Massaron 2015] Boschetti, A. and Massaron, L. (2015). *Python data science essentials*. Packt Publishing Ltd.
- [Coelho 2017] Coelho, A. S. (2017). Introdução a análise de dados com python e pandas. *Anais Eletrônicos ENUCOMP*, pages 862–876.
- [Haslwanter 2016] Haslwanter, T. (2016). *An Introduction to Statistics with Python*. Springer.
- [Jeliahovski 2014] Jeliahovski, E. (2014). *Análise Exploratória de Dados usando o R*. Editus.
- [Jones et al. 2001] Jones, E., Oliphant, T., Peterson, P., et al. (2001). *Scipy: Open source scientific tools for python*.
- [Spiegel and Stephens 2000] Spiegel, M. R. and Stephens, L. J. (2000). *Estatística: Coleção Schaum*. Bookman.

- [Stevenson and De Farias 1981] Stevenson, W. J. and De Farias, A. A. (1981). *Estatística aplicada à administração*.
- [Tukey 1977] Tukey, J. W. (1977). *Exploratory Data Analysis*. Addison-Wesley, 1st edition.
- [Vigni et al. 2013] Vigni, M. L., Durante, C., and Cocchi, M. (2013). Chapter 3 - exploratory data analysis. In Marini, F., editor, *Chemometrics in Food Chemistry*, volume 28 of *Data Handling in Science and Technology*, pages 55 – 126. Elsevier.