

# Python – Pandas – Parte 2

---

## 1. Introdução

Nesta aula utilizaremos algumas coisas novas, a saber:

### 1.1. SciPy

Um ecossistema de ferramentas de computação científica para Python, com muitas rotinas numéricas. Mais detalhes em: <https://www.scipy.org/>. Em nossos código importaremos o SciPy da seguinte forma:

```
import scipy as sp
```

### 1.2. Operações de expressão regular

Uma expressão regular (ou RE) especifica um conjunto de strings que corresponde a ela; as funções neste módulo permitem verificar se uma determinada string corresponde a uma determinada expressão regular (ou se uma determinada expressão regular corresponde a uma determinada string, que se resume à mesma coisa). Mais detalhes em: <https://docs.python.org/3/library/re.html> . Em nossos código importaremos o Re da seguinte forma:

```
import re
```

### 1.3. display

É uma API pública para ferramentas de exibição em IPython. Em nossos código importaremos o display da seguinte forma:

```
from IPython.display import display
```

## 2. Indexação e seleção de dados

Dataframe é uma estrutura de dados bidimensional, ou seja, os dados são alinhados de forma tabular em linhas e colunas, que pode ser criado usando várias entradas:

- Listas;
- Dicionários;
- Séries;
- NumPy ndarrays;
- outro DataFrame.

Indexar no pandas significa simplesmente selecionar linhas e colunas específicas de dados de um DataFrame:

- selecionar todas as linhas e algumas das colunas;
- selecionar algumas linhas e todas as colunas;
- selecionar algumas das linhas e colunas;
- selecionar subconjuntos.

```
import pandas as pd

games = pd.read_csv(
    "vgsales.csv",
    sep=',',
    header=0,
    encoding='latin1',
    names=["Rank", "NomeJogo", "Plataforma", "ano", "Gênero",
           "Fabricante", "Vendas_EUA", "Vendas_Europa",
           "Vendas_Japão", "Vendas_Resto_Mundo", "Total_Vendas"]
)
print(games['Total_Vendas'].head())#primeira forma de fazer
print("\n")
print(games.Total_Vendas.head())#segunda forma de fazer

0    82.74
1    40.24
2    35.82
3    33.00
4    31.37
Name: Total_Vendas, dtype: float64

0    82.74
1    40.24
2    35.82
3    33.00
4    31.37
Name: Total_Vendas, dtype: float64
```

## 2.1. Selecionando mais de uma coluna

Também podemos selecionar mais de uma coluna para análise.

```
games = pd.read_csv(
    "vgsales.csv",
    sep=',',
    header=0,
    encoding='latin1',
    names=["Rank", "NomeJogo", "Plataforma", "ano", "Gênero",
           "Fabricante", "Vendas_EUA", "Vendas_Europa",
           "Vendas_Japão", "Vendas_Resto_Mundo", "Total_Vendas"]
)
print(games[['Total_Vendas', 'Vendas_EUA', 'Fabricante']].head())
```

	Total_Vendas	Vendas_EUA	Fabricante
0	82.74	41.49	Nintendo
1	40.24	29.08	Nintendo
2	35.82	15.85	Nintendo
3	33.00	15.75	Nintendo
4	31.37	11.27	Nintendo

## 2.2. Selecionando trechos de linhas

Você pode selecionar trechos de linhas específicos.

```
games = pd.read_csv(
    "vgsales.csv",
    sep=',',
    header=0,
    encoding='latin1',
    names=["Rank", "NomeJogo", "Plataforma", "ano", "Gênero",
           "Fabricante", "Vendas_EUA", "Vendas_Europa",
           "Vendas_Japão", "Vendas_Resto_Mundo", "Total_Vendas"]
)
print(games[10:15])
```

	Rank	NomeJogo	Plataforma	ano	Gênero	\
10	11	Nintendogs	DS	2005.0	Simulation	
11	12	Mario Kart DS	DS	2005.0	Racing	
12	13	Pokemon Gold/Pokemon Silver	GB	1999.0	Role-Playing	
13	14	Wii Fit	Wii	2007.0	Sports	
14	15	Wii Fit Plus	Wii	2009.0	Sports	

  

	Fabricante	Vendas_EUA	Vendas_Europa	Vendas_Japão	Vendas_Resto_Mundo	\
10	Nintendo	9.07	11.00	1.93	2.75	
11	Nintendo	9.81	7.57	4.13	1.92	
12	Nintendo	9.00	6.18	7.20	0.71	
13	Nintendo	8.94	8.03	3.60	2.15	
14	Nintendo	9.09	8.59	2.53	1.79	

  

	Total_Vendas
10	24.76
11	23.42
12	23.10
13	22.72
14	22.00

## 2.3. Usando loc e iloc

A seleção de subconjunto é uma das tarefas mais frequentemente executadas ao manipular dados. O Pandas fornece maneiras diferentes de selecionar eficientemente subconjuntos de dados do seu DataFrame. Para selecionar um número específico de linhas e colunas, você pode usar o `.loc`.

```
games = pd.read_csv(
    "vgsales.csv",
    sep=',',
    header=0,
    encoding='latin1',
    names=["Rank", "NomeJogo", "Plataforma", "ano", "Gênero",
           "Fabricante", "Vendas_EUA", "Vendas_Europa",
           "Vendas_Japão", "Vendas_Resto_Mundo", "Total_Vendas"]
)
print(games.loc[10:15, ['Vendas_EUA', 'ano']])
```

	Vendas_EUA	ano
10	9.07	2005.0
11	9.81	2005.0
12	9.00	1999.0
13	8.94	2007.0
14	9.09	2009.0
15	14.97	2010.0

```
games = pd.read_csv(
    "vgsales.csv",
    sep=',',
    header=0,
    encoding='latin1',
    names=["Rank", "NomeJogo", "Plataforma", "ano", "Gênero",
           "Fabricante", "Vendas_EUA", "Vendas_Europa",
           "Vendas_Japão", "Vendas_Resto_Mundo", "Total_Vendas"]
)
linhas = [10,12,13,14]
colunas = ["Vendas_EUA", "Vendas_Japão"]
print(games.loc[linhas,colunas])
```

	Vendas_EUA	Vendas_Japão
10	9.07	1.93
12	9.00	7.20
13	8.94	3.60
14	9.09	2.53

Para selecionar um número específico de linhas e colunas, você pode fazer o seguinte usando `.iloc`.

```
games = pd.read_csv(
    "vgsales.csv",
    sep=',',
    header=0,
    encoding='latin1',
    names=["Rank", "NomeJogo", "Plataforma", "ano", "Gênero",
           "Fabricante", "Vendas_EUA", "Vendas_Europa",
           "Vendas_Japão", "Vendas_Resto_Mundo", "Total_Vendas"]
)
print(games.iloc[[10,15],[2,5]])
```

	Plataforma	Fabricante
10	DS	Nintendo
15	X360	Microsoft Game Studios

```
import pandas as pd
from IPython.display import display
games = pd.read_csv(
    "vgsales.csv",
    sep=',',
    header=0,
    encoding='latin1',
    names=["Rank", "NomeJogo", "Plataforma", "ano", "Gênero",
           "Fabricante", "Vendas_EUA", "Vendas_Europa",
           "Vendas_Japão", "Vendas_Resto_Mundo", "Total_Vendas"]
)
display(games.iloc[[10,15],[2,5]])
```

	Plataforma	Fabricante
10	DS	Nintendo
15	X360	Microsoft Game Studios

## 2.4. Usando máscaras booleanas

Podemos selecionar os dados com mascaras booleanas como em NumPy

```
import pandas as pd
from IPython.display import display
games = pd.read_csv(
    "vgsales.csv",
    sep=',',
    header=0,
    encoding='latin1',
    names=["Rank", "NomeJogo", "Plataforma", "ano", "Gênero",
           "Fabricante", "Vendas_EUA", "Vendas_Europa",
           "Vendas_Japão", "Vendas_Resto_Mundo", "Total_Vendas"]
)
amostra = games[(games.ano > 2009) & (games.Total_Vendas > 20)]
display(amostra.mean())
```

Rank	16.500
ano	2011.500
Vendas_EUA	10.990
Vendas_Europa	7.105
Vendas_Japão	0.605
Vendas_Resto_Mundo	2.905
Total_Vendas	21.610
dtype:	float64

## 3. Introdução a estatísticas

Como python você pode manipular dados, no melhor sentido da palavra, e calcular os resultados de várias operações estatísticas usando o arquivo "statistics". Principais funções:

- `mean()`: esta função retorna média dos dados.

- `mode()`: retorna o número com o número máximo de ocorrências.
- `mediana()`: retorna a mediana, ou seja, o elemento do meio dos dados.
- `median_low()`: retorna a mediana dos dados em caso de número ímpar de elementos, mas em caso de número par de elementos, retorna o menor dos dois elementos do meio.
- `median_high()`: retorna a mediana dos dados no caso de número ímpar de elementos, mas no caso de número par de elementos, retorna o maior dos dois elementos.

```
#usando funções estatísticas básicas
import statistics
dados = [3,3,3,5,5,7,9,9,11,11]
print(f"Média = {statistics.mean(dados)}")
print(f"Moda = {statistics.mode(dados)}")
print(f"Mediana = {statistics.median(dados)}")
print(f"Mediana Acima= {statistics.median_high(dados)}")

Média = 6.6
Moda = 3
Mediana = 6.0
Mediana Acima= 7
```

## 4. Usando concat, merge e joins

Um quadro de dados é uma estrutura de dados bidimensional com várias linhas e colunas. Em um quadro de dados, os dados são alinhados apenas na forma de linhas e colunas. Um quadro de dados pode executar operações aritméticas e condicionais. Tem tamanho mutável.

### 4.1.concat()

Realiza a concatenação de DataFrames.

```
from IPython.display import display
dataframe1 = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
                           'B': ['B0', 'B1', 'B2', 'B3'],
                           'C': ['C0', 'C1', 'C2', 'C3'],
                           'D': ['D0', 'D1', 'D2', 'D3']},
                           index=[0, 1, 2, 3])

dataframe2 = pd.DataFrame({'A': ['A4', 'A5', 'A6', 'A7'],
                           'B': ['B4', 'B5', 'B6', 'B7'],
                           'C': ['C4', 'C5', 'C6', 'C7'],
                           'D': ['D4', 'D5', 'D6', 'D7']},
                           index=[4, 5, 6, 7])

dataframe3 = pd.DataFrame({'A': ['A8', 'A9', 'A10', 'A11'],
                           'B': ['B8', 'B9', 'B10', 'B11'],
                           'C': ['C8', 'C9', 'C10', 'C11'],
                           'D': ['D8', 'D9', 'D10', 'D11']},
                           index=[8, 9, 10, 11])

frames = [dataframe1, dataframe2, dataframe3]

display(pd.concat(frames))
```

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

### 4.2.merge()

Pandas fornece uma única função, `merge()`, como o ponto de entrada para todas as operações de junção de banco de dados padrão entre objetos `DataFrame`, que pode utilizar as seguintes propriedades:

- `left`: um `DataFrame` nomeado.
- `right`: outro `DataFrame` nomeado.
- `on`: Nomes de nível de coluna ou índice nos quais participar.
- `left_on`: colunas ou níveis de índice do `DataFrame` esquerdo para usar como chaves.
- `right_on`: colunas ou níveis de índice do `DataFrame` direita para usar como chaves.

- `left_index`: se `True`, use o índice (rótulos de linha) do `DataFrame` esquerdo como sua(s) chave(s) de junção.
- `right_index`: o mesmo uso `left_index` do `DataFrame` direito.
- `how`: Um dos 'left', 'right', 'outer', 'inner'. O padrão é `inner`. Veja abaixo uma descrição mais detalhada de cada método.
- `sort`: classifica o `DataFrame` do resultado pelas chaves de junção em ordem lexicográfica.
- `suffixes`: uma tupla de sufixos de sequência a ser aplicada a colunas sobrepostas.
- `indicator`: adiciona uma coluna ao `DataFrame` de saída chamado `_merge` com informações na origem de cada linha.

```
from IPython.display import display
esquerda = pd.DataFrame({'chave': ['K0', 'K1', 'K2', 'K3'],
                        'A': ['A0', 'A1', 'A2', 'A3'],
                        'B': ['B0', 'B1', 'B2', 'B3']})

direita = pd.DataFrame({'chave': ['K0', 'K1', 'K2', 'K3'],
                        'C': ['C0', 'C1', 'C2', 'C3'],
                        'D': ['D0', 'D1', 'D2', 'D3']})

pd.merge(esquerda, direita, how='inner', on='chave')
```

	chave	A	B	C	D
0	K0	A0	B0	C0	D0
1	K1	A1	B1	C1	D1
2	K2	A2	B2	C2	D2
3	K3	A3	B3	C3	D3

### 4.3. Join

O `join()` usa `merge` internamente para a junção de índice em índice (por padrão) e coluna(s) em índice.

```
from IPython.display import display
esquerda = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
                        'B': ['B0', 'B1', 'B2', 'B3']},
                        index= ['K0', 'K1', 'K2', 'K3'])

direita = pd.DataFrame({'C': ['C0', 'C1', 'C2', 'C3'],
                        'D': ['D0', 'D1', 'D2', 'D3']},
                        index= ['K0', 'K1', 'K2', 'K3'])

display(left.join(direita))
display(right.join(esquerda))
```

	A	B	C	D
K0	A0	B0	C0	D0
K1	A1	B1	C1	D1
K2	A2	B2	C2	D2
K3	A3	B3	C3	D3

  

	C	D	A	B
K0	C0	D0	A0	B0
K1	C1	D1	A1	B1
K2	C2	D2	A2	B2
K3	C3	D3	A3	B3

## 5. Exercícios – base de dados personagens.csv

- Mostre os 5 primeiros registros da base de dados
- Repita o item anterior, mas mostrando apenas nome e poder
- Mostre os 5 primeiros registros da base de dados, que tem poder maior que 90
- Mostre a média do poder

- Mostre a mediana de combate
-