

# Pandas: DataFrame

*DataFrames*: estrutura bidimensional indexada que armazena valores de qualquer tipo.

`pd.DataFrame(valores)` - cria um DataFrame

`pd.DataFrame(valores, index=array1d, columns=array1d)` - cria um DataFrame

Índices: `df.index`

Colunas: `df.columns`

Valores: `df.values`

Dims: `df.shape`

`df.T` - DataFrame Transposto

`pd.read_excel(caminho, index_col=n, header=n, decimal=',')`

**caminho** - localização do arquivo: composto pelo caminho (absoluto/relativo) e nome

**index\_col = n** - O número da coluna do arquivo a ser usada como labels das linhas (índice). O padrão é **None** (o arquivo não possui tal coluna)

**header = n** - O número da linha para os *labels* das colunas (padrão é 0) ou **None** quando não há tal linha

`df.info()` - obter informações

`df.rename_axis(mapper, axis=0, inplace=False)`

`df.index.name = nome` ou `df.columns.name = nome`

Altera o nome do *index* ou *columns*. Retorna um DataFrame com o nome da lista de colunas/linhas alterado

`df.rename(mapper=None, axis= n/str, index=None, columns=None, inplace=False)`

Altera o nome do *index* ou *columns*. Retorna um DataFrame com o nome da lista de colunas/linhas alterado

`df.plot(kind='line', xlim=(li,ls), ylim=(li,ls), grid=False, title=None, subplot=False, figsize=(x,y),... )`

**kind=**      'line' : linha (default)              'bar' : barra vertical              'barh' : barra horizontal  
              'hist' : histograma              'box' : boxplot              'scatter' : dispersão              'pie' : pizza  
              entre outros

**[x|y]lim** = limites do eixo x ou do eixo y

**grid** = True/False, mostrar grade de linhas

**title** = título do gráfico

**subplots** = True/False, criar gráficos separados para cada coluna

**figsize** = (altura,largura) em polegadas

`df[coluna]` ou `df.coluna`

`df[lista de colunas]`

Retorna uma *Series* com os valores da *coluna* ou um *DataFrame* com os elementos da *lista de colunas*

`df.loc[índice]` ou `df.loc[lista de índices]`

`df.iloc[posição]` ou `df.iloc[lista de posições]`

Retorna uma *Serie* com os valores da linha indexada por *índice* ou um *DataFrame* com os elementos da *lista de índices*. **.loc** para os índices criados, **.iloc** para a posição no índice

`df.loc[índice][coluna]` ou `df.loc[lista de índices][lista de colunas]*`

`df[coluna].loc[índice]` ou `df[lista de colunas].loc[lista de índices]*`

Retorna o valor do elemento indexado por *índice*, *coluna* ou uma nova *Series* com os elementos da *lista de índices/colunas*. \* Para posição no índice deve ser usado **.iloc**

**`df.loc[índice]= valor ou Series`**

Altera o valor/valores do elemento(s) indexado(s) por *índice/lista de índices*. Se o índice não existe, é incluído.

**`df.drop(índice) ou df.drop(lista de índices) *`**  
**`df.dropna() *`**

drop: retorna uma cópia do DataFrame sem a linha/linhas especificadas

dropna: retorna uma cópia do DataFrame sem as linhas que tenham colunas com NaN. Com parâmetro `how='all'`, só remove as linhas em que todas as colunas são NaN

\* com `inplace=True`, realiza a operação na Series, não cria uma cópia

**`df[coluna]= valor ou Series`**

Altera o valor/valores do(s) elemento(s) indexado(s) por *coluna/lista de colunas*. Se a coluna não existe, é incluída.

**`df.drop(coluna, axis=1) ou df.drop(lista de colunas, axis=1) *`**  
**`df.dropna(axis=1) *`**

\* com `inplace=True`, realiza a operação na Series, não cria uma cópia

drop: retorna uma cópia do DataFrame sem a coluna/lcolunas especificadas

dropna: retorna uma cópia do DataFrame sem as colunas que tenham linhas com NaN.

Com parâmetro `how='all'`, só remove as linhas em que todas as colunas são NaN

**`df.sort_values(by, axis=0, ascending=True, na_position='last') *`**

Retorna uma cópia do DataFrame ordenado pelos valores

**by** = nomes ou lista de nomes para ordenar. Se `axis=0` os nomes se referem a colunas. Se `axis=1` os nomes se referem a linhas

**na\_position** = 'last' / 'first', coloca NaN no final/ início

\* com `inplace=True`, realiza a operação na Series, não cria uma cópia

**`df.sort_index(axis=0, level=None, ascending=True, na_position='last', sort_remaining=True, by=None) *`**

Retorna uma cópia do DataFrame ordenado pelos labels do índice

\* com `inplace=True`, realiza a operação no DataFrame, não cria uma cópia

**`df.fillna(value=None, axis=None, ascending=True) *`**

Retorna uma cópia do DataFrame substituindo valores Nan

**value** = scalar, dict, Series, or DataFrame

alternately a dict/Series/DataFrame of values specifying which value to use for each index (for a Series) or column (for a DataFrame). (values not in the dict/Series/DataFrame will not be filled).

**axis** = 0 ou 1

\* com `inplace=True`, realiza a operação no DataFrame, não cria uma cópia

**`pd.concat([df1,...dfn], axis=n) -`** n é o eixo (0 ou 1) para a concatenação

## Sumarizações

Média: `df.mean()` [1]

Mediana: `df.median()` [1]

Moda: `df.mode()` [1]

Máximo: `df.max()` [1] [2]

Mínimo: `df.min()` [1] [2]

Índice 1º Mínimo: `df.idxmin()`

Índice 1º Máximo: `df.idxmax()`

Quantil: `df.quantile(q=%)` [1] padrão `q=0.5`

Variância: `df.var()` [1]

Desvio Padrão: `df.std()` [1]

Covariância: `df.cov(series)`

Correlação: `df.corr(series)`

Soma: `df.sum()` [1]

Quantidade: `df.count()` [1]

Contagem de valores exclusivos: `df.index.value_counts()` e `df.columns.value_counts()` (Tabela de frequências)

Resumo: `df.describe()`

[1] Com `axis = 1`, operação por linha

[2] Operação aceita nos atributos `index` e `columns`

### **df.operador\_lógico condição**

Retorna um novo *DataFrame/Series* com valores booleanos True/False.

Pode-se usar o *DataFrame/Series* de booleanos para filtrar os itens selecionados (com valor True).

Normalmente utilizado sobre uma coluna ou linha do *DataFrame*

### **df.isin(lista de valores)**

### **df.índice.isin(lista de índices)**

Retorna um *DataFrame* com True onde o elemento do *DataFrame* ∈ lista de valores.

Para o *index* retorna um vetor booleano, considerando os *labels* de linha e para *columns*, considerando os labels das colunas.

### **df.query('expressão')**

Seleciona as linhas com os valores que satisfazem a expressão, retornando um *DataFrame*

➤ O *index* e o *columns* podem ser utilizados na expressão. A expressão utiliza as colunas do DF

### **df[critério] = valor ou lista de valores**

critério: produz um *DataFrame/Series* booleano

Altera o(s) valor(es) do(s) elemento(s) indexado(s) por posições onde o valor é True

### **df.metodoOperação(obj, fill\_value=valor)**

*obj* pode ser um *DataFrame* ou uma *Series* ou um escalar. Os dados são alinhados pelas colunas e pelos índices. Retorna um *DataFrame* com a união das colunas e dos *labels* das linhas. Se o argumento *fill\_value* está presente e não há sobreposição nos índices, utiliza *valor* para o cálculo, senão o valor é NaN

**df.replace(to\_replace=valor, value=novo)** – Cria uma cópia, substituindo todas as ocorrências de valor por novo

**df.update(df)** – Altera atuais valores pelos valores recebidos, alinhando pelo índice

### **pandas.crosstab(index, columns, margins=False, margins\_name='All', dropna=True)**

Relaciona duas ou mais sequência de valores. Retorna um *DataFrame* com a tabela de frequência dos valores das sequências, a menos que uma função de agregação e um array de valores sejam especificados.

**index, columns** - array, Series ou lista de arrays/Series que definem os valores a agrupar nas linhas/colunas

**margins, margins\_name** – quando *margins* = True é adicionado uma linha/coluna com subtotais e *margins\_name* define o nome da linha/coluna que armazenará os totais.

**grupo1 = df.groupby(chave)**

**grupo2 = df.groupby(chave,axis=1)**

	c1	c2	c3	c4
I1				
I2				
I3				
I4				
I5				

	c1	c2	c3	c4
I1				
I3				
I5				

  

	c1	c2	c3	c4
I2				
I4				

	c1	c2	c3	c4
I1				
I2				
I3				
I4				
I5				

  

	c1	c4
I1		
I2		
I3		
I4		
I5		

  

	c2	c3
I1		
I2		
I3		
I4		
I5		

### **GroupBy.groups**

### **GroupBy.indices**

**GroupBy.groups** dict {nome do grupo -> labels dos índices do grupo}

**GroupBy.indices** dict {nome do grupo -> array com a posição dos índices do grupo}

### **GroupBy.get\_group(nome)**

Constrói um *DataFrame* com os elementos do grupo cujo nome foi fornecido

### **GroupBy.size()**

Retorna a quantidade de elementos de cada grupo

### **GroupBy.agg(função pré-definida ou nome de método)**

### **GroupBy.agg(lista de funções/nome de método)**

As funções fornecidas para agregação reduzem a dimensão do objeto fornecido. São aplicadas sobre os valores do grupo e retornam um resultado para o conjunto.

As mais comuns são *mean*, *sum*, *size*, *count*, *std*, *var*, *describe*, *first*, *last*, *nth*, *min*, *max*.