

# pandas

Funcionalidades essenciais

Reindexação e descarte de entradas de eixo

O método *reindex* cria um novo objeto com os dados de acordo com um novo índice. Primeiro vamos criar um objeto Series.

```
import pandas as pd
```

```
dados = pd.Series([100.0, 75.85, 10.18, -5.4], index=['um', 'dois', 'três', 'quatro'])
```

```
dados
```

```
um      100.00  
dois     75.85  
três     10.18  
quatro   -5.40  
dtype: float64
```

Agora vamos aplicar o método *reindex* ao objeto *Series* criado.

```
dados_reindexados = dados.reindex(['quatro', 'três', 'dois', 'um', 'zero'])
```

```
dados_reindexados
```

quatro	-5.40
três	10.18
dois	75.85
um	100.00
zero	NaN
dtype: float64	

Se um índice não estava presente, é exibido o indicador de dados ausentes, como por exemplo o índice 'zero'.

Podemos reindexar também as linhas e/ou colunas de um DataFrame. Vamos criar um objeto DataFrame.

```
frame = pd.DataFrame(np.arange(9).reshape((3,3)), index=['a','b','c'], columns=['Vitória', 'Vila Velha', 'Viana'])
```

frame

	Vitória	Vila Velha	Viana
a	0	1	2
b	3	4	5
c	6	7	8

Para reindexar as linhas, basta passar apenas uma sequência.

```
frame_reindexado = frame.reindex(['d', 'c', 'b', 'a'])
```

frame\_reindexado

	Vitória	Vila Velha	Viana
d	NaN	NaN	NaN
c	6.0	7.0	8.0
b	3.0	4.0	5.0
a	0.0	1.0	2.0

As colunas podem ser reindexadas com a palavra reservada *columns*.

```
idades = ['Viana', 'Vitória', 'Vila Velha', 'Cariacica']
```

```
frame.reindex(columns=idades)
```

	Viana	Vitória	Vila Velha	Cariacica
a	2	0	1	NaN
b	5	3	4	NaN
c	8	6	7	NaN

Podemos utilizar o método `drop` para devolver um objeto com o valor ou os valores indicados apagados de um eixo.

```
obj = pd.Series(np.arange(5), index=['zero', 'um', 'dois', 'três', 'quatro'])
```

obj	
zero	0
um	1
dois	2
três	3
quatro	4
dtype: int32	

```
nov = obj.drop('dois')
```

nov	
zero	0
um	1
três	3
quatro	4
dtype: int32	

Com DataFrame, podemos apagar os valores dos índices de qualquer eixo. Vamos criar um objeto DataFrame.

```
dados = pd.DataFrame(np.arange(9).reshape((3,3)), index=['Viana', 'Vila Velha', 'Vitória'], columns=['um', 'dois', 'três'])
```

dados

	um	dois	três
Viana	0	1	2
Vila Velha	3	4	5
Vitória	6	7	8



Podemos usar o método `drop` passando uma sequência de rótulos para descartar valores das linhas (eixo 0).

```
dados.drop(['Vila Velha', 'Vitória'])
```

	um	dois	três
Viana	0	1	2

Para descartar valores das colunas usamos `axis=1` ou `axis='columns'`.

```
dados.drop('um', axis=1)
```

	dois	três
Viana	1	2
Vila Velha	4	5
Vitória	7	8

```
dados.drop('dois', axis='columns')
```

	um	três
Viana	0	2
Vila Velha	3	5
Vitória	6	8

## Funcionalidades essenciais – Descartando entradas de um eixo

O *drop*, assim como muitas outras funções que modificam o tamanho ou o formato de uma *Series* ou um *DataFrame*, podem alterar um objeto *in-place*, ou seja, alterar o objeto original sem retornar um novo objeto. Muito cuidado, pois os dados serão realmente descartados do objeto.

```
dados.drop('dois', axis='columns', inplace=True)
```

dados

	um	três
Viana	0	2
Vila Velha	3	5
Vitória	6	8

# FIM