pandas

Funcionalidades essenciais

Aplicação de funções e mapeamento

pandas

Funcionalidades essenciais – Aplicação de funções e mapeamento

Podemos utilizar as *ufuncs* do *NumPy*, que são os métodos de *array* para todos os elementos com objetos do *pandas*.

Veja o uso da função abs().

```
In [3]: import pandas as pd
        import numpy as np

In [4]: df = pd.DataFrame(np.random.randn(4, 3), columns=list('bde'), index=['Brasil', 'Paraguai', 'Uruguai', 'Colômbia'])
```

```
In [5]: df
```

Out[5]:

	b	d	e
Brasil	0.100721	0.197778	0.451556
Paraguai	0.335315	0.583326	-0.034726
Uruguai	0.187077	-0.216061	0.330422
Colômbia	-0.169946	1.631868	0.001770

```
In [6]: np.abs(df)
```

Out[6]:

	b	d	e
Brasil	0.100721	0.197778	0.451556
Paraguai	0.335315	0.583326	0.034726
Uruguai	0.187077	0.216061	0.330422
Colômbia	0.169946	1.631868	0.001770

pandas

Funcionalidades essenciais – Aplicação de funções e mapeamento

Podemos aplicar uma função em *arrays* unidimensionais para cada coluna ou linha, usando o método *apply* de *DataFrame*. Neste caso, a função *f*, calcula a diferença entre o máximo e o mínimo de cada coluna de *df*. O resultado é uma Series com as colunas de *df* como seu índice.

```
In [7]: f = lambda x: x.max() - x.min()

In [8]: df.apply(f)

Out[8]: b    0.505261
        d    1.847929
        e    0.486282
        dtype: float64
```

0.335315 - (-0.169946)

0.505261

```
In [5]: df

Out[5]:
```

	b	d	e
Brasil	0.100721	0.197778	0.451556
Paraguai	0.335315	0.583326	-0.034726
Uruguai	0.187077	-0.216061	0.330422
Colômbia	-0.169946	1.631868	0.001770

pandas

Funcionalidades essenciais – Aplicação de funções e mapeamento

Se passarmos *axis='columns'* para *apply*, a função será chamada uma vez por linha.

```
In [9]:  df.apply(f, axis='columns')

Out[9]:  Brasil      0.350835
         Paraguai    0.618052
         Uruguai     0.546483
         Colômbia    1.801814
         dtype: float64
```

0.451556 -
0.100721

0.350835

In [5]: df

Out[5]:

	b	d	e
Brasil	0.100721	0.197778	0.451556
Paraguai	0.335315	0.583326	-0.034726
Uruguai	0.187077	-0.216061	0.330422
Colômbia	-0.169946	1.631868	0.001770

pandas

Funcionalidades essenciais – Aplicação de funções e mapeamento

Vamos agora devolver uma *Series* em vez de um valor escalar usando uma função passada para *apply*.

```
In [10]: def f(x):
             return pd.Series([x.min(), x.max()], index=['min', 'max'])

In [11]: df.apply(f)
```

Out[11]:

	b	d	e
min	-0.169946	-0.216061	-0.034726
max	0.335315	1.631868	0.451556

pandas

Funcionalidades essenciais – Aplicação de funções e mapeamento

Podemos aplicar funções *Python* também para todos os elementos. Vamos usar *applymap()* para retornar uma *string* formatada para cada valor de ponto flutuante em *df*.

```
In [18]: string_formatada = lambda x: f'{x:.2f}'

In [19]: df.applymap(string_formatada)
```

Out[19]:

	b	d	e
Brasil	0.10	0.20	0.45
Paraguai	0.34	0.58	-0.03
Uruguai	0.19	-0.22	0.33
Colômbia	-0.17	1.63	0.00

Funcionalidades essenciais – Aplicação de funções e mapeamento

O motivo para o nome *applymap* está no fato de que o objeto *Series* possui um método *map* para aplicar uma função em todos os elementos.

```
In [20]: df['d'].map(string_formatada)

Out[20]: Brasil       0.20
         Paraguai     0.58
         Uruguai     -0.22
         Colômbia     1.63
         Name: d, dtype: object
```

FIM