Funcionalidades essenciais

Ordenação e Classificação

#### Funcionalidades essenciais – Ordenação

Podemos ordenar nossos conjuntos de dados pelas linhas ou pelas colunas usando o método sort index(), que devolve um novo objeto ordenado.

```
In [2]: import pandas as pd
        import numpy as np
In [3]: obj = pd.Series(range(5), index=['b','c','a','d','e'])
In [4]: obj.sort_index()
Out[4]
        dtype: int64
```

Para odernar de forma decrescente, podemos adicionar o parâmetro ascending informando False.

```
obj.sort index(ascending=False)
In [10]:
Out[10]: e
         dtype: int64
```

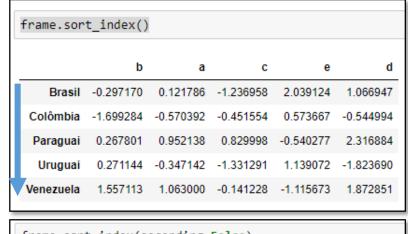


#### Funcionalidades essenciais – Ordenação

Vamos agora trabalhar com ordenação em um

DataFrame.

```
frame = pd.DataFrame(np.random.randn(5,5), columns=list('baced'),
                       index=['Brasil', 'Paraguai', 'Uruguai',
                               'Colômbia', 'Venezuela'])
frame
                 b
          -0.297170
                    0.121786 -1.236958
                                        2.039124
           0.267801
                     0.952138
                              0.829998
                                       -0.540277
                                                 2.316884
           0.271144 -0.347142 -1.331291 1.139072 -1.823690
                    -0.570392 -0.451554
                    1.063000 -0.141228 -1.115673 1.872851
```



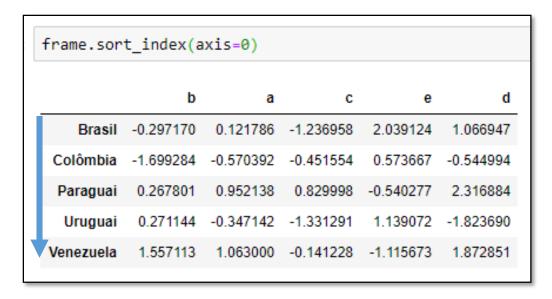
<pre>frame.sort_index(ascending=False)</pre>						
	b	a	С	e	d	
Venezuela	1.557113	1.063000	-0.141228	-1.115673	1.872851	
Uruguai	0.271144	-0.347142	-1.331291	1.139072	-1.823690	
Paraguai	0.267801	0.952138	0.829998	-0.540277	2.316884	
Colômbia	-1.699284	-0.570392	-0.451554	0.573667	-0.544994	
Brasil	-0.297170	0.121786	-1.236958	2.039124	1.066947	

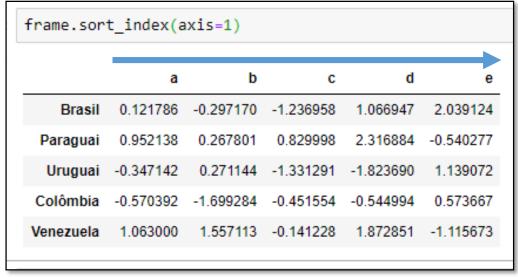




#### Funcionalidades essenciais – Ordenação

### Podemos usar o parâmetro axis para definir o eixo de ordenação.





#### Funcionalidades essenciais – Ordenação

Para ordenar uma Series de acordo com seus valores, utilizamos o método *sort values()*.

```
obj = pd.Series([10, np.nan, 54, 7, -2, np.nan, -4])
obj
    10.0
     NaN
     54.0
     7.0
dtype: float64
```

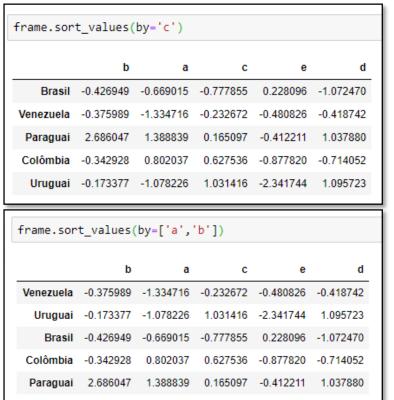
```
obj.sort_values()
     -4.0
     -2.0
     7.0
     10.0
     54.0
      NaN
      NaN
dtype: float64
```

#### Funcionalidades essenciais – Ordenação

Quanto ao DataFrame, podemos ordenar usando os dados de uma ou mais colunas, passando o nome das colunas como parâmetro para a opção by de

sort values. Assim: **sort\_values(by='coluna')** ou sort values(by=['coluna1','coluna2']).

```
frame = pd.DataFrame(np.random.randn(5,5), columns=list('baced'),
                       index=['Brasil', 'Paraguai', 'Uruguai',
                               'Colômbia', 'Venezuela'])
frame
          -0.426949 -0.669015 -0.777855
          -0.173377 -1.078226
                              1.031416 -2.341744
 Venezuela -0.375989 -1.334716 -0.232672 -0.480826 -0.418742
```

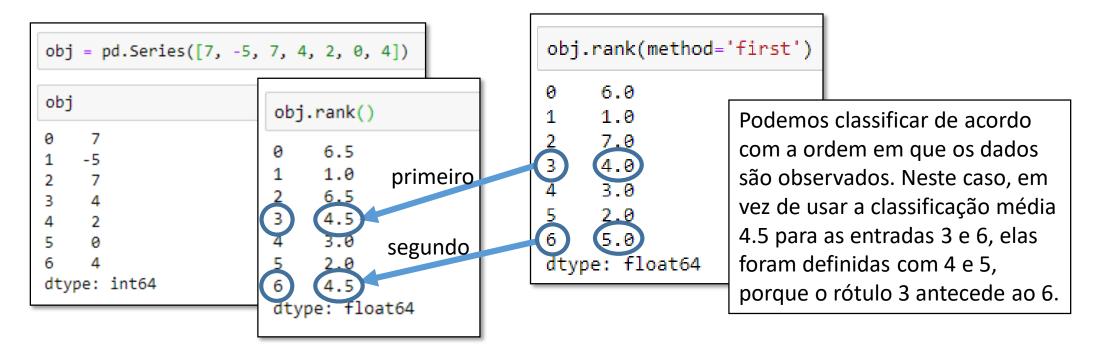






#### Funcionalidades essenciais – Classificação

A classificação (rank) atribui posições iniciando em 1, indo até o número de pontos de dados válidos em um array. Por padrão, rank() resolve empates atribuindo a cada grupo a classificação média.



#### Funcionalidades essenciais – Classificação

Podemos classificar de forma decrescente informando ascending=False.

<pre>obj = pd.Series([0, 1, 2, 3, 4,  obj.rank(ascending=False)  0    7.0 1    6.0</pre>	3, 01)			
2 5.0	Método	Descrição		
4 3.0 5 2.0 6 1.0 dtype: float64	average	Default: Atribui a classificação média para cada entrada no mesmo grupo.		
	min	Utiliza a classificação mínima do grupo todo		
	max	Utiliza a classificação máxima do grupo todo		
	first	Atribui classificações na ordem em que os valores aparecem nos dados		
	dense	Como o 'min', porém as classificações sempre aumentam de 1 entre grupos, em vez do número de elementos iguais em um grupo		

# FIM

