Pandas – Manipulação de Dados

Luiz Celso Gomes-Jr

Agenda

Manipulação e Agregação de Dados

- Manipulação e agregação de dados são funcionalidades essenciais no Pandas.
- O Pandas possui diversos métodos para transformação, agrupamento e agregação dos dados.

Lendo dados de um arquivo

- Usaremos o dataset ‘aluguel.csv’ nos exemplos

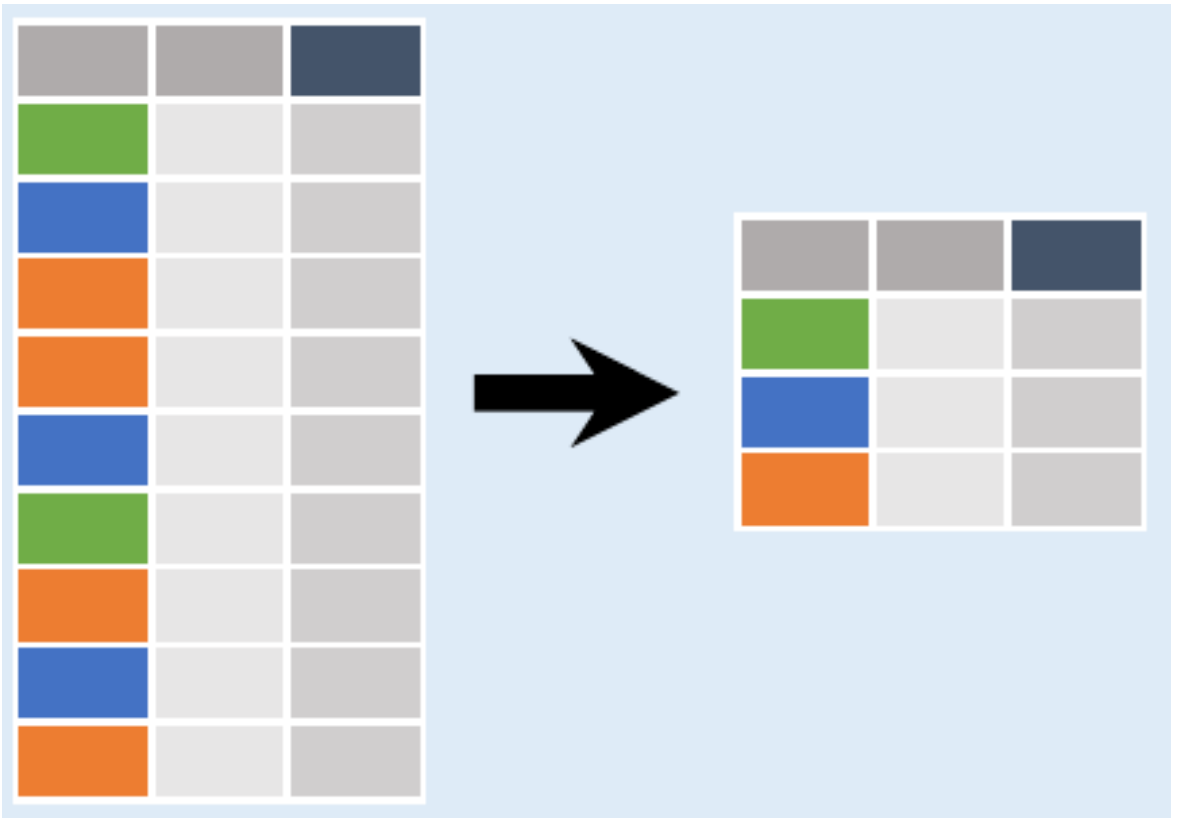
```
# lê o arquivo CSV
df = pd.read_csv('../data/aluguel.csv')

# mostra o conteúdo do DataFrame
df.head()
```

	codigo	endereco	quartos	suite	area	vaga	aluguel	condominio	data
0	34	Rua Desembargador Westphalen	2	0	90	0	900	371	11/10/17
1	167	Rua Jose Loureiro	2	0	64	0	650	428	15/07/17
2	6784	Rua Jose Loureiro	2	0	81	0	1100	400	23/08/17
3	82	Rua Lourenço Pinto	2	0	50	0	1350	300	19/09/17
4	2970	Rua Lourenço Pinto	2	0	63	0	1300	300	05/08/17

Agrupamento

Agrupamentos são usados para se analisar grupos de dados separadamente. O critério de agrupamento é uma lista de colunas do DataFrame.



Agrupamento

Agrupamentos são usados para se analisar grupos de dados separadamente. O critério de agrupamento é uma lista de colunas do DataFrame.

	codigo	endereco	quartos	suite	area	vaga	aluguel	condominio	data
17	66490	Rua Desembargador Westphalen	1	0	80	1	1100	350	29/08/17
11	82343	Avenida Sete de Setembro	1	0	45	0	750	420	11/10/17
16	80	Rua Desembargador Westphalen	1	0	80	1	900	350	12/08/17
4	2970	Rua Lourenço Pinto	2	0	63	0	1300	300	05/08/17
19	44803	Rua Rockefeller	2	0	77	1	950	200	19/07/17

```
df.groupby(['quartos']).mean()
```

quartos	codigo	suite	area	vaga	aluguel	condominio
1	49637.666667	0.0	68.333333	0.666667	916.666667	373.333333
2	23886.500000	0.0	70.000000	0.500000	1125.000000	250.000000

Agrupamento

Podemos agrupar por mais de um campo. E também selecionar as colunas de interesse.

```
df.groupby(['quartos', 'vaga']).mean()[['aluguel']]
```

		aluguel
quartos	**vaga**	
1	**0**	750
	1	1000
2	**0**	1300
	1	950

Agrupamento

Podemos agrupar por mais de um campo. E também selecionar as colunas de interesse.

```
df.groupby(['quartos', 'vaga']).mean()[['aluguel']]
```

		aluguel
quartos	vaga	
1	0	750
	1	1000
2	0	1300
	1	950

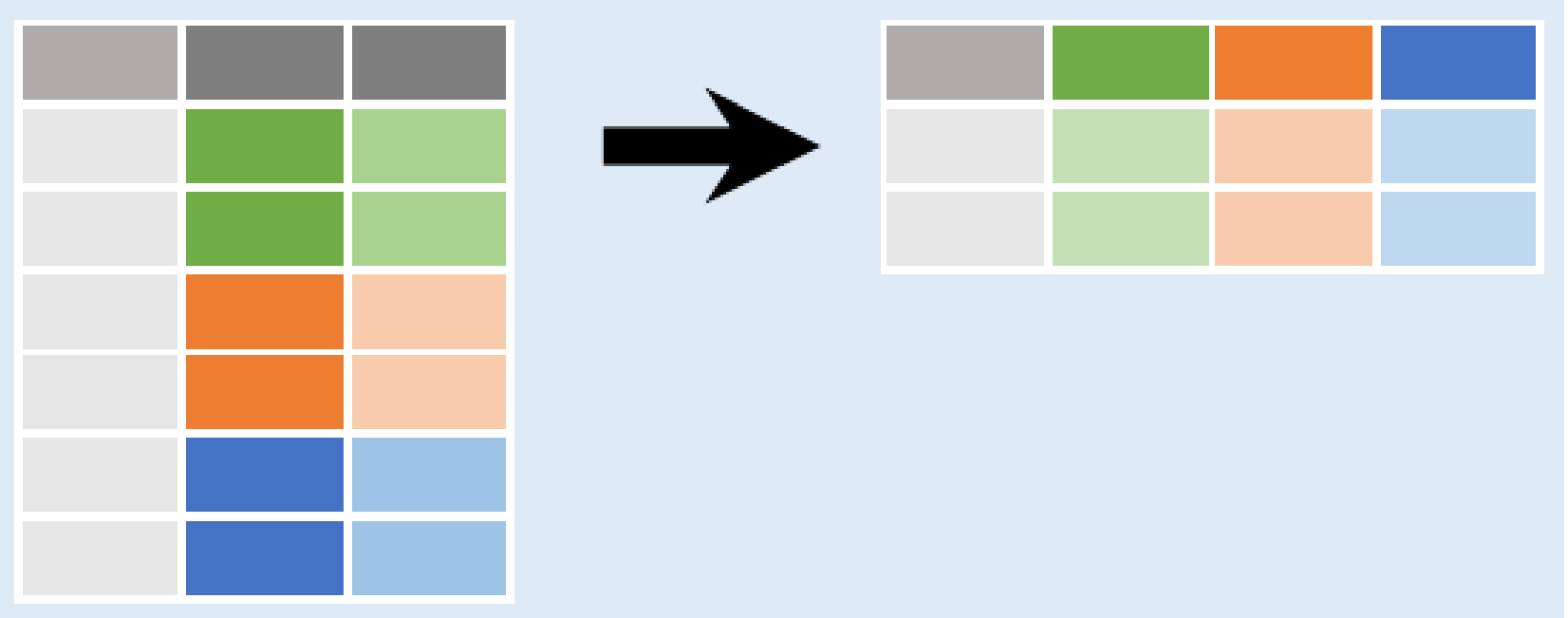
Índice multinível

Agrupamento

Outras funções de agregação podem ser usadas em um agrupamento. Outras funções comuns são size(), sum(), min(), max().

Pivot

Pivoteamento é usado para rearranjar as linhas e colunas.

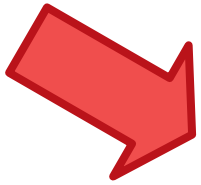


Pivot baseado em índice

Pivoteamento é usado para rearranjar as linhas e colunas. Abaixo fazemos com que o DataFrame seja rearranjado de forma a ter o número de vagas como colunas.

```
df_grouped = df.groupby(['quartos', 'vaga']).mean()[['aluguel']]
df_grouped
```

		aluguel
quartos	**vaga**	
1	**0**	750
	1	1000
2	**0**	1300
	1	950



```
df_grouped.unstack()
```

	aluguel	
vaga	**0**	**1**
quartos		
1	750	1000
2	1300	950

Pivot baseado em colunas

Pivoteamento é usado para rearranjar as linhas e colunas. Abaixo fazemos com que o DataFrame seja rearranjado de forma a ter o número de vagas como colunas.

```
df_grouped = df.groupby(['quartos', 'vaga'])\
               .mean()['aluguel'].reset_index()
df_grouped
```

	quartos	vaga	aluguel
0	1	0	750
1	1	1	1000
2	2	0	1300
3	2	1	950

```
df_grouped.pivot(index='quartos', columns='vaga', values='aluguel')
```

vaga	0	1
quartos		
1	750	1000
2	1300	950

Pivot com agregação

O método **pivot_table** é parecido com o **pivot** mas permite também que se especifique uma função para a agregação de valores.

	codigo	endereco	quartos	suite	area	vaga	aluguel	condominio	data
17	66490	Rua Desembargador Westphalen	1	0	80	1	1100	350	29/08/17
11	82343	Avenida Sete de Setembro	1	0	45	0	750	420	11/10/17
16	80	Rua Desembargador Westphalen	1	0	80	1	900	350	12/08/17
4	2970	Rua Lourenço Pinto	2	0	63	0	1300	300	05/08/17
19	44803	Rua Rockefeller	2	0	77	1	950	200	19/07/17

```
df.pivot_table(index='quartos',
               columns='vaga', values='aluguel', aggfunc=np.mean)
```

vaga	0	1
quartos		
1	750	1000
2	1300	950

Crosstab

O comando **crosstab** relaciona duas colunas para criar um novo DataFrame com valores nas células representando a quantidade de instâncias de cada tipo.

	codigo	endereco	quartos	suite	area	vaga	aluguel	condominio	data
17	66490	Rua Desembargador Westphalen	1	0	80	1	1100	350	29/08/17
11	82343	Avenida Sete de Setembro	1	0	45	0	750	420	11/10/17
16	80	Rua Desembargador Westphalen	1	0	80	1	900	350	12/08/17
4	2970	Rua Lourenço Pinto	2	0	63	0	1300	300	05/08/17
19	44803	Rua Rockefeller	2	0	77	1	950	200	19/07/17

```
pd.crosstab(df['quartos'], df['vaga'])
```

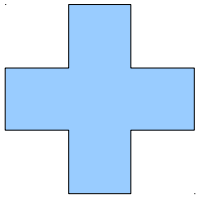
vaga	0	1
quartos		
1	1	2
2	1	1

Junção

Quando temos dois DataFrames e precisamos uni-los, usamos as funcionalidades de junção de dados. Existem vários métodos para fazer junção, dependendo do resultado esperado.

df:

	nome	idade	genero
0	Celso	23	M
1	Josie	32	F
2	Luiz	11	M
3	Helen	18	F



df_s:

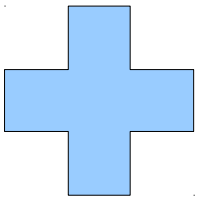
	nome	salario
0	Celso	1230.0
1	Luiz	1100.0
2	Helen	1800.0

Merge

O método mais usado para junção de duas colunas é o **merge**. No exemplo abaixo a aplicação do método retorna um novo DataFrame com as colunas dos dois DataFrames originais unidas e com as linhas associadas de acordo com a coluna em comum (nome).

df:

	nome	idade	genero
0	Celso	23	M
1	Josie	32	F
2	Luiz	11	M
3	Helen	18	F



df_s:

	nome	salario
0	Celso	1230.0
1	Luiz	1100.0
2	Helen	1800.0

```
df_joined = pd.merge(df, df_s)
df_joined
```

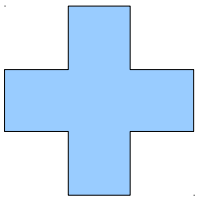
	nome	idade	genero	salario
0	Celso	23	M	1230.0
1	Luiz	11	M	1100.0
2	Helen	18	F	1800.0

Merge - Left/Right/Outer

O **merge** combina linhas com valores em comum e descarta as sem casamento. Para preservar todos os valores, independente do casamento, especifique o tipo Left/Right/Outer.

df:

	nome	idade	genero
0	Celso	23	M
1	Josie	32	F
2	Luiz	11	M
3	Helen	18	F



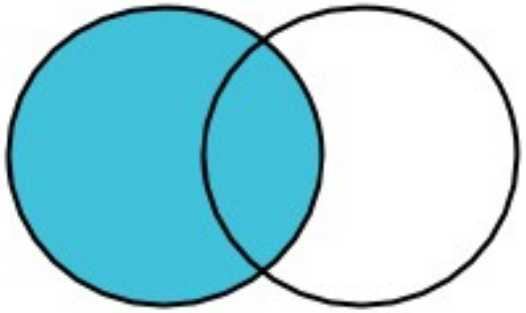
df_s:

	nome	salario
0	Celso	1230.0
1	Luiz	1100.0
2	Helen	1800.0

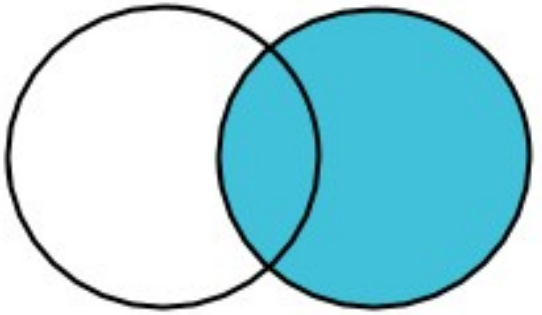
```
df_joined = pd.merge(df, df_s, how='left')
df_joined
```

	nome	idade	genero	salario
0	Celso	23	M	1230.0
1	Josie	32	F	NaN
2	Luiz	11	M	1100.0
3	Helen	18	F	1800.0

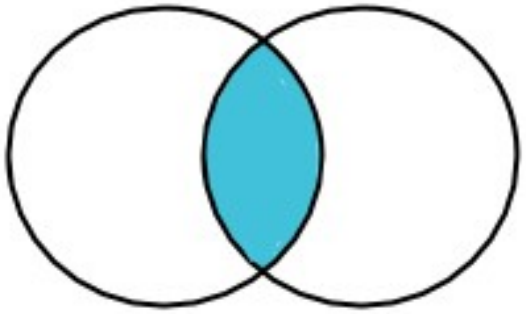
Merge - Left/Right/Outer



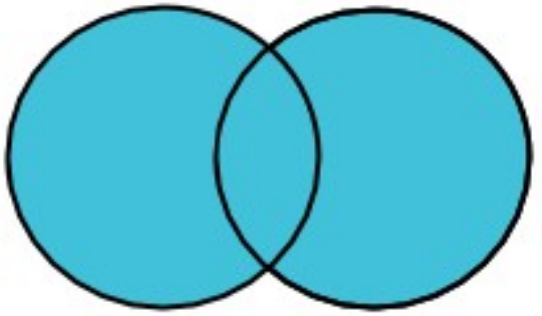
Left Join



Right Join



Inner Join



Full Outer Join

Merge - Opções

Parameters:

right : *DataFrame or named Series*
Object to merge with.

how : *{'left', 'right', 'outer', 'inner'}, default 'inner'*
Type of merge to be performed.

- left: use only keys from left frame, similar to a SQL left outer join; preserve key order.
- right: use only keys from right frame, similar to a SQL right outer join; preserve key order.
- outer: use union of keys from both frames, similar to a SQL full outer join; sort keys lexicographically.
- inner: use intersection of keys from both frames, similar to a SQL inner join; preserve the order of the left keys.

on : *label or list*
Column or index level names to join on. These must be found in both DataFrames. If *on* is None and not merging on indexes then this defaults to the intersection of the columns in both DataFrames.

left_on : *label or list, or array-like*
Column or index level names to join on in the left DataFrame. Can also be an array or list of arrays of the length of the left DataFrame. These arrays are treated as if they are columns.

right_on : *label or list, or array-like*
Column or index level names to join on in the right DataFrame. Can also be an array or list of arrays of the length of the right DataFrame. These arrays are treated as if they are columns.

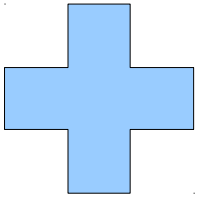
left_index : *bool, default False*
Use the index from the left DataFrame as the join key(s). If it is a MultiIndex, the number of keys in the other DataFrame (either the index or a number of columns) must match the number of levels.

Concat

O **concat** combina dois DataFrames unindo as linhas de ambos.

df1:

	nome	idade	genero
0	Celso	23	M
1	Josie	32	F



df2:

	nome	idade	genero
0	Luiz	11	M
1	Helen	18	F

```
pd.concat([df1, df2])
```

	nome	idade	genero
0	Celso	23	M
1	Josie	32	F
0	Luiz	11	M
1	Helen	18	F

Exercícios!

- Revise o conteúdo e faça os exercícios do notebook:
  03c-Pandas_Manipulação e Agregação de Dados.ipynb