Introdução ao Pandas II

Patrícia Novais

Sumário

- Operações com Dataframes
- União de dataframes: merge e concat
- Agrupamentos: groupby
- Agregações: agg

Operações com Dataframes

» Os dataframes do Pandas são estruturas muito versáteis pois permitem uma série de operações que ajudam nas análises dos dados. Veremos agora as seguintes operações:

- » contagem de valores
- » média
- » soma
- » valores únicos
- » limpeza de dados duplicados
- » limpeza de dados faltantes

Value_counts

» Não poucas vezes precisaremos saber a frequência com que uma dada informação se repete. Para obter tal informação de uma dada variável no Pandas podemos utilizar a função pd.value_counts().

```
df.educacao.value_counts()
```

```
secondary    23202
tertiary     13301
primary       6851
unknown       1857
Name: educacao, dtype: int64
```

```
df['estado civil'].value_counts()
```

```
married     27214
single      12790
divorced     5207
Name: estado civil, dtype: int64
```

Quantidade de dados com estado civil casado, solteiro e divorciado.

Média

» Veremos em mais detalhes nas próximas aulas, mas podemos obter a média simples dos dados utilizando a função pd.mean().

```
df.balance.mean()
```

```
1362.2720576850766
```

```
df.duracao.mean()
```

```
258.16935840707964
```

```
df[['balance','duracao']].mean()
```

Média de uma lista de variáveis

```
balance    1362.272058
duracao     258.169358
dtype: float64
```

Soma

» Quando precisamos saber a soma dos valores de uma determinada variável podemos usar a função pd.sum().

```
df.balance.sum()
```

```
61600855
```

```
df.duration.sum()
```

```
11672773.0
```

```
df[['pdays','duration','balance']].sum()
```

Soma de uma lista de variáveis

```
pdays         1818535.0
duration     11672773.0
balance      61600855.0
dtype: float64
```

Unique e nunique

» Outra importante função é a pd.unique(), que nos mostra todos os valores únicos em uma dada variável.

```
df.Profissao.unique()
```

```
array(['management', 'technician', 'entrepreneur', 'blue-collar',
       'unknown', 'retired', 'admin.', 'services', 'self-employed',
       'unemployed', 'housemaid', 'student'], dtype=object)
```

» Já a função pd.nunique() nos mostra a quantidade de valores únicos em uma variável.

```
df.Profissao.nunique()
```

```
12
```

Dados duplicados

» Outra importação ação aos analisar dados é verificar se há duplicidade nas informações, evitando assim que tiremos conclusões errôneas sobre os dados. Para verificar a existência de duplicidade, podemos usar a função pd.duplicated() e pd.sum().

```
df.duplicated().sum()
```

```
10
```

» Para saber quais os valores duplicados podemos utilizamos o loc.

```
df.loc[df.duplicated() == True]
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	can
45211	44	technician	single	secondary	no	29	yes	no	NaN	5	may	151.0	
45212	33	entrepreneur	married	secondary	no	2	yes	yes	NaN	5	may	76.0	
45213	38	technician	married	secondary	no	557	yes	no	cellular	16	nov	1556.0	

Dados duplicados

» Para criar um novo dataframe sem os dados duplicados utilizamos a função pd.drop_duplicated()

```
df2 = df.drop_duplicates()
```

```
df2.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaig
0	58	management	married	tertiary	no	2143	yes	no	NaN	5	may	261.0	
1	44	technician	single	secondary	no	29	yes	no	NaN	5	may	151.0	
2	33	entrepreneur	married	secondary	no	2	yes	yes	NaN	5	may	76.0	
3	47	blue-collar	married	unknown	no	1506	yes	no	NaN	5	may	92.0	
4	33	unknown	single	unknown	no	1	no	no	NaN	5	may	198.0	

```
df2.duplicated().sum()
```

Verificando que não há dados duplicados

0

Dados faltantes ou nulos

» Dados faltantes ou nulos podem enviesar as análises, por isso é sempre importante eliminá-los ou substituí-los (de acordo com o contexto de análise).

» Uma maneira de saber a quantidade de dados faltantes é utilizando a função pd.isna ou pd.isnull com a função pd.sum().

```
df.isna().sum()
```

```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact     76
day          0
month        0
duration    11
campaign     0
pdays        0
previous     0
poutcome     0
dtype: int64
```

```
df.isnull().sum()
```

```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact     76
day          0
month        0
duration    11
campaign     0
pdays        0
previous     0
poutcome     0
dtype: int64
```

Dados faltantes ou nulos

» Para criar um novo dataframe sem os dados faltantes ou nulos utilizamos a função pd.dropna()

```
df3 = df.dropna()
```

```
df3.isna().sum()
```

```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
dtype: int64
```

Verificando que não há dados faltantes

Existem várias maneiras de retirar os valores faltantes. Para conhecê-los, consulte a documentação da função.

Dados faltantes ou nulos

» Quando for necessário substituir os dados faltantes por um novo valor, podemos usar a função pd.fillna()

```
df4 = df.fillna("desconhecido")
```

```
df4.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	ca
0	58	management	married	tertiary	no	2143	yes	no	desconhecido	5	may	261	
1	44	technician	single	secondary	no	29	yes	no	desconhecido	5	may	151	
2	33	entrepreneur	married	secondary	no	2	yes	yes	desconhecido	5	may	76	
3	47	blue-collar	married	unknown	no	1506	yes	no	desconhecido	5	may	92	
4	33	unknown	single	unknown	no	1	no	no	desconhecido	5	may	198	

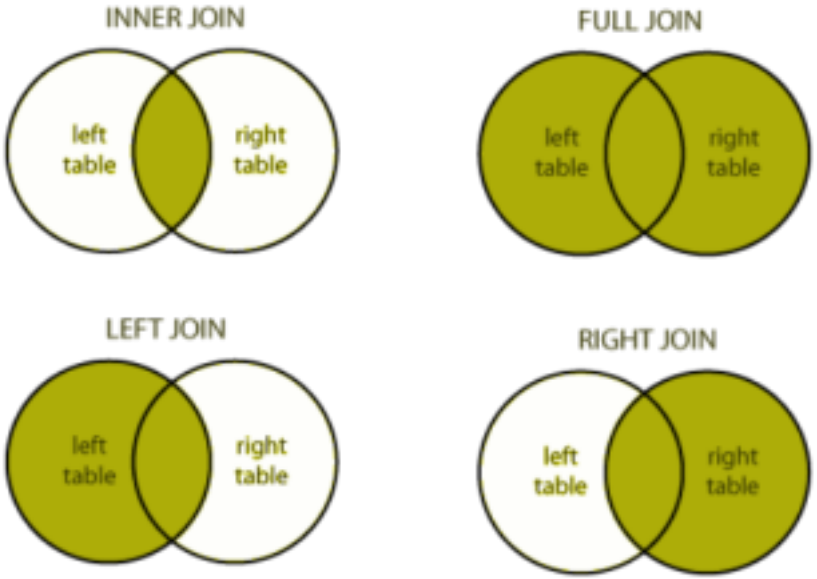
Existem várias maneiras de preencher os valores nulos. Para conhecê-los, consulte a documentação da função.

Exercícios – Operações

» Qual a média do score_felicidade?
» Qual a soma do PIB?
» Qual a soma do freedom e corrupcao?
» Há dados duplicados? Quantos? Verifique quais são eles.
» Verifique a quantidade de dados faltantes.
» Crie um novo dataframe onde os valores faltantes de score_felicidade sejam substituídos por -9999.
» Quantas e quais são regions existentes nos dados?
» Verifique a frequência dos dados segundo suas regiões. Qual a região com maior quantidade de dados? E a região com a menor quantidade?

Unindo amostras de dados

» É muito comum precisarmos unir dados de diferentes conjuntos de dados. O esquema abaixo exemplifica bem alguns tipos de união.



Não iremos explorar aqui todos os tipos de junções que podem ser feitos, mas é importante saber as principais diferenças entre eles.

Unindo amostras de dados

» Com o Pandas temos alguns métodos para unir dados de datasets distintos, dentre os quais podemos citar:

- concat: função que une os datasets ao longo de um eixo.
- merge: método que combina um ou mais datasets baseado em uma coluna em comum.

Veremos o básico de ambos a seguir.

Dataframes

» Vamos antes criar os dataframes que iremos utilizar:

```
dados1 = {
        'ID': ['1', '2', '3', '4', '5'],
        'nome': ['Paula', 'Claudia', 'Joao', 'Carlos', 'Ana'],
        'sobrenome': ['Pereira', 'Silva', 'Silveira', 'Bezerra', 'Souza']}
df1 = pd.DataFrame(dados1, columns = ['ID', 'nome', 'sobrenome'])
display('df1')
display(df1)

dados2 = {
        'ID': ['4', '5', '6', '7', '8'],
        'nome': ['Eder', 'Joana', 'Paulo', 'Pedro', 'Bete'],
        'sobrenome': ['Silva', 'Bezerra', 'Fernandes', 'Brito', 'Oliveira']}
df2 = pd.DataFrame(dados2, columns = ['ID', 'nome', 'sobrenome'])
display('df2')
display(df2)

notas = {
        'ID': ['1', '2', '3', '4', '5', '7', '8', '9', '10', '11'],
        'notas_id': [51, 15, 15, 61, 16, 14, 15, 1, 61, 16]}
df_notas = pd.DataFrame(notas, columns = ['ID','notas_id'])
display('df_notas')
display(df_notas)
```

Dataframes

» Vamos antes criar os dataframes que iremos utilizar:

'df1'

	ID	nome	sobrenome
0	1	Paula	Pereira
1	2	Claudia	Silva
2	3	Joao	Silveira
3	4	Carlos	Bezerra
4	5	Ana	Souza

'df2'

	ID	nome	sobrenome
0	4	Eder	Silva
1	5	Joana	Bezerra
2	6	Paulo	Fernandes
3	7	Pedro	Brito
4	8	Bete	Oliveira

'df_notas'

	ID	notas_id
0	1	51
1	2	15
2	3	15
3	4	61
4	5	16
5	7	14
6	8	15
7	9	1
8	10	61
9	11	16

concat

» A função pd.concat() irá unir os dados ao longo de um eixo (o default é pelo índice).

```
df_1_2 = pd.concat([df1,df2])
df_1_2
```

	ID	nome	sobrenome
0	1	Paula	Pereira
1	2	Claudia	Silva
2	3	Joao	Silveira
3	4	Carlos	Bezerra
4	5	Ana	Souza
0	4	Eder	Silva
1	5	Joana	Bezerra
2	6	Paulo	Fernandes
3	7	Pedro	Brito
4	8	Bete	Oliveira

concat

» Unindo pelas colunas, utilizando o parâmetro axis=1:

```
df_1_2 = pd.concat([df1,df2], axis=1)
df_1_2
```

	ID	nome	sobrenome	ID	nome	sobrenome
0	1	Paula	Pereira	4	Eder	Silva
1	2	Claudia	Silva	5	Joana	Bezerra
2	3	Joao	Silveira	6	Paulo	Fernandes
3	4	Carlos	Bezerra	7	Pedro	Brito
4	5	Ana	Souza	8	Bete	Oliveira

concat

» Unindo pelas colunas, utilizando o parâmetro axis=1:

```
df_1_notas = pd.concat([df1,df_notas], axis=1)
df_1_notas
```

	ID	nome	sobrenome	ID	notas_id
0	1	Paula	Pereira	1	51
1	2	Claudia	Silva	2	15
2	3	Joao	Silveira	3	15
3	4	Carlos	Bezerra	4	61
4	5	Ana	Souza	5	16
5	NaN	NaN	NaN	7	14
6	NaN	NaN	NaN	8	15
7	NaN	NaN	NaN	9	1
8	NaN	NaN	NaN	10	61
9	NaN	NaN	NaN	11	16

merge

» A função pd.merge também une datasets, porém utilizando explicitamente a coluna de interesse.

```
df_1_2_notas = pd.merge(df_1_2, df_notas, on='ID')
df_1_2_notas
```

	ID	nome	sobrenome	notas_id
0	1	Paula	Pereira	51
1	2	Claudia	Silva	15
2	3	Joao	Silveira	15
3	4	Carlos	Bezerra	61
4	4	Eder	Silva	61
5	5	Ana	Souza	16
6	5	Joana	Bezerra	16
7	7	Pedro	Brito	14
8	8	Bete	Oliveira	15

Os dataframes foram unidos com base na coluna ID.

Repare na quantidade de registros. O que há de diferente?

merge

» A função pd.merge permite fazer diversos tipos de união, como os mostrado no diagrama anterior, bastando ajustar o parâmetro how:

Merge method	SQL Join Name	Description
left	LEFT OUTER JOIN	Use keys from left frame only
right	RIGHT OUTER JOIN	Use keys from right frame only
outer	FULL OUTER JOIN	Use union of keys from both frames
inner	INNER JOIN	Use intersection of keys from both frames

```
pd.merge(df_esq, df_dir, on='coluna_em_comum', how='metodo de junção')
```

```
pd.merge(df_esq, df_dir, on='coluna_em_comum', how='left')
```

```
pd.merge(df_esq, df_dir, on='coluna_em_comum', how='right')
```

```
pd.merge(df_esq, df_dir, on='coluna_em_comum', how='outer')
```

```
pd.merge(df_esq, df_dir, on='coluna_em_comum', how='inner')
```

Documentações

» Todas os tipos de possibilidades de união de dados, por si só, já renderia um curso.

» Existe uma extensa documentação sobre os usos e especificidades das funções pd.concat() e pd.merge(). Estudar essas documentações te fará um melhor analista de dados.

» Material do Pydata: https://pandas.pydata.org/pandas-docs/stable/user_guide/merging.html

Exercícios – Unindo dataframes

1. Crie um dataframe para cada um dos arquivos nba_2015_a.csv, nba_2015_b.csv, nba_2015_c.csv, e bust_nba_2015.csv. Chame esses dataframes de df_a, df_b, df_c, bust, respectivamente.
2. Visualize o head() de cada um dos dataframes.
3. Concatene os arquivos df_a, df_b e df_c usando a função concat() usando os índices. Salve um dataframe chamado df_total.
4. Faça a concatenação do dataframe df_total com o dataframe bust utilizando a função merge() e a variável ID.
5. Busque a documentação das funções concat() e merge() e veja que outros parâmetros podem ser utilizados.

Agrupando os dados: groupby

» Diversas vezes precisamos analisar os dados agrupados, ao invés de um a um, para entender o comportamento do todo.

» Podemos querer saber a soma de valores, a frequência com que eles ocorrem, as médias agrupadas, dentre outras operações.

» No Pandas temos uma função excelente para isso, a função pd.groupby(), que agrupa os dados, calcula algumas propriedades dos grupos formados e sumariza os resultados.

Agrupando os dados: groupby

» Para apenas separarmos os dados de acordo com uma variável:

```
df.groupby('emprestimo')
```

Agrupando pela variável emprestimo

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x0000018DC4F17E80>
```

```
df.groupby('emprestimo').groups
```

Visualizando os grupos

```
{'no': Int64Index([    0,     1,     3,     4,     5,     7,     8,     9,    1
0,
                11,
            ...
            45209, 45210, 45211, 45213, 45214, 45215, 45216, 45217, 45218,
            45219],
           dtype='int64', length=37975),
 'yes': Int64Index([    2,     6,    20,    22,    24,    27,    29,    30,
32,
                54,
            ...
            45074, 45103, 45108, 45122, 45151, 45153, 45194, 45205, 45212,
            45220],
           dtype='int64', length=7246)}
```

Agrupando os dados: groupby

» Para apenas separarmos os dados de acordo com múltiplas variáveis, usando uma lista de variáveis:

Agrupando pelas variáveis emprestimo e profissao.

```
df.groupby(['emprestimo','profissao'])
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x0000018DC00388D0>
```

```
df.groupby(['emprestimo','profissao']).groups
```

```
{('no',
  'admin.'): Int64Index([   10,    11,    16,    25,    38,    39,    45,    5
3,    60,
               80,
            ...
            45142, 45144, 45147, 45162, 45167, 45171, 45173, 45176, 45177,
            45202],
           dtype='int64', length=4180),
 ('no',
  'blue-collar'): Int64Index([    3,    17,    33,    36,    42,    50,    57,
58,    62,
               64,
            ...
            45100, 45124, 45127, 45135, 45174, 45178, 45181, 45190, 45199,
            45209],
           dtype='int64', length=8048),
 ('no',
  'entrepreneur'): Int64Index([    7,    94,   172,   222,   232,   241,   256,
264,   273,
              357,
            ...
```

Agrupando os dados: groupby

» Podemos ainda aplicar funções sobre os agrupamentos

```
df.groupby('emprestimo').mean()
```

Média dos valores agrupados

emprestimo	idade	balance	dia	duracao	campanha	pdays	previous
no	41.011244	1474.428993	15.764450	259.587868	2.750125	41.204687	0.592153
yes	40.555479	774.139387	16.020563	250.864163	2.834391	35.024427	0.522219

```
df.groupby('emprestimo').count()
```

Contagem dos valores agrupados

emprestimo	idade	profissao	estado civil	educacao	default	balance	housing	contato	dia
no	37975	37975	37975	37975	37975	37975	37975	37910	37975
yes	7246	7246	7246	7246	7246	7246	7246	7235	7246

Agrupando os dados: groupby

» Podemos ainda aplicar funções sobre os agrupamentos

```
df.groupby(['emprestimo','estado civil']).mean()
```

Média dos valores agrupados

emprestimo	estado civil	idade	balance	dia	duracao	campanha	pdays	p
no	divorced	46.012133	1278.097993	15.688521	262.818585	2.602193	43.178721	(
	married	43.663446	1548.183990	15.810647	255.551408	2.838305	38.417136	(
	single	33.708034	1400.515277	15.700036	266.525791	2.628325	46.095884	(
yes	divorced	44.716612	717.111835	16.298588	261.144408	2.764387	30.870793	(
	married	42.185368	834.025531	16.061360	243.327039	2.863120	35.792534	(
	single	33.686899	637.957332	15.752404	266.291040	2.792668	35.171875	(

Agrupando os dados: Groupby e aggregation

» Podemos ainda querer mais do que apenas uma informação dos grupos, então podemos usar a função agg().

```
df.groupby(['emprestimo','housing']).agg(['mean','sum'])
```

		idade		balance		dia		
		mean	sum	mean	sum	mean	sum	mea
emprestimo	housing							
no	no	43.264280	744535	1737.451566	29899804	16.061538	276403	257.9042
	yes	39.144130	812867	1256.459453	26091637	15.518251	322252	260.9834
yes	no	42.417449	122035	752.713243	2165556	16.093848	46302	249.4928
	yes	39.329366	171830	788.248569	3443858	15.972305	69783	251.7665

```
df.groupby(['emprestimo','housing']).agg(['count','mean'])
```

		idade		balance		dia		dur
		count	mean	count	mean	count	mean	count	m
emprestimo	housing							
no	no	17209	43.264280	17209	1737.451566	17209	16.061538	17207	257.90
	yes	20766	39.144130	20766	1256.459453	20766	15.518251	20759	260.98
yes	no	2877	42.417449	2877	752.713243	2877	16.093848	2875	249.49

Agrupando os dados: Groupby e aggregation

```
df.groupby(['emprestimo','housing']).agg(['count','mean']).T
```

	emprestimo	no		yes	
	housing	no	yes	no	yes
idade	count	17209.000000	20766.000000	2877.000000	4369.000000
	mean	43.264280	39.144130	42.417449	39.329366
balance	count	17209.000000	20766.000000	2877.000000	4369.000000
	mean	1737.451566	1256.459453	752.713243	788.248569
dia	count	17209.000000	20766.000000	2877.000000	4369.000000
	mean	16.061538	15.518251	16.093848	15.972305
duracao	count	17207.000000	20759.000000	2875.000000	4369.000000
	mean	257.904225	260.983429	249.492870	251.766537
campanha	count	17209.000000	20766.000000	2877.000000	4369.000000
	mean	2.822767	2.689926	2.980188	2.738384
pdays	count	17209.000000	20766.000000	2877.000000	4369.000000
	mean	27.902144	52.228643	16.828989	47.006180
previous	count	17209.000000	20766.000000	2877.000000	4369.000000

» Para facilitar a visualização, podemos usar a função transposta T.

Exercícios - Agrupando os dados

1. Abra o arquivo preferencias.csv como um dataframe chamado pref.
2. Visualize os 5 primeiras linhas do arquivo.
3. Agrupe os dados pela variável Gender (gênero).
4. Verifique a contagem de itens por cada gênero.
5. Agrupe os dados pelas variáveis Gender e Favorite Color (cor favorita).
6. Quantos itens de gênero F também possuem cor favorita Cool?
7. Agrupe os dados pelas variáveis Gender e Favorite Color e Favorite Beverage (bebida favorita).
8. Verifique a quantidade de itens de gênero M que têm cor preferida Warm e que preferem Beer.