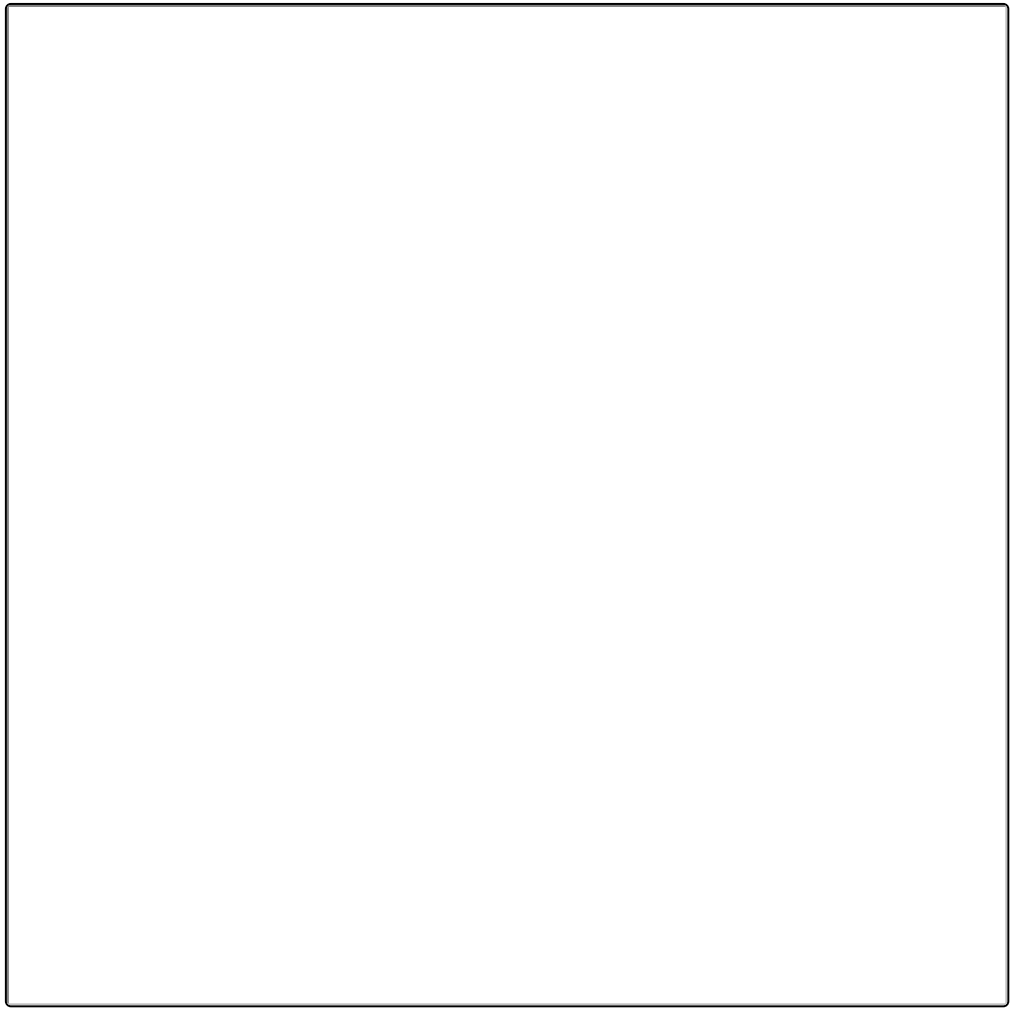
04 - Pandas (Pt3)

Docupedia Export

Author:Lima Queila (CtP/ETS)
Date:27-Jan-2026 17:10

Table of Contents

1 TREINAMENTO DE PYTHON (PANDAS) - Parte 3



Vamos continuar usando o **Pandas** para trabalhar nosso dados.
Vamos abrir o dataset que foi salvo no final da aula anterior.

```
import pandas as pd
titanic = pd.read_csv("data/titanic_1_aula.csv")
```

1.1 Tratando valores nulos

1.1.1 Porque tratar valores nulos?

Valores nulos atrapalham a utilização de algoritmos de ML e podem atrapalhar algumas análises também.

```
titanic.head(20)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Relatives	AgeRange
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	1	adulto
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	1	adulto
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	0	adulto
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	1	adulto
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	0	adulto
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q	0	nada
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S	0	adulto
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S	4	criança
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S	2	adulto
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C	1	adolescente
10	11	1	3	Sandstrom, Miss.	female	4.0	1	1	PP 9549	16.7000	G6	S	2	criança

Assim a gente consegue ver todos os dados nulos por coluna:

```
titanic.isnull().sum()
```

```
PassengerId     0
Survived        0
Pclass          0
Name            0
Sex             0
Age           177
SibSp           0
Parch           0
Ticket          0
Fare            0
Cabin         687
Embarked        2
Relatives       0
AgeRange        0
dtype: int64
```

Como funciona? O que acontece é que se somarmos todos os dados True/False ele vai dar a quantidade de True (que equivale a 1)

```
titanic.Age.isnull()
```

```
0      False
1      False
2      False
3      False
4      False
5       True
6      False
7      False
8      False
9      False
10     False
11     False
12     False
```

```
13    False
14    False
15    False
16    False
17     True
18    False
19     True
20    False
21    False
22    False
23    False
24    False
25    False
26     True
27    False
28     True
29     True
      ...
861   False
862   False
863    True
864   False
865   False
866   False
867   False
868    True
869   False
870   False
871   False
872   False
873   False
874   False
875   False
876   False
877   False
878    True
```

```
879    False
880    False
881    False
882    False
883    False
884    False
885    False
886    False
887    False
888     True
889    False
890    False
Name: Age, Length: 891, dtype: bool
```

1.2 Podemos tratar valores nulos por duas abordagens

1.2.1 1ª: Apagando Nulos

```
titanic.Age.head(10)
```

```
0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
5     NaN
6    54.0
7     2.0
8    27.0
9    14.0
Name: Age, dtype: float64
```

```
idades_nao_nulas = titanic.Age.dropna()
idades_nao_nulas.head(10)
```

```
0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
6    54.0
7     2.0
8    27.0
9    14.0
10    4.0
Name: Age, dtype: float64
```

```
qnt_idades_nulas_dropna = idades_nao_nulas.isnull().sum()
print("Quantidade de valores nulos: {}".format(qnt_idades_nulas_dropna))
```

```
Quantidade de valores nulos: 0
```

Size x Count :

- **size** - O tamanho total da série
- **count** - a quantidade de valores não nulos na série

```
titanic.Age.size
```

```
891
```

```
titanic.Age.count()
```

```
714
```

```
# idades_nao_nulas.size é igual a titanic.Age.count()
idades_nao_nulas.size
```

714

Atribuir uma série com os valores nulos apagados a uma coluna do DataFrame não faz sentido! A coluna vai voltar a possuir valores nulos
Como o tamanho da Série vai ser diferente da quantidade de linhas do Dataframe, os valores que faltam na série serão colocados como **nulos** conforme o índice.

```
titanic["AgeNotNull"] = idades_nao_nulas
```

```
titanic[["AgeNotNull","Age"]].head(10)
```

	AgeNotNull	Age
0	22.0	22.0
1	38.0	38.0
2	26.0	26.0
3	35.0	35.0
4	35.0	35.0
5	NaN	NaN
6	54.0	54.0

	AgeNotNull	Age
7	2.0	2.0
8	27.0	27.0
9	14.0	14.0

Isso acontece por causa do índice

```
idades_nao_nulas.head(10) # procure o 5
```

```
0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
6    54.0
7     2.0
8    27.0
9    14.0
10    4.0
Name: Age, dtype: float64
```

2ª: Preenchendo com algum valor

Se desejamos preencher os valores nulos com algum valor, podemos usar o **fillna**.

Nesse caso, se criarmos uma coluna com o resultado do `fillna`, ela realmente não terá mais valores nulos!

```
# Preenchendo as idades nulas com -1
```

```
serie_idade_sem_nulo = titanic.Age.fillna(-1)
serie_idade_sem_nulo.isnull().sum()
```

0

```
serie_idade_sem_nulo.count()
```

891

```
titanic["AgeFillNa-1"] = serie_idade_sem_nulo
```

```
titanic["AgeFillNa-1"].head(10)
```

0 22.0
1 38.0
2 26.0
3 35.0
4 35.0
5 -1.0
6 54.0
7 2.0
8 27.0
9 14.0
Name: AgeFillNa-1, dtype: float64

1.2.1.1 Preencher com `-1` não parece ser a melhor escolha.

Uma técnica muito usada é trocar o valor dos dados numéricos é usar a média

Vamos preencher os valores de idade nulo com a médias das idades

---

```
titanic["AgeFillNaMean"] = titanic.Age.fillna(titanic.Age.mean())
```

```
titanic[['Age','AgeFillNaMean']].head(10)
```

	Age	AgeFillNaMean
0	22.0	22.000000
1	38.0	38.000000
2	26.0	26.000000
3	35.0	35.000000
4	35.0	35.000000
5	NaN	29.699118
6	54.0	54.000000
7	2.0	2.000000
8	27.0	27.000000
9	14.0	14.000000

1.3 Agrupando

Podemos agrupar os dados baseados em uma coluna.
O agrupamento aplicando uma função nos dados que estão no mesmo grupo
O **groupby** seguido de uma operação funciona da seguinte forma:

1. Agrupa as linhas com valores iguais na coluna do agrupamento. [no código a coluna do agrupamente é colocada `groupby(COLUNA_DO_AGRUPAMENTO)`]
2. Aplica uma função em cada grupo gerado no passo (1) que transforme todas as linhas em apenas um valor.
3. Na imagem abaixo, estamos aplicando o groupby na coluna "X", e aplicando uma função de MÉDIA na coluna "Y". `df.groupby("X")["Y"].mean()`

```
# Agrupando pessoas por sexo
pessoas_por_sexo = titanic.groupby("Sex").count()
pessoas_por_sexo.head()
```

Sex	PassengerId	Survived	Pclass	Name	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Relatives	faixa_etaria
female	314	314	314	314	261	314	314	314	314	97	312	314	314
male	577	577	577	577	453	577	577	577	577	107	577	577	577

O resultado foi um Dataframe, que mostra a quantidade de dados não nulos em cada coluna dos grupos das cabines.

Se quisermos apenas o valor de pessoas por cabine, basta selecionarmos uma coluna que tenha todos os dados não nulos (Já que vimos que o **count()** conta apenas os nãos nulos)

```
# Quantas pessoas tem em cada cabine?
pessoas_por_cabine = titanic.groupby("Cabin")["PassengerId"].count()
pessoas_por_cabine.head(20)
```

```
Cabin
A10    1
A14    1
A16    1
A19    1
A20    1
A23    1
```

```
A24    1
A26    1
A31    1
A32    1
A34    1
A36    1
A5     1
A6     1
A7     1
B101   1
B102   1
B18    2
B19    1
B20    2
Name: PassengerId, dtype: int64
```

1.3.1 Outra maneira é substituir o `count` por `size`, então podemos contar a quantidade de elementos em qualquer coluna, mesmo com valores nulos

```
# A coluna Age tem valores nulos. Podemos usar o "size" para contar por ela
pessoas_por_cabine = titanic.groupby("Cabin")["Age"].size()
pessoas_por_cabine.head(20)
```

```
Cabin
A10    1
A14    1
A16    1
A19    1
A20    1
A23    1
A24    1
A26    1
A31    1
A32    1
```

```
A34    1
A36    1
A5     1
A6     1
A7     1
B101   1
B102   1
B18    2
B19    1
B20    2
Name: Age, dtype: int64
```

1.3.2 Outra maneira de agrupar é pela soma

1.3.3 Qual a idade somada das pessoas em cada cabine?

```
soma_idade = titanic.groupby("Cabin")["Age"].sum()
soma_idade.head(20)
```

```
Cabin
A10    36.0
A14     0.0
A16    48.0
A19     0.0
A20    49.0
A23    80.0
A24    31.0
A26    56.0
A31    40.0
A32     0.0
A34     4.0
A36    39.0
A5     71.0
A6     28.0
A7     56.0
```

```
B101    35.0
B102     0.0
B18     60.0
B19     61.0
B20     48.0
Name: Age, dtype: float64
```

1.3.4 Também podemos agrupar pela média

```
# Qual a idade média das pessoas de cada sexo?
media_idade = titanic.groupby("Sex")["Age"].mean()
media_idade.head(20)
```

```
Sex
female    27.915709
male      30.726645
Name: Age, dtype: float64
```

1.3.5 Existe uma maneira mais livre para agruparmos. Podemos usar o apply!

1.3.6 Usando o apply para contar os passageiros por cabine

```
def contar_pessoas(registros_agrupados):
    return registros_agrupados["PassengerId"].size


pessoas_por_cabine_apply = titanic.groupby("Cabin").apply(contar_pessoas)
pessoas_por_cabine_apply.head(10)
```

```
Cabin
A10    1
A14    1
A16    1
A19    1
A20    1
```

```
A23    1
A24    1
A26    1
A31    1
A32    1
dtype: int64
```

1.3.7 Vamos fazer o agrupamento por cabine somando as idades usando o apply

```python
def contar_pessoas(registros_agrupados):
    return registros_agrupados.Age.sum()


soma_idade_apply_certo = titanic.groupby("Cabin").apply(contar_pessoas)
soma_idade_apply_certo.describe()
```

```
count    147.000000
mean      45.091293
std       26.172014
min        0.000000
25%       28.500000
50%       44.000000
75%       59.250000
max      130.000000
dtype: float64
```

1.3.8 Vamos voltar ao caso de contar quantos homens e mulheres temos

```python
# Usando o count()
soma_idade_apply_certo = titanic.groupby("Sex")["PassengerId"].count()
soma_idade_apply_certo
```

```
Sex
female    314
```

male 577
Name: PassengerId, dtype: int64

```
# Usando apply
def count_sex(registros):
    return registros["Sex"].count()

titanic.groupby("Sex").apply(count_sex)
```

Sex
female 314
male 577
dtype: int64

2 Desafio

Lembram que não ficou muito bom quando colocamos a média geral da idade nas idades nulas?
Vamos tentar melhorar a média das idades!

Vamos preencher os nulos de maneira que se o passageiro for mulher esse recebe a idade media das mulheres.
Se o passageiro for homem ele recebe a idade média dos homens

Resultado

```
media_mulher = titanic[titanic.Sex == "female"].Age.mean()
media_homen = titanic[titanic.Sex == "male"].Age.mean()

print("""
Media de idade das mulheres: {}
Media de idade dos homens: {}""".format(media_mulher,media_homen))
```

Media de idade das mulheres: 27.915708812260537
Media de idade dos homens: 30.72664459161148

```
def preenche_idade(linha):
    idade = linha["Age"]
    sexo = linha["Sex"]

    if pandas.isnull(idade): #funcao padrão do pandas para verificar se na linha tem dado nulo

        if sexo == "female":
            return media_mulher
        else:
            return media_homen

    else:
        return idade


idades_preenchidas = titanic.apply(preenche_idade,axis=1)
```

```
print("""
Quantidade de nulos após o preenchimento: {}
""".format(idades_preenchidas.isnull().sum()))
```

Quantidade de nulos após o preenchimento: 0

```
# Vamos criar a coluna idade preenchida pra usar o resultado do apply
titanic["AgeFillNaSexMean"] = idades_preenchidas
```

```
titanic[["Age", "AgeFillNaSexMean"]]
```

	Age	AgeFillNaSexMean
0	22.0	22.000000
1	38.0	38.000000
2	26.0	26.000000
3	35.0	35.000000
4	35.0	35.000000
5	NaN	30.726645
6	54.0	54.000000
7	2.0	2.000000

	Age	AgeFillNaSexMean
8	27.0	27.000000
9	14.0	14.000000
10	4.0	4.000000
11	58.0	58.000000
12	20.0	20.000000
13	39.0	39.000000
14	14.0	14.000000
15	55.0	55.000000
16	2.0	2.000000
17	NaN	30.726645
18	31.0	31.000000
19	NaN	27.915709
20	35.0	35.000000
21	34.0	34.000000
22	15.0	15.000000

	Age	AgeFillNaSexMean
23	28.0	28.000000
24	8.0	8.000000
25	38.0	38.000000
26	NaN	30.726645
27	19.0	19.000000
28	NaN	27.915709
29	NaN	30.726645
...	...	...
861	21.0	21.000000
862	48.0	48.000000
863	NaN	27.915709
864	24.0	24.000000
865	42.0	42.000000
866	27.0	27.000000
867	31.0	31.000000

	Age	AgeFillNaSexMean
868	NaN	30.726645
869	4.0	4.000000
870	26.0	26.000000
871	47.0	47.000000
872	33.0	33.000000
873	47.0	47.000000
874	28.0	28.000000
875	15.0	15.000000
876	20.0	20.000000
877	19.0	19.000000
878	NaN	30.726645
879	56.0	56.000000
880	25.0	25.000000
881	33.0	33.000000
882	22.0	22.000000

	Age	AgeFillNaSexMean
883	28.0	28.000000
884	25.0	25.000000
885	39.0	39.000000
886	27.0	27.000000
887	19.0	19.000000
888	NaN	27.915709
889	26.0	26.000000
890	32.0	32.000000

891 rows × 2 columns

2.1 Salvando no trabalho :D

```
titanic.to_csv("titanic_2_aula.csv", index=False)
```

```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
```

```
Data columns (total 18 columns):
PassengerId        891 non-null int64
Survived           891 non-null int64
Pclass             891 non-null int64
Name               891 non-null object
Sex                891 non-null object
Age                714 non-null float64
SibSp              891 non-null int64
Parch              891 non-null int64
Ticket             891 non-null object
Fare               891 non-null float64
Cabin              204 non-null object
Embarked           889 non-null object
Relatives          891 non-null int64
AgeRange           891 non-null object
AgeNotNull         714 non-null float64
AgeFillNa-1        891 non-null float64
AgeFillNaMean      891 non-null float64
AgeFillNaSexMean   891 non-null float64
dtypes: float64(6), int64(6), object(6)
memory usage: 125.4+ KB
```