



# Aula 3

## **Limpeza e preparação de dados em Python**

---

PROF. MAURÍCIO DUARTE

FATEC - POMPÉIA

# Tratando dados ausentes

---

Dados ausentes são comuns em muitas aplicações de análise de dados. Um dos objetivos do pandas é deixar o trabalho com dados ausentes o menos problemático possível.

A forma como dados ausentes são representados em objetos pandas, de certo modo, não é perfeita, porém é funcional para muitos usuários.

Para dados numéricos, o pandas utiliza o valor de ponto flutuante NaN (Not a Number) para representá-los.

# Tratando dados ausentes – exemplo

---

```
import pandas as pd
import numpy as np

str = pd.Series(['Mauricio', 'FATEC', np.nan, 'Professor'])
str
```

```
0    Mauricio
1         FATEC
2          NaN
3   Professor
dtype: object
```

# Observação: Series em Python

---

Uma Series é um objeto em Python, do tipo array unidimensional, que contém uma sequência de valores. Esses valores possuem rótulos (índices). A Series mais simples é composta apenas de um array de dados... e nestes casos, os rótulos iniciam-se em 0...  
Veja:

```
A = pd.Series([4, 5, -8, 9])
```

Porém, é possível criar Series e também definir quais rótulos cada elemento irá possuir.

```
B = pd.Series ([4, 5, -8, 9], index = ['d', 'a', 'c', 'b'] )
```

# Tratando dados ausentes – exemplo

---

No pandas, é comum adotarmos a notação usada na linguagem R, referenciando os dados como NA (Not Available – indisponível). A notação NA representa dados inexistentes ou dados que existem, porém, não foram observados.

O valor embutido None do Python também é tratado como NA em arrays e objetos... Vejamos um exemplo...

# Tratando dados ausentes – exemplo

---

```
str[0] = None  
str.isnull()
```

```
0      True  
1     False  
2      True  
3     False  
dtype: bool
```

# Filtrando dados ausentes com dropna()

---

Em Python há algumas maneiras de filtrar dados ausentes. O método `dropna()` pode ser muito útil para isso. Considerando uma `Series`, o `dropna()` devolve a `Series` somente com os dados diferentes de null e os valores dos índices. Veja:

```
from numpy import nan as NA
notas = pd.Series([7.5, NA, 3.5, NA, 9.3])
notas.dropna()
```

```
0    7.5
2    3.5
4    9.3
dtype: float64
```

# Filtrando dados ausentes com dropna()

---

Com objetos DataFrame, a situação é um pouco mais complexa. Talvez você queira descartar linhas ou colunas que contenham somente NA ou apenas aquelas que contenham algum NA.

Por padrão, o dropna() descarta qualquer linha contendo um valor ausente.... Veja um exemplo



# Exemplo de dropna() com DataFrame

---

```
dados = pd.DataFrame([[ 'Ana', 7.5, 8.5], [ 'Luis', NA, NA],  
                      [NA, NA, NA], [ 'Maria', 6.5, NA]])
```

dados

	0	1	2
0	Ana	7.5	8.5
1	Luis	NaN	NaN
2	NaN	NaN	NaN
3	Maria	6.5	NaN



# Exemplo de dropna() com DataFrame

---

```
dados_limpos = dados.dropna()  
dados_limpos
```

	0	1	2
0	Ana	7.5	8.5




OBS. Percebe-se que todas as linhas que possuíam em algum lugar NA, foram descartadas... Porém, esta operação poderia não ser a desejada...

# Exemplo de dropna() com DataFrame

---

Ao usar o `dropna()` e passar como parâmetro **`how='all'`**, o `dropna()` descartará apenas as linhas que possuem todas as colunas iguais a NA. Veja...

```
dados.dropna(how='all')
```



	0	1	2
0	Ana	7.5	8.5
1	Luis	NaN	NaN
3	Maria	6.5	NaN

# Exemplo de dropna() com DataFrame

---

Caso fosse para descartar uma coluna inteira de NAs, use **axis=1**. Veja...

```
dados[4] = NA
```

```
dados
```

	0	1	2	4
0	Ana	7.5	8.5	NaN
1	Luis	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN
3	Maria	6.5	NaN	NaN

# Exemplo de dropna() com DataFrame

---

Removendo a coluna de  
NAs... Veja...

```
dados.dropna(axis=1, how='all')
```

	0	1	2
0	Ana	7.5	8.5
1	Luis	NaN	NaN
2	NaN	NaN	NaN
3	Maria	6.5	NaN



# Limpando dados com o parâmetro thresh

---


Em um DataFrame, pode ocorrer que desejamos manter somente as linhas contendo determinado número de observações. Para isso, usaremos o argumento `thresh`.

Primeiramente, em nosso exemplo, iremos criar um DataFrame com valores randômicos em 6 linhas e 3 colunas... Veja:

# Limpando dados com o parâmetro thresh

---

```
df = pd.DataFrame(np.random.randn(6,3))  
df
```



	0	1	2
0	0.258713	1.147135	-0.927219
1	-1.132042	-0.942985	1.598466
2	-0.627109	-0.085061	-0.693106
3	0.401265	-0.329551	0.875167
4	0.394148	-0.453748	-1.271954
5	-0.005204	1.075107	-0.276887

# Limpendo dados com o parâmetro thresh

Na sequência, iremos colocar NA em 4 linhas da coluna 1 e em 2 linhas da coluna 2...  
Veja:

```
df.iloc[:4, 1] = NA  
df.iloc[:2, 2] = NA  
df
```

	0	1	2
0	0.258713	NaN	NaN
1	-1.132042	NaN	NaN
2	-0.627109	NaN	-0.693106
3	0.401265	NaN	0.875167
4	0.394148	-0.453748	-1.271954
5	-0.005204	1.075107	-0.276887



# Limpando dados com o parâmetro thresh

---

Usando o thresh para remover apenas as linhas que possuem 2 valores iguais a NA... Veja:

```
df.dropna(thresh=2)
```

	0	1	2
2	-0.627109	NaN	-0.693106
3	0.401265	NaN	0.875167
4	0.394148	-0.453748	-1.271954
5	-0.005204	1.075107	-0.276887

# Preenchendo dados ausentes

Ao invés de filtrar dados ausentes e descartá-los... Poderemos preenche-los de várias maneiras... Na maioria dos casos, o método **fillna ()** poderá ser usado... Vamos considerar o mesmo DataFrame anterior...

```
df.iloc[:4, 1] = NA
df.iloc[:2, 2] = NA
df
```

	0	1	2
0	-0.507275	NaN	NaN
1	-0.382485	NaN	NaN
2	-1.657780	NaN	0.613173
3	0.185148	NaN	-1.051864
4	-0.446178	0.063136	-0.441251
5	-0.930104	-0.095638	0.119468

# Preenchendo dados ausentes

---

Ao invocar o método **fillna ()** com uma **constante** os campos NAs serão substituídos por esse valor. Veja...

```
df.fillna(0)
```

	0	1	2
0	-0.507275	0.000000	0.000000
1	-0.382485	0.000000	0.000000
2	-1.657780	0.000000	0.613173
3	0.185148	0.000000	-1.051864
4	-0.446178	0.063136	-0.441251
5	-0.930104	-0.095638	0.119468

# Exercício 1

---

Crie uma Series com 10 notas reais... Coloque em algumas delas NA (dados ausentes)... Mostre a Series... Após isso, usando o método fillna, substitua os dados ausentes pela média das notas.

Tente usar para o cálculo da média, apenas as notas válidas.

## Exercício 2

---

Há várias maneiras para realizarmos substituição de valores em um determinado DataFrame. Uma delas é o método **replace**. Faça uma pesquisa no Google para saber como usá-lo e, em seguida, crie uma Series contendo 10 idades. Nestas idades, coloque umas 3 ou 4 idades inválidas com o valor -99 e usando replace substitua-as para NaN.

# Detectando valores Discrepantes

Filtrar ou transformar valores discrepantes (outliers) é, em boa medida, uma questão de aplicar operações em um array. Para exemplificar, vamos criar um DataFrame com alguns dados normalmente distribuídos:

```
numeros = pd.DataFrame(np.random.randn (1000, 4))  
numeros.describe()
```

	0	1	2	3
count	1000.000000	1000.000000	1000.000000	1000.000000
mean	0.004684	0.016057	0.018281	-0.035890
std	1.014325	1.008602	1.014088	0.980465
min	-3.175390	-3.400836	-4.044290	-3.491889
25%	-0.714180	-0.622666	-0.683977	-0.678321
50%	0.028962	-0.013969	0.019440	-0.021614
75%	0.698154	0.713128	0.736053	0.606191
max	2.867111	3.252273	4.384666	2.970184

# Detectando valores Discrepantes

---

Vamos agora, supor que desejamos encontrar os valores que excedem 3 em valor absoluto em uma das colunas... Veja:

```
col = numeros[2]  
col [np.abs(col) > 3]
```

```
23      -4.044290  
717      4.384666  
836      3.230657  
Name: 2, dtype: float64
```

# Detectando valores Discrepantes

Para selecionar todas as linhas que tenham um valor que exceda 3 ou -3, podemos usar o método **any** em um DataFrame ... Veja:

```
numeros[(np.abs(numeros) > 3).any(1)]
```

	0	1	2	3
23	0.020116	0.346029	-4.044290	-0.086682
59	-3.175390	-0.321210	1.269869	-0.276933
66	0.048507	3.094782	1.181756	0.466019
170	0.267194	-3.194057	0.444610	0.874398
291	-0.114597	3.242207	-0.920286	-1.244770
314	1.754696	0.935938	0.650707	-3.124155
505	0.913116	1.905692	-0.208995	-3.327910
717	-0.519372	0.268065	4.384666	-0.079835



# Detectando valores Discrepantes

Supomos ainda que desejamos eliminar os valores que estejam fora do intervalo -3 a 3 ... Veja:

```
numeros[np.abs(numeros)>3] = np.sign(numeros) *3  
numeros.describe()
```

	0	1	2	3
<b>count</b>	1000.000000	1000.000000	1000.000000	1000.000000
<b>mean</b>	0.004859	0.016392	0.017710	-0.034629
<b>std</b>	1.013790	1.003863	1.004668	0.976402
<b>min</b>	-3.000000	-3.000000	-3.000000	-3.000000
<b>25%</b>	-0.714180	-0.622666	-0.683977	-0.678321
<b>50%</b>	0.028962	-0.013969	0.019440	-0.021614
<b>75%</b>	0.698154	0.713128	0.736053	0.606191
<b>max</b>	2.867111	3.000000	3.000000	2.970184

## Obs. `np.sign()`

---

A instrução `np.sign(numeros)` gera valores -1 ou 1 com base no fato dos valores em números serem positivos ou negativos

# Atividade para entregar - Avaliativa

---

Crie uma Series contendo 20 notas de alunos. As notas devem estar entre 0 e 10. Notas fora deste intervalo são discrepantes.... Em um primeiro momento, localize todos as notas discrepantes e em seguida substitua-as: as maiores que 10 para 10 e as menores que 0 para 0.