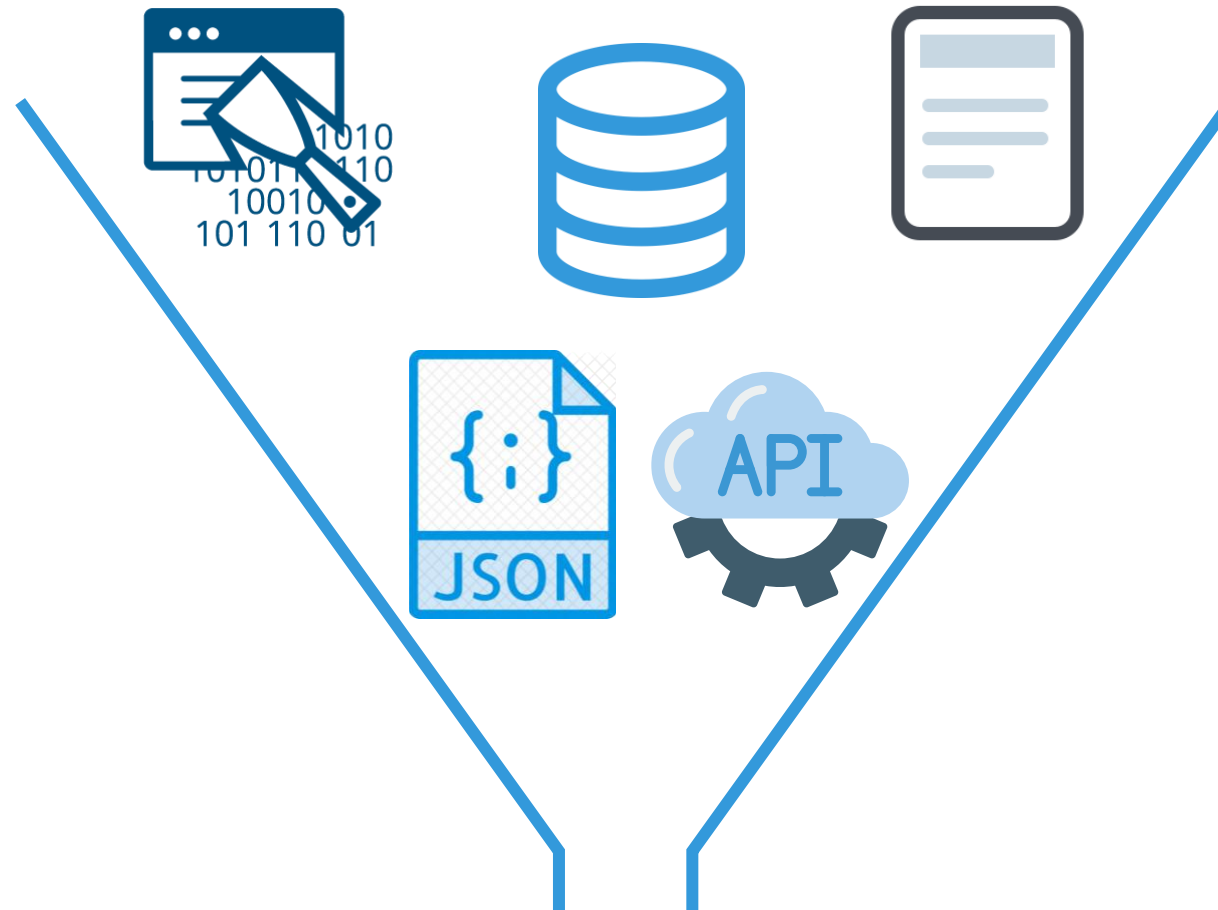
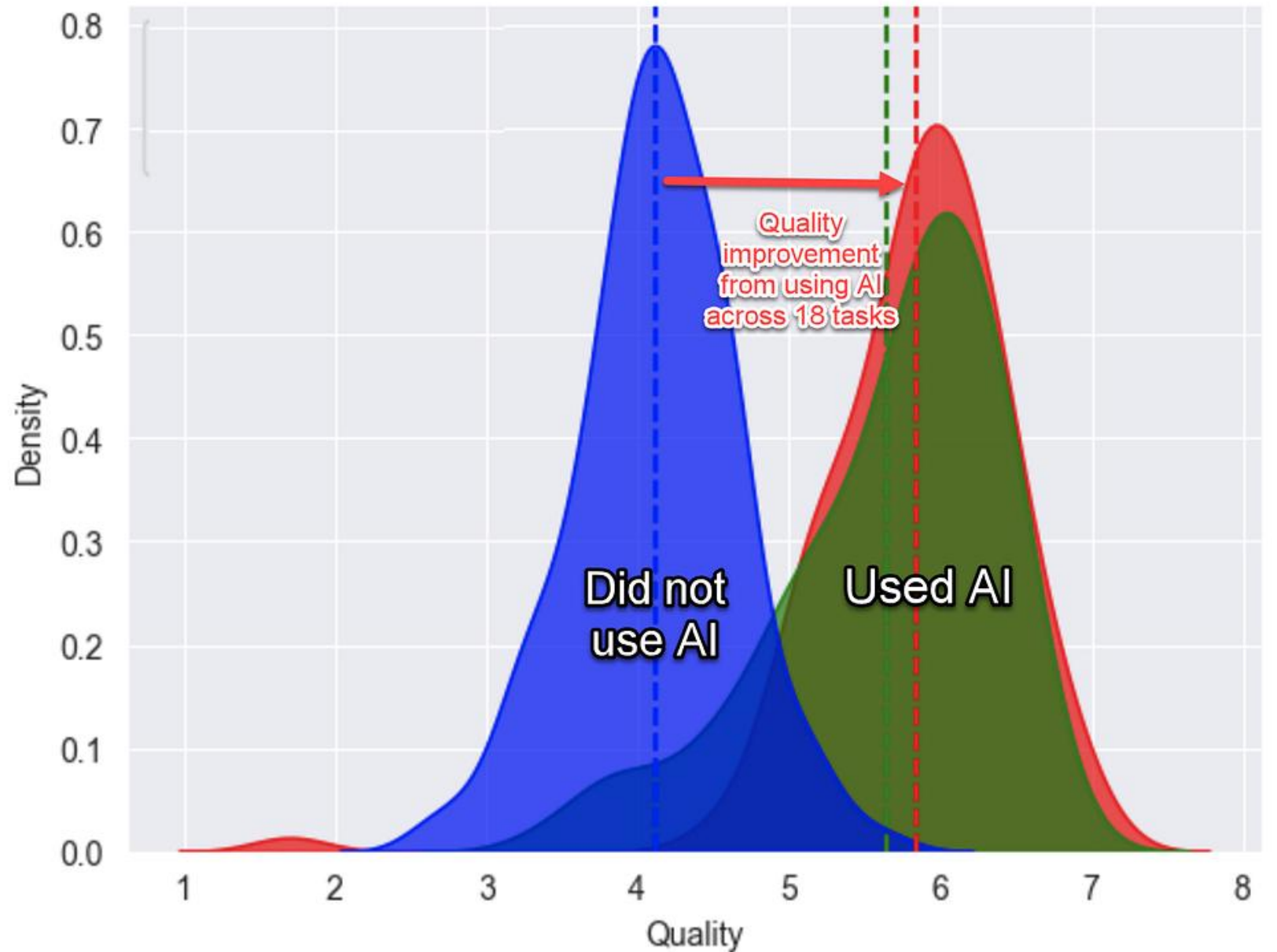


Pandas e Fontes de dados



Produtividade com o uso da IA

- Ethan Mollick mediu
 - Com um RCT



Função apply

- Aplica uma função ao longo de um eixo (linhas ou colunas) de um DataFrame
 - padrão: axis=0 (rows/index)

	b	d	e
Utah	0.807185	-1.135001	-1.071895
Ohio	1.943592	0.151843	1.507981
Texas	-0.374554	0.076108	1.625286
Oregon	0.252045	1.726434	-0.300007

```
3 f = lambda x: x.max()*2
4 frame.apply(f, axis='rows')
<
b      3.887184
d      3.452868
e      3.250571
dtype: float64
```

```
2 frame.apply(f, axis='columns')
<
Utah      1.614371
Ohio      3.887184
Texas     3.250571
Oregon    3.452868
dtype: float64
```

Função map

- Aplica uma função a cada elemento (element-wise)

	b	d	e
Utah	-1.551399	0.617767	0.045852
Ohio	1.079766	0.075621	1.174516
Texas	-0.339443	1.589292	-0.064619
Oregon	1.450546	-2.259592	0.348973

```
format = lambda x: '%.2f' % x  
frame.map(format)
```

	b	d	e
Utah	-1.55	0.62	0.05
Ohio	1.08	0.08	1.17
Texas	-0.34	1.59	-0.06
Oregon	1.45	-2.26	0.35



Sumarização e Estatística Descritiva

	one	two
a	1.40	NaN
b	7.10	-4.5
c	NaN	NaN
d	0.75	-1.3

```
In [232]: df.sum()  
Out[232]:  
one      9.25  
two     -5.80
```

```
In [233]: df.sum(axis='columns')  
Out[233]:  
a      1.40  
b      2.60  
c      NaN  
d     -0.55
```

mean (média)

- Os valores de NA são excluídos,
 - a menos que toda a fatia
 - ✓ linha ou coluna seja NA.
 - Isso pode ser desativado com a opção skipna

	one	two
a	1.40	NaN
b	7.10	-4.5
c	NaN	NaN
d	0.75	-1.3

```
1 df.mean(axis='columns')
```

a	1.400
b	1.300
c	NaN
d	-0.275

```
In [234]: df.mean(axis='columns', skipna=False)
```

```
Out[234]:
```

a	NaN
b	1.300
c	NaN
d	-0.275

describe (resumo de várias estatísticas)

- Computa várias estatísticas

	one	two
a	1.40	NaN
b	7.10	-4.5
c	NaN	NaN
d	0.75	-1.3



```
In [237]: df.describe()
```

```
Out[237]:
```

	one	two
count	3.000000	2.000000
mean	3.083333	-2.900000
std	3.493685	2.262742
min	0.750000	-4.500000
25%	1.075000	-3.700000
50%	1.400000	-2.900000
75%	4.250000	-2.100000
max	7.100000	-1.300000

Exemplo de uso do read_csv / read_excel

```
pd.read_excel('./pib_municipios.xlsx', sheet_name='Tabela', skiprows=3)
```

	Unnamed: 0	Unnamed: 1	Unnamed: 2	2007	2009	2011	2013
0	MU	1100015.0	Alta Floresta D'Oeste (RO)	191364	256986	280510	341325.0
1	MU	1100023.0	Ariquemes (RO)	905203	1133095	1651885	1799853.0
2	MU	1100031.0	Cabixi (RO)	49166	69776	77217	96365.0
3	MU	1100049.0	Cacoal (RO)	814890	985479	1259024	1433254.0
4	MU	1100056.0	Cerejeiras (RO)	143270	190902	260142	353270.0

Exemplo de uso do read_csv / read_excel

- Neste curso, descubra a URL na página do github

```
1 pd.read_excel('https://github.com/alexlopespereira/curso_ciencia_dados2020/r
```

	Unnamed: 0	Unnamed: 1	Unnamed: 2	2007	2009	2011	2013
0	MU	1100015.0	Alta Floresta D'Oeste (RO)	191364	256986	280510	341325.0
1	MU	1100023.0	Ariquemes (RO)	905203	1133095	1651885	1799853.0
2	MU	1100031.0	Cabixi (RO)	49166	69776	77217	96365.0
3	MU	1100049.0	Cacoal (RO)	814890	985479	1259024	1433254.0
4	MU	1100056.0	Cerejeiras (RO)	143270	190902	260142	353270.0

Solução para os principais problemas de leitura de arquivos read_csv/read_excel

- sep="," (somente no read_csv)
 - caracter de separação entre as colunas. Padrão é a vírgula
- skiprows=*N*
 - Ignora as primeiras *N* linhas do arquivo
 - ✓ Encontre o valor de *N* por tentativa e erro.
- skipfooter=*N*
 - Ignora as últimas *N* linhas do arquivo
- na_values="-" ou na_values=["-", "..."]
 - Especifica o(s) valor(es) que devem ser interpretados como NA
- encoding="UTF-8" (somente no read_csv)
 - Especifica o padrão de codificação dos caracteres do arquivo texto
- dtype={'cod_munic': str, 'cod_uf': str}
 - Especifica o tipo de dado que uma dada coluna deve ter no dataframe
- decimal=', ' (o padrão é o ponto)
 - Use a vírgula quando as casas decimais estiverem separadas por vírgula)
- Ver exemplo no Jupyter notebook

Descartando valores faltantes (NA ou NaN)

- Aceita o argumento `inplace=True`

```
In [15]: from numpy import nan as NA
```

```
In [16]: data = pd.Series([1, NA, 3.5, NA, 7])
```

```
In [17]: data.dropna()
```

```
Out[17]:
```

```
0    1.0
```

```
2    3.5
```

```
4    7.0
```

```
In [18]: data[data.notnull()]
```

```
Out[18]:
```

```
0    1.0
```

```
2    3.5
```

```
4    7.0
```



Equivalentes

Preenchendo valores faltantes

- `fillna` também aceita o argumento `inplace=True`

```
In [27]: df = pd.DataFrame(np.random.randn(7, 3))
```

```
In [33]: df.fillna(0)
```

```
Out[33]:
```

	0	1	2
0	-0.204708	0.000000	0.000000
1	-0.555730	0.000000	0.000000
2	0.092908	0.000000	0.769023
3	1.246435	0.000000	-1.296221
4	0.274992	0.228913	1.352917
5	0.886429	-2.001637	-0.371843
6	1.669025	-0.438570	-0.539741

```
In [34]: df.fillna({1: 0.5, 2: 0})
```

```
Out[34]:
```

	0	1	2
0	-0.204708	0.500000	0.000000
1	-0.555730	0.500000	0.000000
2	0.092908	0.500000	0.769023
3	1.246435	0.500000	-1.296221
4	0.274992	0.228913	1.352917
5	0.886429	-2.001637	-0.371843
6	1.669025	-0.438570	-0.539741

Preenchendo valores faltantes

- `fillna` possui um método de preenchimento `ffill`

```
In [37]: df = pd.DataFrame(np.random.randn(6, 3))
```

```
In [38]: df.iloc[2:, 1] = NA
```

```
In [39]: df.iloc[4:, 2] = NA
```

```
In [40]: df
```

```
Out[40]:
```

	0	1	2
0	0.476985	3.248944	-1.021228
1	-0.577087	0.124121	0.302614
2	0.523772	NaN	1.343810
3	-0.713544	NaN	-2.370232
4	-1.860761	NaN	NaN
5	-1.265934	NaN	NaN

```
In [41]: df.fillna(method='ffill')  
Out[41]:
```

	0	1	2
0	0.476985	3.248944	-1.021228
1	-0.577087	0.124121	0.302614
2	0.523772	0.124121	1.343810
3	-0.713544	0.124121	-2.370232
4	-1.860761	0.124121	-2.370232
5	-1.265934	0.124121	-2.370232

Remover duplicatas

```
In [45]: data = pd.DataFrame({'k1': ['one', 'two'] * 3 + ['two'],  
.....:                      'k2': [1, 1, 2, 3, 3, 4, 4]})
```

```
In [46]: data
```

```
Out[46]:
```

	k1	k2
0	one	1
1	two	1
2	one	2
3	two	3
4	one	3
5	two	4
6	two	4

```
In [47]: data.duplicated()
```

```
Out[47]:
```

0	False
1	False
2	False
3	False
4	False
5	False
6	True

dtype: bool

```
In [48]: data.drop_duplicates()
```

```
Out[48]:
```

	k1	k2
0	one	1
1	two	1
2	one	2
3	two	3
4	one	3
5	two	4

Indexação Hierárquica

- Possibilita mais de um nível de indexação num eixo

```
data = pd.Series(np.random.randn(9),  
                 index=[['a', 'a', 'a', 'b', 'b', 'c', 'c', 'd', 'd'],  
                       [1, 2, 3, 1, 3, 1, 2, 2, 3]))
```

a	1	-0.204708
	2	0.478943
	3	-0.519439
b	1	-0.555730
	3	1.965781
c	1	1.393406
	2	0.092908
d	2	0.281746
	3	0.769023

Filtro com lista

```
In [14]: data.loc[['b', 'd']]  
Out[14]:  
b 1 -0.555730  
   3  1.965781  
d 2  0.281746  
   3  0.769023
```

Filtro no 2º Nível

```
In [15]: data.loc[:, 2]  
Out[15]:  
a    0.478943  
c    0.092908  
d    0.281746
```

Resumo estatístico por nível

state		Ohio		Colorado
color		Green	Red	Green
key2	key1			
1	a	0	1	2
	b	6	7	8
2	a	3	4	5
	b	9	10	11

```
In [27]: frame.sum(level='key2')
```

```
Out[27]:
```

state	Ohio		Colorado
color	Green	Red	Green
key2			
1	6	8	10
2	12	14	16

Erros que vocês vão cometer

- Fazer uma operação que espera uma string
 - sobre uma coluna que tem dados numéricos (int ou float)
- Fazer uma operação que espera um dado numérico
 - sobre uma coluna que tem dados em formato de string
- Para sair desse problema, investigue usando `pd.DataFrame.info()`
 - Ou seja, `df.info()`

Data columns (total 9 columns):

#	Column	Non-Null Count	Dtype
0	nivel	5570 non-null	object
1	cod_ibge7	5570 non-null	int64
2	municipio	5570 non-null	object
3	2007	5570 non-null	object
4	2009	5570 non-null	object
-	-	-	-

string

inteiro

Depois de encontrar a causa do problema você precisará usar um desses argumentos para reler o arquivo

- `sep=","` (somente no `read_csv` ou `sep=";"`)
 - caracter de separação entre as colunas. Padrão é a vírgula
- `skiprows=N`
 - Ignora as primeiras *N* linhas do arquivo
 - ✓ Encontre o valor de *N* por tentativa e erro.
- `skipfooter=N`
 - Ignora as últimas *N* linhas do arquivo
- `na_values="-"` ou `na_values=["-", "..."]`
 - Especifica o(s) valor(es) que devem ser interpretados como NA
- `encoding="UTF-8"` (somente no `read_csv`)
 - Especifica o padrão de codificação dos caracteres do arquivo texto
- `dtype={'cod_munic': str, 'cod_uf': str}`
 - Especifica o tipo de dado que uma dada coluna deve ter no dataframe
- `decimal=','` (o padrão é o ponto)
 - Use a vírgula quando as casas decimais estiverem separadas por vírgula)

A IA ainda não está boa carregar arquivos tabulares

- A IA ainda não está boa para inferir os argumentos das funções do pandas que leem arquivos e transformam num DataFrame.
- **Você precisará, a priori, descobrir** quais são esses argumentos e solicitar que a IA os utilize para ler os arquivos.
 - Por exemplo, visualizando as primeiras e ultimas linhas dos arquivos

Prática no Colab Notebook

- Faça o restante dos exercícios da aula;
- Há exercícios extra.