

Aula 01 | PosTech | Tratamento, exploração e visualização de dados

Anotações sobre a primeira aula da PosTech FIAP ✨✨

<https://on.fiap.com.br/mod/conteudoshtml/view.php?id=307796&c=8729&sesskey=ZuKoJQwSR0>

Temas abordados:

- Como ler dados de fontes diferentes;
- Introdução à união de Dataframes com pequenas manipulações e tratamentos;
- Como comparar os gastos em relação à determinado estado;
- Como fazer a leitura de dados no formato Excel com Pandas;
- Como tornar uma string em (com formato de tabela) legível em Pandas usando o String.IO e como usar o Join em Pandas.

Pré-requisitos:

- Base de dados:

https://github.com/alura-cursos/agendamento-hospitalar/blob/main/dados/estimativa_dou_2020.xls?raw=true

- Importar essa base de dados no Colaboratory

- Ler esse artigo sobre Pandas:

<https://www.analyticsvidhya.com/blog/2021/06/data-manipulation-using-pandas-essential-functionalities-of-pandas-you-need-to-know/#:~:text=Pandas is an open-source,work on%2C DataFrame and Series>

Em um ambiente prático, é comum trabalharmos com situações em que temos que buscar dados de diversas fontes como csv, excel, html, pdf e muito mais. Como trazer esses dados e por onde eu posso lê-los?

Você sabia que as opções de “read” do Pandas aceitam que você leia arquivos que estejam na internet?

😊 É bem simples, imagine que você quer ler um arquivo excel de um repositório github, dessa forma você pode:

```
ibge_estimativa = pd.read_excel("https://github.com/alura-cursos/
    agendamento-hospitalar/blob/main/dados/estimativa_dou_2020.xls?
    raw=true")
```

Agora, o caminho não é um arquivo Excel em minha máquina local, mas um “caminho” dentro da web.

Posso só copiar e colar de uma página web o conteúdo de uma tabela direto para o Python?

Sim! Não é recomendado, a não ser que seja uma última instância de desespero. Já que, se você precisa ler dados de páginas web que não estão contidos em arquivos, você tem o **read_html()** do Pandas ou até mesmo **bibliotecas auxiliares com requests, urllib e BeautifulSoup**, que fazem um **web scraping** (em algumas fases futuras) que é uma beleza.

Um exemplo, de forma simplificada, é usar **StringIO do Python**, já contida na primeira parte da nossa aula. Outro ponto é olharmos para um universo dos tratamentos de dados de maneira mais incisiva. Não é comum na realidade seus dados virem do jeito que você imagina, com os campos certos, nenhum valor nulo, campos com o seu “tipo” correto.

Por isso, existem funções que nos ajudam, como **.dropna()** que dropa tudo o que é nulo (aconselho examinar com calma, porque ela deleta registros caso algo nulo seja identificado, podendo tirar mais do que deveria).

Outro caso é a função **.info()** que nos traz uma descrição do Dataframe, nos datando a criação de nascimento desse objeto, mostrando as colunas e seus tipos nativos.

```
ibge_estimativa = pd.read_excel(" https://github.com/alura-cursos/
    agendamento-hospitalar/blob/main/dados/estimativa_dou_2020.xls?
    raw=true")
ibge_estimativa.head()
```

```
[5] import pandas as pd

ibge_estimativa = pd.read_excel('/content/estimativa dou 2020.xls')

ibge_estimativa.head()
```

	ESTIMATIVAS DA POPULAÇÃO RESIDENTE NO BRASIL E UNIDADES DA FEDERAÇÃO COM DATA DE REFERÊNCIA EM 1º DE JULHO DE 2020	Unnamed: 1	Unnamed: 2
0	BRASIL E UNIDADES DA FEDERAÇÃO	NaN	POPULAÇÃO ESTIMADA
1	Brasil	NaN	211755692
2	Região Norte	NaN	18672591
3	Rondônia	NaN	1796460
4	Acre	NaN	894470

```
dados_da_populacao = """Posição Unidade federativa População % da pop. total País comparável
1 São Paulo 46 649 132 21,9% Espanha (46 439 864)

2 Minas Gerais 21 411 923 10,1% Sri Lanka (20 675 000)

3 Rio de Janeiro 17 463 349 8,2% Países Baixos (16 922 900)

4 Bahia 14 985 284 7,1% Chade (14 037 000)

5 Paraná 11 597 484 5,4% Bolívia (11 410 651)

6 Rio Grande do Sul 11 466 630 5,4% Bélgica (11 250 659)

7 Pernambuco 9 674 793 4,5% Bielorrússia (9 485 300)

8 Ceará 9 240 580 4,3% Emirados Árabes Unidos (9 157 000)

9 Pará 8 777 124 4,1% Áustria (8 602 112)

10 Santa Catarina 7 338 473 3,4% Sérvia (7 114 393)

11 Goiás 7 206 589 3,4% Paraguai (7 003 406)

12 Maranhão 7 153 262 3,4% Paraguai (7 003 406)

13 Amazonas 4 269 995 2,0% Líbano (4 168 000)

14 Espírito Santo 4 108 508 1,9% Líbano (4 168 000)

15 Paraíba 4 059 905 1,9% Líbano (4 168 000)

16 Mato Grosso 3 567 234 1,7% Uruguai (3 415 866)

17 Rio Grande do Norte 3 560 903 1,7% Uruguai (3 415 866)

18 Alagoas 3 365 351 1,6% Uruguai (3 415 866)

19 Piauí 3 289 290 1,6% Kuwait (3 268 431)

20 Distrito Federal 3 094 325 1,4% Lituânia (2 900 787)
```

```

21 Mato Grosso do Sul 2 839 188 1,3% Jamaica (2 717 991)

22 Sergipe 2 338 474 1,1% Namíbia (2 280 700)

23 Rondônia 1 815 278 0,8% Gabão (1 725 000)

24 Tocantins 1 607 363 0,7% Bahrein (1 359 800)

25 Acre 906 876 0,4% Fiji (859 178)

26 Amapá 877 613 0,4% Fiji (859 178)

27 Roraima 652 713 0,3% Luxemburgo (562 958)"""

```

```

dados da populacao = """Posição Unidade federativa População % da pop. total País comparável

1 São Paulo 46 649 132 21,9% Espanha (46 439 864)

2 Minas Gerais 21 411 923 10,1% Sri Lanka (20 675 000)

3 Rio de Janeiro 17 463 349 8,2% Países Baixos (16 922 900)

4 Bahia 14 985 284 7,1% Chade (14 037 000)

5 Paraná 11 597 484 5,4% Bolívia (11 410 651)

6 Rio Grande do Sul 11 466 630 5,4% Bélgica (11 250 659)

7 Pernambuco 9 674 793 4,5% Bielorrússia (9 485 300)

8 Ceará 9 240 580 4,3% Emirados Árabes Unidos (9 157 000)

9 Pará 8 777 124 4,1% Áustria (8 602 112)

10 Santa Catarina 7 338 473 3,4% Sérvia (7 114 393)

11 Goiás 7 206 589 3,4% Paraguai (7 003 406)

12 Maranhão 7 153 262 3,4% Paraguai (7 003 406)

13 Amazonas 4 269 995 2,0% Líbano (4 168 000)

14 Espírito Santo 4 108 508 1,9% Líbano (4 168 000)

15 Paraíba 4 059 905 1,9% Líbano (4 168 000)

```

Fonte desse trecho de código:

https://pt.wikipedia.org/wiki/Lista_de_unidades_federativas_do_Brasil_por_popula%C3%A7%C3%A3o#cite_note-IBGE_POP-1

No código acima, na primeira parte lemos a base de dados em Excel, enquanto que na segunda parte estamos preocupados em trazer essa base de dados em forma de string.

Mas, para fazer essa ação, devemos saber manipular isso. Então, em seguida podemos realizar:

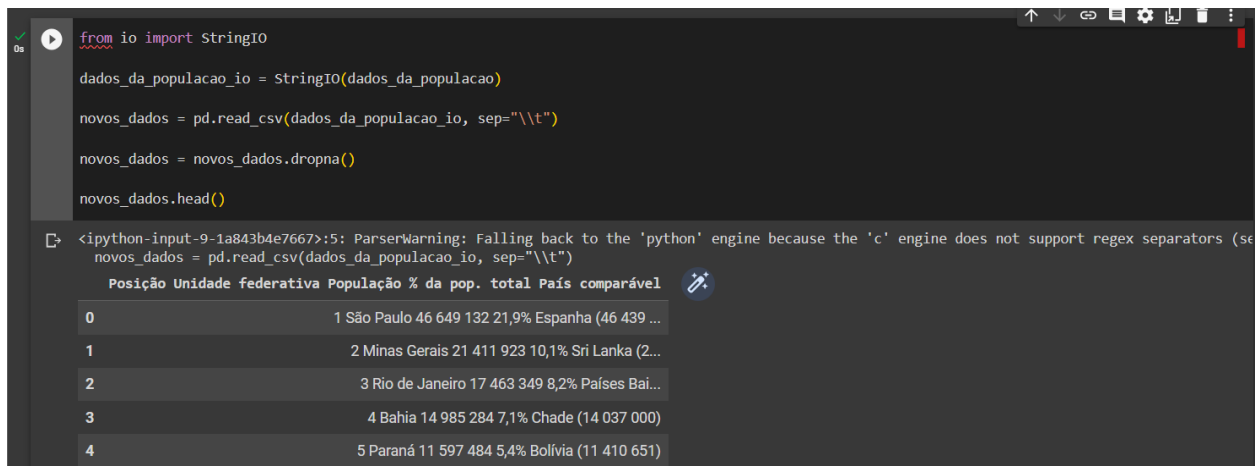
```
from io import StringIO

dados_da_populacao_io = StringIO(dados_da_populacao)

novos_dados = pd.read_csv(dados_da_populacao_io, sep="\t")

novos_dados = novos_dados.dropna()

novos_dados.head()
```



The screenshot shows a Jupyter Notebook interface. The top part displays the code from the previous block, which reads a CSV file from a StringIO object and drops missing values. Below the code, the output is shown as a DataFrame with 5 rows and 5 columns. The columns are labeled 'Posição', 'Unidade federativa', 'População', '% da pop. total', and 'País comparável'. The data shows the top 5 most populous states in Brazil, with São Paulo being the most populous.

Posição	Unidade federativa	População	% da pop. total	País comparável
0	1 São Paulo	46 649 132	21,9%	Espanha (46 439 ...)
1	2 Minas Gerais	21 411 923	10,1%	Sri Lanka (2...
2	3 Rio de Janeiro	17 463 349	8,2%	Países Bai...
3	4 Bahia	14 985 284	7,1%	Chade (14 037 000)
4	5 Paraná	11 597 484	5,4%	Bolívia (11 410 651)

```
populacao.columns = ["posicao", "uf", "populacao", "porcentagem",  
"pais_comparavel"]  
populacao["populacao"] = populacao["populacao"].str.replace(" ", "").astype(int)  
populacao = populacao[["uf", "populacao"]]  
populacao.head()
```

```

populacao.info ()
populacao.describe()

display(gastos_do_mais_recente.head())
display(populacao.head())

populacao = populacao.set_index("uf")
populacao.head()

populacao.index = populacao.index.str.strip()
populacao.head()

gastos_do_mais_recente.index = gastos_do_mais_recente.index.str[3:]
gastos_do_mais_recente.head()

for estado in gastos_do_mais_recente.index:
    populacao.index = populacao.index.str.replace(f"{estado} {estado}", estado)
    populacao.index

gastos_e_populacao = populacao.join(gastos_do_mais_recente)
gastos_e_populacao.head()

```

Parte 1 | Fontes externas, limpeza e manipulação de dados I

Vamos no Censo 2022 IBGE para termos previsão de períodos de acordo com o crescimento da população.

<https://www.ibge.gov.br/estatisticas/sociais/populacao/9103-estimativas-de-populacao.html>

gouv.br

CORONAVÍRUS (COVID-19)

ACESSO À INFORMAÇÃO

PARTICIPE

LEGISLAÇÃO

ÓRGÃOS DO GOVERNO

IBGE

Instituto Brasileiro de Geografia e Estatística

Buscar

Estadísticas

Sociais

População

Estimativas da População

Estimativas da População

O que é

Edições

2021

Tabelas

Conceitos e métodos

Dúvidas e Contestações

Downloads

Informações técnicas

Notícias e Releases

Outras informações

Links

Tabelas - 2021

Estimativas de população enviadas ao TCU

Tabelas de estimativas para 1º de julho de 2021, atualizadas e enviadas ao TCU após a publicação no DOU:

PDF | ODS | XLS

Arquivos atualizados em 12/12/2022:

- Devido à revogação da decisão judicial que alterava a estimativa da população do município de Tapauá-AM.

Arquivos atualizados em 09/11/2022:

- Devido ao cumprimento de decisões judiciais que altera a estimativa da população do município de Fonte Boa-AM.

- Devido à revogação de decisão judicial que alterava a estimativa da população do município de Itacoatiara-AM.

Arquivos atualizados em 05/09/2022:

- Devido ao cumprimento de decisões judiciais que alteram as estimativas da população dos municípios de Novo Airão-AM, Borba-AM, Manacapuru-AM, Pauini-AM e Ribeirão do Pinhal-PR.

- Devido à revogação de decisão judicial que alterava a estimativa da população do município de Fonte Boa-AM.

Arquivos atualizados em 11/07/2022:

- Devido ao cumprimento de decisões judiciais que alteram as estimativas da população dos municípios de Barcelos, AM e Planaltina, GO.

- Devido à revogação de decisões judiciais que alteravam as populações dos municípios de Manquiri, AM e Vera Cruz, BA.

Posso **baixar estimativas** quando clico em “pdf”.

Aula 01 | PosTech | Tratamento, exploração e visualização de dados

7

ESTIMATIVAS DA POPULAÇÃO RESIDENTE NO BRASIL E UNIDADES DA FEDERAÇÃO COM DATA DE REFERÊNCIA EM 1º DE JULHO DE 2021	
BRASIL E UNIDADES DA FEDERAÇÃO	
Brasil	213.317.639
Região Norte	18.906.962
Rondônia	1.815.278
Acre	906.876
Amazonas	4.269.995
Roraima	652.713
Pará	8.777.124
Amapá	877.613
Tocantins	1.607.363
Região Nordeste	57.667.842
Maranhão	7.153.262
Piauí	3.289.290 ⁽¹⁾
Ceará	9.240.580 ⁽¹⁾
Rio Grande do Norte	3.560.903
Paraíba	4.059.905
Pernambuco	9.674.793 ⁽²⁾
Alagoas	3.365.351 ⁽²⁾
Sergipe	2.338.474 ⁽³⁾
Bahia	14.985.284 ⁽³⁾
Região Sudeste	89.632.912
Minas Gerais	21.411.923
Espírito Santo	4.108.508
Rio de Janeiro	17.463.349
São Paulo	46.649.132
Região Sul	30.402.587
Paraná	11.597.484
Santa Catarina	7.338.473
Rio Grande do Sul	11.466.630
Região Centro-Oeste	16.707.336
Mato Grosso do Sul	2.839.188
Mato Grosso	3.567.234
Goiás	7.206.589 ⁽⁴⁾
Distrito Federal	3.094.325 ⁽⁴⁾

Fonte: IBGE. Diretoria de Pesquisas - DPE - Coordenação de População e Indicadores Sociais - COPIS.

Notas:

(1) Diferença de 786 pessoas entre os Estados do Piauí e Ceará com relação a Projeção da População para o Brasil e Unidades da Federação 2021, para o ano de 2021, em virtude de alteração de limites entre municípios na fronteira interestadual.

(2) Diferença de 456 pessoas entre os Estados de Alagoas e Pernambuco com relação a Projeção da População para o Brasil e Unidades da Federação 2021, para o ano de 2021, em virtude de

Posso usar ela de várias formas...

Tem-se previsão por município também.

Se abrir em .xls, posso usar esse arquivo fazendo o upload dele no Colab, fazendo o mesmo que no módulo 01:

- ☐ `ibge_estimativa = pd.read_excel('/content/POP2021_20221212.xls')`
- ☐ `ibge_estimativa.head()`

```
[80] ibge_estimativa = pd.read_excel('/content/POP2021_20221212.xls')
```

```
ibge_estimativa.head()
```

	ESTIMATIVAS DA POPULAÇÃO RESIDENTE NO BRASIL E UNIDADES DA FEDERAÇÃO COM DATA DE REFERÊNCIA EM 1º DE JULHO DE 2021	Unnamed: 1	Unnamed: 2
0	BRASIL E UNIDADES DA FEDERAÇÃO	NaN	POPULAÇÃO ESTIMADA
1	Brasil	NaN	213317639
2	Região Norte	NaN	18906962
3	Rondônia	NaN	1815278
4	Acre	NaN	906876

A primeira linha é considerada "lixo", temos 3 colunas: estimativas, unnamed 1 e unnamed 2.

Algo que a gente pode fazer também é entrar no wikipedia procurando por "lista de unidades federativas do Brasil por população", selecionar essa lista toda, copiá-la e armazenar tudo em uma variável chamada `dados_da_populacao`:

Lista





Posição	Unidade federativa	População	% da pop. total	País comparável (habitantes)
1	São Paulo	46 649 132	21,9%	Espanha (46 439 864)
2	Minas Gerais	21 411 923	10,1%	Sri Lanka (20 675 000)
3	Rio de Janeiro	17 463 349	8,2%	Países Baixos (16 922 900)
4	Bahia	14 985 284	7,1%	Chade (14 037 000)
5	Paraná	11 597 484	5,4%	Bolívia (11 410 651)
6	Rio Grande do Sul	11 466 630	5,4%	Bélgica (11 250 659)
7	Pernambuco	9 674 793	4,5%	Bielorrússia (9 485 300)
8	Ceará	9 240 580	4,3%	Emirados Árabes Unidos (9 157 000)
9	Pará	8 777 124	4,1%	Áustria (8 602 112)
10	Santa Catarina	7 338 473	3,4%	Sérvia (7 114 393)
11	Goiás	7 206 589	3,4%	Paraguai (7 003 406)
12	Maranhão	7 153 262	3,4%	Paraguai (7 003 406)
13	Amazonas	4 269 995	2,0%	Libano (4 168 000)
14	Espírito Santo	4 108 508	1,9%	Libano (4 168 000)
15	Paraíba	4 059 905	1,9%	Libano (4 168 000)
16	Mato Grosso	3 567 234	1,7%	Uruguai (3 415 866)
17	Rio Grande do Norte	3 560 903	1,7%	Uruguai (3 415 866)

Detalhe: string em python começa com "" (três aspas).

☐ dados_da_populacao = """ <tabela abaixo> """

# Posição	Unidade federativa	Aa População(Censo de 2010)[2]	População(Prévia 2022)	Mudança	% da pop. total	País comparável(habitantes)	Files
1	<u>São Paulo</u>	<u>41 262 199</u>	46 024 937	+11.5%	22,2%	<u>Espanha</u> (46 754 778)	

# Posição	≡ Unidade federativa	Aa População(Censo de 2010)[2]	≡ População(Prévia 2022)	≡ Mudança	≡ % da pop. total	≡ País comparável(habitantes)	📎 Files
2	Minas Gerais	19 597 330	20 732 660	+5.8%	10,0%	Burquina Fasso (20 903 273)	
3	Rio de Janeiro	15 989 929	16 615 526	+3.9%	8,0%	Camboja (16 718 965)	
4	Bahia	14 016 906	14 659 023	+4.6%	7,1%	Zimbabwe (14 862 924)	
5	Paraná	10 444 526	11 835 379	+13.3%	5,7%	Tunísia (11 818 619)	
6	Rio Grande do Sul	10 693 929	11 088 065	+3.7%	5,3%	Cuba (11 326 616)	
7	Pernambuco	8 796 448	9 051 113	+2.9%	4,4%	Áustria (9 006 398)	
8	Ceará	8 452 381	8 936 431	+5.7%	4,3%	Papua-Nova Guiné (8 947 024)	
9	Pará	7 581 051	8 442 962	+11.4%	4,1%	Suíça (8 654 622)	
10	Santa Catarina	6 248 436	7 762 154	+24.2%	3,7%	Paraguai (7 132 538)	
11	Goiás	6 003 788	6 950 976	+15.8%	3,3%	Bulgária (6 948 445)	
12	Maranhão	6 574 789	6 800 605	+3.4%	3,3%	Bulgária (6 948 445)	
13	Espírito Santo	3 514 952	4 108 508	+13.1%	1,9%	Geórgia (3 989 167)	
14	Paraíba	3 766 528	4 030 961	+7.0%	1,9%	Geórgia (3 989 167)	
15	Amazonas	3 483 985	3 952 262	+13.4%	1,9%	Geórgia (3 989 167)	
16	Mato Grosso	3 035 122	3 784 239	+24.7%	1,8%	Geórgia (3 989 167)	
17	Rio Grande do Norte	3 168 027	3 303 953	+4.3%	1,6%	Uruguai (3 473 730)	
18	Piauí	3 118 360	3 270 174	+4.9%	1,6%	Bósnia e Herzegovina (3 280 819)	
19	Alagoas	3 120 494	3 125 254	+0.2%	1,5%	Mónaco (3 278 290)	
20	Distrito Federal	2 570 160	2 923 369	+13.7%	1,4%	Armênia (2 963 243)	
21	Mato Grosso do Sul	2 449 024	2 833 742	+15.7%	1,4%	Albânia (2 877 797)	
22	Sergipe	2 068 017	2 211 868	+7.0%	1,1%	Gabão (2 225 734)	
23	Rondônia	1 562 409	1 616 379	+3.5%	0,8%	Bahrein (1 701 575)	

# Posição	≡ Unidade federativa	Aa População(Censo de 2010)[2]	≡ População(Prévia 2022)	≡ Mudança	≡ % da pop. total	≡ País comparável(habitantes)	📎 Files
24	Tocantins	1 383 445	1 584 306	+14.5%	0,8%	Guiné Equatorial (1 402 985)	
25	Acre	733 559	829 780	+13.1%	0,4%	Comores (869 601)	
26	Amapá	669 526	774 268	+15.6%	0,4%	Guiné (786 552)	
27	Roraima	450 479	634 805	+40.9%	0,3%	Montenegro (649 335)	

☐ Colocar no final do código:

#fonte: https://pt.wikipedia.org/wiki/Lista_de_unidades_federativas_do_Brasil_por_popula%C3%A7%C3%A3o

#fonte indireta IBGE

☐ Deve ficar assim:

```

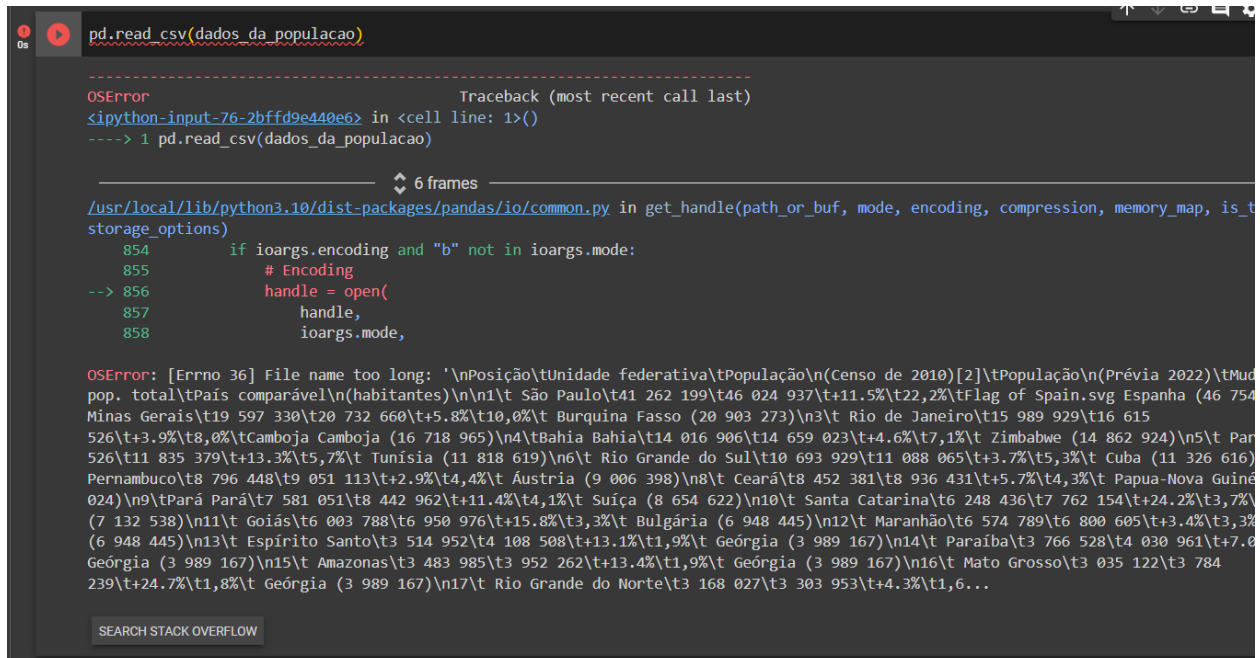
0s ▶ dados_da_populacao = ""
Posição Unidade federativa População
(Censo de 2010)[2] População
(Prévia 2022) Mudança % da pop. total País comparável
(habitantes)

1 São Paulo 41 262 199 46 024 937 +11.5% 22,2% Flag of Spain.svg Espanha (46 754 778)
2 Minas Gerais 19 597 330 20 732 660 +5.8% 10,0% Burquina Fasso (20 903 273)
3 Rio de Janeiro 15 989 929 16 615 526 +3.9% 8,0% Camboja Camboja (16 718 965)
4 Bahia Bahia 14 016 906 14 659 023 +4.6% 7,1% Zimbábue (14 862 924)
5 Paraná 10 444 526 11 835 379 +13.3% 5,7% Tunísia (11 818 619)
6 Rio Grande do Sul 10 693 929 11 088 065 +3.7% 5,3% Cuba (11 326 616)
7 Pernambuco 8 796 448 9 051 113 +2.9% 4,4% Áustria (9 006 398)
8 Ceará 8 452 381 8 936 431 +5.7% 4,3% Papua-Nova Guiné (8 947 024)
9 Pará Pará 7 581 051 8 442 962 +11.4% 4,1% Suíça (8 654 622)
10 Santa Catarina 6 248 436 7 762 154 +24.2% 3,7% Paraguai (7 132 538)
11 Goiás 6 003 788 6 950 976 +15.8% 3,3% Bulgária (6 948 445)
12 Maranhão 6 574 789 6 800 605 +3.4% 3,3% Bulgária (6 948 445)
13 Espírito Santo 3 514 952 4 108 508 +13.1% 1,9% Geórgia (3 989 167)
14 Paraíba 3 766 528 4 030 961 +7.0% 1,9% Geórgia (3 989 167)
15 Amazonas 3 483 985 3 952 262 +13.4% 1,9% Geórgia (3 989 167)
16 Mato Grosso 3 035 122 3 784 239 +24.7% 1,8% Geórgia (3 989 167)
17 Rio Grande do Norte 3 168 027 3 303 953 +4.3% 1,6% Uruguai (3 473 730)
18 Piauí 3 118 360 3 270 174 +4.9% 1,6% Bósnia e Herzegovina (3 280 819)
19 Alagoas 3 120 494 3 125 254 +0.2% 1,5% Mónaco (3 278 290)
20 Distrito Federal 2 570 160 2 923 369 +13.7% 1,4% Armênia (2 963 243)
21 Mato Grosso do Sul 2 449 024 2 833 742 +15.7% 1,4% Albânia (2 877 797)
22 Sergipe 2 068 017 2 211 868 +7.0% 1,1% Gabão (2 225 734)
23 Rondônia 1 562 409 1 616 379 +3.5% 0,8% Bahrein (1 701 575)
24 Tocantins 1 383 445 1 584 306 +14.5% 0,8% Guiné Equatorial (1 402 985)
25 Acre 733 559 829 780 +13.1% 0,4% Comores (869 601)

```

Se eu passar uma string para `read_csv`, o programa vai achar que é um arquivo local, não dá certo isso. Vai achar que é um identificador de requisição / recurso:

```
pd.read_csv(dados_da_populacao)
```



```
pd.read_csv(dados_da_populacao)

OSError                                Traceback (most recent call last)
<ipython-input-76-2bffd9e440e6> in <cell line: 1>()
----> 1 pd.read_csv(dados_da_populacao)

6 frames
/usr/local/lib/python3.10/dist-packages/pandas/io/common.py in get_handle(path_or_buf, mode, encoding, compression, memory_map, is_t
storage_options)
    854         if ioargs.encoding and "b" not in ioargs.mode:
    855             # Encoding
--> 856             handle = open(
    857                 handle,
    858                 ioargs.mode,

OSError: [Errno 36] File name too long: '\nPosição\tUnidade federativa\tPopulação\n(Censo de 2010)[2]\tPopulação\n(Prévia 2022)\tMud
pop. total\tPaís comparável\n(habitantes)\n\n1\t São Paulo\t41 262 199\t46 024 937\t+11.5%\t22,2%\tFlag of Spain.svg Espanha (46 754
Minas Gerais\t19 597 330\t20 732 660\t+5.8%\t10,0%\t Burquina Fasso (20 903 273)\n3\t Rio de Janeiro\t15 989 929\t16 615
526\t+3.9%\t8,0%\t Camboja Camboja (16 718 965)\n4\t Bahia Bahia\t14 016 906\t14 659 023\t+4.6%\t7,1%\t Zimbabwe (14 862 924)\n5\t Par
526\t11 835 379\t+13.3%\t5,7%\t Tunísia (11 818 619)\n6\t Rio Grande do Sul\t10 693 929\t11 088 065\t+3.7%\t5,3%\t Cuba (11 326 616)
Pernambuco\t8 796 448\t9 051 113\t+2.9%\t4,4%\t Áustria (9 006 398)\n8\t Ceará\t8 452 381\t8 936 431\t+5.7%\t4,3%\t Papua-Nova Guiné
024)\n9\t Pará Pará\t7 581 051\t8 442 962\t+11.4%\t4,1%\t Suíça (8 654 622)\n10\t Santa Catarina\t6 248 436\t7 762 154\t+24.2%\t3,7%\t
(7 132 538)\n11\t Goiás\t6 003 788\t6 950 976\t+15.8%\t3,3%\t Bulgária (6 948 445)\n12\t Maranhão\t6 574 789\t6 800 605\t+3.4%\t3,3%\t
(6 948 445)\n13\t Espírito Santo\t3 514 952\t4 108 508\t+13.1%\t1,9%\t Geórgia (3 989 167)\n14\t Paraíba\t3 766 528\t4 030 961\t+7.0
Geórgia (3 989 167)\n15\t Amazonas\t3 483 985\t3 952 262\t+13.4%\t1,9%\t Geórgia (3 989 167)\n16\t Mato Grosso\t3 035 122\t3 784
239\t+24.7%\t1,8%\t Geórgia (3 989 167)\n17\t Rio Grande do Norte\t3 168 027\t3 303 953\t+4.3%\t1,6...
```

Temos, portanto, que transformar essa string em algo de entrada e saída (IO), como se fosse a entrada de um arquivo. Isso existe no python, se chama StringIO! 😊 Você passa uma string e devolve um fluxo de entrada.

```
from io import StringIO

dados_da_populacao_io = StringIO(dados_da_populacao)

novos_dados = pd.read_csv(dados_da_populacao_io)

novos_dados.head()
```

Porém, ainda dará erro, porque nossa tabela toda não está sendo separada por vírgulas e sim por espaços né? Ela não está bem configurada. Vamos, portanto, colocar um separador.

```
novos_dados = pd.read_csv(dados_da_populacao_io, sep="\t")
```

```
[84] from io import StringIO
[85] dados_da_populacao_io = StringIO(dados_da_populacao)
[86] novos_dados = pd.read_csv(dados_da_populacao_io, sep="\t")

novos_dados.head()
```

	Posição	Unidade federativa	População	% da pop. total	País comparável
1	São Paulo	41 262 199	46 024 937	+11.5%	22,2% Flag of Spain.svg Espanha (46 754 778)
2	Minas Gerais	19 597 330	20 732 660	+5.8%	10,0% Burquina Fasso (20 903 273)
3	Rio de Janeiro	15 989 929	16 615 526	+3.9%	8,0% Camboja Camboja (16 718 965)
4	Bahia Bahia	14 016 906	14 659 023	+4.6%	7,1% Zimbábwe (14 862 924)
5	Paraná	10 444 526	11 835 379	+13.3%	5,7% Tunísia (11 818 619)

Agora, vamos ignorar todas as linhas que tenham Nan:

- ☐ `from io import StringIO`
- ☐ `dados_da_populacao_io = StringIO(dados_da_populacao)`
- ☐ `novos_dados = pd.read_csv(dados_da_populacao_io, sep="\t")`
- ☐ `novos_dados = novos_dados.dropna()`
- ☐ `novos_dados.head()`

DESAFIO 01: pesquisar a documentação da StringIO (e IO).

DESAFIO 02: limpar o excel do IBGE, assim como a tabela do Wikipedia, com nomes de colunas apropriados e linhas somente representando as unidades federativas.

DESAFIO 03: ler diretamente da wikipedia a versão html (ver isso na documentação do pandas).

Parte 2 | Fontes externas, limpeza e manipulação de dados II

Em vez de usar `read_html()`, também podemos usar `read_table()`! é uma outra possibilidade. 😊

Agora, podemos notar que há vários espaços entre os números né? Temos muitas coisas para corrigir / limpar na tabela novos_dados. Em vez de novos_dados, vou alterar o nome para populacao:

- ☐ `dados_da_populacao_io = StringIO(dados_da_populacao)`
- ☐ `populacao = pd.read_csv(dados_da_populacao_io, sep="\t")`
- ☐ `populacao = populacao.dropna()`
- ☐ `populacao.head()`

- ☐ `populacao.columns = ["posicao", "uf", "populacao", "porcentagem", "pais_comparavel"]` # aqui estamos mudando o nome das colunas porque no pandas não é uma boa prática trabalhar com colunas com espaço e, principalmente, trabalhar com colunas com acentos. ⚠

- ☐ `populacao.head()`

0s

```
1 populacao.columns = ["posicao", "uf", "populacao", "porcentagem", "pais_comparavel"]
2 populacao.head()
```

	posicao	uf	populacao	porcentagem	pais_comparavel
1	1	São Paulo	46 649 132	21,9%	Flag of Spain.svg Espanha (46 439 864)
2	2	Minas Gerais	21 411 923	10,1%	Sri Lanka (20 675 000)
3	3	Rio de Janeiro	17 463 349	8,2%	Países Baixos (16 922 900)
4	4	Bahia Bahia	14 985 284	7,1%	Chade (14 037 000)
5	5	Paraná	11 597 484	5,4%	Bolívia (11 410 651)

Agora, quero substituir esses espaços por nada. Quero apenas “devolver” esses espaços.

Vou pegar a coluna “populacao” e dar um `.replace(" ", "")`, trocando espaço vazio por espaço sem nada:

- ☐ `populacao.columns = ["posicao", "uf", "posicao", "porcentagem", "pais_comparavel"]`
- ☐ `populacao["populacao"] = populacao["populacao"].replace(" ", "")`
- ☐ `populacao.head()`

Isso não dá em nada, porque “replace” é uma função / método de Dataframe. Não queremos pegar uma célula de espaço em branco e substituir por uma célula sem valor, não é isso, queremos pegar o valor de uma string de uma célula e aplicar a função de string em Python nela. (?). Confuso.

Quero trabalhar com essa coluna “populacao” em python e transformá-la em string, basicamente.

Dado uma série, o `.str` disponibiliza diversas funções que estamos acostumados a aplicar em uma string pontualmente, como transformar em maiúsculo, minúsculo, trocar caracteres... tudo isso aplicamos em uma string para manipulá-la.

☐ `populacao["populacao"] = populacao["populacao"].str.replace(" ", "")`

```
1 populacao.columns = ["posicao", "uf", "populacao", "porcentagem", "pais_comparavel"]
2 populacao["populacao"] = populacao["populacao"].str.replace(" ", "")
3 populacao.head()
```

	posicao	uf	populacao	porcentagem	pais_comparavel
1	1	São Paulo	46649132	21,9%	Flag of Spain.svg Espanha (46 439 864)
2	2	Minas Gerais	21411923	10,1%	Sri Lanka (20 675 000)
3	3	Rio de Janeiro	17463349	8,2%	Países Baixos (16 922 900)
4	4	Bahia Bahia	14985284	7,1%	Chade (14 037 000)
5	5	Paraná	11597484	5,4%	Bolívia (11 410 651)

Agora sim, os espaços foram tirados. Mas não só isso, também quero **transformar em um inteiro**.

☐ `populacao.info()`


```
1 populacao.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 27 entries, 1 to 27
Data columns (total 5 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   posicao                27 non-null    object
1   uf                    27 non-null    object
2   populacao             27 non-null    object
3   porcentagem           27 non-null    object
4   pais_comparavel       27 non-null    object
dtypes: object(5)
memory usage: 1.3+ KB
```

Aqui, podemos ver que o campo “populacao” é **um objeto**, não é um inteiro.

```
populacao["populacao"] = populacao["populacao"].str.replace(" ", "").astype(int)
```

1 `populacao.describe()`

populacao	
count	27.00
mean	7900653.30
std	9380643.82
min	652713.00
25%	2966756.50
50%	4108508.00
75%	9457686.50
max	46649132.00

Agora sim, usando o `astype(int)`, transformamos esses valores em um inteiro. 😊 Podemos trocar para qualquer outro tipo, isso é muito importante!

DESAFIO 01: estudar o `read_table`.

DESAFIO 02: quais são os dtypes que o pandas possui?

Vamos continuar limpando ainda mais as colunas:

- ☐ `populacao.columns = ["posicao", "uf", "posicao", "porcentagem", "pais_comparavel"]`
- ☐ `populacao["populacao"] = populacao["populacao"].replace(" ", "")`
- ☐ `populacao = populacao[["uf", "populacao"]] # aqui`
- ☐ `populacao.head()`

DESAFIO 03: explorar 'str' do pandas.

Agora, vou conectar minha tabela de população com o último mês. A tabela "gastos_do_mais_recente" tem dados da unidade da federação e os gastos em milhões.

☐ `gastos_do_mais_recente.head()`

☐ `populacao.head()`



The image shows a Jupyter Notebook interface. On the left, there is a green checkmark and the text "0s". In the center, there is a play button icon. To the right of the play button, the following code is displayed:

```
1 gastos_do_mais_recente.head()  
2 populacao.head()
```

Below the code, there is a copy icon (two overlapping squares) and a table output. The table has two columns: "uf" and "populacao". The rows are numbered 1 to 5.

	uf	populacao
1	São Paulo	46649132
2	Minas Gerais	21411923
3	Rio de Janeiro	17463349
4	Bahia Bahia	14985284
5	Paraná	11597484

☐ `display(gastos_do_mais_recente.head())`

☐ `display(populacao.head())`

```
↳ Unidade da Federação
35 São Paulo          301.99
31 Minas Gerais       139.16
41 Paraná             91.19
43 Rio Grande do Sul  74.95
33 Rio de Janeiro     94.14
Name: 2021/Ago, dtype: float64
```

	uf	populacao
1	São Paulo	46649132
2	Minas Gerais	21411923
3	Rio de Janeiro	17463349
4	Bahia Bahia	14985284
5	Paraná	11597484

Usar `display()` é melhor nesse sentido de visualização. 😊

A tabela “`gastos_do_mais_recente`” e “`populacao`” têm coisas em comum: as “**unidades de federação**” e “`uf`”. Portanto, quero juntas uma coisa com a outra usando `.join()`.

```
↳ populacao.join(gastos_do_mais_recente)
```

	uf	populacao	2021/Ago
1	São Paulo	46649132	nan
2	Minas Gerais	21411923	nan
3	Rio de Janeiro	17463349	nan
4	Bahia Bahia	14985284	nan
5	Paraná	11597484	nan
6	Rio Grande do Sul	11466630	nan
7	Pernambuco	9674793	nan
8	Ceará	9240580	nan
9	Pará Pará	8777124	nan

Porém, não queremos esses valores de “nan” de 2021 de Agosto. Ele pegou do lado esquerdo (populacao) com o lado direito (gastos_do_mais_recente). Se não encontrou: “not a number”.

☐ `gastos_do_mais_recente.index.str[3:]`

```
1 gastos_do_mais_recente.index.str[3:]
```

```
Index(['São Paulo', 'Minas Gerais', 'Paraná', 'Rio Grande do Sul',
      'Rio de Janeiro', 'Bahia', 'Pernambuco', 'Santa Catarina', 'Ceará',
      'Goiás', 'Pará', 'Maranhão', 'Espírito Santo', 'Rio Grande do Norte',
      'Paraíba', 'Distrito Federal', 'Mato Grosso do Sul', 'Piauí', 'Alagoas',
      'Mato Grosso', 'Amazonas', 'Sergipe', 'Tocantins', 'Rondônia', 'Acre',
      'Roraima', 'Amapá'],
      dtype='object', name='Unidade da Federação')
```

Continua devolvendo pra gente o índice, mas agora do mais recente.

Sobrescrevendo...

☐ `gastos_do_mais_recente.index = gastos_do_mais_recente.index.str[3:]`

☐ `gastos_do_mais_recente.head()`

O join, por padrão, dá um join de índices, buscando índice em índice. Então, em `populacao.join`, está buscando o índice de população (1 2 3...) dentro de `gastos_do_mais_recente`, que é SP, MG...

Então, faltou fazer a criação desse código:

☐ `populacao.join(gastos_do_mais_recente)`

1 `populacao.join(gastos_do_mais_recente)`

	populacao	2021/Ago
uf		
São Paulo	46649132	nan
Minas Gerais	21411923	nan
Rio de Janeiro	17463349	nan
Bahia Bahia	14985284	nan
Paraná	11597484	nan
Rio Grande do Sul	11466630	nan
Pernambuco	9674793	nan
Ceará	9240580	nan
Pará Pará	8777124	nan
Santa Catarina	7338473	nan

Tem algo errado...

☐ `populacao.index`

```
1 populacao.index  
  
Index(['São Paulo', 'Minas Gerais', 'Rio de Janeiro', 'Bahia Bahia',  
      'Paraná', 'Rio Grande do Sul', 'Pernambuco', 'Ceará', 'Pará Pará',  
      'Santa Catarina', 'Goiás', 'Maranhão', 'Amazonas',  
      'Espírito Santo', 'Paraíba', 'Mato Grosso', 'Rio Grande do Norte',  
      'Alagoas', 'Piauí', 'Distrito Federal', 'Mato Grosso do Sul',  
      'Sergipe', 'Rondônia', 'Tocantins', 'Acre', 'Amapá', 'Roraima'],  
      dtype='object', name='uf')
```

Aqui conseguimos ver espaços em branco! É por isso.

- ☐ `populacao.index = populacao.index.str.strip()`
- ☐ `populacao.head`
- ☐ `populacao.join(gastos_do_mais_recente)`

Assim, tiraremos esses espaços em branco que não queremos, tanto na esquerda quanto na direita.

Ainda assim, tem algo errado. Vemos ali “Bahia Bahia” e “Pará Pará”. O problema está na imagem da bandeira no Wikipedia, porque é diferente das bandeiras dos demais estados. 😬 Porque são links, não imagens e o texto alternativo desse link é justamente o nome do estado, é uma questão de acessibilidade na web.

	populacao	2021/Ago
uf		
São Paulo	46649132	301.99
Minas Gerais	21411923	139.16
Rio de Janeiro	17463349	9
Bahia Bahia	14985284	nan
Paraná	11597484	91.19
Rio Grande do Sul	11466630	74.95
Pernambuco	9674793	71.63
Ceará	9240580	35.65
Pará Pará	8777124	nan
Santa Catarina	7338473	48.88
Goiás	7206589	44.94
Maranhão	7153262	25.78

Posição ↕	Unidade federativa ↕	População ↕	% da pop. total ↕	País comparável (habitantes) ↕
1	 São Paulo	46 649 132	21,9%	 Espanha (46 439 864)
2	 Minas Gerais	21 411 923	10,1%	 Sri Lanka (20 675 000)
3	 Rio de Janeiro	17 463 349	8,2%	 Países Baixos (16 922 900)
4	 Bahia	14 985 284	7,1%	 Chade (14 037 000)
5	 Paraná	11 597 484	5,4%	 Bolívia (11 410 651)
6	 Rio Grande do Sul	11 466 630	5,4%	 Bélgica (11 250 659)
7	 Pernambuco	9 674 793	4,5%	 Bielorrússia (9 485 300)
8	 Ceará	9 240 580	4,3%	 Emirados Árabes Unidos (9 157 000)
9	 Pará	8 777 124	4,1%	 Áustria (8 602 112)
10	 Santa Catarina	7 338 473	3,4%	 Sérvia (7 114 393)
11	 Goiás	7 206 589	3,4%	 Paraguai (7 003 406)
12	 Maranhão	7 153 262	3,4%	 Paraguai (7 003 406)
13	 Amazonas	4 269 995	2,0%	 Líbano (4 168 000)
14	 Espírito Santo	4 108 508	1,9%	 Líbano (4 168 000)
15	 Paraíba	4 059 905	1,9%	 Líbano (4 168 000)
16	 Mato Grosso	3 567 234	1,7%	 Uruguai (3 415 866)
17	 Rio Grande do Norte	3 560 903	1,7%	 Uruguai (3 415 866)
18	 Alagoas	3 365 351	1,6%	 Uruguai (3 415 866)
19	 Piauí	3 289 290	1,6%	 Kuwait (3 268 431)
20	 Distrito Federal	3 094 325	1,4%	 Lituânia (2 900 787)

Como resolver isso?

- ☐ `for estado in gastos_do_mais_recente.index:`
- ☐ `populacao.index = populacao.index.str.replace(f"{estado} {estado}", estado)`
- ☐ `populacao.index`
- ☐ `gastos_e_populacao = populacao.join(gastos_do_mais_recente)`

□ `gastos_e_populacao.head()`

DESAFIO 04: é possível fazer o processo de outra forma sem utilizar a palavra 'for'?