**FUNDAÇÃO GETULIO VARGAS**
**ESCOLA DE MATEMÁTICA APLICADA**

**GIANLUCCA DEVIGILI**

# USO DOS DADOS DO IBGE EM DOIS FORMATOS: UMA APLICAÇÃO COM DADOS DO CENSO DEMOGRÁFICO

Rio de Janeiro
2023

**GIANLUCCA DEVIGILI**

# USO DOS DADOS DO IBGE EM DOIS FORMATOS: UMA APLICAÇÃO COM DADOS DO CENSO DEMOGRÁFICO

Trabalho de conclusão de curso apresentada para a Escola de Matemática Aplicada (FGV/EMAp) como requisito para o grau de bacharel em Ciência de Dados e Inteligência Artificial.

Área de estudo: ciência de dados.

Orientador: Júlio César Chaves

Rio de Janeiro

2023

Ficha catalográfica elaborada pela BMHS/FGV

Devigili, Gianlucca
Uso dos dados do IBGE em dois formatos: uma aplicação com dados do censo demográfico: / Gianlucca Devigili. – 2023.
33f.

Trabalho de Conclusão de Curso – Escola de Matemática Aplicada.

Advisor: Júlio César Chaves.
Includes bibliography.

1. Matemática 2. Aplicada 3. na matemática I. Chaves, Júlio César II. Escola de Matemática Aplicada III. Uso dos dados do IBGE em dois formatos: uma aplicação com dados do censo demográfico

**GIANLUCCA DEVIGILI**

# USO DOS DADOS DO IBGE EM DOIS FORMATOS: UMA APLICAÇÃO COM DADOS DO CENSO DEMOGRÁFICO:

Trabalho de conclusão de curso apresentada para a Escola de Matemática Aplicada (FGV/EMAp) como requisito para o grau de bacharel em Ciência de Dados e Inteligência Artificial.

Área de estudo: ciência de dados.

E aprovado em 07/12/2023
Pela comissão organizadora

---

Júlio César Chaves
Escola de Matemática Aplicada da Fundação
Getulio Vargas - EMAp/FGV

---

Rodolpho Guedon Tobler
Instituto Brasileiro de Economia da
Fundação Getulio Vargas - IBRE/FGV

---

Andrea Diniz da Silva
Escola Nacional de Ciências Estatísticas do
Instituto Brasileiro de Geografia e
Estatística - ENCE/IBGE

Dedico esta dissertação aos meus pais, Ricardo e Karina, sem vocês eu não teria chegado até aqui.

# Agradecimentos

Aos meus pais, Ricardo e Karina, por todo o apoio e incentivo desde criança, estando presentes em cada uma das minhas conquistas. Agradeço também meu orientador, professor Júlio, pelo auxílio e pelos conselhos ao longo de minha jornada acadêmica e profissional, tanto na iniciação científica quanto no desenvolvimento do presente trabalho. Ao CDMC, por ter me proporcionado a possibilidade de estudar na FGV. Por fim, agradeço todos os amigos e professores que de alguma forma fizeram parte de minha vida.

*"So once you do know what the question actually is, you'll know what the answer means."*

*Douglas Adams*

# Resumo

O presente trabalho objetiva melhorar o acesso aos dados públicos, disponibilizando *datasets* coletados através de duas diferentes interfaces de dados do Instituto Brasileiro de Geografia e Estatística (IBGE): a API do serviço de dados e os microdados dos Censos Demográficos. Utilizando a API, foram coletados dados referentes às localidades, indicadores socioeconômicos de diversos países, os chamados agregados e seus metadados. O último grupo consiste da consolidação de respostas das pesquisas promovidas pelo instituto de acordo com diversos critérios, variáveis as quais são armazenadas em arquivos de microdados. Estes arquivos contém de forma anonimizada as respostas individuais de cada entrevistado durante a pesquisa, permitindo análises mais detalhadas do que os agregados. Durante o trabalho, foram tratados de dados do censo demográfico brasileiro, contudo os códigos desenvolvidos são genéricos e podem ser utilizados em outras pesquisas do IBGE desde que sigam o mesmo formato possuam os mesmos dados. Também foi realizado um estudo de caso, no qual foi exemplificado o uso de ambos os tipos de dados e comparando eles através de visualizações.

Palavras-chave: Censo Demográfico. IBGE. Microdados. Dados públicos.

# Abstract

This work aims to improve public data access, making available datasets collected from two different sources from Brazilian Institute of Geography and Statistics (IBGE): the data service API and the microdata from the brazilian demographic census. Using the API, data related to locations, socioeconomic indicators from various countries, the so-called aggregations, and their metadata were collected. The last group consists of the consolidation of many variables from surveys made by the institute according to diverse criteria, whose variables are stored in text files called microdata files. These files anonymize individual answers from each of the intervewee that took part of the census and they allow their user to make more detailed data analysis than they could if using the aggregates. Throughout the project, Brazilian demographic census data were processed and treated; however, the codes are generic and can be used in other IBGE surveys as long as they follow the same format and have the same files. Also, there were made data visualizations in order to exemplify and compare both datasets.

Keywords: Brazilian demographic census. IBGE. microdata. Public data.

# Lista de ilustrações

# Lista de tabelas

# Lista de abreviaturas e siglas

| | |
|---|---|
| API | *Application Programming Interface* (Interface de programação de aplicações) |
| ASCII | *American Standard Code for Information Interchange* (Código Padrão Americano para o Intercâmbio de Informação) |
| CSV | *Comma separated values* |
| HTTP | *Hypertext Transfer Protocol* |
| IBGE | Instituto Brasileiro de Geografia e Estatística |
| ID | Identificador |
| IPUMS | *Integrated Public Use Microdata Series* |
| JSON | *JavaScript Object Notation* (Notação de Objeto JavaScript) |
| MB | *Megabyte* |
| NoSQL | *Not Only Structured Query Language* |
| ODS | *Open Document Spreadsheet* |
| ONU | Organização das Nações Unidas |
| PNAD | Pesquisa Nacional por Amostra de Domicílios |
| SC | Estado de Santa Catarina |
| SSL | *Secure Sockets Layer* |
| TSV | *Tab separated values* |
| UF | Unidade Federativa ou Unidade da Federação |
| URL | *Uniform Resource Locator* (localizador uniforme de recursos) |

# Sumário

# 1 Introdução

No cenário altamente digitalizado no qual a humanidade se insere atualmente, um volume gigantesco de informação é gerado diariamente e a análise de dados é cada vez mais uma ferramenta crucial para inúmeras atividades dos mais diversos setores da sociedade. Por exemplo, empresas utilizam os dados para a tomada de decisões, análises de mercado e criação de estratégias de atuação; governos se baseiam neles para a aplicação de políticas públicas e direcionamento de verbas; pesquisadores para embasar suas pesquisas e programadores para treinar modelos de aprendizado de máquina. Em suma, a análise de dados auxilia indivíduos e organizações, gerando *insights*, conhecimento e valor para eles, servindo como base para responder as mais diversas perguntas.

Contudo, a abundância de dados não implica em sua qualidade e acessibilidade; pelo contrário, frequentemente, o caminho entre a informação e a geração de conhecimento a partir dela é longo e complexo e, raramente, se encontra um conjunto de dados prontos para a análise e uso imediato. Além disso, documentação escassa ou até mesmo inexistente e a falta de metadados é uma realidade comum, que acaba por gerar diversas tentativas frustradas de se utilizarem *datasets* com o intuito de responder uma pergunta e percebe-se, apenas após considerável esforço, que o conjunto de dados que se tem em mãos não será capaz de proporcionar valor.

Outro problema comum é quanto à estrutura de armazenamento: Bancos de dados transacionais ou *Not Only Structured Query Language* (NoSQL) tem um formato focado em funcionalidade, velocidade de transação e normalização, contudo estes formatos geralmente não atendem a necessidade de analistas e cientistas de dados, já que nesses casos é melhor o sacrifício da economia em espaço de armazenamento em prol da simplificação e eficiência de consulta devido ao seu volume maior. E ainda, às vezes a estrutura na qual os dados são armazenados é ainda mais inóspita que um *dataset* "sujo" ou um modelo com dezenas de tabelas, fazendo com que o usuário deles seja obrigado à ter conhecimento técnico e realizar um trabalho que às vezes dura meses apenas para conseguir iniciar a sua pesquisa.

Tendo em vista as possíveis dificuldades ao se deparar com um problema de dados, bem como a importância dos dados públicos para os mais diversos tipos de usuário, o presente trabalho objetiva a exploração e tratamento de duas diferentes formas de coleta de dados do Instituto Brasileiro de Geografia e Estatística (IBGE): A *Application Programming Interface* (API) de serviço de dados (IBGE, 2017) e os microdados, que representam os dados das respostas de cada um dos entrevistados, facilitando o estudo dessas bases e possibilitando um encurtamento do caminho entre a informação e a geração de conhecimento.

## 1.1 O formato dos dados do IBGE

Os dados do Instituto Brasileiro de Geografia e Estatística são disponibilizados em dois formatos: os microdados e os agregados que, em suma, representam a mesma informação, diferindo apenas em termos de granularidade.

Os microdados[1] "consistem no menor nível de desagregação dos dados de uma pesquisa, retratando, sob a forma de códigos numéricos, o conteúdo dos questionários, preservado o sigilo estatístico com vistas à não individualização das informações." (IBGE, 201-?). Cada linha de um arquivo de microdados representa o conjunto de respostas de um único entrevistado dentro de uma determinada pesquisa, bem como informações calculadas à partir do que foi amostrado, como renda *per capita* e valor do aluguel em salários mínimos, por exemplo.

Por sua vez, os agregados são agrupamentos desses microdados de acordo com determinados critérios. A *Application Programming Interface* (API) do serviço de dados do IBGE[2] disponibiliza cada variável consolidada de acordo com a pesquisa realizada e respectiva periodicidade, além de agregações de caráter geográfico, indo desde grande região (p. ex. Norte, Sul, *etc.*) até o grão de município ou distrito municipal, quando aplicável.

Em termos geográficos, os agregados alcançam apenas até o nível de município, enquanto os microdados chegam ao grão de setor censitário, sendo descritos por Encarnação (2010) no Guia do Censo como "unidades territoriais estabelecidas para fins de controle cadastral, situadas em um único quadro urbano ou rural, com dimensão e número de domicílios [...]".

Nesse contexto, a principal distinção dos dois formatos é o seu nível de detalhe e sua principal vantagem e desvantagem residem no *tradeoff* entre detalhamento e volume. Os microdados, por apresentarem uma granularidade menor, permitem análises mais específicas porém requisitando um maior poder computacional e ocupando um maior espaço de armazenamento. Enquanto isso, os dados agregados já se encontram pré-processados de acordo com seus respectivos pesos amostrais e são mais leves em termos de memória e armazenamento, contudo reduzindo as possibilidades de análise conforme a agregação aumenta.

O repositório contendo os arquivos de código utilizados para o desenvolvimento do trabalho, bem como os dados preparados, se encontra em: <github.com/GDevigili/TCC-IBGE>.

[1] Disponíveis em: https://anda.ibge.gov.br/estatisticas/sociais/populacao/22827-censo-demografico-2022.html?edicao=37415&t=microdados

[2] Disponível em <https://servicodados.ibge.gov.br/api/docs/>. Acessado em 30 de set. de 2023.

# 2 Objetivo Geral

Tendo em vista a importância dos dados públicos, em especial os do censo demográfico realizado pelo IBGE, o presente trabalho objetiva facilitar o acesso aos dados censitários através da criação de um repositório público de dados contendo os conjuntos de dados devidamente tratados e transformados em formato tabular.

## 2.1 Objetivos Específicos

- Estudar diferentes estratégias de acesso aos dados censitários do IBGE;
- Carregar os dados do IBGE via API e processá-los de forma a reestruturar eles em formato tabular;
- Codificar um *script* capaz de ler e associar *labels* aos respectivos microdados de modo a gerar um arquivo de dados do *Stata*;
- Demonstrar o uso de ambos conjuntos de dados através de um estudo de caso, demonstrando as diferenças de abrangência e utilização dos dois formatos estudados (API e microdados), gerando visualizações e comparando os dois conjuntos.

# 3 Métodos

Como discutido anteriormente, as diferentes formas nas quais os dados censitários são disponibilizados se diferem basicamente quanto ao seu grão e estrutura, sendo os agregados disponibilizados via API na notação *Javascript Object Notation* (JSON) e os microdados em arquivos de texto em um formato compactado que será explicado em detalhes na seção 3.2.

Desta forma, duas diferentes abordagens foram adotadas no decorrer do desenvolvimento do trabalho: a de coletar os dados através da API e a de baixar os arquivos de microdados do IBGE e processá-los para então criar os conjuntos de dados disponíveis no repositório.

O código utilizado para a carga e tratamento dos dados, bem como os arquivos de dados criados, podem ser encontrados no repositório do Github <github.com/GDevigili/TCC-IBGE>, incluindo o *script* que automatiza a criação do DoFile e o que tranforma os resultados da requisição à API em dados tabulares.

## 3.1 APIs do serviço de dados do IBGE

O IBGE oferece ao todo 17 APIs em seu serviço de dados, incluindo a de Agregados, citada anteriormente, a de Localidades, que inclui os códigos dos vários níveis de localidade (p. ex. unidade federativa (UF), macrorregião, município), a de Metadados, incluindo informações como periodicidade e variáveis disponíveis em cada uma das pesquisas disponíveis na API, entre outras.

Nessas APIs é possível, através de *Uniform Resource Locators* (URLs) gerar requisições por meio de bibliotecas como a *requests* do *python* para obter os dados da API no formato JSON. A construção dessas URLs pode ser auxiliada pela ferramenta de *Querie Builder* integrada à plataforma. Por exemplo, ao fazer uma requisição à API de localidades utilizando o caminho `servicodados.ibge.gov.br/api/v1/localidades/distritos` obteremos uma lista de valores no formato exemplificado em *Listing* 1:

Por mais que a formatação JSON tenha uma praticidade maior do que dados em formato tabular, analisando por um ponto de vista transacional ou orientado à objetos, ele não é ideal para o uso analítico por conta de uma complexidade intrínseca para a navegação pelos dados, necessitando de um certo trabalho para a interpretação dos mesmos.

Dado que normalização não é uma prioridade quando se trata de análise, sendo mais importante a simplicidade na hora de se criarem consultas, foi aplicado à *string* de resultados da requisição um processo de *unnesting* (em português desaninhamento), cujo

```
{"id":421400310, "nome":"Mirador",
  "municipio":{"id":4214003,"nome":"Presidente Getúlio",
    "microrregiao":{"id":42011,"nome":"Rio do Sul",
      "mesorregiao":{"id":4204,"nome":"Vale do Itajaí",
        "UF":{"id":42, "sigla":"SC","nome":"Santa Catarina",
          "regiao":{"id":4, "sigla":"S", "nome":"Sul"}}}},
    "regiao-imediata":{"id":420023, "nome":"Ibirama - Presidente
       Getúlio",
      "regiao-intermediaria":{"id":4207, "nome":"Blumenau",
        "regiao":{"id":4, "sigla":"S","nome":"Sul"}}}}
}
```

Listing 1 – Exemplo de resultado de uma requisição da API de localidades.

algoritmo se encontra no *listing* 3 do apêndice A. Tal processo é o oposto do conceito conhecido por *nesting* e consiste de, recursivamente, subir o nível de aninhamento (*nesting-level*) de uma entrada, para o nível imediatamente superior, até que todos os valores estejam alinhados em um único nível.

Por exemplo, aplicando o algoritmo de *unnesting* juntamente com a função `make_df()` (*listing* A.4), gera um *DataFrame* como a tabela 1, com todos os dados em um mesmo *nesting-level*.

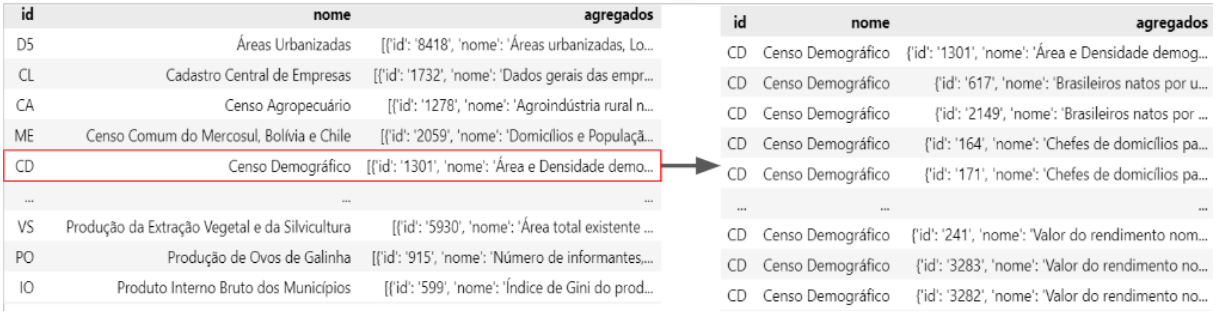
| id-distrito | nome-distrito | id-municipio | nome-municipio | ··· | nome-regiao |
|---|---|---|---|---|---|
| 421400310 | Mirador | 4214003 | Presidente Getúlio | ··· | Sul |
| 420690010 | Dalbérgia | 4206900 | Ibirama | ··· | Sul |

Tabela 1 – Exemplo dos dados de localidade em formato tabular. Fonte: Censo Demográfico, IBGE (2010).

Um problema que pode ainda ocorrer é que, mesmo após o *unnesting*, os dados em JSON podem conter também elementos em formato de lista, gerando assim uma coluna na tabela de resultado que possui uma lista de valores. Portanto se faz necessário um segundo tratamento para estes casos, que é facilmente resolvido armazenando o conjunto de dados dentro de um objeto do tipo *DataFrame* da biblioteca *pandas* da linguagem *python* e utilizar o método explode()[1] de modo a converter elementos *list-like* que possam existir dentro das colunas em linhas, copiando os demais valores, como é exemplificado pela figura 1.

Nota-se que os valores "explodidos" continuam aninhados, portanto nesse caso foi necessário reaplicar a função unnest_json() de modo a separar adequadamente os dados. O

[1] Documentação disponível em <pandas.pydata.org/docs/reference/api/pandas.DataFrame.explode.html>, código fonte disponível em <github.com/pandas-dev/pandas/blob/v2.1.1/pandas/core/frame.py#L9432-L9558>. Acessado em 30 de set. 2023.



| id | nome | agregados |
|---|---|---|
| D5 | Áreas Urbanizadas | [{'id': '8418', 'nome': 'Áreas urbanizadas, Lo... |
| CL | Cadastro Central de Empresas | [{'id': '1732', 'nome': 'Dados gerais das empr... |
| CA | Censo Agropecuário | [{'id': '1278', 'nome': 'Agroindústria rural n... |
| ME | Censo Comum do Mercosul, Bolívia e Chile | [{'id': '2059', 'nome': 'Domicílios e Populaçã... |
| CD | Censo Demográfico | [{'id': '1301', 'nome': 'Área e Densidade demo... |
| ... | ... | ... |
| VS | Produção da Extração Vegetal e da Silvicultura | [{'id': '5930', 'nome': 'Área total existente ... |
| PO | Produção de Ovos de Galinha | [{'id': '915', 'nome': 'Número de informantes,... |
| IO | Produto Interno Bruto dos Municípios | [{'id': '599', 'nome': 'Índice de Gini do prod... |

| id | nome | agregados |
|---|---|---|
| CD | Censo Demográfico | {'id': '1301', 'nome': 'Área e Densidade demog... |
| CD | Censo Demográfico | {'id': '617', 'nome': 'Brasileiros natos por u... |
| CD | Censo Demográfico | {'id': '2149', 'nome': 'Brasileiros natos por ... |
| CD | Censo Demográfico | {'id': '164', 'nome': 'Chefes de domicílios pa... |
| CD | Censo Demográfico | {'id': '171', 'nome': 'Chefes de domicílios pa... |
| ... | ... | ... |
| CD | Censo Demográfico | {'id': '241', 'nome': 'Valor do rendimento nom... |
| CD | Censo Demográfico | {'id': '3283', 'nome': 'Valor do rendimento no... |
| CD | Censo Demográfico | {'id': '3282', 'nome': 'Valor do rendimento no... |

Figura 1 – Exemplo do uso do método pandas.DataFrame().explode(). Fonte: Censo Demográfico, IBGE (2010).

processo completo é uma sequência de aplicações dessas duas funções conforme aparecem listas e dicionários, até ser obtido um *DataFrame* completamente desaninhado.

Das 17 APIs disponíveis, foram carregados dados das APIs de localidades, países, agregados e metadados. A API de localidades retorna as informações dos identificadores geográficos do país definidos pelo IBGE, utilizados para identificar a área na qual foram coletados. Esse serviço de dados possui ao todo quatro possíveis requisições nas quais os dados são separados por distritos, subdistritos, região metropolitana e região integrada de desenvolvimento. Os conjuntos de dados derivados podem ser juntados entre si através do identificador mais específico presentes nelas, sendo o ID do distrito nas duas primeiras e o id do município nas outras duas.

Contendo 34 indicadores socioeconômicos dos 193 países membros da Organização das Nações Unidas (ONU) a API de países é a única que não utiliza apenas dados coletados pelo próprio IBGE, trazendo bases do banco de dados *Integrated Public Use Microdata Series* (IPUMS). Como uma requisição comum solicita a especificação dos países e/ou das variáveis desejadas, a estratégia adotada foi primeiramente obter todos os códigos distintos de países para posteriormente realizar uma requisição para cada país contendo todos os possíveis indicadores e concatenando todos os resultados em um único conjunto de dados. O código completo pode ser encontrado no *listing* A.6.

A carga mais elaborada foi a de agregados cuja API totaliza 8442 agregados de 68 pesquisas, cada qual contendo diversas variáveis e séries temporais. Para o projeto foi apenas utilizado o Censo, porém o código desenvolvido é genérico e pode ser aproveitado para transformar dados de qualquer uma das pesquisas disponibilizadas no serviço de dados. Primeiramente foram obtidos os identificadores dos agregados para que posteriormente fossem usados como parâmetro na hora de requisitar os metadados de cada pesquisa. Tendo os dados de nível geográfico (vindo dos metadados) e o ID do agregado foi então possível usar a URL 'https://servicodados.ibge.gov.br/api/v3/agregados/ {agregado}/variaveis?localidades={nivel}[all]', onde `{agregado}` e `{nivel}` são parâmetros passados conforme os metadados obtidos.

O resultado da requisição acima, transformado em *DataFrame* é o mostrado na

figura 2. A coluna mais importante é de resultados, que irá conter duas listas de valores: classificações, contendo metadados acerca da variável, e, novamente, resultados, que por sua vez também contem listas, mas dessa vez com as séries dos valores numéricos.

Aplica-se então o método `explode()` ao *DataFrame* até termos os valores das séries distribuídos em uma coluna por ano, que, para normalizar a tabela, é executado o método `melt()` para criar uma coluna de período e outra de valor. O resultado final é um conjunto de dados com as colunas variável, unidade, id, nome e nível geográfico da localidade (p. ex. Município, UF, *etc.*), ano e valor.

| | id | variavel | unidade | resultados |
|---|---|---|---|---|
| 0 | 1488 | Mulheres de 10 anos ou mais de idade que vivia... | Pessoas | [{'classificacoes': [{'id': '1', 'nome': 'Situ... |
| 1 | 1489 | Mulheres de 10 anos ou mais de idade que vivia... | Pessoas | [{'classificacoes': [{'id': '1', 'nome': 'Situ... |
| 2 | 1514 | Filhos tidos pelas mulheres de 10 anos ou mais... | Pessoas | [{'classificacoes': [{'id': '1', 'nome': 'Situ... |
| 3 | 1515 | Filhos tidos nascidos vivos pelas mulheres de ... | Pessoas | [{'classificacoes': [{'id': '1', 'nome': 'Situ... |
| 4 | 1525 | Filhos tidos nascidos vivos pelas mulheres de ... | Pessoas | [{'classificacoes': [{'id': '1', 'nome': 'Situ... |
| 5 | 1526 | Filhos tidos nascidos mortos pelas mulheres de... | Pessoas | [{'classificacoes': [{'id': '1', 'nome': 'Situ... |
| 6 | 1527 | Filhos tidos pelas mulheres de 10 anos ou mais... | Pessoas | [{'classificacoes': [{'id': '1', 'nome': 'Situ... |

Figura 2 – Tabela de dados após primeira requisição para a carga dos agregados. Fonte: Censo Demográfico, IBGE (2010).

## 3.2 Microdados do Censo Demográfico

Os dados fornecidos pela API de agregados consistem essencialmente na consolidação das respostas individuais de cada um dos entrevistados durante o processo de recenseamento. Esses agrupamentos são calculados com base nos microdados, que estão disponíveis para *download* no portal de produtos estatísticos da instituição[2], estando armazenados em diversas pastas compactadas em formato `.zip` que contém os arquivos de texto nos quais se encontram os dados. No entanto, arquivos não estão imediatamente prontos para a utilização tal qual uma planilha ou um arquivo de valores separados por vírgulas (em inglês *comma separated values* ou CSV), mas sim em um formato comprimido, onde os valores categóricos são codificados através de identificadores (IDs) numéricos, enquanto valores que originalmente possuíam casas decimais tem seus pontos flutuantes removidos. Dessa forma, os dados se apresentam conforme ilustrado pela figura 3.

Juntamente com os microdados, é fornecido uma planilha no formato *Open Document Spreadsheet* (ODS) ou Excel que descreve o que cada caractere do arquivo significa, relacionando as categorias e identificadores, assim como definindo a formatação referente às casas decimais das variáveis numéricas.

[2] Para mais detalhes, consulte <www.ibge.gov.br/estatisticas/todos-os-produtos-estatisticas.html>. Acesso em 03 out. 2023

src > data > microdados > amostra_domicilios_2010_SC.txt

```
1   4200051420005100100100000745003486869475141940300900201115  3060030
2   4200051420005100100100029472003523739097269840300900201111  3060030
3   4200051420005100100100049362002259614086741440300900201111  1070040
4   4200051420005100100100072173003709996388955640300900101111  3050040
5   4200051420005100100100073380003724660209393340300900201111  1080030
6   4200051420005100100100076708004465496693169140300900101111  3050060
7   4200051420005100100100091389003724660209393340300900201111  1090020
8   4200051420005100100100094224004702878043781240300900101111  1070070
9   4200051420005100100100102234002906904385943840300900101111  1070040
10  4200051420005100100100112068002776637222533540300900201111  3090030
```

Figura 3 – Exemplo de microdados: primeiras 10 linhas da pesquisa de Domicílios do Censo de 2010 no estado de Santa Catarina. Fonte: Dados do Autor (2023)

Tomando como exemplo a amostra da pesquisa de domicílios do Censo Demográfico de 2010, o arquivo de *layout* [3] (exemplificado na figura 4), temos que as duas primeiras posições do arquivo referem-se variável V0001 à UF, cujo valor 42 corresponde ao Estado de Santa Catarina. Outro exemplo significativo é a variável numérica V0010, Peso Amostral, engloba os dígitos da posição 29 até 44, sendo os 3 primeiros (coluna INT) os dígitos anteriores ao ponto decimal, e as 13 subsequentes sendo as casas de precisão após a vírgula (coluna DEC).

| VAR | NOME | POSIÇÃO INICIAL | POSIÇÃO FINAL | INT | DEC | TIPO |
|---|---|---|---|---|---|---|
| V0001 | UNIDADE DA FEDERAÇÃO:<br>41- Paraná<br>42- Santa Catarina<br>43- Rio Grande do Sul<br>50- Mato Grosso do Sul | 1 | 2 | 2 | | A |
| V0002 | CÓDIGO DO MUNICÍPIO | 3 | 7 | 5 | | A |
| V0010 | PESO AMOSTRAL | 29 | 44 | 3 | 13 | N |
| V1006 | SITUAÇÃO DO DOMICÍLIO:<br>1- Urbana<br>2- Rural | 53 | 53 | 1 | | C |

Figura 4 – Recorte da planilha de *layout* da pesquisa de domicílios do Censo Demográfico de 2010. Fonte: Censo Demográfico, IBGE (2010)

Variáveis padrões contidas nas pesquisas são as referentes à localização geográfica como município, UF e área de ponderação, assim como o peso amostral, que por sua vez é uma medida de representatividade daquela resposta dentro do escopo da pesquisa.

Os arquivos de microdados ficam disponíveis através da URL <www.ibge.gov.br/estatisticas/sociais/trabalho/22827-censo-demografico-2022.html?edicao=37225&t=microdados>, separados por Estado e por ano, hoje[4] estando disponíveis os microdados Censos de 2000 e 2010 no *website*. Com o intuito de facilitar o processo de

[3] Arquivo /Documentação/Layout/Layout_microdados_amostra.xls. Download em: <ftp.ibge.gov.br/Censos/Censo_Demografico_2010/Resultados_Gerais_da_Amostra/Microdados/Documentacao.zip>. Acesso em 03 out. 2023.

[4] Considerando o último acesso em novembro de 2023.

baixar os arquivos, foi codificado um *script python*[5] para tal, que realiza o *download* de todos os arquivos `.zip`, descompacta eles e então os renomeia, de modo a padronizar os nomes dos arquivos de ambas pesquisas e separar as diferentes amostras (questionários de domicílios, pessoas, mortalidade e emigração), além de separar arquivos de dados dos arquivos auxiliares que estão inclusos nos diretórios compactados. Vale ressaltar que os arquivos de microdados não processados não foram incluídos no repositório por conta do limite máximo de 100 megabytes (MB) que o *Github* impõe para os arquivos.

Após extraídos os `zip`s, tem-se uma coleção de arquivos de texto (de extensão `.txt`) contendo os microdados codificados como na figura 3. Para transformá-los em dados utilizáveis, os microdados precisam ser associados aos respectivos valores, descritos no arquivo de *layout*.

Inicialmente, tentou-se fazer tal "tradução" via *python*[6], manualmente programando tal a associação dos IDs com suas respectivas *labels*. Primeiramente destrinchando o arquivo de descrição das variáveis em pares chave-valor e dividindo os números dos microdados em colunas de acordo com as respectivas posições inicial e final, armazenando ambos os resultados deste pré-processamento em `DataFrame` da biblioteca `pandas`. Tendo em mãos os objetos necessários, foi iterado através de cada linha e coluna, substituindo os identificadores por suas respectivas *labels*.

Porém, o custo computacional de processar os dados dessa forma, ainda mais utilizando uma linguagem *python*, conhecida por um tempo de execução elevado, além da presença de *bugs* no código devido ao funcionamento de algumas estruturas de dados da `pandas`, optou-se por realizar o mesmo trabalho utilizando o *software Stata* (em sua versão 18.0), que possui funções já implementadas para o processamento de arquivos como os do IBGE.

Com comandos da linguagem própria do *Stata*, é possível fazer a mesma associação de identificadores e labels com alguns poucos comandos, para então gerar um arquivo de dados de extenção `.dta` que posteriormente poderá ser utilizado como *dataset*. Por exemplo, no *Listing* 2 definimos a posição das variáveis, sua tipagem, respectivas *labels* e possíveis valores, nos casos de colunas categóricas, ou formatação, em caso de colunas numéricas.

O código exemplificado é um pequeno recorte do arquivo[7] completo da pesquisa de domicílios, que conta com 76 variáveis e mais de 6 mil *labels*, em sua maioria referentes à localização geográfica da coleta da amostra, já que estão inclusos mais de 5 mil municípios

[5] Código completo no arquivo download-arquivos-microdados.ipynb, disponível em: <github.com/GDevigili/TCC-IBGE/blob/main/src/notebooks/download-arquivos-microdados.ipynb>

[6] Código completo no arquivo translate-microdata.ipynb, disponível em <github.com/GDevigili/TCC-IBGE/blob/main/src/notebooks/old-notebooks/2-translate-microdata.ipynb>

[7] Arquivo amostra_domicilios_2010.do, disponível em <https://github.com/GDevigili/TCC-IBGE/blob/main/src/do-files/amostra_domicilios_2010.do>

```
* Define as posições onde se encontram cada dado
quietly infix                  ///
  byte     V1006    53-53      ///
  double     V6204    82-84    ///
using `"amostra_domicilios_2010_RJ.txt"', clear
* Define a formatação dos dados numéricos
* Neste caso,  o dado deve assumir a formatação 0000.0
format V6204 %04.1f
* Define as labels para os dados
label var V1006    `"SITUAÇÃO DO DOMICÍLIO"'
label var V6204    `"DENSIDADE DE MORADOR / DORMITÓRIO  "'
* Define os pares chave-valor para cada variável
label define V1006_lbl 1 `" Urbana"', add
label define V1006_lbl 2 `" Rural"', add
* Associa os valores da variável V1006 com os pares chave-valor
   armazenados em V1006_lbl
label values V1006 V1006_lbl
```

Listing 2 – Exemplo de comandos Stata utilizados para"traduzir" os microdados.

e 500 microrregiões.

Para não ser necessário definir manualmente cada uma das variáveis, aproveitou-se parte do código que seria descartada na tentativa de realizar a "tradução" dos microdados via *python* e se utilizou do processamento da tabela de *layout* para gerar um *DataFrame* contendo os pares chave-valor utilizados para mapear via comando `label define` as possíveis *labels* de cada uma das colunas utilizando um *script* que itera sobre a tabela processada.

Além disso, como os códigos de município, micro e mesorregião ficam em planilhas auxiliares, optou-se por utilizar dos identificadores de localidade coletados via API para associar os respectivos códigos no *Dofile* que seria executado no *Stata* para gerar o conjunto de dados. A única transformação necessária foi o uso de uma coluna auxiliar que juntava o código da UF com os códigos de município e regiões, já que a API utiliza chaves concatenadas como identificadores.

Após gerado o *Dofile* através do *script python* descrito acima, só é necessário carregar o arquivo no *Stata* e executá-lo, por fim salvando o novo arquivo de dados. Caso aberto através de um editor de texto, o arquivo `.dta` se parece com o arquivo de microdados, porém carrega as configurações de *label* e formatação que foram definidas anteriormente. Para a leitura do arquivo em um amiente diferente do Stata, é necessário um leitor específico. Além disso é possível a conversão do `.dta` para formatos mais comuns de armazenamento de dados, como *Excel* ou arquivos de texto com delimitadores como *Comma* ou *Tab separated values* (CSV e TSV respectivamente).

# 4 Estudo de Caso: Acesso à água encanada no Estado do Rio de Janeiro

Exemplificando um caso de uso dos conjuntos de dados estudados, foi realizada uma análise da distribuição de água encanada no Estado do Rio de Janeiro.

Para o carregamento do arquivo de dados do *Stata* no *python*, foi necessário abrir os arquivos na forma de um objeto do tipo `StataReader` que permite o acesso às *labels* definidas na execução do *Dofile* para posterior aplicação no conjunto de dados. A carga completa se encontra no *listing* 7 do apêndice A.

A variável utilizada para o cálculo do valor foi a coluna "V0209: Abastecimento de Água, Canalização" da amostra de domicílios do Censo de 2010. Foram consideradas no cálculo do percentual as respostas "Sim, em pelo menos um cômodo" e "Sim, só na propriedade ou terreno" e a ponderação da representatividade da resposta foi dada através da coluna de peso amostral (V0010). Por se tratarem de dados com uma média de 98.74% e mediana em 99.54%, uma escala contínua deixaria o mapa gerado de difícil interpretação, então foi optado por subdividir os dados em 5 categorias: Menor que 80%, entre 80% e 90%, entre 90% e 95%, entre 95% e 98% e entre 98% e 100%.

Para a criação de uma visualização em forma de mapa (Figura 5) referente aos dados, foi utilizado um arquivo *Shapefile*[1], um arquivos vetorial que armazena forma e atributos de localizações geográficas (ARCGIS, 2022). Através a biblioteca *GeoPandas* é possível manipular esse conjunto de dados geográficos, juntá-los com outros *datasets* e também gerar gráficos. Acrescentando as porcentagens calculadas com os microdados ao *dataframe*, foi utilizado o método `plot()` com algumas configurações comuns da biblioteca `matplotlib` para exibir as categorias acima citadas.

Como comparativo, foi também utilizada a base de agregados, mais especificamente a variável 1000096: "Existência de água canalizada" agregado no nível de localidade de município (N6). A variável em questão já está calculada em percentual do total e o valor utilizado foi, novamente, a soma das respostas afirmativas à questão. Comparando o gráfico com os dados da API lado à lado com o da figura 5, podemos observar pequenas variações de valor:

Pouco muda entre um gráfico e outro, já que eles representam em tese a mesma coisa, mas algumas poucas variações mostram como o nível de detalhe dos microdados

[1] Arquivos *shapefile* referentes aos Estados brasileiros disponíveis em: https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_de_setores_censitarios__divisoes_intramunicipais/censo_2010/setores_censitarios_shp/. O arquivo utilizado foi de extensão `.shp da pasta compactada rj_setores_censitarios.zip`

[2] Com base nos microdados da amostra de domicílios de 2010 do Estado do Rio de Janeiro.

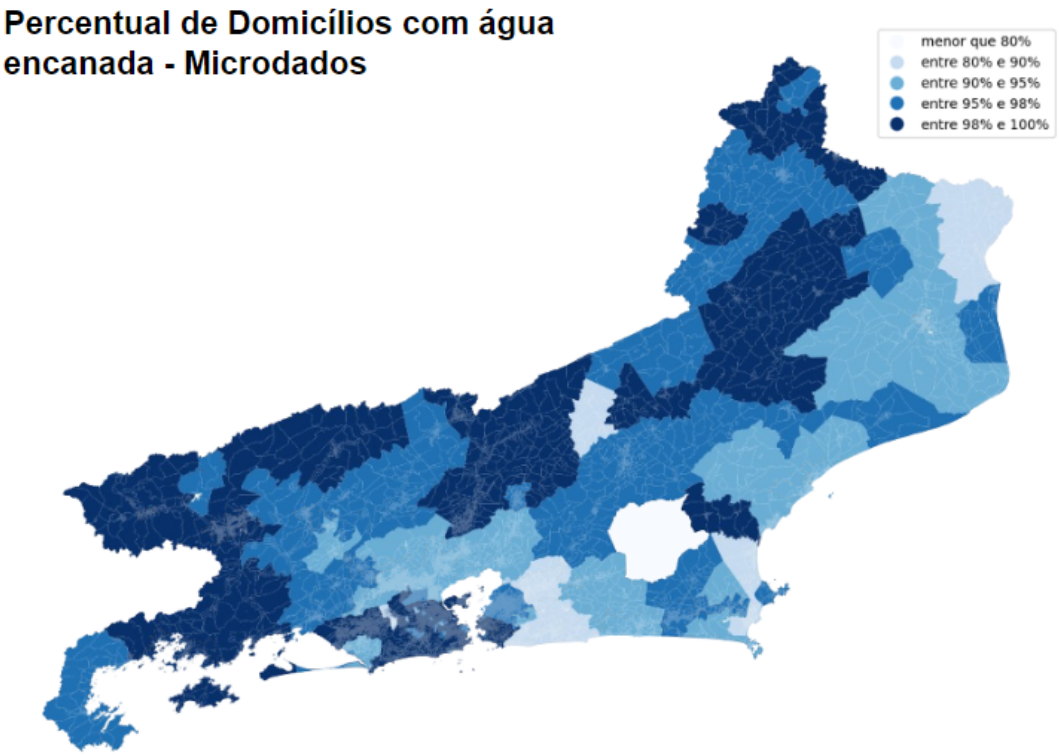



Figura 5 – Percentual de domicílios com água encanada no Estado do Rio de Janeiro com base nos microdados da amostra de domicílios. Fonte: Censo Demográfico, IBGE (2010).[2]

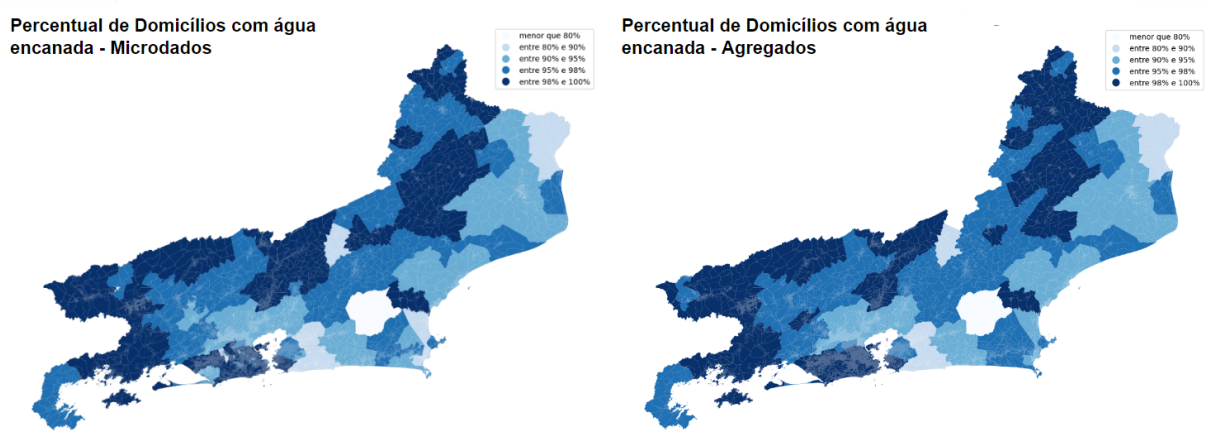



Figura 6 – Percentual de domicílios com água canalizada no Estado do Rio de Janeiro. Na esquerda o gráfico com base nos microdados e na direita com base nos dados agregados da API. Fonte: Censo Demográfico, IBGE (2010).

revela informações que são perdidas no gráfico de agregados[3], como na região mais ao norte do Estado, que aparenta ter mais regiões com água encanada porém a agregação "mascara" certos pontos menores que entram na categoria "entre 95% e 98%" ao invés da categoria superior.

[3] Gráfico em tamanho maior na imagem 7 do apêndice B.1

# 5 Conclusão

Os dados do Instituto Brasileiro de Geografia e Estatística são de enorme valor para diversos setores do país, tanto no âmbito público quanto privado. Tais dados podem ser obtidos em dois formatos distintos: fazendo o *download* dos arquivos de microdados codificados de acordo com o arquivo de *layout* e o formato agregado, carregado via API de serviço de dados do instituto, onde os dados vem em notação JSON.

Através de requisições feitas à API do IBGE, usuários podem carregar dados em formato JSON utilizando URLs personalizadas com as variáveis desejadas, contudo ainda é necessário realizar uma recursão para que todos os valores fiquem em um mesmo nível de aninhamento. As dificuldades encontradas são a engenharia de uma URL que retorne os dados desejados e a transformação deles para a utilização. Mas após o entendimento de como usar o serviço efetivamente, a criação de *datasets* a partir dele se torna uma tarefa simples. Além disso, as APIs, principalmente a de agregados, possuem uma vasta gama de pesquisas de diversas áreas de conhecimento, assim sendo uma ferramenta valiosa para trabalhos de dados.

Por sua vez, os microdados se destacam por sua riqueza em detalhamento, já que representam o grão mais atômico possível quando se trata de uma resposta individual. Contudo, grande detalhe implica em um grande volume de dados, e além disso, o formato em que são disponibilizados é complexo e demanda um processamento demorado. O *software Stata* foi de extrema ajuda para a leitura e relacionamento dos identificadores com respectivas *labels*, tornando um procedimento que demorado em algo quase instantâneo, e a automatização da criação do *Dofile* reduziu substâncialmente o trabalho necessário para programar arquivos para novas pesquisas.

Ao analisar os dados através de visualizações, foi possível perceber que os microdados conseguem explicar alguns detalhes que passam despercebidos em níveis de agregação maiores. Essa vantagem vem com a desvantagem da haver necessidade da realização de alguns cálculos e ponderações antes do uso, o que não é necessário para os agregados.

Conclui-se que ambas as formas de ingestão de dados, por mais que necessário um trabalho prévio para sua utilização, são valiosas fontes de dados. Os microdados tendo como maior vantagem o seu detalhamento e possibilidades de análises dentro de municípios, enquanto os agregados, apesar de não alcançarem um grão tão baixo, tem seus registros já previamente calculados, permitindo maior agilidade em sua utilização.

Vale ainda ressaltar que todo o processo aqui documentado foi construído de forma que os códigos desenvolvidos possam ser reutilizados para diferentes conjuntos de dados, podendo ser replicados e adaptados conforme necessário.

# Referências

ARCGIS. Shapefiles, 2022. Disponível em: <https://enterprise.arcgis.com/en/portal/latest/use/shapefiles.htm>.

ENCARNAÇÃO, Paulo; Maurício Schueler da. **Guia do Censo: Operação Censitária**. [S.l.]: Coordenação de Comunicação Social - IBGE., 2010. Disponível em: <https://censo2010.ibge.gov.br/materiais/guia-do-censo/operacao-censitaria.html>.

IBGE. **API de dados agregados do IBGE**. [S.l.]: IBGE (Instituto Brasileiro de Geografia e Estatística), 2017. Disponível em: <https://servicodados.ibge.gov.br/api/docs/agregados?versao=3>.

_____. **Censo Demográfico: Microdados**. [S.l.]: IBGE (Instituto Brasileiro de Geografia e Estatística), 201-? Disponível em: <https://www.ibge.gov.br/estatisticas/sociais/trabalho/22827-censo-demografico-2022.html?=&t=microdados>.

# Apêndices

# APÊNDICE A – Códigos

Este apêndice contém os algoritmos, códigos, *scripts* e funções utilizadas para o desenvolvimento do trabalho que são citadas o longo do decorrer da dissertação. O conjunto completo dos arquivos de código se encontra no diretório `/src` do repositório do *Github* <github.com/GDevigili/TCC-IBGE>.

## A.1 Algoritmo de *Unnesting*

O presente algoritmo realiza o processo de *unnesting*, que visa o nivelamento de uma estrutura de dados em nível único, duplicando dados quando necessário. O processo de desaninhamento sobe os dados de um valor "filho" (de grau inferior) até o nível de seu valor "pai" (de grau imediatamente superior) de forma recursiva.

```python
def unnest_json(json_dict: dict, col_name: str = '') -> dict:
    output = {}
    def flatten(column, col_name:str = ''):
        # Se a coluna é do tipo dict, continua a recursão
        if type(column) == dict:
            for key in column:
                # Determina o nome da coluna para cada chave (
                    key)
                if col_name == '':
                    new_col_name = str(key)
                else:
                    # adiciona o pai da coluna ao seu nome
                    parent_col = col_name.split('_')[-1]
                    new_col_name = parent_col + '_' + str(key)
                # Chama recursivamente a função flatten para a
                    chave atual
                flatten(column[key], new_col_name)
        else:
            # Se a coluna não é do tipo dict, salva o valor dela
                no output
            output[col_name] = column
    flatten(json_dict, col_name)
    return output
```

Listing 3 – Algoritmo de *unnesting* em *python*.

## A.2 Algoritmo de conversão de *string* JSON para um objeto do tipo *pandas.DataFrame*

O *script* abaixo converte um objeto do tipo *string* contendo um código JSON para um objeto do tipo `DataFrame` da biblioteca `pandas`, para tal, utilizando a função de *unnesting*, presente em *listing* 3. Um exemplo de uso é a conversão do exemplo em *listing* 1 da sessão 3.1 para a tabela 1.

```
import pandas as pd

def make_df(json_dict: dict) -> pd.DataFrame:
    """Return a DataFrame from a json dict.

    Args:
        json_dict (dict): A dict containing the json data.

    Returns:
        pd.DataFrame: A DataFrame with the data.
    """
    # Create a list to store the rows
    rows = []
    # Iterate over each item on the json dict
    for item in json_dict:
        # Unpack the json dict into a single row dict
        row = unpack_json(item)
        # Add the row to the list
        rows.append(row)
    # Create a DataFrame from the rows list
    df = pd.DataFrame(rows)
    return df
```

Listing 4 – Algoritmo de transformação de *string* JSON para um objeto do tipo *pandas.DataFrame*.

## A.3 Adaptador HTTP customizado

Devido ao erro `UNSAFE_LEGACY_RENEGOTIATION_DISABLED` encontrado ao tentar realizar requisições às APIs do IBGE utilizando sistemas operacionais baseados em *Linux* (foi utilizado Ubuntu 20 nos testes), foi necessário criar um adaptador customizado do

*Hypertext Transfer Protocol* (HTTP) utilizando um contexto legado do protocolo *Secure Sockets Layer* (SSL).

```
import requests
import urllib3
import ssl

class CustomHttpAdapter (requests.adapters.HTTPAdapter):
    """Transport adapter that allows us to use custom
       ssl_context."""
    def __init__(self, ssl_context=None, **kwargs):
        self.ssl_context = ssl_context
        super().__init__(**kwargs)

    def init_poolmanager(self, connections, maxsize, block=False
        ):
        self.poolmanager = urllib3.poolmanager.PoolManager(
            num_pools=connections, maxsize=maxsize,
            block=block, ssl_context=self.ssl_context)

def get_request():
    """Return a requests session with a custom SSL context."""
    ctx = ssl.create_default_context(ssl.Purpose.SERVER_AUTH)
    ctx.options |= 0x4  # OP_LEGACY_SERVER_CONNECT
    session = requests.session()
    session.mount('https://', CustomHttpAdapter(ctx))
    return session

def get_request_json(url: str)-> dict:
    """Return a json from a request."""
    response = get_request().get(url)
    return response.json()
```

Listing 5 – Algoritmo de transformação de *string* JSON para um objeto do tipo *pandas.DataFrame*.

## A.4 *Script* para a carga de indicadores da API de países

O código a seguir realiza requisições para a API de países do serviço de dados do IBGE, retornando um conjunto de dados com 34 indicadores de 193 países. O módulo

`utils` importado no *script* está no diretório `src` do repositório do *Github*. As funções utilizadas são `make_df()` e `get_request_json()` (*listings* 4 e 5, respectivamente).

```
import pandas as pd
from utils import *

url_paises = "https://servicodados.ibge.gov.br/api/v1/paises/"
df_paises = make_df(get_request_json(url_paises))
df_paises = df_paises.rename(columns={
    'id_M49': 'id-pais'
    ,'id_ISO-3166-1-ALPHA-2': 'sigla-2'
    ,'id_ISO-3166-1-ALPHA-3': 'sigla-3'
})
url_indicadores = "https://servicodados.ibge.gov.br/api/v1/
   paises/indicadores/"
df_indicadores = make_df(get_request_json(url_indicadores))

country_list = df_paises['sigla'].unique()
indicator_list = df_indicadores['id'].unique()

df_country_indicators = pd.DataFrame()
for country in country_list:
    # making the request
    url_country = f"https://servicodados.ibge.gov.br/api/v1/
       paises/{country}/indicadores"
    df_country = make_df(get_request_json(url_country))
    # unpacking the lists
    # take the series dict out of the list (there will be always
        one or zero values on the list)
    df_country = df_country.explode('series')
    # get the country code (e.g. BR) and country name from the
       series country and concat it into each of the df_country
       lines
    df_country = pd.concat([df_country, pd.json_normalize(
       df_country['series'])], axis = 1)
    # drop the column series as it won't be used anymore
    df_country = df_country.drop('series', axis = 1)
    # unpack the values for each date
    df_country = df_country.explode('serie')
    df_country['ano'] = df_country['serie'].apply(lambda x: list
```

```
        (x.keys())[0] if type(x) == dict else None)
    df_country['valor_indicador'] = df_country['serie'].apply(
        lambda x: list(x.values())[0] if type(x) == dict else
        None)
    df_country = df_country.drop('serie', axis = 1)
    df_country_indicators = pd.concat([df_country_indicators,
        df_country], ignore_index=True)
```

Listing 6 – *Script* de carga dos indicadores da API de países.

## A.5 Leitura de arquivo *Stata* no *python*

Para realizar a leitura de um arquivo de extenção `.dta`, às vezes se faz necessário carregar ele como um objeto do tipo *StataReader* da biblioteca `pandas.io.stata`.

No primeiro bloco de código, a variável `data_micro` recebe os dados no formato de microdados lidos pelo *reader*, `col_labels` os nomes de coluna e `value_lables` os valores identificados pelos IDs que estão nos microdados. Tendo isto em mãos, os valores são substituídos no laço de repetição para obter-se o *DataFrame* final.

```
# Read the file and the labels
with StataReader('../../dados/microdados-processados/
    amostra_domicilios_2010_RJ.dta') as reader:
    data_micro = reader.read(convert_categoricals=False)
    col_labels = reader.variable_labels()
    value_labels = reader.value_labels()

# Remove '_lbl' from the keys in value_labels
value_labels = {k.replace('_lbl', ''): v for k, v in
    value_labels.items()}

# Apply the labels to the data
for col, labels in value_labels.items():
    if col in data_micro:
        data_micro[col] = data_micro[col].map(labels)

# Apply the labels to the columns
data_micro = data_micro.rename(columns=col_labels)
```

Listing 7 – *Script* de carga dos indicadores da API de países.

# APÊNDICE B – Visualização dos dados

## B.1 Mapa do percentual de domicílios com acesso à água encanada no Estado do Rio de Janeiro - Dados Agregados

Mapa análogo ao da figura 5 que tem como fonte de dados os agregados da API. O mapa também é utilizado para comparação na figura 6.

Os dados coletados da API são os referentes à variável 100009: "Existência de água canalizada" em unidades percentuais. A variável foi filtrada para apenas conter valores referentes ao Estado do Rio de Janeiro agregados pelo nível de município (N6). A seguinte URL redireciona para o JSON contendo os dados: https://servicodados.ibge.gov.br/api/v3/agregados/2065/periodos/-6/variaveis/96%7C1000096?localidades=N1[all]|N13[330101]|N6[N3[33]]&classificacao=471[0,12219].

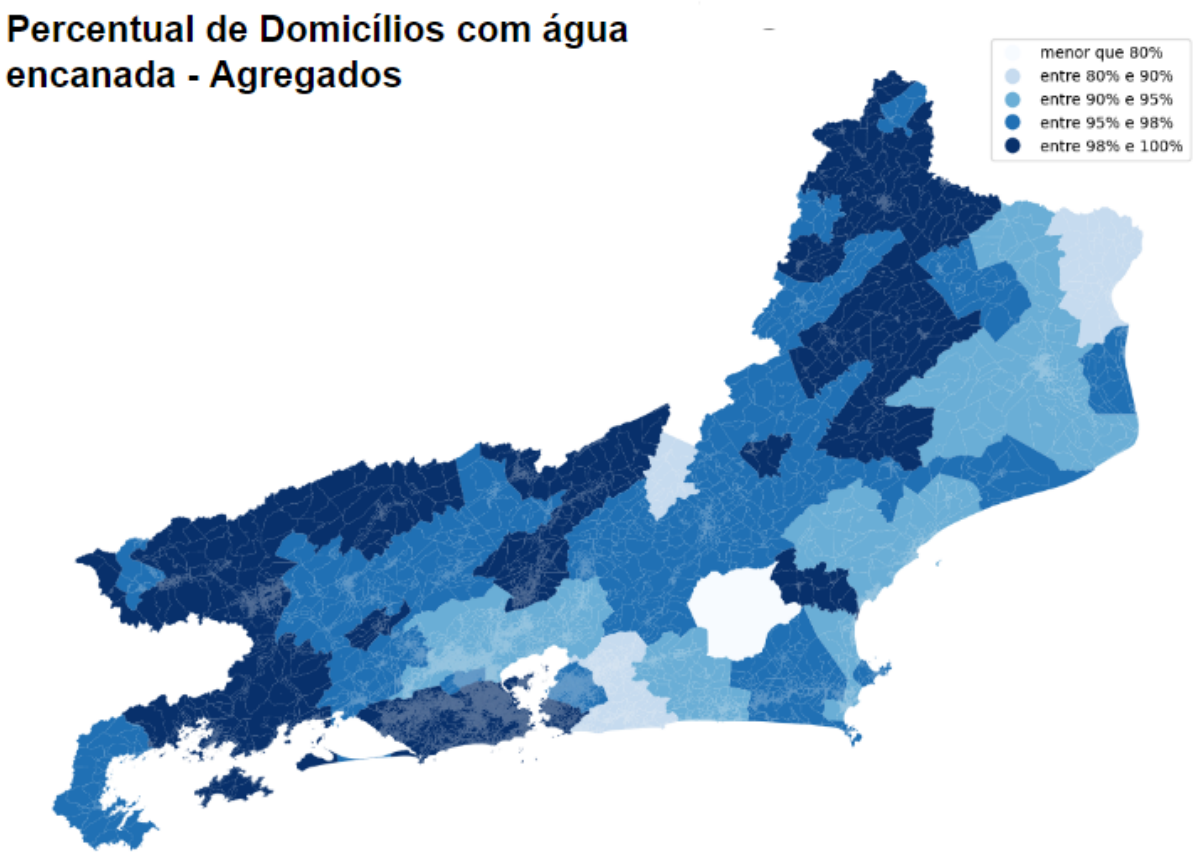



Figura 7 – Mapa do percentual de domicílios com água encanada no Estado do Rio de Janeiro. Fonte: Dados do Autor (2023)[1]

[1] Com base nos dados agregados do Censo Demográfico de 2010.