

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**  
**NÚCLEO DE EDUCAÇÃO A DISTÂNCIA**  
**Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data**

**ANDRÉ LUÍS THERESA**

**PREDIÇÃO DA ARRECADAÇÃO DE TRIBUTOS FEDERAIS, POR MUNICÍPIO  
BRASILEIRO, A PARTIR DE INFORMAÇÕES SOCIAIS, ECONÔMICAS,  
DEMOGRÁFICAS E GEOPOLÍTICAS.**

Belo Horizonte

2021

**ANDRÉ LUÍS THERESA**

**PREDIÇÃO DA ARRECADAÇÃO DE TRIBUTOS FEDERAIS, POR MUNICÍPIO  
BRASILEIRO, A PARTIR DE INFORMAÇÕES SOCIAIS, ECONÔMICAS,  
DEMOGRÁFICAS E GEOPOLÍTICAS.**

Trabalho de Conclusão de Curso apresentado  
ao Curso de Especialização em Ciência de  
Dados e Big Data como requisito parcial à  
obtenção do título de especialista.

Belo Horizonte  
2021

# SUMÁRIO

<b>1. INTRODUÇÃO.....</b>	<b>6</b>
1.1. CONTEXTUALIZAÇÃO .....	6
- Fatores que influenciam o comportamento de conformidade tributária e a atuação das Administrações Tributárias .....	6
- Atuação das Administrações Tributárias, <i>Big Data</i> e Ciência de Dados .....	8
- Contribuições gerais deste projeto.....	9
1.2. O PROBLEMA PROPOSTO.....	10
1.3. LINGUAGEM DE PROGRAMAÇÃO E AMBIENTE COMPUTACIONAL.....	11
<b>2. COLETA DE DADOS.....</b>	<b>12</b>
2.1. BASES DE DADOS PRINCIPAIS.....	12
i. Arrecadação das Receitas Administradas pela RFB, por Município ( <i>arrecadacao-da-receita-administrada-pela-rfb-por-municipio-2018.xlsx</i> ) .....	12
ii. Tabela 1685 - Unidades locais, empresas e outras organizações atuantes, pessoal ocupado total, pessoal ocupado assalariado, pessoal assalariado médio, salários e outras remunerações e salário médio mensal ( <i>tabela1685.csv</i> ) .....	13
iii. Produto Interno Bruto dos Municípios (“ <i>base_de_dados_2010_2018.xls.zip</i> ”).....	14
iv. Tabela 6579 - População residente estimada ( <i>tabela6579.csv</i> ).....	14
v. Tabela 200 - População residente, por sexo, situação e grupos de idade ( <i>tabela200.csv</i> ) .....	15
vi. Tabela 1554 - Pessoas de 10 anos ou mais de idade, por nível de instrução ( <i>tabela1554.csv</i> ) .....	16
vii. Tabela 2030 - Pessoas de 10 anos ou mais de idade, por classes de rendimento nominal mensal ( <i>tabela2030.csv</i> ) .....	16
2.2. BASE AUXILIAR.....	17
viii. Tabela Auxiliar: Códigos dos Municípios - 2020 ( <i>DTB_2020_v2.zip</i> ) .....	17
<b>3. PROCESSAMENTO/TRATAMENTO DE DADOS .....</b>	<b>18</b>
3.1. TRANSFORMAÇÕES PRELIMINARES .....	18
- Descompactação de arquivos.....	18
- Visualização, análise e importação das bases originais .....	19
- Criação da variável padronizada, chamada “município-uf” .....	23
- Manutenção, apenas, das informações relativas ao ano de interesse.....	24
- Renomeação de variáveis.....	24
- Criação, em alguns casos, de novas colunas derivadas (enriquecimento dos dados) .....	25
- Remoção de variáveis redundantes e com valores nulos ou constantes .....	27
3.2. BASES DE DADOS TRANSFORMADAS.....	28
- ntbk 01_01 - rfb arrecadacao 2018.csv .....	28
- ntbk 01_02 - ibge empresas 2018.csv .....	28
- ntbk 01_03 – ibge pib 2018.csv .....	29
- ntbk 01_04 – ibge população residente 2018.csv .....	30
- ntbk 01_05 – ibge censo 2010 idades.csv .....	31
- ntbk 01_06 - ibge censo instrucao.csv.....	32
- ntbk 01_07 – ibge censo rendimentos.csv .....	32
3.3. INTEGRAÇÃO DAS BASES TRANSFORMADAS E CRIAÇÃO DO DATASET CONSOLIDADO .....	33
- Integração ( <i>merge/join</i> ) dos <i>datasets</i> transformados .....	33
- Saneamento das divergências em relação aos nomes dos municípios .....	33
- Geração do <i>dataset</i> consolidado “ntbk 02 – dataset consolidado.csv” .....	38
3.4. TRANSFORMAÇÃO DAS VARIÁVEIS NUMÉRICAS (CONVERSÃO PARA BASE LOGARÍTMICA NATURAL).....	40
3.5. CRIAÇÃO DA VARIÁVEL CATEGÓRICA "FAIXA_ARRECADACAO" .....	42
3.6. TRANSFORMAÇÃO DAS VARIÁVEIS CATEGÓRICAS (ONE HOT ENCODING) .....	44

3.7. NOVAS EXCLUSÕES DE VARIÁVEIS .....	45
- Exclusão de variáveis com multicolinearidade identificada. ....	45
- Exclusão de variáveis com correlação inferior a 0,01 com a variável dependente ( <i>label</i> ). ....	46
- Exclusão de variáveis que não serão utilizadas na aplicação dos modelos de machine learning. ....	47
3.8. GERAÇÃO DO DATASET CONSOLIDADO “NTBK 03 – DATASET CONSOLIDADO – MAIORES CORRELACOES.CSV” .....	47
<b>4. ANÁLISE E EXPLORAÇÃO DOS DADOS .....</b>	<b>48</b>
4.1. SUMÁRIOS ESTATÍSTICOS.....	48
4.2. ANÁLISES DA VARIÁVEL DEPENDENTE ( <i>LABEL</i> ): 01_ARRECADACAO_2018. ....	50
- Análise dos <i>outliers</i> .....	52
4.3. ANÁLISE VISUAL DAS VARIÁVEIS NUMÉRICAS. ....	54
- <i>Boxplots</i> por UF .....	55
- <i>Boxplots</i> por faixa de arrecadação .....	58
4.4. ANÁLISE DAS CORRELAÇÕES E TRATAMENTO DE MULTICOLINEARIDADES. ....	66
- Impressão de Matrizes de Gráficos de Dispersão para análise dos relacionamentos entre as variáveis .....	66
- Análise das correlações .....	69
- Identificação de multicolinearidades .....	71
- Identificação das variáveis com baixa correlação.....	76
- Correlação não implica necessariamente causalidade. ....	77
<b>5. CRIAÇÃO E AVALIAÇÃO DE MODELOS DE MACHINE LEARNING.....</b>	<b>77</b>
5.1. AÇÕES PRELIMINARES.....	78
- Importação do <i>dataset</i> com os dados de interesse.....	78
- Criação dos <i>dataframes</i> de estudo.....	79
- Separação dos <i>dataframes</i> em base de treinamento e de teste.....	82
- Definição da métrica e de uma <i>baseline</i> para avaliação dos resultados .....	83
5.2. APLICAÇÃO DOS ALGORITMOS DE MACHINE LEARNING .....	86
- Regressão Linear.....	88
- Lasso .....	90
- Ridge.....	91
- ElasticNet.....	92
- Random Forest .....	94
- Melhor modelo identificado para este primeiro experimento .....	95
5.3. ANÁLISE DOS RESÍDUOS .....	96
- Resíduos gerados pelo melhor modelo ( <i>RandomForestRegressor</i> e <i>df_log</i> ) .....	96
- Comparação com o segundo melhor modelo ( <i>ElasticNet</i> e <i>df_original</i> ) .....	101
- Resultado da exclusão das maiores arrecadações para o <i>ElasticNet</i> .....	103
5.4. OTIMIZAÇÃO DE HIPERPARÂMETROS E CROSS VALIDATION .....	105
<b>6. APRESENTAÇÃO DOS RESULTADOS .....</b>	<b>107</b>
<b>7. LINKS.....</b>	<b>110</b>
<b>REFERÊNCIAS .....</b>	<b>111</b>
<b>APÊNDICES.....</b>	<b>112</b>
Apêndice I – Resumo do projeto (Canvas)	
Apêndice II – Detalhes adicionais para a coleta dos <i>datasets</i> originais	
Apêndice III – Resumo das transformações nos <i>datasets</i> originais	
Apêndice IV – Matriz de correlações	
Apêndice V – Correlações entre as variáveis independentes e o <i>label</i>	
Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb	

**Apêndice VII – Ntbk 02. Criação do *dataset* consolidado - merges e saneamentos.ipynb**

**Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb**

**Apêndice IX – Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb**

## 1. Introdução

### 1.1. Contextualização

#### - Fatores que influenciam o comportamento de conformidade tributária e a atuação das Administrações Tributárias

O conhecimento dos fatores que determinam o comportamento de conformidade dos contribuintes é ponto fundamental às Administrações Tributárias para o estabelecimento de estratégias que visem o alcance da arrecadação potencial de um país, ao mesmo tempo que consolida uma relação de confiança com a sociedade a que servem.

O bem-estar e a prosperidade de uma nação se dão, em boa parte, por meio de políticas sociais implementadas pelo Estado cujo financiamento depende, principalmente, da arrecadação dos tributos legalmente impostos aos contribuintes (pessoas físicas e pessoas jurídicas).

Define-se conformidade tributária como o cumprimento das obrigações principais e acessórias pelos contribuintes (pessoas físicas e jurídicas). Essas obrigações podem ser divididas em quatro grandes categorias:

- inscrição e habilitação cadastral;
- entrega de declarações tributárias e aduaneiras dentro do prazo;
- prestação de informações completas e exatas; e
- pagamento de impostos e contribuições dentro do prazo.

A gestão da conformidade e a implementação de medidas que maximizem o cumprimento (seja atuando de maneira a incentivar a conformidade, seja de modo a dissuadir a não conformidade) é papel das Administrações Tributárias, essencial, portanto: ao planejamento das políticas governamentais e à decisão sobre o que poderá ser realizado ou sobre o que se deixará de fazer diante de demandas sociais crescentes e de recursos escassos.

A conformidade tributária nem sempre ocorre de maneira uniforme. Os aspectos comportamentais específicos de cumprimento das obrigações tributárias devem ser entendidos dentro de uma perspectiva mais ampla sobre o comportamento humano em geral. As diferentes interações de cada um dos contribuintes com fatores a ele individualmente relacionados leva a diferentes atitudes diante das obrigações tributárias. Distinguem-se quatro perfis de contribuintes, diante das obrigações tributárias (OCDE, 2004):

- decididos a não cumprir;

- não querem cumprir, mas o farão se o risco de ser detecção e as consequências com atuação pela Administração Tributária forem altos;
- tentam, mas nem sempre conseguem cumprir; e
- dispostos a fazer o certo.

É impossível, porém, às AT controlarem e verificarem todo e cada contribuinte individualmente. Além disso, não é razoável alocar os recursos, limitados, em exames rotineiros em contribuintes que cumprem com suas obrigações (de baixo risco). Os custos de oportunidade para tais exames são elevados.

Assim como uma empresa privada aloca seus recursos para as áreas que têm mais potencial para gerar receitas e lucros, uma moderna Administração Tributária deve ter por fundamento o conhecimento do comportamento de conformidade dos contribuintes (e, principalmente, das causas desse comportamento) e então concentrar suas auditorias naqueles de maior risco.

A figura abaixo demonstra essa relação entre o comportamento dos contribuintes e a atuação esperada por parte da Administração Tributária. Para tanto, consideram-se os fatores que influenciam o comportamento dos contribuintes a fim de compreender as suas atitudes em relação às obrigações tributárias e aduaneiras e, então, serem estabelecidas as correspondentes estratégias para fomentar o cumprimento.

### Comportamento ↔ Atuação da administração tributária e aduaneira.



Figura 1. Comportamento x Atitudes x Estratégias x Medidas de tratamento

Fonte: RFB/AUDIT, 2016 (Guia)– adaptado de OCDE (2004, p.40)

Medidas de tratamento direcionadas às causas do comportamento de não-conformidade são muito mais efetivas (e quase sempre menos custosas) que aquelas voltadas simplesmente aos efeitos da não-conformidade.

Nessa linha, a literatura internacional identifica uma série de fatores que moldam o comportamento e impactam as atitudes em relação à conformidade, a exemplo de: oportunidade para fraudar, percepção de equidade e justiça, normas sociais, normas pessoais, dissuasão ou probabilidade percebida de detecção e sanções (UNIÃO EUROPEIA, 2010).

O quadro abaixo fornece uma lista mais abrangente de prováveis fatores que podem ser utilizados como variáveis explicativas úteis em um modelo de risco de não conformidade.

Categoría	Características gerais
Fatores empresariais	Natureza jurídica (empresário individual, sociedade simples limitada, sociedade anônima aberta, etc.), porte e data de abertura, atividades econômicas principal e secundárias, foco de atuação (local <i>versus</i> internacional), capital social, intermediários de negócios
Fatores setoriais	Definição e tamanho do setor econômico, empresa líder de mercado, margem de lucro, estrutura de custos, regulamentação da atividade, parceiros comerciais, nível de competição, fatores sazonais e questões de infraestrutura
Fatores sociológicos	Normas culturais, origem étnica, atitude em relação ao governo, idade, gênero, nível educacional
Fatores econômicos	Investimento, taxas demográficas, sistema tributário, políticas governamentais, influência internacional, inflação, mercados
Fatores psicológicos	Ambição, apetite de risco, medo, confiança, equidade e justiça, oportunidade de evadir

*Quadro 1. Fatores que influenciam o comportamento de conformidade.*

Fonte: RFB/AUDIT, 2016 (Guia) – adaptado de OCDE (2004, p.40)

Assim, escolher a melhor forma de atuar (se por meio da educação, da assistência ou da sanção, por exemplo) exige análise profunda sobre os fatores que determinam os comportamentos dos contribuintes de tal forma que estes enxerguem a justiça procedural da Administração Tributária e que os contribuintes conformes ou aqueles com tendência à conformidade não percam a confiança na instituição e alterem o seu modo de agir de forma negativa.

### **- Atuação das Administrações Tributárias, Big Data e Ciência de Dados**

Dito isso, conhecer esses comportamentos e suas variações para os contribuintes individuais ou segmentos de contribuintes é um desafio sobre o qual as Administrações Tributárias de todo o mundo têm se debruçado. A construção do conhecimento necessário à identificação dos riscos de conformidade passa pelo uso equilibrado e combinado de todas as fontes de informação disponíveis (TADAT, 2019):

- legislação tributária;
- declarações de impostos e demonstrações financeiras;
- informações de terceiros, inclusive de outras administrações tributárias e aduaneiras;
- indicadores (ex. macroeconômicos);
- resultados de auditorias, inclusive aleatórias;
- resultados de pesquisa de opinião da sociedade;
- estudos sobre gap tributário;

- estudos sobre os fatores ambientais que influenciam ou podem influenciar o comportamento dos contribuintes e dos intervenientes do comércio exterior;
- estudos tópicos sobre temas relacionados ao cumprimento (ex. preços de transferência e planejamento tributário), estudos sobre atividades econômicas ocultas (ex. economia informal);
- gestores e equipes de trabalho.

O desafio de extrair conhecimento de uma ampla base de dados, estruturados e não estruturados, impõem às administrações tributárias voltarem seus olhos em direção ao *Big Data* e à Ciência de Dados. A aplicação de técnicas de análise de dados em busca de padrões, para revelar correlações entre variáveis ou para fazer previsões a partir de massas de dados cada vez maiores e do conhecimento de especialistas nos processos de trabalho, já é realidade. O *Big Data* restringe o alcance de análises puramente empíricas e destaca o valor de técnicas e ferramentas avançadas suportadas por uma adequada infraestrutura de tecnologia da informação (*hardware* e *software*). (RFB/Audit – Guia, 2016)

### **- Contribuições gerais deste projeto**

A contribuição deste projeto é buscar demonstrar, ainda que como um passo inicial e de forma agregada por município (e não para cada contribuinte), a existência de relação entre a arrecadação de tributos federais e de variáveis não necessariamente relacionadas às bases de cálculo tradicionais dos tributos (como por exemplo renda ou receita bruta) mas também com outras, por exemplo, de cunho social (como a distribuição da população por nível de instrução). Para tanto, serão combinadas informações públicas disponibilizadas pela Receita Federal do Brasil - RFB e pelo Instituto Brasileiro de Geografia e Estatística – IBGE

Desse modo, o que se pretende com o presente projeto é contribuir, como um ponto de partida, na análise da influência dos diversos fatores vistos acima (quadro 1) e a arrecadação tributária ao se examinar as seguintes questões:

- Há correlação entre fatores sociais, econômicos, demográficos e geopolíticos, e a arrecadação de tributos federais agregada por município?**
- É possível prever a arrecadação para cada um dos municípios brasileiros tão-somente a partir de dados públicos relacionados a esses fatores?**

## 1.2. O problema proposto

Neste projeto serão criados modelos de aprendizado supervisionado de máquina para, por meio da aplicação de algoritmos de Regressão, avaliar a capacidade de predição da arrecadação de tributos federais, por municípios brasileiros, a partir de dados públicos, divulgados pela Receita Federal do Brasil – RFB e pelo Instituto Brasileiro de Geografia e Estatística (IBGE), relacionados a aspectos sociais, econômicos, demográficos e geopolíticos.

Embora fosse possível, e talvez até mesmo com maior grau de eficácia, a previsão da arrecadação de determinado ano a partir da série histórica das arrecadações (de anos anteriores), ponderada por indicadores macroeconômicos (como inflação, por exemplo), o que se quer aqui buscar evidenciar é se é possível predizer essa **arrecadação (nossa variável independente, ou *label*)** com base em outros **atributos (nossas variáveis independentes, ou *features*)**, de cunho:

- econômico: PIB, dos valores agregados por cada setor econômico, das principais atividades econômicas etc.;
- social: faixa etária da população, nível de escolaridade, total da população ocupada etc.;
- socioeconômico: PIB *per capita*, salário médio mensal da população, distribuição da população por faixas de rendimento etc.;
- demográfico: número de residentes;
- geopolítico: município, estado da federação etc.

Isso porque o objetivo maior não é simplesmente predizer a arrecadação (ainda mais passada, já que os dados aqui trabalhados foram os do ano de 2018), mas sim identificar fatores não diretamente vinculados aos tributos (como seriam os fatores geradores ou as bases de cálculo) que tenham correlação com a arrecadação esperada.

Como mencionado no título 1.1., acima, o exame dos fatores que determinam o comportamento de conformidade é um campo importante a ser considerado pelas Administrações Tributárias para a definição de estratégias que impulsionem o cumprimento das obrigações pelos contribuintes.

A opção de se adotar uma análise segmentada por município deve-se principalmente pela disponibilidade da informação - vez que, em razão do sigilo fiscal e mesmo por sua abrangência, não há dados públicos para cada contribuinte

individualmente (foram localizadas apenas informações agregadas), sendo o município a agregação com maior capilaridade dentre as disponíveis.

Ainda sobre a informação de interesse, para o estudo foram utilizadas bases de dados disponibilizadas pela RFB e pelo IBGE. A análise terá por foco principal os dados do ano de 2018 - período mais recente para o qual as informações completas estão disponíveis (com exceção do Censo, cujas informações são apenas de 2010).

### **1.3. Linguagem de programação e ambiente computacional**

Para a manipulação, transformação e exploração dos dados e aplicação dos algoritmos de aprendizado de máquina (*machine learning*), será utilizado a linguagem de programação Python e o ambiente computacional Jupyter Notebook, com as seguintes características (metadados):

```
{"kernelspec": {"name": "python3","display_name": "Python 3","language": "python"},  
"language_info": {"name": "python","version": "3.8.5","mimetype": "text/x-python", "codemirror_mode": {"name": "ipython","version": 3 },"pygments_lexer": "ipython3", "nbconvert_exporter": "python", "file_extension": ".py" } }
```

Como estratégia de programação e organização do estudo, decidiu-se por criar quatro Jupyter Notebooks (cujas íntegras seguem nos Apêndices VI a IX):

- Ntbk 01. Coleta dos dados e transformações preliminares.ipynb;**
- Ntbk 02. Criação do dataset consolidado - merges e saneamentos;**
- Ntbk 03. Análise e exploração dos dados.ipynb;**
- Ntbk 04. Criação dos modelos de machine learning-Rregressão.ipynb.**

Por fim, para permitir a execução dos códigos em Python nesses notebooks, foi necessária a utilização de funcionalidades das seguintes bibliotecas, classes e módulos da biblioteca padrão do Python:

- pandas: fornece estruturas de dados de alto nível e uma grande variedade de ferramentas para análise (<https://pandas.pydata.org/>);
- numpy: destinado ao processamento de grandes matrizes e matrizes multidimensionais, e uma extensa coleção de funções matemáticas de alto nível e métodos implementados possibilitam a execução de várias operações com esses objetos (<https://numpy.org/>);

- matplotlib: útil à criação de diagramas e gráficos bidimensionais (<https://matplotlib.org/>);
- seaborn: contém configurações padrão mais adequadas para o processamento de gráficos (<https://seaborn.pydata.org/>);
- sklearn (Scikit-learn): fornece algoritmos para muitas tarefas padrão de aprendizado de máquina e mineração de dados, como *clustering*, regressão, classificação, redução de dimensionalidade e seleção de modelo (<https://scikit-learn.org/stable/>);
- chardet: útil para a detecção do *encoding* (conjunto de caracteres utilizados para escrever o documento/arquivo) - <https://pypi.org/project/chardet/>;
- zipfile: para trabalhar com arquivos ZIP (<https://docs.python.org/pt-br/3/library/zipfile.html>);
- csv: módulo da biblioteca padrão do Python, implementa classes para ler e gravar dados tabulares no formato CSV (<https://docs.python.org/pt-br/3/library/>);
- sys: módulo da biblioteca padrão do Python, fornece acesso a algumas variáveis usadas ou mantidas pelo interpretador e a funções que interagem fortemente com o interpretador (<https://docs.python.org/pt-br/3/library/>);
- math: módulo da biblioteca padrão do Python, fornece acesso às funções matemáticas definidas pelo padrão C (<https://docs.python.org/pt-br/3/library/>).

## 2. Coleta de Dados

### 2.1. Bases de dados principais

As informações utilizadas neste projeto foram coletadas dos sites do Governo Federal Brasileiro ([www.gov.br](http://www.gov.br)) e do Instituto Brasileiro de Geografia e Estatística – IBGE ([www.ibge.gov.br](http://www.ibge.gov.br)).

#### i. Arrecadação das Receitas Administradas pela RFB, por Município (*arrecadação-da-receita-administrada-pela-rfb-por-municipio-2018.xlsx*)

- Descrição:

Arrecadação total (DARF+GPS) das receitas administradas pela Receita Federal do Brasil- RFB em 2018, por Município.

O documento possui três planilhas, a de interesse é a denominada “Total” (que é a soma entre os valores pagos por meio de DARF e por GPS – que são tipos de guias para pagamento).

- Fonte: Site do Governo Federal Brasileiro na Internet;
- Endereço eletrônico: [https://www.gov.br/receitafederal/pt-br/acesso-a-informacao/dados-abertos/receitadata/arrecadacao/copy\\_of\\_arrecadacao-das-receitas-administradas-pela-rfb-por-municipio/arrecadacao-das-receitas-administradas-pela-rfb/arrecadacao-da-receita-administrada-pela-rfb-por-municipio-2018.xlsx](https://www.gov.br/receitafederal/pt-br/acesso-a-informacao/dados-abertos/receitadata/arrecadacao/copy_of_arrecadacao-das-receitas-administradas-pela-rfb-por-municipio/arrecadacao-das-receitas-administradas-pela-rfb/arrecadacao-da-receita-administrada-pela-rfb-por-municipio-2018.xlsx);
- Informações adicionais sobre a obtenção dos dados: para esta base, nenhum detalhe adicional para obtenção dos dados é necessário, vez que o download do arquivo é feito diretamente a partir do link acima.
- Data da coleta: 19/3/2021;
- Possui metadados: não.

## **ii. Tabela 1685 - Unidades locais, empresas e outras organizações atuantes, pessoal ocupado total, pessoal ocupado assalariado, pessoal assalariado médio, salários e outras remunerações e salário médio mensal (*tabela1685.csv*)**

- Descrição:

O Cadastro Central de Empresas - Cempre constitui um importante acervo de dados sobre o universo das empresas e outras organizações formais e suas respectivas unidades locais existentes no Brasil, reunindo informações cadastrais e econômicas oriundas de pesquisas anuais do IBGE, nas áreas de Indústria, Construção, Comércio e Serviços, e de registros administrativos do Ministério do Trabalho e Previdência Social, como a Relação Anual de Informações Sociais - RAIS.

Tabela 1685 - Unidades locais, empresas e outras organizações atuantes, pessoal ocupado total, pessoal ocupado assalariado, pessoal assalariado médio, salários e outras remunerações e salário médio mensal.

- Fonte: Site do Instituto Brasileiro de Geografia e Estatística na Internet (Sistema IBGE de Recuperação Automática - SIDRA);
- Endereço eletrônico:  
<https://sidra.ibge.gov.br/geratabela?format=us.csv&name=tabela1685.csv&terr=N&rank=-&query=t/1685/n6/all/v/all/p/last%201/d/v662%200,v1606%201,v5944%202,v10143%202/l,v,t%2Bp>
- Informações adicionais sobre a obtenção dos dados: vide **Apêndice II – Detalhes adicionais para a coleta dos datasets originais**.
- Data da coleta: 19/3/2021;

- Possui metadados: sim, mas sem descrição das variáveis (<https://metadados.ibge.gov.br/consulta/estatisticos/operacoes-estatisticas/CL>).

### **iii. Produto Interno Bruto dos Municípios ("base\_de\_dados\_2010\_2018\_xls.zip")**

- Descrição:

Fornece estimativas do Produto Interno Bruto - PIB dos Municípios, a preços correntes, e do valor adicionado bruto da Agropecuária, da Indústria, dos Serviços e da Administração, saúde e educação públicas e segurança social, a preços correntes, compatível com as metodologias das Contas Regionais e Nacionais do Brasil, sendo as estimativas obtidas comparáveis entre si.

Trata-se de um arquivo compactado que contém nossa base de interesse: "PIB dos Municípios - base de dados 2010-2018.xls"

- Fonte: Site do Instituto Brasileiro de Geografia e Estatística na Internet;
- Endereço eletrônico: <https://www.ibge.gov.br/estatisticas/economicas/contas-nacionais/2036-np-produto-interno-bruto-dos-municipios/9088-produto-interno-bruto-dos-municipios.html?=&t=downloads>;
- Informações adicionais sobre a obtenção dos dados: vide **Apêndice II – Detalhes adicionais para a coleta dos datasets originais**.
- Data da coleta: 19/3/2021;
- Possui metadados: sim, mas sem descrição das variáveis (<https://metadados.ibge.gov.br/consulta/estatisticos/operacoes-estatisticas/IO> ).

### **iv. Tabela 6579 - População residente estimada (*tabela6579.csv*)**

- Descrição: As estimativas populacionais têm fundamental importância para o cálculo de indicadores sociodemográficos nos períodos intercensitários, bem como alimentam as bases de informações de Ministérios e Secretarias Estaduais e Municipais da área social para a implementação de políticas públicas e posterior avaliação de seus respectivos programas. Além disso, em cumprimento ao dispositivo constitucional, as estimativas da população constituem o principal parâmetro para a distribuição, conduzida pelo Tribunal de Contas da União, das quotas partes relativas ao Fundo de Participação de Estados e Municípios.
- Fonte: Site do Instituto Brasileiro de Geografia e Estatística na Internet (Sistema IBGE de Recuperação Automática - SIDRA);

- Endereço eletrônico:

<https://sidra.ibge.gov.br/geratabela?format=us.csv&name=tabela6579.csv&terr=N&rank=-&query=t/6579/n6/all/v/all/p/2018/l/,v,t%2Bp>

- Informações adicionais sobre a obtenção dos dados: vide **Apêndice II – Datas adiconais para a coleta dos datasets originais.**

- Data da coleta: 19/3/2021;

- Possui metadados:

(<https://metadados.ibge.gov.br/consulta/estatisticos/operacoes-estatisticas/XF>).

## v. Tabela 200 - População residente, por sexo, situação e grupos de idade (*tabela200.csv*)

- Descrição:

Censo Demográfico - 2010 O Censo Demográfico tem por objetivo contar os habitantes do território nacional, identificar suas características e revelar como vivem os brasileiros, produzindo informações imprescindíveis para a definição de políticas públicas e a tomada de decisões de investimentos da iniciativa privada ou de qualquer nível de governo. Também constitui a única fonte de referência sobre a situação de vida da população nos municípios e em seus recortes internos, como distritos, bairros e localidades, rurais ou urbanas, cujas realidades dependem de seus resultados para serem conhecidas e terem seus dados atualizados.

Tabela 200 - População residente, por sexo, situação e grupos de idade - Amostra - Características Gerais da População.

- Fonte: Site do Instituto Brasileiro de Geografia e Estatística na Internet (Sistema IBGE de Recuperação Automática - SIDRA);

- Endereço eletrônico:

<https://sidra.ibge.gov.br/geratabela?format=us.csv&name=tabela200.csv&terr=N&rank=-&query=t/200/n6/all/u/y/v/allxp/p/last%201/c2/0/c1/0/c58/0,1140,1141,1142,1143,1144,1145,1146,1147,1148,1149,1150,1151,1152,1153,1154,1155,6802,6803,92963,92964,92965/d/v93%200/l/v.c58.t%2Bp%2Bc1%2Bc2>

- Informações adicionais sobre a obtenção dos dados: vide **Apêndice II – Datas adiconais para a coleta dos datasets originais.**

- Data da coleta: 19/3/2021;

- Possui metadados: sim, mas sem descrição das variáveis

(<https://metadados.ibge.gov.br/consulta/estatisticos/operacoes-estatisticas/CD>).

**vi. Tabela 1554 - Pessoas de 10 anos ou mais de idade, por nível de instrução  
(*tabela1554.csv*)**

- Descrição:

Censo Demográfico – 2010. O Censo Demográfico tem por objetivo contar os habitantes do território nacional, identificar suas características e revelar como vivem os brasileiros, produzindo informações imprescindíveis para a definição de políticas públicas e a tomada de decisões de investimentos da iniciativa privada ou de qualquer nível de governo. Também constitui a única fonte de referência sobre a situação de vida da população nos municípios e em seus recortes internos, como distritos, bairros e localidades, rurais ou urbanas, cujas realidades dependem de seus resultados para serem conhecidas e terem seus dados atualizados.

Tabela 1554 - Pessoas de 10 anos ou mais de idade, por nível de instrução - Resultados Gerais da Amostra

- Fonte: Site do Instituto Brasileiro de Geografia e Estatística na Internet (Sistema IBGE de Recuperação Automática - SIDRA);

- Endereço eletrônico:

<https://sidra.ibge.gov.br/geratabela?format=us.csv&name=tabela1554.csv&terr=N&rank=-&query=t/1554/n6/all/v/allxp/p/all/c1568/all/d/v140%200/l/v.c1568.t%2Bp>

- Informações adicionais sobre a obtenção dos dados: vide **Apêndice II – Datas adicionais para a coleta dos datasets originais**.

- Data da coleta: 19/3/2021;

- Possui metadados: sim, mas sem descrição das variáveis

( <https://metadados.ibge.gov.br/consulta/estatisticos/operacoes-estatisticas/CD>).

**vii. Tabela 2030 - Pessoas de 10 anos ou mais de idade, por classes de rendimento nominal mensal (*tabela2030.csv*)**

- Descrição:

Censo Demográfico – 2010. O Censo Demográfico tem por objetivo contar os habitantes do território nacional, identificar suas características e revelar como vivem os brasileiros, produzindo informações imprescindíveis para a definição de políticas públicas e a tomada de decisões de investimentos da iniciativa privada ou de qualquer nível de governo. Também constitui a única fonte de referência sobre a situação de vida da população nos municípios e em seus recortes internos, como distritos, bairros e localidades, rurais ou urbanas, cujas realidades dependem de seus resultados para serem conhecidas e terem seus dados atualizados.

Tabela 2030 - Pessoas de 10 anos ou mais de idade, por classes de rendimento nominal mensal - Resultados Gerais da Amostra.

- Fonte: Site do Instituto Brasileiro de Geografia e Estatística na Internet (Sistema IBGE de Recuperação Automática - SIDRA);

- Endereço eletrônico:

<https://sidra.ibge.gov.br/geratabela?format=us.csv&name=tabela2030.csv&terr=N&rank=-&query=t/2030/n6/all/v/allxp/p/last%201/c11570/all/d/v140%200/l/v,c11570,t%2Bp>

- Informações adicionais sobre a obtenção dos dados: vide **Apêndice II – Detalhes adicionais para a coleta dos datasets originais**.

- Data da coleta: 19/3/2021;

Possui metadados: sim, mas sem descrição das variáveis

(<https://metadados.ibge.gov.br/consulta/estatisticos/operacoes-estatisticas/CD>).

## 2.2. Base auxiliar

Para que fosse possível a integração das bases acima, foi necessária a criação, em cada uma delas, de um campo padronizado, chamado ‘municipio-uf’. Esse campo é resultado da transformação de campos existentes nos próprios datasets. Como, no entanto, as bases têm origem diversa, esse campo padronizado não apresentou informação totalmente idêntica entre as bases, dada a existência, por exemplo, de grafias diversas, de alterações significativas nos nomes em diferentes momentos etc. Diante disso, foi necessária a coleta da base abaixo, que contém a relação mais recente dos municípios do Brasil publicada pelo IBGE, de modo a ser esta a referência a partir da qual os dados serão uniformizados.

### viii. Tabela Auxiliar: Códigos dos Municípios - 2020 (DTB\_2020\_v2.zip)

- Descrição:

A Tabela de Códigos de Municípios do IBGE apresenta a lista dos municípios brasileiros associados a um código composto de 7 dígitos, sendo os dois primeiros referentes ao código da Unidade da Federação.

É atualizada sistematicamente de forma a incluir as alterações decorrentes do desdobramento de municípios e, consequentemente, da criação de novos municípios, mudanças de nome dos municípios, como também de processos de fusão que resultam na extinção ou modificação de nome de algum município.

A informação mais recente é a do ano de 2020.

Trata-se de um arquivo compactado que contém nossa base de interesse: “RELATORIO\_DTB\_BRASIL\_MUNICIPIO.xls”

- Fonte: Site do Instituto Brasileiro de Geografia e Estatística na Internet ([https://geoftp.ibge.gov.br/organizacao\\_do\\_territorio/estrutura\\_territorial/divisao\\_territorial/](https://geoftp.ibge.gov.br/organizacao_do_territorio/estrutura_territorial/divisao_territorial/))
- Endereço eletrônico:  
[https://geoftp.ibge.gov.br/organizacao\\_do\\_territorio/estrutura\\_territorial/divisao\\_territorial/2020/DTB\\_2020\\_v2.zip](https://geoftp.ibge.gov.br/organizacao_do_territorio/estrutura_territorial/divisao_territorial/2020/DTB_2020_v2.zip)
- Informações adicionais sobre a obtenção dos dados: para esta base, nenhum detalhe adicional para obtenção dos dados é necessário, vez que o download do arquivo é feito diretamente a partir do link acima.
- Data da coleta: 19/3/2021;
- Possui metadados: não.

### **3. Processamento/Tratamento de Dados**

#### **3.1. Transformações preliminares**

Para que as bases descritas no título anterior pudessem ser integradas umas às outras e, então, comporem um único *dataset* (íntegro, sem dados ausentes, sem inconsistências) uma série de transformações preliminares foram necessárias.

As principais ações preparatórias promovidas sobre os arquivos originais seguem mencionadas abaixo e seus detalhes podem ser consultados no **Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb**.

##### **- Descompactação de arquivos.**

Como visto, as bases “**iii. Produto Interno Bruto dos Municípios (“base\_de\_dados\_2010\_2018.xls.zip”)**” e “**viii. Tabela Auxiliar: Códigos dos Municípios - 2020 (DTB\_2020\_v2.zip)**” são disponibilizadas no formato compactado “.zip”.

Para sua descompactação foi criada a função “*descompactar* (*nome\_arquivo\_compactado*)”, abaixo, que se vale da biblioteca “*zipfile*”, e recebe como argumento o nome do arquivo a ser descompactado.

```
In [6]: from zipfile import ZipFile

def descompactar(nome_arquivo_compactado):
    # lê o arquivo compactado e extrai o conteúdo
    with ZipFile (nome_arquivo_compactado, 'r') as zip:
        zip.extractall()
        zip.printdir()
    return "Concluído"
```

Figura. Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

Exemplo de chamado à função.

```
In [29]: arquivo_a_descompactar = 'base_de_dados_2010_2018.xls.zip'
descompactar(arquivo_a_descompactar) ## chama a função declarada no início do notebook
```

File Name	Modified	Size
PIB dos Municípios - base de dados 2010-2018.xls	2020-12-11 14:42:38	29066752

```
Out[29]: 'Concluído'
```

Figura. Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

## - Visualização, análise e importação das bases originais

Após as descompactações acima, passa-se a ter apenas arquivos com as extensões “.csv”, “.xls” e “xlsx”.

De modo a instanciar os *dataframes* apenas com as informações de interesse, importante, primeiro, analisar o conteúdo dos arquivos para identificar e excluir possíveis resíduos (cabeçalhos, rodapés, linhas em branco e totalizadores).

Em relação aos arquivos “.csv” a exibição de uma prévia de seu conteúdo se dá pela função “*ler\_csv(nome\_arquivo)*”, abaixo, e se vale do módulo “*csv*” da biblioteca padrão do Python.

```
In [2]: ## Função utilizada para visualizar a estrutura do arquivo '.csv' a ser importado.

import csv
import chardet

def ler_csv(nome_arquivo):

    with open(nome_arquivo, 'rb') as rawdata:
        result = chardet.detect(rawdata.read(100000))
        encoding_arquivo = result['encoding']

    with open(nome_arquivo, newline='', encoding = encoding_arquivo) as f:
        reader = csv.reader(f)
        try:
            print(reader.line_num)
            for row in reader:
                print(reader.line_num, row)
        except csv.Error as e:
            sys.exit('file {}, line {}: {}'.format(filename, reader.line_num, e))
    return encoding_arquivo
```

Figura. Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

Ao ser chamada a função, cujo argumento é o nome do arquivo a ser visualizado, o resultado é similar ao da figura abaixo.

```
In [51]: nome_arquivo = 'tabela200.csv'
ler_csv(nome_arquivo) #Chama a função para a leitura do arquivo '.csv'

0
1 ['\ufe0f" Tabela 200 - População residente', ' por sexo', ' situação e grupos de idade - Amostra - Características Gerais da População"]]
2 ['Variável - População residente (Pessoas)']
3 ['Município', 'Ano', 'Situação do domicílio', 'Sexo', 'Grupo de idade']
4 ['Município', 'Ano', 'Situação do domicílio', 'Sexo', 'Total', '0 a 4 anos', '5 a 9 anos', '10 a 14 anos', '15 a 19 anos', '20 a 24 anos', '25 a 29 anos', '30 a 34 anos', '35 a 39 anos', '40 a 44 anos', '45 a 49 anos', '50 a 54 anos', '55 a 59 anos', '60 a 64 anos', '65 a 69 anos', '70 a 74 anos', '75 a 79 anos', '80 a 84 anos', '85 a 89 anos', '90 a 94 anos', '95 a 99 anos', '100 anos ou mais']
5 ['Alta Floresta D'Oeste (RO)', '2010', 'Total', '24392', '1851', '2107', '2401', '2588', '2200', '2092', '1867', '1795', '1715', '1566', '1178', '888', '676', '551', '457', '287', '115', '35', '10', '12', '-']
6 ['Ariquemes (RO)', '2010', 'Total', '90353', '7342', '8286', '9256', '9041', '8981', '8458', '7668', '6944', '6416', '5042', '3982', '2915', '2009', '1665', '1089', '567', '513', '109', '39', '19', '13']
7 ['Cabixi (RO)', '2010', 'Total', '6313', '515', '507', '604', '595', '448', '522', '462', '457', '467', '415', '406', '278', '228', '148', '134', '73', '44', '7', '1', '1', '3']
8 ['Cacoal (RO)', '2010', 'Total', '78574', '5923', '6275', '7473', '7859', '7758', '7018', '6617', '6067', '5647', '4832', '4072', '2671', '2048', '1582', '1154', '854', '434', '237', '46', '7', '-']
9 ['Cerejeiras (RO)', '2010', 'Total', '17029', '1218', '1357', '1635', '1719', '1560', '1410', '1321', '1249', '1268', '1129', '836', '629', '508', '478', '302', '225', '111', '43', '24', '6', '-']
10 ['Colorado do Oeste (RO)', '2010', 'Total', '18501', '1401', '1421', '1668', '1787', '1639', '1628', '1481', '136
```

Figura. Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

Nesta figura é possível perceber que antes do cabeçalho (linha 4) há 3 (três) outras linhas que deverão ser ignoradas.

Movendo a barra de rolagem vertical para baixo, é possível ver o conteúdo final do mesmo arquivo (figura abaixo), e se constata que a última linha de interesse é a 5570 e que, assim, há 14 linhas (5571 a 5592 – linhas em branco não são numeradas) a serem descartadas.

```
In [51]: nome_arquivo = 'tabela200.csv'
ler_csv(nome_arquivo) #Chama a função para a leitura do arquivo '.csv'

'327', '281', '282', '225', '142', '116', '66', '32', '3', '2', '3', '-']
5570 ['Brasília (DF)', '2010', 'Total', 'Total', '2570160', '189171', '200087', '219091', '220178', '245521', '267638', '2524
32', '212348', '187844', '157673', '124462', '95703', '69884', '48720', '36479', '21644', '12252', '5946', '2425', '530', '13
2']
5571 ['Fonte: IBGE - Censo Demográfico']
5572 []
5573 ['Notas']
5574 ['1 - Para o ano de 1991, dados do Universo. Para os demais anos, dados da Amostra']
5575 ['2 - Até o ano de 1991 os grupos de idade vão até 80 anos ou mais; a partir de 2000, vão até 100 anos ou mais.']
5576 ['']
5577 ['Legenda']
5578 ['Símbolo', 'Significado']
5580 ['- ', 'Zero absoluto, não resultante de um cálculo ou arredondamento.\nEx: Em determinado município não existem pessoas
de 14 anos de idade sem instrução.']
5583 ['0 ', 'Zero resultante de um cálculo ou arredondamento.\nEx: A inflação do feijão em determinada Região Metropolitana fo
i.\nDeterminado município produziu 400 kg de sementes de girassol e os dados da tabela são expressos em toneladas.']
5585 ['X ', 'Valor inibido para não identificar o informante.\nEx: Determinado município só possui uma empresa produtora de ci
mento, logo o valor de sua produção deve ser inibido.']
5587 ['.. ', 'Valor não se aplica.\nEx: Não se pode obter o total da produção agrícola em determinado município quando os prod
```

Figura. Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

Cientes do intervalo de interesse, é possível, então, comandar a leitura do arquivo (`pd.read_csv`) definindo os parâmetros (`skiprows=3` e `skipfooter=14`).

```
In [52]: IBGE_Censo_Idade=pd.read_csv(nome_arquivo, delimiter=',', engine='python', skiprows=3, skipfooter=14)
IBGE_Censo_Idade
```

Out[52]:

	Município	Ano	Situação do domicílio	Sexo	Total	0 a 4 anos	5 a 9 anos	10 a 14 anos	15 a 19 anos	20 a 24 anos	25 a 29 anos	30 a 34 anos	35 a 39 anos	40 a 4 anos
0	Alta Floresta D'Oeste (RO)	2010	Total	Total	24392	1851	2107	2401	2588	2200	2092	1867	1795	171
1	Ariquemes (RO)	2010	Total	Total	90353	7342	8286	9256	9041	8981	8458	7668	6944	641
2	Cabixi (RO)	2010	Total	Total	6313	515	507	604	595	448	522	462	457	46
3	Cacoal (RO)	2010	Total	Total	78574	5923	6275	7473	7859	7758	7018	6617	6067	564
4	Cerejeiras (RO)	2010	Total	Total	17029	1218	1357	1635	1719	1560	1410	1321	1249	126
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5561	Vianópolis (GO)	2010	Total	Total	12548	786	1008	1129	1085	1039	1056	1127	1028	95
5562	Vicentínópolis (GO)	2010	Total	Total	7371	592	587	622	640	697	669	655	607	58
5563	Vila Boa (GO)	2010	Total	Total	4735	447	481	475	429	456	567	436	317	29
5564	Vila Propício (GO)	2010	Total	Total	5145	357	490	508	468	360	363	382	369	36
5565	Brasília (DF)	2010	Total	Total	2570160	189171	200087	219091	220178	245521	267638	252432	212348	18784

5566 rows × 26 columns

Figura. Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

Em relação aos arquivos com extensão “.xls” e “xlsx”, a visualização prévia do conteúdo, de modo a analisar sua estrutura original e as linhas que efetivamente deverão ser lidas, foi feito diretamente por meio do comando `pd.read_excel`.

```
In [7]: nome_arquivo = 'arrecadacao-da-receita-administrada-pela-rfb-por-municipio-2018.xlsx'
planilha_interesse = 'TOTAL'
arrecadacao_2018 = pd.read_excel(nome_arquivo,sheet_name=planilha_interesse)
arrecadacao_2018
```

Out[7]:

		Unnamed: 0	Unnamed: 1	Unnamed: 2
0	ARRECADAÇÃO DAS RECEITAS ADMINISTRADAS PELA RF...		NaN	NaN
1		TOTAL	NaN	NaN
2		PERÍODO: 2018	NaN	NaN
3		UNIDADE: R\$ 1,00	NaN	NaN
4		MUNICÍPIOS	UF	ARRECADAÇÃO
...		...	...	...
5578		ZE DOCA	MA	1.326e+07
5579		ZERADO	-	1.0381e+10
5580		ZORTEA	SC	4.23176e+06
5581		{Ñ CLASS}	-	5.15863e+07
5582		TOTAL	-	1.39896e+12

5583 rows × 3 columns

Figura. Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

Como se observa no exemplo acima, as quatro primeiras linhas do arquivo são apenas informativas e não deverão ser carregadas (assim será necessário configurar o parâmetro *skiprows*). Vê-se igualmente que, no fim do arquivo, há linhas que não são de interesse (valores não classificados e o total), de modo que também será necessário limitar a importação dos dados, configurando o parâmetro *skipfooter*.

```
In [8]: arrecadacao_2018 = pd.read_excel(nome_arquivo,sheet_name=planilha_interesse,skiprows=5, skipfooter=2)
arrecadacao_2018
```

Out[8]:

	MUNICÍPIOS	UF	ARRECADAÇÃO
0	ABADIA DE GOIAS	GO	1.483209e+07
1	ABADIA DOS DOURADOS	MG	8.950170e+06
2	ABADIANIA	GO	1.779163e+07
3	ABAETE	MG	3.525459e+07
4	ABAETETUBA	PA	5.956002e+07
...	...	...	...
5571	ZABELE	PB	1.588994e+06
5572	ZACARIAS	SP	3.651978e+06
5573	ZE DOCA	MA	1.325997e+07
5574	ZERADO	-	1.038097e+10
5575	ZORTEA	SC	4.231760e+06

5576 rows × 3 columns

Figura. Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

## - Criação da variável padronizada, chamada “município-uf”

O passo seguinte foi a criação de uma nova variável, padronizada, chamada ‘municipio-uf’ que será a chave comum para *joins/merges* entre os *dataframes*. Essa variável será derivada de outras já existentes nos *datasets* originais.

Abaixo, um exemplo. Veja que, originalmente, os nomes dos municípios (variável Município) possuem tanto maiúsculas como minúsculas, há caracteres especiais e a sigla da unidade da federação aparece entre parêntesis. O padrão adotado foi converter para maiúsculas, remover os caracteres especiais e incluir a sigla da unidade da federação após um hífen.

```
In [20]: # Criando a nova coluna 'municipio-uf'.
IBGE_Empresas['municipio-uf']=IBGE_Empresas['Município'].str[:5].str.upper() \
+'-'+IBGE_Empresas['Município'].str[-3:-1].str.upper()

# Removendo caracteres especiais da coluna recém-criada.
IBGE_Empresas['municipio-uf'] = IBGE_Empresas['municipio-uf'].apply(remover_caracteres_especiais)

IBGE_Empresas.head()
```

Out[20]:

	Município	Ano	Número de unidades locais (Unidades)	Número de empresas e outras organizações atuantes (Unidades)	Pessoal ocupado total (Pessoas)	Pessoal ocupado assalariado (Pessoas)	Pessoal assalariado médio (Pessoas)	Salários e outras remunerações (Mil Reais)	Salário médio mensal (Salários mínimos)	Salário médio mensal em reais (Reais)	municipio-uf
0	Alta Floresta D'Oeste (RO)	2018	521	513	3271	2584	2557.86	57902	1.8	1741.31	ALTA FLORESTA D'OESTE-RO
1	Ariquemes (RO)	2018	2507	2407	19896	16578	16683.88	422650	2.0	1948.68	ARIQUEMES-RO
2	Cabixi (RO)	2018	79	77	607	535	540.03	11744	1.8	1672.78	CABIXI-RO
3	Cacoal (RO)	2018	2230	2139	19942	16815	17093.05	431101	2.0	1940.07	CACOAL-RO
4	Cerejeiras (RO)	2018	389	377	2482	2044	2076.75	52751	2.0	1953.89	CEREJEIRAS-RO

Figura. Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

Para garantir a uniformidade, após a criação da nova variável, procedeu-se à remoção de caracteres especiais por meio da função “*remover\_caracteres\_especiais(nome)*”, abaixo.

```
In [4]: ## Função utilizada para remover caracteres especiais.
```

```
def remover_caracteres_especiais(nome):
    nome = nome.replace('Á', 'A').replace('Ã', 'A').replace('À', 'A').replace('À', 'A')\
    .replace('É', 'E').replace('Ê', 'E').replace('Í', 'I').replace('Ó', 'O').replace('Ô', 'O')\
    .replace('Ô', 'O').replace('Ú', 'U').replace('Ü', 'U').replace('Ҫ', 'C')
    return nome
```

Figura. Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

Essa função é então aplicada a todo o conteúdo da recém-criada variável (‘municipio-uf’), conforme exemplo a seguir.

```
In [20]: # Criando a nova coluna 'municipio-uf'.
IBGE_Empresas['municipio-uf']=IBGE_Empresas['Município'].str[:-5].str.upper() \
+'-' +IBGE_Empresas['Município'].str[-3:-1].str.upper()

# Removendo caracteres especiais da coluna recém-criada.
IBGE_Empresas['municipio-uf'] = IBGE_Empresas['municipio-uf'].apply(remover_caracteres_especiais)

IBGE_Empresas.head()
```

Figura. Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

## - Manutenção, apenas, das informações relativas ao ano de interesse.

Foram mantidos nos *datasets* apenas os valores relativos ao ano de interesse (2018), exceto para os *datasets* relativos ao Censo realizado pelo IBGE, para os quais as informações disponíveis são apenas as do ano de 2010. Exemplo abaixo.

```
In [32]: # Vamos agora manter no dataframe exclusivamente o ano de 2018

IBGE_PIB = IBGE_PIB.loc[IBGE_PIB['Ano'] == 2018]
print (IBGE_PIB.shape)
```

(5570, 43)

Figura. Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

## - Renomeação de variáveis

As variáveis foram, então, renomeadas para que lhes fosse atribuído um prefixo com o número de ordem da base a que pertencem (“01\_” para as variáveis cuja origem é o *dataset i. Arrecadação das Receitas Administradas pela RFB, por Município*; “02\_”, para aquelas com origem em *ii. Tabela 1685 - Unidades locais, empresas e outras organizações atuantes, pessoal ocupado total, pessoal ocupado assalariado, pessoal assalariado médio, salários e outras remunerações e salário médio mensal (tabe-la1685.csv)*; e assim por diante).

Também foram atribuídos, em alguns casos, nomes mais sucintos. Assim, por exemplo, a variável “**Valor adicionado bruto total, a preços correntes (R\$ 1.000)**” foi renomeada para “**03\_vlr\_adic\_bruto\_total**”.

```

IBGE_PIB.columns

Out[36]: Index(['Ano', 'Nome da Grande Região', 'Sigla da Unidade da Federação',
       'Nome da Unidade da Federação', 'Código do Município',
       'Nome do Município', 'Nome da Mesorregião', 'Nome da Microrregião',
       'Nome da Região Geográfica Imediata',
       'Nome da Região Geográfica Intermediária',
       'Valor adicionado bruto da Agropecuária, \na preços correntes\n(R$ 1.000)',
       'Valor adicionado bruto da Indústria,\nna preços correntes\n(R$ 1.000)',
       'Valor adicionado bruto dos Serviços,\nna preços correntes \n- exceto Administração, defesa, educação e saúde públicas e
seguridade social\n(R$ 1.000)',
       'Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, \nna preços correntes\n
(R$ 1.000)',
       'Valor adicionado bruto total, \nna preços correntes\n(R$ 1.000)',
       'Impostos, líquidos de subsídios, sobre produtos, \nna preços correntes\n(R$ 1.000)',
       'Produto Interno Bruto, \nna preços correntes\n(R$ 1.000)',
       'Produto Interno Bruto per capita, \nna preços correntes\n(R$ 1,00)',
       'Atividade com maior valor adicionado bruto',
       'Atividade com segundo maior valor adicionado bruto',
       'Atividade com terceiro maior valor adicionado bruto', 'municipio-uf'],
      dtype='object')

In [37]: # Vamos também renomear as colunas para nomes mais sucintos...

IBGE_PIB.columns=['03_ano','03_grande_regiao','03_uf','03_nome_uf','03_cod_municipio','03_municipio',
      '03_mesorregiao','03_microregiao','03_Regiao_imediata','03_Regiao_intermediaria',
      '03_vlr_adic_bruto_agropecuaria','03_vlr_adic_bruto_industria','03_vlr_adic_bruto_servicos',
      '03_vlr_adic_bruto_administracao','03_vlr_adic_bruto_total','03_impostos_sobre_produtos',
      '03_pib','03_pib_per_capita','03_atividade_principal_primeira',
      '03_atividade_principal_segunda','03_atividade_principal_terceira', 'municipio-uf']

```

Figura. Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

### **- Criação, em alguns casos, de novas colunas derivadas (enriquecimento dos dados).**

De modo a permitir comparações, em mesma base, entre os diversos municípios, algumas variáveis foram transformadas para valores relativos. Assim, por exemplo, foi criada a variável “02\_unidades\_atuantes\_%”, como resultado da operação:

$$02.\text{unidades}.\text{atuantes}.\% = \frac{02.\text{unidades}.\text{atuantes}}{02.\text{unidades}} \times 100$$

Na sequência, as variáveis transformadas puderam ser excluídas (“02\_unidades\_atuantes”, no exemplo acima).

```
In [77]: # Temos, então, 5565 municípios com informações completas.

#Vamos, agora, transformar a informação como um percentual do total.
IBGE_Censo_Instrucao['06_Sem_Instrução_e_Fundamental_Incompleto%'] = \
IBGE_Censo_Instrucao['Sem instrução e fundamental incompleto']/IBGE_Censo_Instrucao['Total']

IBGE_Censo_Instrucao['06_Medio_Incompleto%'] = \
IBGE_Censo_Instrucao['Fundamental completo e médio incompleto']/IBGE_Censo_Instrucao['Total']*100

IBGE_Censo_Instrucao['06_Superior_Incompleto%'] = \
IBGE_Censo_Instrucao['Médio completo e superior incompleto']/IBGE_Censo_Instrucao['Total']*100

IBGE_Censo_Instrucao['06_Superior_Completo%'] = \
IBGE_Censo_Instrucao['Superior completo']/IBGE_Censo_Instrucao['Total']*100

IBGE_Censo_Instrucao['06_Indeterminado%'] = \
IBGE_Censo_Instrucao['Não determinado']/IBGE_Censo_Instrucao['Total']*100

#Podemos agora apagar as colunas originais.
cols=['Sem instrução e fundamental incompleto',
      'Fundamental completo e médio incompleto',
      'Médio completo e superior incompleto', 'Superior completo',
      'Não determinado']
IBGE_Censo_Instrucao = IBGE_Censo_Instrucao.drop(cols,1)

IBGE_Censo_Instrucao.head()

Out[77]:
   Total  município-uf  06_Sem_Instrução_e_Fundamental_Incompleto%  06_Medio_Incompleto%  06_Superior_Incompleto%  06_Superior_Completo%
0  20434  ALTA FLORESTA D'OESTE-RO  0.659195  14.314378  15.273564  3.919937
1  100000  ARIQUIFIMES-
```

Figura. Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

Outro caso de criação de novas variáveis teve por objetivo reduzir o número de faixas de valores em que os dados originalmente haviam sido divididos. No exemplo abaixo, as 21 faixas de valor originais foram substituídas pelas 4 novas faixas derivadas.

```
In [62]: # Temos, então, 5565 municípios com informações completas.

#Vamos, agora, reduzir o número de faixas de valores considerando ranges de 20 anos em vez de 5 anos e
#transformar a informação como um percentual do total.
IBGE_Censo_Idade['05_0-19_anos%'] = (IBGE_Censo_Idade['0 a 4 anos']+IBGE_Censo_Idade['5 a 9 anos']\ 
+IBGE_Censo_Idade['10 a 14 anos']+IBGE_Censo_Idade['15 a 19 anos'])/IBGE_Censo_Idade['Total']*100

IBGE_Censo_Idade['05_20-39_anos%'] = (IBGE_Censo_Idade['20 a 24 anos']+IBGE_Censo_Idade['25 a 29 anos']\ 
+IBGE_Censo_Idade['30 a 34 anos']+IBGE_Censo_Idade['35 a 39 anos'])/IBGE_Censo_Idade['Total']*100

IBGE_Censo_Idade['05_40-59_anos%'] = (IBGE_Censo_Idade['40 a 44 anos']+IBGE_Censo_Idade['45 a 49 anos']\ 
+IBGE_Censo_Idade['50 a 54 anos']+IBGE_Censo_Idade['55 a 59 anos'])/IBGE_Censo_Idade['Total']*100

IBGE_Censo_Idade['05_60+anos%'] = (IBGE_Censo_Idade['60 a 64 anos']+IBGE_Censo_Idade['65 a 69 anos']\ 
+IBGE_Censo_Idade['70 a 74 anos']+IBGE_Censo_Idade['75 a 79 anos']+IBGE_Censo_Idade['80 a 84 anos']\ 
+IBGE_Censo_Idade['85 a 89 anos']+IBGE_Censo_Idade['90 a 94 anos']+IBGE_Censo_Idade['95 a 99 anos']\ 
+IBGE_Censo_Idade['100 anos ou mais'])/IBGE_Censo_Idade['Total']*100

# Agora apagar as colunas originais.
cols=['0 a 4 anos', '5 a 9 anos', '10 a 14 anos', '15 a 19 anos',
      '20 a 24 anos', '25 a 29 anos', '30 a 34 anos', '35 a 39 anos',
      '40 a 44 anos', '45 a 49 anos', '50 a 54 anos', '55 a 59 anos',
      '60 a 64 anos', '65 a 69 anos', '70 a 74 anos', '75 a 79 anos',
      '80 a 84 anos', '85 a 89 anos', '90 a 94 anos', '95 a 99 anos',
      '100 anos ou mais']
IBGE_Censo_Idade = IBGE_Censo_Idade.drop(cols,1)

IBGE_Censo_Idade.info()
```

Figura. Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

## - Remoção de variáveis redundantes e com valores nulos ou constantes

Foram identificadas e removidas as variáveis que possuíam valores nulos e que não agregariam valor ao presente estudo.

Para essa identificação, definiu-se a função “*identificar\_faltantes(df)*”, abaixo.

```
In [5]: ## Função para identificar se há valores faltantes, ou NaN em um dataframe

def identificar_faltantes(df):

    #Mostra o tamanho do meu dataset
    print("Shape do dataframe:",df.shape)

    #mostra em que coluna estão esses elementos faltantes
    colunas_faltantes = df.columns[df.isna().any()].tolist()
    print("Colunas que contêm valores faltantes:",colunas_faltantes)

    #traz o número de linhas que restariam se eu tirasse os que seriam excluídos pelo dropna
    print("Número de linhas com valores faltantes:",df.shape[0]-df.dropna().shape[0],"\n")

    return colunas_faltantes
```

Figura. Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

A seguir, um exemplo da identificação dessas variáveis com valores faltantes e de sua exclusão.

```
In [35]: # Chama a função para identificar se há valores faltantes ou NaN.
colunas_faltantes = identificar_faltantes(IBGE_PIB)

Shape do dataframe: (5570, 44)
Colunas que contêm valores faltantes: ['Região Metropolitana', 'Código Concentração Urbana', 'Nome Concentração Urbana', 'Tipo Concentração Urbana', 'Código Arranjo Populacional', 'Nome Arranjo Populacional']
Número de linhas com valores faltantes: 5164
```

```
In [36]: # Além de possuírem valores nulos, as colunas acima não agregam valor ao modelo e serão excluídas.
IBGE_PIB = IBGE_PIB.drop(colunas_faltantes,1)

# Além delas, também serão excluídas as colunas que se referem a códigos para os quais os nomes estão em colunas específicas.
# Ex. O Código de mesorregião será excluído e mantido o Nome da Mesorregião
# Apenas manterei, temporariamente, o código do município para futuro uso.
colunas_a_excluir=['Código da Grande Região','Código da Unidade da Federação',
'Código da Mesorregião','Código da Microrregião','Código da Região Geográfica Imediata',
'Município da Região Geográfica Imediata','Código da Região Geográfica Intermediária',
'Município da Região Geográfica Intermediária',
'Hierarquia Urbana','Hierarquia Urbana (principais categorias)',
'Código da Região Rural','Nome da Região Rural','Região rural (segundo classificação do núcleo)',
'Amazônia Legal','Semiárido','Cidade/Região de São Paulo']

IBGE_PIB = IBGE_PIB.drop(colunas_a_excluir,1)
IBGE_PIB.columns
```

```
Out[36]: Index(['Ano', 'Nome da Grande Região', 'Sigla da Unidade da Federação',
'Nome da Unidade da Federação', 'Código do Município',
'Nome do Município', 'Nome da Mesorregião', 'Nome da Microrregião',
'Nome da Região Geográfica Imediata',
'Nome da Região Geográfica Intermediária'])
```

Figura. Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

A figura também exemplifica remoções de variáveis de colunas redundantes (quando havia o código e o nome de determinada variável, manteve-se apenas o nome).

Por fim, foram também removidas as variáveis com valores constantes (fixos) nos *datasets* (por exemplo, a variável ‘Ano’ tinha sempre o valor “2010” nos *datasets* relacionados ao Censo de 2010).

### 3.2. Bases de dados transformadas

Assim, após promovidas as transformações preparatórias necessárias acima descritas (vide **Apêndice III – Resumo das transformações nos datasets originais** e **Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb**), as bases originais deram origem, respectivamente, aos seguintes arquivos:

#### - ntbk 01\_01 - rfb arrecadacao 2018.csv

Resultado de transformações no *dataset* original: i. **Arrecadação das Receitas Administradas pela RFB, por Município (arrecadacao-da-receita-administrada-pela-rfb-por-municipio-2018.xlsx)**.

Esse novo *dataset* foi, então, instanciado como *arrecadacao\_2018*, vide abaixo.

```
In [97]: #Ntbk 01_01 - RFB Arrecadacao 2018.csv
arrecadacao_2018.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5576 entries, 0 to 5575
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   municipio-uf    5576 non-null   object  
 1   01_arrecadacao_2018 5576 non-null   float64 
dtypes: float64(1), object(1)
memory usage: 87.2+ KB
```

Figura. Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

#### - ntbk 01\_02 - ibge empresas 2018.csv

Resultado de transformações no *dataset* original: ii. **Tabela 1685 - Unidades locais, empresas e outras organizações atuantes, pessoal ocupado total, pes-**

**soal ocupado assalariado, pessoal assalariado médio, salários e outras remunerações e salário médio mensal (*tabela1685.csv*)**

Esse novo *dataset* foi, então, instanciado como IBGE\_empresas, vide abaixo.

```
In [98]: #Ntbk 01_02 - IBGE Empresas 2018.csv
IBGE_Empresas.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 5570 entries, 0 to 5569
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   municipio-uf      5570 non-null   object  
 1   02_unidades       5570 non-null   int64  
 2   02_unidades_atuantes_% 5570 non-null   float64 
 3   02_ocupados_total  5570 non-null   int64  
 4   02_ocupados_assalariados_% 5570 non-null   float64 
 5   02_ocupados_assalariados_medio 5570 non-null   float64 
 6   02_salario_medio_mensal_em_SM 5570 non-null   float64 
 7   02_salario_medio_mensal_em_Reais 5570 non-null   float64 
dtypes: float64(5), int64(2), object(1)
memory usage: 391.6+ KB
```

Figura. Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

**- ntbk 01\_03 – ibge pib 2018.csv**

Resultado de transformações no *dataset* original: **iii. Produto Interno Bruto dos Municípios ('PIB dos Municípios - base de dados 2010-2018.xls).**

Esse novo *dataset* foi, então, instanciado como IBGE\_PIB, vide abaixo.

```
In [99]: #Ntbk 01_03 - IBGE PIB 2018.csv
IBGE_PIB.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 5570 entries, 44545 to 50114
Data columns (total 20 columns):
 #   Column           Non-Null Count Dtype  
 --- 
 0   municipio-uf      5570 non-null   object  
 1   03_nome_uf        5570 non-null   object  
 2   03_uf              5570 non-null   object  
 3   03_cod_municipio    5570 non-null   object  
 4   03_grande_regiao    5570 non-null   object  
 5   03_mesorregiao     5570 non-null   object  
 6   03_microregiao     5570 non-null   object  
 7   03_regiao_imediata 5570 non-null   object  
 8   03_regiao_intermediaria 5570 non-null   object  
 9   03_vlr_adic_bruto_total 5570 non-null   float64 
 10  03_vlr_adic_bruto_agropecuaria_% 5570 non-null   float64 
 11  03_vlr_adic_bruto_industria_% 5570 non-null   float64 
 12  03_vlr_adic_bruto_servicos_% 5570 non-null   float64 
 13  03_vlr_adic_bruto_administracao_% 5570 non-null   float64 
 14  03_impostos_sobre_produtos 5570 non-null   float64 
 15  03_pib              5570 non-null   float64 
 16  03_pib_per_capita    5570 non-null   float64 
 17  03_atividade_principal_primeira 5570 non-null   object  
 18  03_atividade_principal_segunda 5570 non-null   object  
 19  03_atividade_principal_terceira 5570 non-null   object  
dtypes: float64(8), object(12)
memory usage: 913.8+ KB
```

Figura. Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

### - ntbk 01\_04 – ibge população residente 2018.csv

Resultado de transformações no *dataset* original: **iv. Tabela 6579 - População residente estimada (tabela6579.csv)**.

Esse novo *dataset* foi, então, instanciado como IBGE\_Residentes, vide abaixo.

```
In [100]: #Ntbk 01_04 - IBGE População Residente 2018.csv
IBGE_Residentes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5570 entries, 0 to 5569
Data columns (total 2 columns):
 #   Column            Non-Null Count  Dtype  
---  --  
 0   municipio-uf      5570 non-null   object  
 1   04_populacao_residente_2018  5570 non-null   int64  
dtypes: int64(1), object(1)
memory usage: 87.2+ KB
```

Figura. Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

### - ntbk 01\_05 – ibge censo 2010 idades.csv

Resultado de transformações no *dataset* original: **v. Tabela 200 - População residente, por sexo, situação e grupos de idade (tabela200.csv)**.

Esse novo *dataset* foi, então, instanciado como IBGE\_Censo\_Idade, vide abaixo.

```
In [101]: #Ntbk 01_05 - IBGE Censo 2010 Idades.csv
IBGE_Censo_Idade.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5565 entries, 0 to 5565
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype    
---  --  
 0   municipio-uf      5565 non-null   object  
 1   05_total_residentes 5565 non-null   float64  
 2   05_0-19_anos_%     5565 non-null   float64  
 3   05_20-39_anos_%    5565 non-null   float64  
 4   05_40-59_anos_%    5565 non-null   float64  
 5   05_60+_anos_%     5565 non-null   float64  
dtypes: float64(5), object(1)
memory usage: 304.3+ KB
```

Figura. Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

### - ntbk 01\_06 - ibge censo instrucao.csv

Resultado de transformações no *dataset* original: **vi. Tabela 1554 - Pessoas de 10 anos ou mais de idade, por nível de instrução (tabela1554.csv)**.

Esse novo *dataset* foi, então, instanciado como IBGE\_Censo\_Instrucao, vide abaixo.

```
In [102]: #Ntbk 01_06 - IBGE Censo Instrucao.csv
IBGE_Censo_Instrucao.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5565 entries, 0 to 5564
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   municipio-uf      5565 non-null   object  
 1   06_total_instrução 5565 non-null   int64  
 2   06_Sem_Instrução_e_Fundamental_Incompleto% 5565 non-null   float64 
 3   06_Medio_Incompleto_%    5565 non-null   float64 
 4   06_Superior_Incompleto_% 5565 non-null   float64 
 5   06_Superior_Completo_%   5565 non-null   float64 
 6   06_Indeterminado_%     5565 non-null   float64 
dtypes: float64(5), int64(1), object(1)
memory usage: 304.5+ KB
```

Figura. Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

### - ntbk 01\_07 – ibge censo rendimentos.csv

Resultado de transformações no *dataset* original: **vii. Tabela 2030 - Pessoas de 10 anos ou mais de idade, por classes de rendimento nominal mensal (tabela2030.csv)**.

Esse novo *dataset* foi, então, instanciado como IBGE\_Censo\_Rendimento, vide abaixo.

```
In [103]: #Ntbk 01_07 - IBGE Censo Rendimentos.csv
IBGE_Censo_Rendimento.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5565 entries, 0 to 5564
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   municipio-uf      5565 non-null   object  
 1   07_total_rendimentos 5565 non-null   int64   
 2   07_Ate_1SM_%       5565 non-null   float64 
 3   07_Ate_2SM_%       5565 non-null   float64 
 4   07_Ate_3SM_%       5565 non-null   float64 
 5   07_Ate_5SM_%       5565 non-null   float64 
 6   07_Ate_10SM_%      5565 non-null   float64 
 7   07_Ate_20SM_%      5565 non-null   float64 
 8   07_20SM+_%        5565 non-null   float64 
 9   07_Sem_Rendimento_% 5565 non-null   float64 
dtypes: float64(8), int64(1), object(1)
memory usage: 434.9+ KB
```

Figura. Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

### 3.3. Integração das bases transformadas e criação do *dataset consolidado*

#### - Integração (*merge/join*) dos *datasets* transformados

O objetivo agora é criar um único *dataset* consolidado, gerado a partir da integração (*join/merge*) dos *datasets* gerados pelo “*Ntbk 01. Coleta dos dados e transformações preliminares.ipynb*” (que são resultado dos ajustes e transformações dos *datasets* originais coletados).

Para isso, foi necessário promover transformações adicionais, agora em relação ao conteúdo dos *datasets*, de modo a garantir a integridade das informações.

Os detalhes desse processo podem ser consultados no **Apêndice VII – Ntbk 02. Criação do dataset consolidado - merges e saneamentos.ipynb**.

#### - Saneamento das divergências em relação aos nomes dos municípios

Adotou-se, por premissa neste estudo, garantir a preservação do maior número possível de municípios. Diante disso, a exclusão de ocorrências somente deveria ocorrer em último caso.

Em primeiro lugar, chama a atenção as diferentes dimensões dos *datasets* de interesse.

```
In [6]: arrecadacao_2018=pd.read_csv('ntbk 01_01 - rfb arrecadacao 2018.csv')
IBGE_Empresas=pd.read_csv('ntbk 01_02 - ibge empresas 2018.csv')
IBGE_PIB=pd.read_csv('ntbk 01_03 - ibge pib 2018.csv')
IBGE_Residentes=pd.read_csv('ntbk 01_04 - ibge população residente 2018.csv')
IBGE_Censo_Idade=pd.read_csv('ntbk 01_05 - ibge censo 2010 idades.csv')
IBGE_Censo_Instrucao=pd.read_csv('ntbk 01_06 - ibge censo instrucao.csv')
IBGE_Censo_Rendimento=pd.read_csv('ntbk 01_07 - ibge censo rendimentos.csv')

print ("Shapes:")
print ('-arrecadacao_2018',arrecadacao_2018.shape)
print ('-IBGE_Empresas',IBGE_Empresas.shape)
print ('-IBGE_PIB',IBGE_PIB.shape)
print ('-IBGE_Residentes',IBGE_Residentes.shape)
print ('-IBGE_Censo_Idade',IBGE_Censo_Idade.shape)
print ('-IBGE_Censo_Instrucao',IBGE_Censo_Instrucao.shape)
print ('-IBGE_Censo_Rendimento',IBGE_Censo_Rendimento.shape)

Shapes:
-arrecadacao_2018 (5576, 2)
-IBGE_Empresas (5570, 8)
-IBGE_PIB (5570, 20)
-IBGE_Residentes (5570, 2)
-IBGE_Censo_Idade (5565, 6)
-IBGE_Censo_Instrucao (5565, 7)
-IBGE_Censo_Rendimento (5565, 10)
```

Figura. Vide Apêndice VII – Ntbk 02. Criação do dataset consolidado - merges e saneamentos.ipynb.

Com a leitura das dimensões (*shapes*) das bases transformadas, observa-se, de pronto, que há divergências entre os valores e, possivelmente, inconsistências.

Sabe-se que atualmente, há, no Brasil, 5570 municípios, mesmo número contido nas bases indicadas por IBGE\_Empresas, IBGE\_PIB e IBGE\_Residentes. No entanto a base da RFB (indicada por arrecadacao\_2018) apresentava 5576 ocorrências. Por sua vez, as bases indicadas por IBGE\_Censo\_Idade, IBGE\_Censo\_Instrucao, IBGE\_Censo\_Rendimento continham apenas 5565 linhas.

De modo a uniformizar nossas bases a partir de uma referência comum e, assim, sanear as possíveis divergências foi importada a tabela *Códigos de Municípios* mais recente do site do IBGE (instanciado como IBGE\_Municipios), que é a do ano de 2020 (aqui referenciada por **viii. Tabela Auxiliar: Códigos dos Municípios – 2020 ('DTB\_2020\_v2.zip')**, conforme anteriormente descrita no título **2.2. Base auxiliar**).

Como já mencionado, a integração das bases de interesse foi precedida pela criação da variável padronizada, chamada ‘municipio-uf’.

Assim, para a criação do *dataset* consolidado, foram feitas uma sequência de integrações (*merges*), com os saneamentos necessários. O primeiro merge foi entre os *dataframes* **IBGE\_municipios** ('Tabela Auxiliar: Códigos dos Municípios – 2020 ('DTB\_2020\_v2.zip')') e **arrecadacao\_2018** ('ntbk 01\_01 - rfb arrecadacao 2018.csv'). Os demais *merges* continuaram a ser feitos entre dois *datasets*, sendo que um deles era o gerado com o último *merge* efetivado.

Por ser a informação mais atualizada, de modo a uniformizar os resultados, privilegiou-se sempre as informações contidas em **IBGE\_municipios**. Assim:

- municípios que não mais constem dessa relação foram excluídos;
- municípios com grafias diferentes tiveram seus nomes ajustados a partir dessa referência.

### Divergências apuradas com os merges.

Como primeiro passo, foram feitos *merges*, com parâmetro “*outer join*” de modo a preservar integralmente as bases sob integração e evidenciar as possíveis divergências. O exemplo abaixo é do primeiro *merge*, entre **IBGE\_municipios** e **arrecadacao\_2018** (*dataframes* instanciados com as bases viii e i, respectivamente).

```
In [12]: # Iniciamos com um outer join para identificar as ocorrências que não são mútuas entre as tabelas originais
print('Shape de IBGE_municipios',IBGE_municipios.shape)
print('Shape de arrecadacao_2018',arrecadacao_2018.shape)
Merge_01 = pd.merge(IBGE_municipios['municipio-uf'], arrecadacao_2018['municipio-uf'],how="outer", on='municipio-uf',indicator=Merge_01.shape
print('Shape de merge_01', Merge_01.shape)

## Identificando as divergências
diferencias=Merge_01[Merge_01['_merge']!='both']
print("\n", "Diferenças encontradas", diferencias.shape)
print (diferencias['_merge'].value_counts())
diferencias

< IPython.core.display.HTML>

Shape de IBGE_municipios (5570, 14)
Shape de arrecadacao_2018 (5576, 2)
Shape de merge_01 (5609, 2)

Diferenças encontradas (72, 2)
right_only    39
left_only     33
both         0
Name: _merge, dtype: int64

out[12]:
      municipio-uf   _merge
197 ELDORADO DO CARAJAS-PA left_only
259 SANTA IZABEL DO PARA-PA left_only
436 SAO VALERIO-TO left_only
440 TABOCAO-TO left_only
1076 ACU-RN left_only
```

Figura. Vide Apêndice VII – Ntbk 02. Criação do dataset consolidado - merges e saneamentos.ipynb.

A partir dos comandos acima, nota-se que há, nesse exemplo, a princípio, 33 municípios que aparecem somente em **IBGE\_municipios** (indicados com o 'left\_only') e outros 39 municípios que constam apenas em **arrecadacao\_2018** (indicados com 'right\_only').

Uma análise mais detida, porém, mostra que parte das inconsistências se dá por divergência de grafia (por exemplo: ACU-RN vs. ASSU-RN, ou GRAO-PARA-SC vs. GRAO PARA-SC) ou por uma mudança mais significativa de nomenclatura (como TABOCAO-TO vs. FORTALEZA DO TABOCAO-TO, ou EMBU DAS ARTES-SP vs. EMBU-SP). Há ainda situação curiosa de município com alteração total de seu nome (como o município de 'AUGUSTO SEVERO-RN' que passou a se chamar 'CAMPO GRANDE-RN')

Outra uma situação interessante: oficialmente, o município de 'BOA SAUDE-RN', que passou a se chamar 'JANUARIO CICCO' em 1953 (quando elevado a município). Emenda da Câmara Municipal n.º 01, de 02-02-1991, porém, retornou seu nome para BOA SAUDE, mas como não houve até o momento homologação por Lei Estadual, o IBGE desconsidera essa alteração. Porém todas as referências são para Boa Saúde (RN) - vide o site da prefeitura local <http://www.boasaude.rn.gov.br/historia-do-municipio.html>. Como se está privilegiando o definido pelo IBGE, foi mantido o nome JANUARIO CICCO.

Para esses ajustes de grafia, foi declarada a função “*corrigir\_nomes\_municípios(nome)*”, abaixo, que consolida e corrige as 37 divergências no conteúdo do campo “município-uf” apuradas com os *merges* entre os diversos *datasets*.

```
In [4]: ## Função utilizada para uniformizar os nomes dos municípios.
## A grafia à direita ser refere à atualização mais recente publicada pelo IBGE acerca dos municípios do Brasil.

def corrigir_nomes_municípios(nome):
    nome = nome.replace('AMPARO DA SERRA-MG', 'AMPARO DO SERRA-MG')\
    .replace('AMPARO DE SAO FRANCISCO-SE', 'AMPARO DO SAO FRANCISCO-SE')\
    .replace('AUGUSTO SEVERO-RN', 'CAMPO GRANDE-RN') \
    .replace('ASSU-RN', 'ACU-RN') \
    .replace('BALNEARIO DE PICARRAS-SC', 'BALNEARIO PICARRAS-SC').replace('BARAO DO MONTE ALTO-MG', 'BARAO DE MONTE ALTO-MG') \
    .replace('BELEM DE SAO FRANCISCO-PE', 'BELEM DO SAO FRANCISCO-PE').replace('BIRITIBA-MIRIM-SP', 'BIRITIBA MIRIM-SP') \
    .replace('BOA SAUDE-RN', 'JANUARIO CICCO-RN') \
    .replace('BOM JESUS-GO', 'BOM JESUS DE GOIAS-GO').replace('BRASOPOLIS-MG', 'BRAZOPOLIS-MG') \
    .replace('DONA EUSEBIA-MG', 'DONA EUZEBIA-MG').replace('ELDORADO DOS CARAJAS-PA', 'ELDORADO DO CARAJAS-PA') \
    .replace('EMBU-SP', 'EMBU DAS ARTES-SP').replace('FLORINIA-SP', 'FLORINEA-SP').replace('FORTALEZA DO TABOCAO-TO', 'TABOCAO-TO') \
    .replace('GRAO PARA-SC', 'GRAO-PARA-SC') \
    .replace('IGUARACI-PE', 'IGUARACY-PE').replace('LAGOA DO ITAENGA-PE', 'LAGOA DE ITAENGA-PE') \
    .replace('MOGI-MIRIM-SP', 'MOGI MERIM-SP').replace('MUNHOZ DE MELO-PR', 'MUNHOZ DE MELO-PR') \
    .replace('MUQUEM DE SAO FRANCISCO-BA', 'MUQUEM DO SAO FRANCISCO-BA') \
    .replace('OLHO D'AGUA DOS BORGES-RN', 'OLHO D'AGUA DO BORGES-RN') \
    .replace('PACAEMBU DAS ARTES-SP', 'PACAEMBU-SP').replace('PARATI-RJ', 'PARATY-RJ') \
    .replace('PASSA-VINTE-MG', 'PASSA VINTE-MG').replace('PINGO D'AGUA-MG', 'PINGO-D'AGUA-MG') \
    .replace('POXOREO-MT', 'POXOREU-MT').replace('PRESIDENTE CASTELO BRANCO-SC', 'PRESIDENTE CASTELLO BRANCO-SC') \
    .replace('SANTA CRUZ DO MONTE CASTELO-PR', 'SANTA CRUZ DE MONTE CASTELO-PR') \
    .replace('SANTA ISABEL DO PARA-PA', 'SANTA IZABEL DO PARA-PA').replace('SANTA TERESINHA-BA', 'SANTA TEREZINHA-BA') \
    .replace('SANTANA DO LIVRAMENTO-RS', 'SANTANA DO LIVRAMENTO-RS').replace('SAO DOMINGOS DE POMBAL-PB', 'SAO DOMINGOS-PB') \
    .replace('SAO THOME DAS LETRAS-MG', 'SAO TOME DAS LETRAS-MG') \
    .replace('SAO VALERIO DA NATIVIDADE-TO', 'SAO VALERIO-TO').replace('TRAJANO DE MORAIS-RJ', 'TRAJANO DE MORAES-RJ')

    return nome
```

Figura. Vide Apêndice VII – Ntbk 02. Criação do dataset consolidado - merges e saneamentos.ipynb.

Como a decisão é de privilegiar o contido em IBGE\_municipios, será alterada sempre a grafia do *dataframe* à direita no comando *pd.merge*.

```
In [13]: # Como no merge1 o dataframe à direita é arrecadacao_2018, será nele que faremos as adequações.
arrecadacao_2018['municipio-uf'] = arrecadacao_2018['municipio-uf'].apply(corriger_nomes_municípios)
```

Figura. Vide Apêndice VII – Ntbk 02. Criação do dataset consolidado - merges e saneamentos.ipynb.

Mais uma categoria de divergências apurada a partir dos *merges* é a duplicidade indevida de ocorrências para o mesmo município, mas sob a aparência de municípios distintos.

Explica-se melhor. Tome por exemplo o município de 'ASSIS CHATEAUBRIAND - PB' que passou a se chamar 'RIACHAO DO BACAMARTE-PB'. Ocorre que em arrecadacao\_2018 há informação de valores arrecadados tanto para um como para outro, quando, na verdade, trata-se do mesmo município.

```
In [14]: municipios_atuais = ['RIACHAO DO BACAMARTE-PB', 'TACIMA-PB', 'LIVRAMENTO DE NOSSA SENHORA-BA', 'JOCA CLAUDINO-PB']
municipios_extintos = ['ASSIS CHATEAUBRIAND-PB', 'CAMPO DE SANTANA-PB', 'LIVRAMENTO DO BRUMADO-BA', 'SANTAREM-PB']
duplas=zip(municipios_atuais, municipios_extintos)

for i in duplas:
    print("\n",i)

    ind_1 = arrecadacao_2018.index[arrecadacao_2018['municipio-uf']==i[0]].tolist()
    arrec_munic_1 = arrecadacao_2018.at[ind_1[0], '01_arrecadacao_2018']
    print (i[0]," - ",arrec_munic_1)

    ind_2 = arrecadacao_2018.index[arrecadacao_2018['municipio-uf']==i[1]].tolist()
    arrec_munic_2 = arrecadacao_2018.at[ind_2[0], '01_arrecadacao_2018']
    print (i[1]," - ",arrec_munic_2)

    soma_arrec = arrec_munic_1.astype(float) + arrec_munic_2.astype(float)
    print ("Soma"," - ", soma_arrec)

    arrecadacao_2018.at[ind_1[0], '01_arrecadacao_2018'] = soma_arrec
    print ("Novo valor para",i[0]," - ",arrecadacao_2018.at[ind_1[0],'01_arrecadacao_2018'])

('RIACHAO DO BACAMARTE-PB', 'ASSIS CHATEAUBRIAND-PB')
RIACHAO DO BACAMARTE-PB - 422067.12
ASSIS CHATEAUBRIAND-PB - 2696361.84000009
Soma - 3118428.9600000903
Novo valor para RIACHAO DO BACAMARTE-PB - 3118428.9600000903
```

Figura. Vide Apêndice VII – Ntbk 02. Criação do dataset consolidado - merges e saneamentos.ipynb.

Para sanear essa inconsistência, somou-se os dois valores e atribuiu-se o resultado a 'RIACHAO DO BACAMARTE-PB', excluindo, na sequência, a ocorrência para 'ASSIS CHATEAUBRIAND - PB'.

O mesmo foi feito com CAMPO DE SANTANA-PB e TACIMA-PB; com LIVRAMENTO DO BRUMADO - BA e LIVRAMENTO DE NOSSA SENHORA -BA; e com SANTAREM-PB e JOCA CLAUDINO-PB – sendo sempre mantidos os últimos e lhes atribuído a soma das arrecadações.

As inconsistências acima mencionadas, após seu saneamento, não impactaram o número total de municípios, que se mantinha 5570. No entanto, ao se fazer o *merge* com as tabelas relativas ao censo, foi necessário excluir 5 (cinco municípios), o que **fez com que no dataset final restassem 5565 municípios**. O motivo dessa exclusão é que esses cinco municípios tiveram sua instalação oficializada somente após 2010, isto é, após a data do último censo realizado pelo

IBGE, e assim caso mantidos implicariam em informações nulas para as variáveis com origem nos demais *datasets*.

**out[34]:**

		municipio-uf	_merge
224	MOJUI DOS CAMPOS-PA	left_only	
4341	BALNEARIO RINCAO-SC	left_only	
4505	PESCARIA BRAVA-SC	left_only	
4923	PINTO BANDEIRA-RS	left_only	
5160	PARAISO DAS AGUAS-MS	left_only	

Figura. Vide Apêndice VII – Ntbk 02. Criação do dataset consolidado - merges e saneamentos.ipynb.

### - Geração do *dataset consolidado* “ntbk 02 – dataset consolidado.csv”

A partir, então, da integração (*join/merge*) dos *datasets* acima, foi gerado o dataset “ntbk 02 – dataset consolidado.csv”, cujo conteúdo segue abaixo.

Variável	Não-nulas	Tipo	Descrição
municipio-uf	5565	object	Nome do município e sigla do Estado da Federação
01_arrecadacao_2018	5565	float64	Valor da arrecadação em 2018.
02_unidades	5565	int64	Número de unidades locais (Unidades)
02_unidades_atuantes_%	5565	float64	Número de empresas e outras organizações atuantes (Unidades) dividido pelo Número de unidades locais (Unidades)
02_ocupados_total	5565	int64	Pessoal ocupado total (Pessoas)
02_ocupados_assalariados_%	5565	float64	Pessoal ocupado assalariado (Pessoas) dividido pelo Pessoal ocupado total (Pessoas)
02_ocupados_assalariados_medio	5565	float64	Pessoal assalariado médio (Pessoas)
02_salario_medio_mensal_em_SM	5565	float64	Salário médio mensal (Salários mínimos)
02_salario_medio_mensal_em_Reais	5565	float64	Salário médio mensal em reais (Reais)
03_nome_uf	5565	object	Nome da Unidade da Federação
03_uf	5565	object	Sigla da Unidade da Federação
03_cod_municipio	5565	object	Código do Município
03_grande_regiao	5565	object	Nome da Grande Região
03_mesorregiao	5565	object	Nome da Mesorregião
03_microregiao	5565	object	Nome da Microrregião
03_regiao_imediata	5565	object	Nome da Região Geográfica Imediata
03_regiao_intermediaria	5565	object	Nome da Região Geográfica Intermediária
03_vlr_adic_bruto_total	5565	object	Valor adicionado bruto total, a preços correntes (R\$ 1.000)

03_vlr_adic_bruto_agropecuaria_%	5565	float64	Valor adicionado bruto da Agropecuária, a preços correntes (R\$ 1.000)
03_vlr_adic_bruto_industria_%	5565	float64	Valor adicionado bruto da Indústria, a preços correntes (R\$ 1.000)
03_vlr_adic_bruto_servicos_%	5565	float64	Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R\$ 1.000)
03_vlr_adic_bruto_administracao_%	5565	float64	Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R\$ 1.000)
03_impostos_sobre_produtos	5565	int64	Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R\$ 1.000)
03_pib	5565	int64	Produto Interno Bruto, a preços correntes (R\$ 1.000)
03_pib_per_capita	5565	float64	Produto Interno Bruto per capita, a preços correntes (R\$ 1,00)
03_atividade_principal_primeira	5565	object	Atividade com maior valor adicionado bruto
03_atividade_principal_segunda	5565	object	Atividade com segundo maior valor adicionado bruto
03_atividade_principal_terceira	5565	object	Atividade com terceiro maior valor adicionado bruto
04_populacao_residente_2018	5565	int64	População residente estimada para 2018
05_total_residentes	5565	float64	População residente em 2018
05_0-19_anos_%	5565	float64	Participação relativa da população com esta faixa etária em relação à população total
05_20-39_anos_%	5565	float64	Participação relativa da população com esta faixa etária em relação à população total
05_40-59_anos_%	5565	float64	Participação relativa da população com esta faixa etária em relação à população total
05_60+_anos_%	5565	float64	Participação relativa da população com esta faixa etária em relação à população total
06_total_instrucao	5565	int64	População residente em 2018
06_Sem_Instrucao_e_Fundamental_I_ncompleto%	5565	float64	Participação relativa da população com este nível educacional em relação à população total
06_Medio_Incompleto_%	5565	float64	Participação relativa da população com este nível educacional em relação à população total
06_Superior_Incompleto_%	5565	float64	Participação relativa da população com este nível educacional em relação à população total
06_Superior_Completo_%	5565	float64	Participação relativa da população com este nível educacional em relação à população total
06_Indeterminado_%	5565	float64	Participação relativa da população com este nível educacional em relação à população total
07_total_rendimentos	5565	int64	População residente em 2018
07_Ate_1SM_%	5565	float64	Participação relativa da população com esta faixa de renda em relação à população total
07_Ate_2SM_%	5565	float64	Participação relativa da população com

			esta faixa de renda em relação à população total
07_Ate_3SM_%	5565	float64	Participação relativa da população com esta faixa de renda em relação à população total
07_Ate_5SM_%	5565	float64	Participação relativa da população com esta faixa de renda em relação à população total
07_Ate_10SM_%	5565	float64	Participação relativa da população com esta faixa de renda em relação à população total
07_Ate_20SM_%	5565	float64	Participação relativa da população com esta faixa de renda em relação à população total
07_20SM+_%	5565	float64	Participação relativa da população com esta faixa de renda em relação à população total
07_Sem_Rendimento_%	5565	float64	Participação relativa da população com esta faixa de renda em relação à população total

Esse novo *dataset* será a fonte das informações sobre a qual serão promovidas as análises a partir de agora.

### 3.4. Transformação das variáveis numéricas (conversão para base logarítmica natural)

Há expressiva diferença de grandezas entre as variáveis, o que impõe dificuldades para a comparação entre elas, especialmente quando se analisam os relacionamentos com a variável dependente, a ser predita ("01\_arrecadacao\_2018").

Isso fica evidente quando se comparam as variáveis 01\_arrecadacao\_2018 e 02\_unidades (a segunda, em ordem, das que contêm os maiores valores). Enquanto aquela possui uma amplitude de mais de R\$ 280,44 bilhões (diferença entre 416.536,96 e 280.440.706.625,21); a amplitude desta é de “apenas” 556.032 (diferença entre 5 e 566.037) unidades.

Essa diferença de magnitudes faz com que o gráfico que compare ambas se pareça com o abaixo, a partir do qual uma análise válida fica prejudicada.

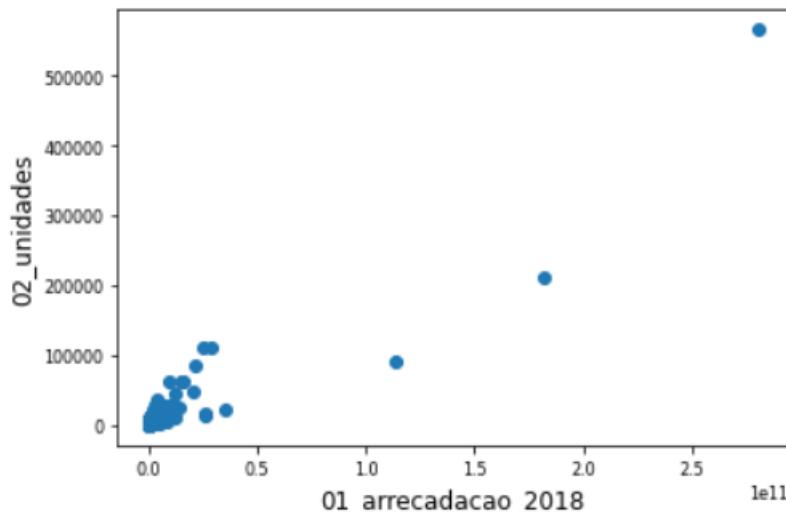


Figura: Gráfico de dispersão para as variáveis 01\_arrecadação\_2018 e 02\_unidades  
(Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb)

Com vistas a, assim, reduzir os efeitos da diferença de magnitude entre as variáveis, optou-se por transformar os valores numéricos para uma escala logarítmica natural, o que acaba por linearizar o gráfico. Para tanto, foi declarada a função “converte\_para\_ln(df)”, abaixo.

```
In [4]: # Converte variáveis numéricas para a base logarítmica
# Pode ser passado por parâmetro uma única variável, ou mesmo o dataframe inteiro.

def converte_para_ln(df):
    cols = df.columns.tolist()
    contador = 0
    for col in cols:

        if df[col].dtype in [np.int64, np.int32, np.float64]:
            nova_coluna = col + "_ln"
            # A linha abaixo calcula o log de cada valor das colunas numéricas
            # O abs para x é porque o np.log retorna erro para valores negativos, usaremos, assim, o valor do módulo.
            # A função Log retornará um erro (-inf) quando o x = 0
            # Assim, quando o valor for zero, a função retornará -13,82 (que é o ln de 0,000001).
            df_transformado[nova_coluna]=df[col].apply(lambda x: np.log(abs(x)) if x!= 0 else -13,82).astype(float)
            contador = contador + 1
    print ('Colunas criadas:',contador)
    return('Concluído')
```

Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

Com a transformação para das variáveis numéricas para a base logarítmica natural, o novo gráfico impresso passa a exibir mais claramente a relação entre as duas variáveis sob atenção.

No exemplo a seguir, repete-se o gráfico acima, combinando as mesmas variáveis, porém convertidas para a base logarítmica

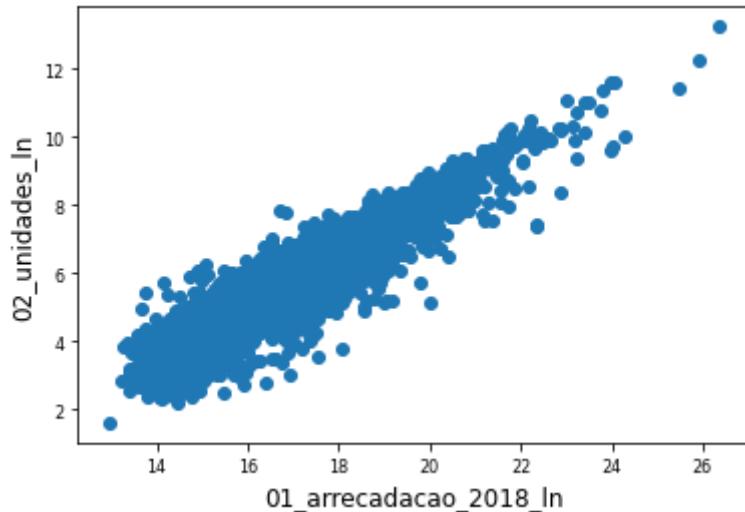


Figura: gráfico de dispersão para as variáveis 01\_arrecadação\_2018\_In e 02\_unidades\_In  
(Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb)

Maiores informações sobre a utilidade e os efeitos dessa transformação serão apresentadas no título 4.1, quando exibidos os sumários estatísticos dos dados transformados, e no título 4.3., quando promovida a análise visual de variáveis numéricas.

### **3.5. Criação da variável categórica "faixa\_arrecadacao".**

Para contribuir com a investigação de características comuns entre as ocorrências do *dataset*, criou-se a variável categórica “faixa\_arrecadacao” cujo objetivo é classificar o *dataset* a partir do conteúdo da variável 01\_arrecadacao\_2018\_In (resultado da conversão da variável 01\_arrecadacao\_2018 para a base logarítmica natural).

Para isso foram definidas 5 faixas:

- K = 5
- Amplitude =  $26.36 - 12.94 = 13.42$
- Intervalo = Amplitude / K
- Faixa\_1:  $0 \mid- 15.623999999999999$
- Faixa\_2:  $15.623999999999999 \mid- 18.308$
- Faixa\_3:  $18.308 \mid- 20.991999999999997$
- Faixa\_4:  $20.99199999999997 \mid- 23.6760000000000000002$
- Faixa\_5:  $23.6760000000000000002 \mid- 26.36$

Após a classificação, a distribuição dos municípios segundo as faixas de arrecadação é a que segue.

```
Número de municípios por Faixa
1    2152
2    2676
3    633
4    94
5    10
Name: faixa_arrecadacao, dtype: int64
```

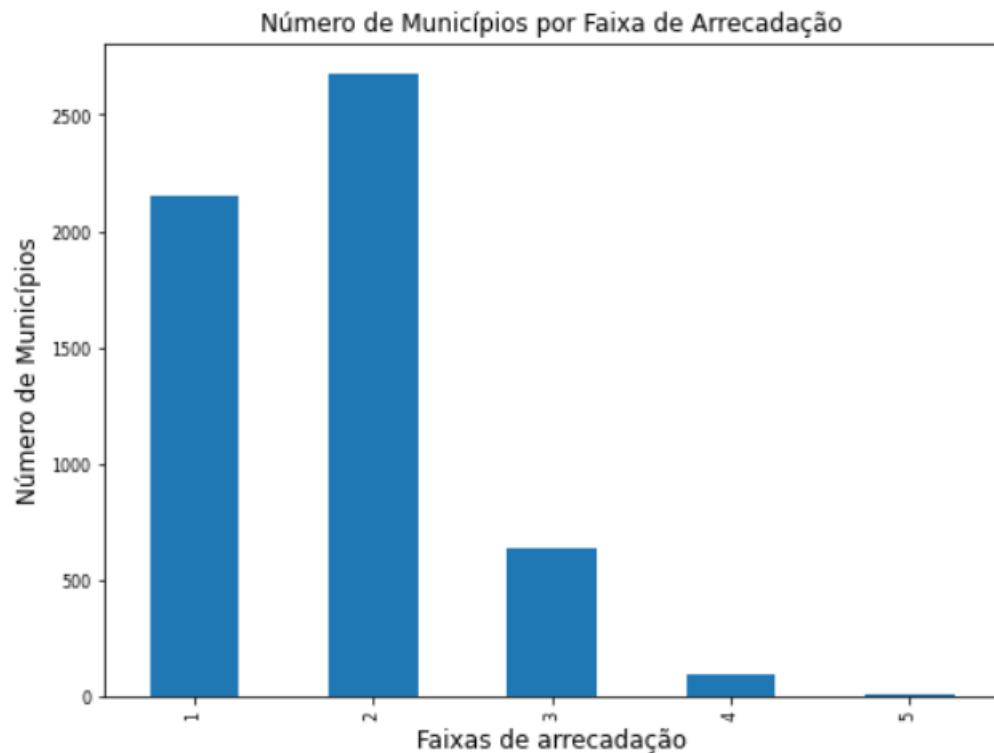


Figura: número de municípios em cada faixa de arrecadação.  
(Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb)

Vê-se acima que as duas primeiras faixas de arrecadação compreendem a maior parte dos municípios (aproximadamente 86,75% dos 5.565 aqui sob análise).

Embora o critério adotado não garanta uma distribuição equilibrada entre as cinco faixas de arrecadação, buscou-se acompanhar a característica da curva da arrecadação (vide abaixo) e, assim, consolidar em cada faixa os municípios com valores similares.

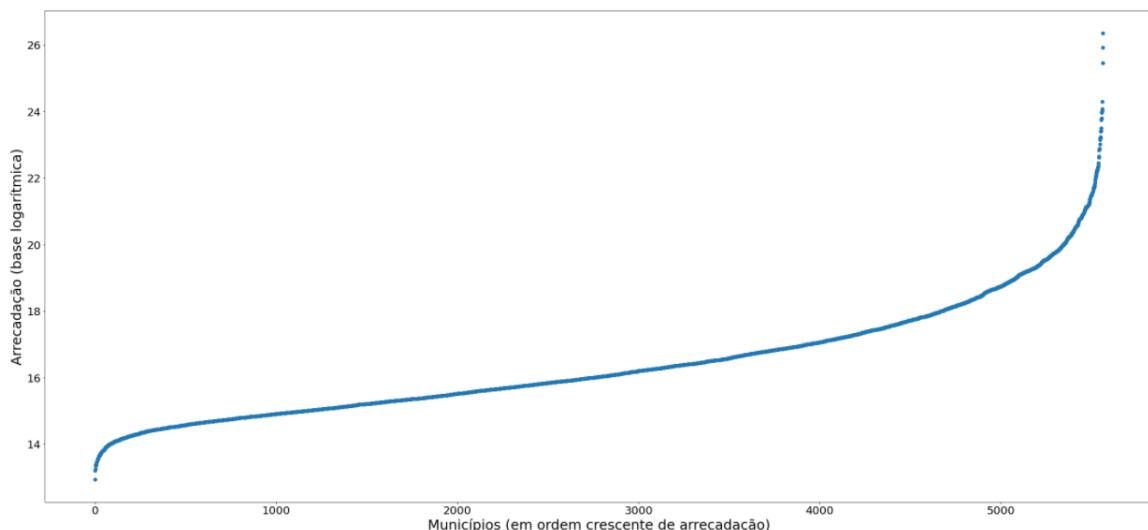


Figura: gráfico das arrecadações (na base logarítmica natural) em ordem ascendente  
 (Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb)

### 3.6. Transformação das variáveis categóricas (One Hot Encoding)

O dataset possui 13 (treze) variáveis categóricas nominais (não ordinais). Como os algoritmos de aprendizado de máquina para regressão exigem que os atributos sejam todos numéricos, necessária a conversão.

```
In [31]: for i in variaveis_categoricas:
    print ('Valores únicos para',i,df_transformado[i].nunique())

```

Valores únicos para municipio-uf 5565  
 Valores únicos para 03\_nome\_uf 27  
 Valores únicos para 03\_uf 27  
 Valores únicos para 03\_cod\_municipio 5565  
 Valores únicos para 03\_grande\_regiao 5  
 Valores únicos para 03\_mesorregiao 137  
 Valores únicos para 03\_microregiao 554  
 Valores únicos para 03\_regiao\_imediata 508  
 Valores únicos para 03\_regiao\_intermediaria 133  
 Valores únicos para 03\_atividade\_principal\_primeira 10  
 Valores únicos para 03\_atividade\_principal\_segunda 10  
 Valores únicos para 03\_atividade\_principal\_terceira 10  
 Valores únicos para faixa\_arrecadacao 5

Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

Diante da grande quantidade de valores únicos vista acima e da relevância para o presente estudo, dentre as variáveis relativas à classificação geopolítica, foi mantida apenas a '03\_uf'. As demais variáveis categóricas de interesse, então,

foram: 03\_atividade\_principal\_primeira, 03\_atividade\_principal\_segunda, 03\_atividade\_principal\_terceira.

Para essa transformação utilizou-se a técnica One Hot Encoding, em que, na prática, cria uma nova variável (coluna) para cada categoria (atribuindo o valor 1 quando pertinente; e zero, quando não). Foram criadas 57 (sessenta e duas) novas variáveis; em seguida, excluídas aquelas a partir das quais foram originadas.

```
In [33]: variaveis_a_transformar = ['03_uf','03_atividade_principal_primeira',
                                '03_atividade_principal_segunda', '03_atividade_principal_terceira']

for i in variaveis_a_transformar:
    df_transformado = pd.concat([df_transformado,pd.get_dummies(df_transformado[i],prefix=i)],axis=1)
    df_transformado = df_transformado.drop(i,axis=1)

## as linhas abaixo totalizam o número de colunas criadas
cols = df_transformado.columns.tolist()

print(df_transformado.shape)
#print(df_transformado.columns)

## as linhas abaixo exibem as colunas criadas
colunas_criadas=[]

for i in variaveis_a_transformar:
    x=0
    while x < len(cols):
        if cols[x][0:len(i)] == i: #procura pelas colunas que se iniciam com o nome da coluna identificada por i
            colunas_criadas.append(cols[x])
        x = x+1

print("Colunas criadas:",len(colunas_criadas))
print(colunas_criadas)

(5565, 141)
Colunas criadas: 57
['03_uf_AC', '03_uf_AL', '03_uf_AM', '03_uf_AP', '03_uf_BA', '03_uf_CE', '03_uf_DF', '03_uf_ES', '03_uf_GO', '03_uf_MA', '03_uf_MG', '03_uf_MS', '03_uf_MT', '03_uf_PA', '03_uf_PB', '03_uf_PE', '03_uf_PI', '03_uf_PR', '03_uf_RJ', '03_uf_RN', '03_uf_RO', '03_uf_RR', '03_uf_RS', '03_uf_SC', '03_uf_SE', '03_uf_SP', '03_uf_TO', '03_atividade_principal_primeira_Administração, defesa, educação e saúde públicas e seguridade social', '03_atividade_principal_primeira_Agricultura, inclusive apoio à agricultura e a pós colheita', '03_atividade_principal_primeira_Comércio e reparação de veículos automotores e motocicletas', '03_atividade_pri
```

Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

### 3.7. Novas exclusões de variáveis

#### - Exclusão de variáveis com multicolinearidade identificada.

Se considerada elevada uma correlação de valor acima de 0.95, então foram identificados dois grupos de variáveis com multicolinearidade (vide detalhes no capítulo 4.4.). Nesse caso, foram mantidas apenas duas variáveis - 03\_pib e 02\_salario\_medio\_mensal\_em\_Reais (pois, em cada grupo, eram as que apresentavam maior correlação com a variável dependente) - e excluídas dez (em itálico, abaixo).

Grupo 1	Grupo 2
<b>03_pib,</b> <i>02_ocupados_assalariados_medio,</i> <i>02_ocupados_total,</i> <i>02_unidades,</i> <i>03_vlr_adic_bruto_total,</i> <i>03_impostos_sobre_produtos,</i> <i>04_populacao_residente_2018,</i> <i>05_total_residentes,</i> <i>06_total_instrucao,</i> <i>07_total_rendimentos</i>	<b>02_salario_medio_mensal_em_Reais,</b> <i>02_salario_medio_mensal_em_SM</i>

**- Exclusão de variáveis com correlação inferior a 0,01 com a variável dependente (*label*).**

Considerou-se como baixas as correlações de valor inferior a 0.01, em relação à variável dependente.

```
In [43]: correlacoes=df_transformado[colunas_a_manter].corr().abs()
print (df_transformado[colunas_a_manter].shape)
correlacoes=pd.DataFrame(correlacoes['01_arrecadacao_2018'].sort_values(ascending=False))
baixas_correlacoes = correlacoes.index[correlacoes['01_arrecadacao_2018']<=.01]
print(baixas_correlacoes.shape)
baixas_correlacoes

(5565, 99)
(42,)

Out[43]: Index(['03_ef_BA',
   '03_atividade_principal_segunda_Pecuária, inclusive apoio à pecuária',
   '03_ef_PI', '03_ef_PB',
   '03_atividade_principal_terceira_Produção florestal, pesca e aquicultura',
   '03_ef_MA', '03_ef_RN', '03_ef_TO', '03_ef_GO', '03_ef_RS',
   '03_atividade_principal_primeira_Pecuária, inclusive apoio à pecuária',
   '03_atividade_principal_primeira_Eletrociidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação',
   '03_atividade_principal_terceira_Indústrias de transformação',
   '03_atividade_principal_segunda_Indústrias extractivas', '03_ef_AL',
   '03_ef_MT', '03_ef_PA', 'faixa_arrecadacao_3', '03_ef_CE',
   '03_atividade_principal_segunda_Produção florestal, pesca e aquicultura',
   '03_ef_SE', '03_ef_PR', '03_atividade_principal_terceira_Construção',
   '03_ef_PE',
   '03_atividade_principal_primeira_Produção florestal, pesca e aquicultura',
   '03_atividade_principal_terceira_Indústrias extractivas', '03_ef_RO',
   '03_ef_MS',
   '03_atividade_principal_terceira_Eletrociidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação',
   '03_atividade_principal_primeira_Comércio e reparação de veículos automotores e motocicletas',
   '03_atividade_principal_segunda_Eletrociidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação',
   '03_ef_AC', '03_atividade_principal_primeira_Indústrias extractivas',
   '03_ef_SC', '03_atividade_principal_segunda_Construção', '03_ef_AP',
   '03_ef_RR', '03_atividade_principal_primeira_Construção', '03_ef_ES',
   '03_atividade_principal_primeira_Indústrias de transformação',
   '03_atividade_principal_segunda_Administração, defesa, educação e saúde públicas e seguridade social',
   '03_ef_AM'],
  dtype='object')
```

Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

Foram identificadas 42 variáveis sob esse critério, veja-se acima, que foram, então, excluídas de nosso *dataset* de modo a se evitar/reduzir a “maldição da dimensionalidade” (vide detalhes no capítulo 4.4).

**- Exclusão de variáveis que não serão utilizadas na aplicação dos modelos de machine learning.**

Dentre as variáveis com informações geopolíticas nos *datasets* optou-se por manter apenas a 03\_uf (por ser bastante conhecida e possuir boa capilaridade) e por excluir as seguintes:

'03\_nome\_uf'  
'03\_cod\_municipio'  
'03\_grande\_regiao',  
'03\_mesorregiao',  
'03\_microregiao',  
'03\_regiao\_imediata',  
'03\_regiao\_intermediaria'.

**3.8. Geração do *dataset* consolidado “ntbk 03 – dataset consolidado – maiores correlacoes.csv”**

Reforçando o caráter iterativo do processo para extração de conhecimento, depois das novas transformações nos dados (como a criação de novas variáveis derivadas das variáveis categóricas) e da exclusão de outras variáveis (como decorrência da etapa de exploração das informações, especialmente a análise de multicolinearidade), foi gerado o *dataset* “ntbk 03 – dataset consolidado – maiores correlacoes.csv” (com 5565 linhas e 45 variáveis), que será a fonte dos dados sobre os quais os modelos de aprendizado de máquina serão aplicados.

```
In [49]: print(df_transformado.shape)
df_transformado.columns

(5565, 45)

Out[49]: Index(['municipio-uf', 'faixa_arrecadacao', '01_arrecadacao_2018',
       '02_unidades_atuantes_%', '02_ocupados_assalariados_%',
       '02_salario_medio_mensal_em_Reais', '03_nf_DF', '03_nf_MG', '03_nf_RJ',
       '03_nf_SP', '03_vlr_adic_bruto_agropecuaria_%',
       '03_vlr_adic_bruto_industria_%', '03_vlr_adic_bruto_servicos_%',
       '03_vlr_adic_bruto_administracao_%', '03_pib', '03_pib_per_capita',
       '03_atividade_principal_primeira_Administração, defesa, educação e saúde públicas e segurança social',
       '03_atividade_principal_primeira_Agricultura, inclusive apoio à agricultura e a pós colheita',
       '03_atividade_principal_primeira_Demais serviços',
       '03_atividade_principal_segunda_Agricultura, inclusive apoio à agricultura e a pós colheita',
       '03_atividade_principal_segunda_Comércio e reparação de veículos automotores e motocicletas',
       '03_atividade_principal_segunda_Demais serviços',
       '03_atividade_principal_segunda_Indústrias de transformação',
       '03_atividade_principal_terceira_Administração, defesa, educação e saúde públicas e segurança social',
       '03_atividade_principal_terceira_Agricultura, inclusive apoio à agricultura e a pós colheita',
       '03_atividade_principal_terceira_Comércio e reparação de veículos automotores e motocicletas',
       '03_atividade_principal_terceira_Demais serviços',
       '03_atividade_principal_terceira_Pecuária, inclusive apoio à pecuária',
       '05_0-19_anos_%', '05_20-39_anos_%', '05_40-59_anos_%', '05_60+anos_%',
       '06_Sem_Instrução_e_Fundamental_Incompleto%', '06_Médio_Incompleto_%',
       '06_Superior_Incompleto_%', '06_Superior_Completo_%',
       '06_Indeterminado_%', '07_Ate_15M_%', '07_Ate_25M_%', '07_Ate_35M_%',
       '07_Ate_55M_%', '07_Ate_105M_%', '07_Ate_205M_%', '07_205M+_%',
       '07_Sem_Rendimento_%'],
      dtype='object')

In [50]: df_transformado.to_csv('ntbk 03 - dataset consolidado - maiores correlações.csv', index=False)
```

Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

## 4. Análise e Exploração dos Dados

### 4.1. Sumários estatísticos

Para iniciar a exploração dos dados será apresentado um sumário estatístico das variáveis com dados numéricos. Essa análise tem por fonte de informação o dataset “ntbk 02 – dataset consolidado.csv”, cuja origem e composição estão descritas no título 3.3, acima.

```
In [10]: df_original.describe().round(2).astype(str)

Out[10]:
   01_arrecadacao_2018  02_unidades  02_unidades_atuantes_%  02_ocupados_total  02_ocupados_assalariados_%  02_salario_medio_mensal_em_SM
count            5565.0        5565.0             5565.0        5565.0                  5565.0        5565.0
mean           249435087.56        978.82            97.44        9382.44                  85.31         2.02
std            4865354052.58       8958.4             2.95        93409.53                  6.76         0.49
min            416535.96          5.0              56.1          59.0                 21.12         0.8
25%           3775315.98         73.0             96.55          564.0                 81.53         1.7
50%           9175716.74        173.0             97.98         1233.0                 85.86         1.9
75%          31501112.41        471.0             99.21         3536.0                 89.88         2.2
max          280440706625.21      566037.0            100.0        5571893.0                99.49         6.9
```

Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

Interessante notar a diferença da magnitude entre as variáveis. Por exemplo: os valores da coluna 01\_arrecadacao\_2018 chegam à ordem das centenas de bilhões (vide o max 280.440.706.625,21); os da coluna 02\_ocupados\_total, à ordem dos milhões (max 5.571.893); os da coluna 02\_salario\_medio\_mensal\_em\_SM são inferiores a uma dezena (vide o max 6,9); e os valores da coluna 02\_ocupados\_assalariados\_% estão, apenas, entre 0 e 100.

Mesmo a amplitude dos valores da própria coluna 01\_arrecadacao\_2018 é significativa. Enquanto a menor arrecadação é a de Santo Antonio dos Milagres - PI, com o valor de R\$ 416.535,96 (ordem das centenas de milhares de Reais), a maior é a do município de São Paulo - SP, com o valor de R\$ 280.440.706.625,21 (centenas de bilhões de Reais).

```
In [11]: print("Município com maior valor para 01_arrecadacao_2018:", "\n", \
df_original[['municipio-uf','01_arrecadacao_2018']][df_original['01_arrecadacao_2018']\
==max(df_original['01_arrecadacao_2018'])])

print("\n")

print("Município com menor valor para 01_arrecadacao_2018:", "\n", \
df_original[['municipio-uf','01_arrecadacao_2018']][df_original['01_arrecadacao_2018']\
==min(df_original['01_arrecadacao_2018'])])
```

	Município com maior valor para 01_arrecadacao_2018:
3830	SAO PAULO-SP            2.804407e+11

	Município com menor valor para 01_arrecadacao_2018:
847	SANTO ANTONIO DOS MILAGRES-PI            416535.96

Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

Agora, um sumário estatístico das colunas com dados não numéricos. Vemos, por exemplo, que o Nordeste é a região brasileira com maior número de municípios com mais de 32% do total (variável 03\_grande\_regiao), embora seja Minas Gerais o estado com maior concentração (variável 03\_nome\_uf). Interessante também notar que a maior parte das unidades (estabelecimentos de pessoas jurídicas localizados nos municípios) têm por atividade principal "Administração, defesa, educação e saúde pública" (variável 03\_atividade\_principal\_primeira).

In [12]:	categ = df_original.dtypes[df_original.dtypes == "object"].index df_original[categ].describe()
Out[12]:	
	municipio-uf 03_nome_uf 03_uf 03_cod_municipio 03_grande_regiao 03_mesorregiao 03_atividade_principal_primeira
count	5565 5565 5565 5565 5565 5565 5565
unique	5565 27 27 5565 5 137 10
top	AGROLANDIA-SC Minas Gerais MG 1721109 Nordeste Noroeste Rio-grandense Administração, defesa, educação e saúde pública...
freq	1 853 853 1 1794 216 2737

Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

#### 4.2. Análises da variável dependente (*label*): 01\_arrecadacao\_2018.

O estudo tem por interesse a predição da arrecadação dos municípios brasileiros em 2018 (indicado pela variável dependente 01\_arrecadacao\_2018).

Para conhecer melhor nossa variável de interesse (*label*), passa-se a examiná-la.

Consultando as estatísticas básicas, vemos que a variável original (01\_arrecadacao\_2018) possui uma amplitude significativa (valor mínimo de R\$ 416,5 mil e valor máximo de R\$ 280,4 bi), com média de R\$ 249,5 mi e mediana de R\$ 9,1 mi.

```
In [14]: plotstats(df_original, '01_arrecadacao_2018')
df_transformado['01_arrecadacao_2018'].describe().round(2).astype(str)

Out[14]: count      5565.0
mean     249435087.56
std      4865354052.58
min      416535.96
25%     3775315.98
50%     9175716.74
75%    31501112.41
max    280440706625.21
Name: 01_arrecadacao_2018, dtype: object
```

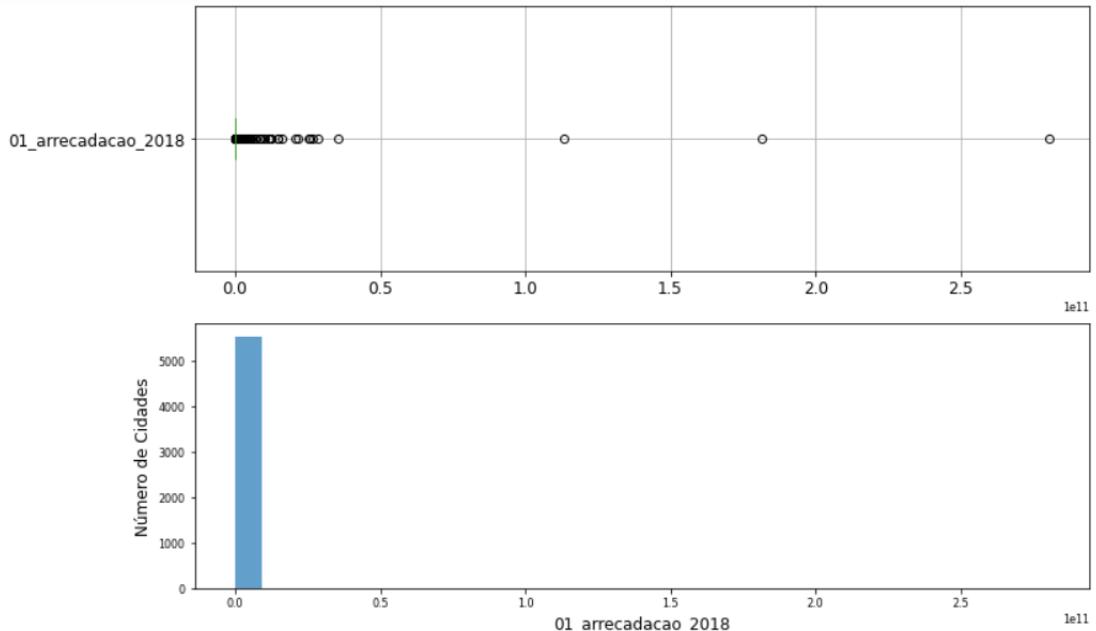


Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

Essa grande variação prejudica a visualização gráfica tanto no *boxplot* como no histograma e, como mencionado anteriormente, dificulta a análise comparativa com as demais variáveis, de menores grandezas.

Em razão disso, a variável foi convertida para a escala logarítmica natural.

```
In [15]: df_transformado['01_arrecadacao_2018_ln']=np.log(df_transformado['01_arrecadacao_2018']).astype(float)
```

Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

Após essa transformação, tanto o *boxplot* como o histograma podem ser melhor analisados.

```
In [17]: plotstats(df_transformado, '01_arrecadacao_2018_ln')
df_transformado['01_arrecadacao_2018_ln'].describe().round(8).astype(str)

Out[17]: count      5565.0
mean     16.40321133
std      1.69714142
min     12.93972808
25%    15.14399464
50%    16.03207107
75%    17.26553342
max     26.35962815
Name: 01_arrecadacao_2018_ln, dtype: object
```

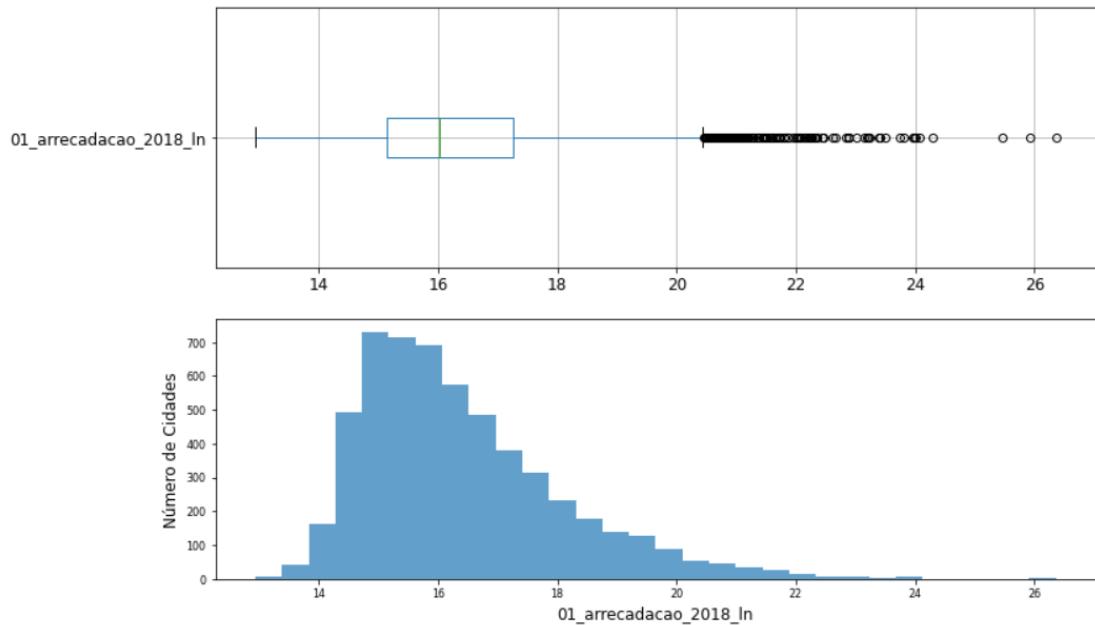


Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

O histograma mostra que a distribuição da variável (convertida para a base logarítmica) se aproxima da curva Normal (assimétrica à direita, ou positiva), com média de 16,4 e mediana de 16,03

Pelo *boxplot*, nota-se a presença de *outliers*.

### - Análise dos *outliers*

A partir dos valores exibidos pelo *boxplot*, acima, foram identificados 158 *outliers* assim consideradas as arrecadações dos municípios que atendessem a um dos seguintes critérios:

- 01\_arrecadacao\_2018\_ln < 12.94 (R\$ 416.535,96)
- 01\_arrecadacao\_2018\_ln > 20.45 (R\$ 757.532.350,88)

```
In [18]: #A linha abaixo utiliza a função limites_min_max_boxplot declarada anteriormente.
limites_outliers=limites_min_max_boxplot(df_transformado,'01_arrecadacao_2018_ln')
print ('Serão outliers os valores:')
print ('-acima de :',limites_outliers[0], "(base logarítmica) ou",limites_outliers[2],"(em Reais)")
print ('-abaixo de :',limites_outliers[1],"(base logarítmica) ou",limites_outliers[3],"(em Reais)")

Serão outliers os valores:
-acima de : 20.445576801868192 (base logarítmica) ou 757532350.8799902 (em Reais)
-abaixo de : 12.93972807539753 (base logarítmica) ou 416535.96000001614 (em Reais)
```

Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

Todas as 158 arrecadações extrapolaram o limite superior acima indicado (não houve, com base nesse critério, *outlier* de valores menores ao limite inferior).

Abaixo, apresentam-se gráficos de dispersão para esses *outliers*: no primeiro gráfico, está a distribuição por estado da federação; o gráfico na sequência exibe a distribuição em ordem ascendente das arrecadações.

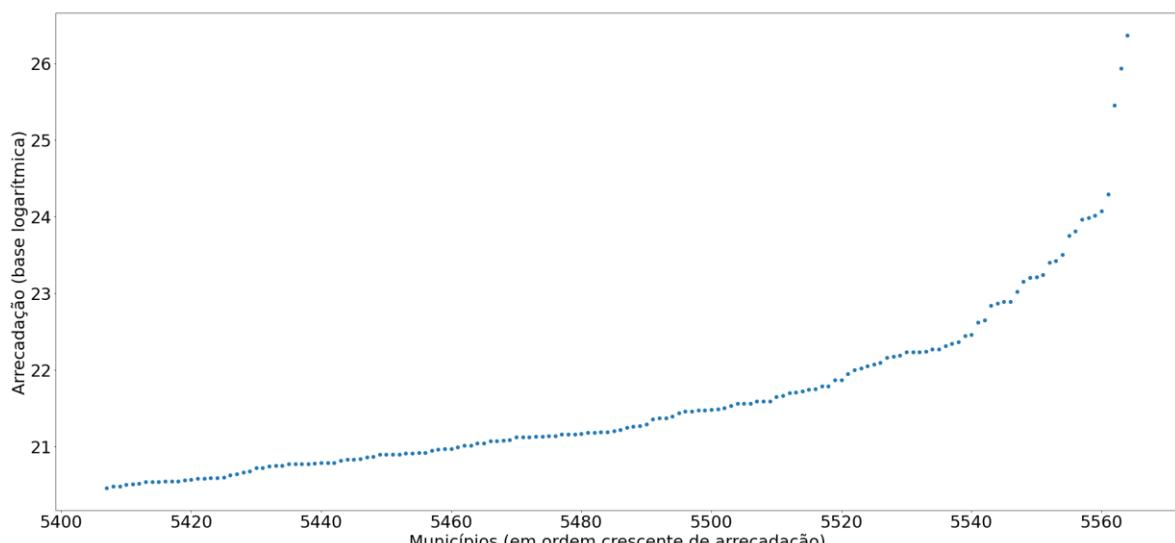
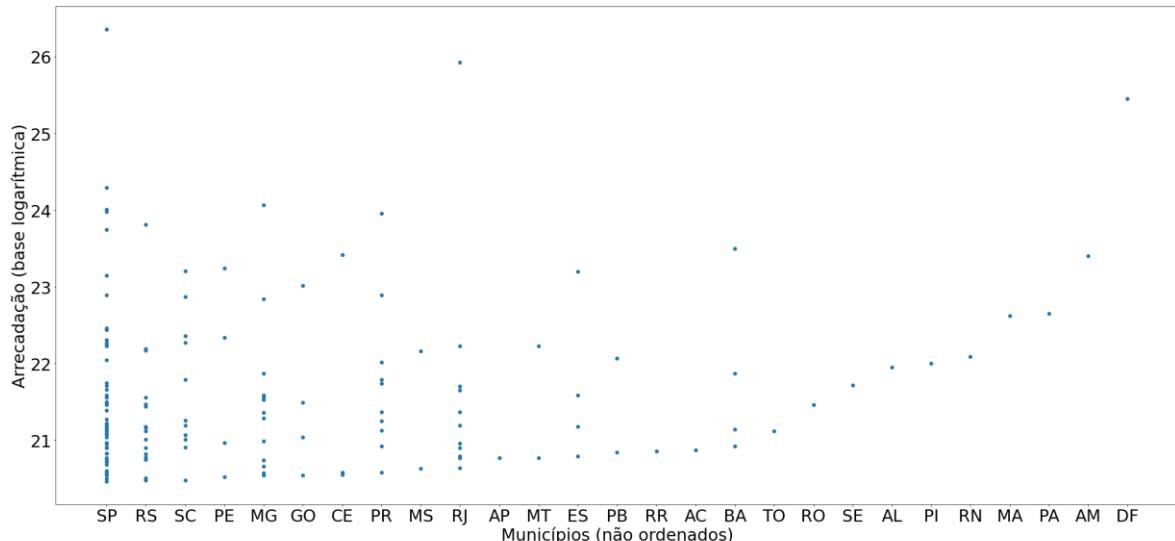


Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

Vê-se com o primeiro gráfico que há uma maior concentração de *outliers* no estado de SP, e que as três maiores arrecadações estão em municípios do estado de SP, RJ e no Distrito Federal, respectivamente. Já a partir do segundo gráfico pode-se observar uma importante inflexão da curva para as 25 maiores arrecadações, com um visível *gap* em relação aos três municípios com maiores arrecadações.

A importância de se analisar os *outliers* diz respeito ao seu possível efeito negativo aos modelos de aprendizado de máquina, influenciando-os de maneira indevida em alguns casos. Assim, os *outliers* precisam ser avaliados e, se for o caso, tratados (transformados, ou excluídos).

No entanto, a caracterização de um valor como um outlier não é tão simples. Embora as arrecadações dos 158 municípios acima realmente destoem das demais (lembrando que a média total das arrecadações é de 16,4, em bases logarítmicas, ou em torno de R\$ 13,30 mi, e aqui estamos falando de arrecadações superiores a R\$ 772,26 mi), não se tratam de *outliers* causados, por exemplo, por erros de valores (o que ocorreria, por exemplo, se para uma variável que se refira à altura da pessoa, em vez de uma escala em metros - em que a altura máxima possível fosse pouco superior a 2,0 metros, na ordem das unidades - alguns valores estivessem informados em cm - o que levaria os valores à ordem das centenas).

Diante disso, ainda que possam influenciar os resultados dos modelos, esses valores destoantes refletem as características intrínsecas do objeto em estudo (há, de fato, municípios tanto com baixas, como com altíssimas arrecadações) e se decidiu, assim, por mantê-los. Os efeitos da manutenção ou não desses *outliers* ficaram evidentes quando se compararam os resultados da aplicação do algoritmo ElasticNet sobre a base com e sem tais valores – vide capítulo 5.3., títulos *Comparação com o segundo melhor modelo* e *Resultado da exclusão das maiores arrecadações*.

#### **4.3. Análise visual das variáveis numéricas.**

Como várias vezes mencionado, há expressiva diferença de grandezas entre as variáveis, o que impõe dificuldades para a comparação entre elas, especialmente quando analisamos os relacionamentos com a variável a ser predita ("01\_arrecadacao\_2018").

Viu-se, no título 3.4, que para reduzir os efeitos da diferença de magnitude entre as variáveis, optou-se por transformar os valores numéricos para uma escala logarítmica natural.

Nota-se, abaixo, que após a transformação dos dados para essa escala, o menor valor entre todas as variáveis (indicadas pelo sufixo “\_ln” passou a ser -13,82 e o maior valor apenas 26,35, trazendo-os, assim, para uma mesma magnitude facilitando a análise comparada e reduzindo vieses decorrentes unicamente da diferença de grandezas.

In [24]:	df_transformado.describe()				
out[24]:	01_arrecadacao_2018	01_arrecadacao_2018_ln	02_unidades_ln	02_unidades_atuantes_%_ln	02_ocupados_totais
<b>count</b>	5.565000e+03	5565.000000	5565.000000	5565.000000	5565.000000
<b>mean</b>	2.494351e+08	16.403211	5.330180	4.578749	7.397
<b>std</b>	4.865354e+09	1.697141	1.421578	0.032886	1.417
<b>min</b>	4.165360e+05	12.939728	1.609438	4.027194	4.077
<b>25%</b>	3.775316e+06	15.143995	4.290459	4.570079	6.338
<b>50%</b>	9.175717e+06	16.032071	5.153292	4.584794	7.117
<b>75%</b>	3.150111e+07	17.265533	6.154858	4.597202	8.170
<b>max</b>	2.804407e+11	26.359628	13.246415	4.605170	15.533

Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

### - **Boxplots por UF**

Após a transformação para a base logarítmica natural, a visualização gráfica passa a exibir mais claramente a relação entre as diversas variáveis.

Para ilustrar algumas dessas relações, serão apresentados, a seguir, *boxplots* que exibem o comportamento de variáveis numéricas e as variações relevantes entre os diversos Estados da Federação.

Este primeiro gráfico retrata o comportamento da variável '01\_arrecadacao\_2018\_ln', agrupada por '03\_uf'.

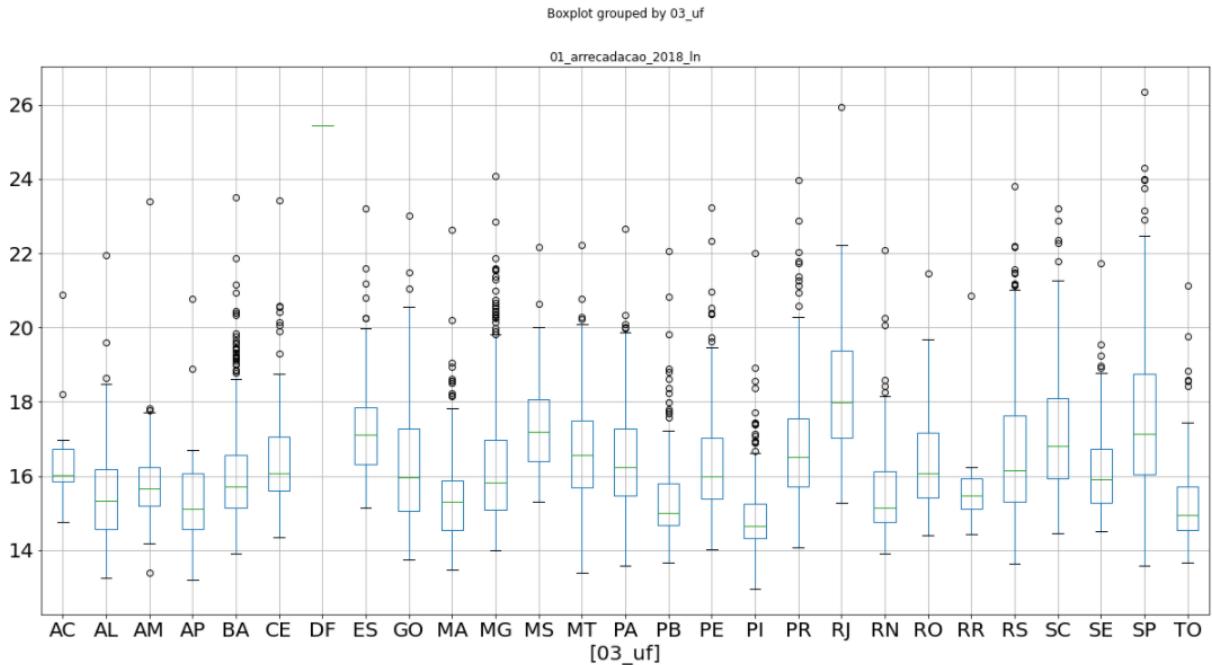


Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

Embora os municípios com maior arrecadação nacional estejam no estado de São Paulo (observado a partir do valor máximo da distribuição, desconsiderados os outliers) - estado em que também se observa a maior dispersão (diferença entre o terceiro e o primeiro quartis) e maior amplitude (diferença entre o valor máximo e o valor mínimo) -, é o estado do Rio de Janeiro que apresenta a proporção mais significativa de municípios com alta arrecadação (mediana e segundo e terceiro quartis com valores mais elevados dentre os *boxplots*). Por outro lado, é do estado do Piauí a distribuição de municípios com menor arrecadação (embora seja a Paraíba o estado com menor valor mínimo do *boxplot*).

Examinando o conjunto de *boxplots* desse primeiro gráfico, nota-se que as distribuições são, em sua maioria, simétricas ou levemente assimétricas positivas (em razão de a mediana aproximar-se da indicação do primeiro quartil). Há outliers superiores para todos os estados da federação.

Observam-se características semelhantes nos dois gráficos abaixo (o primeiro apresenta a variável '02\_ocupados\_total\_In' agrupada por '03\_uf' e, o segundo, a variável '02\_unidades\_In' agrupada por '03\_uf'), com a diferença de que, neste último, o menor número de unidades está no estado do Tocantins (e não do Piauí).

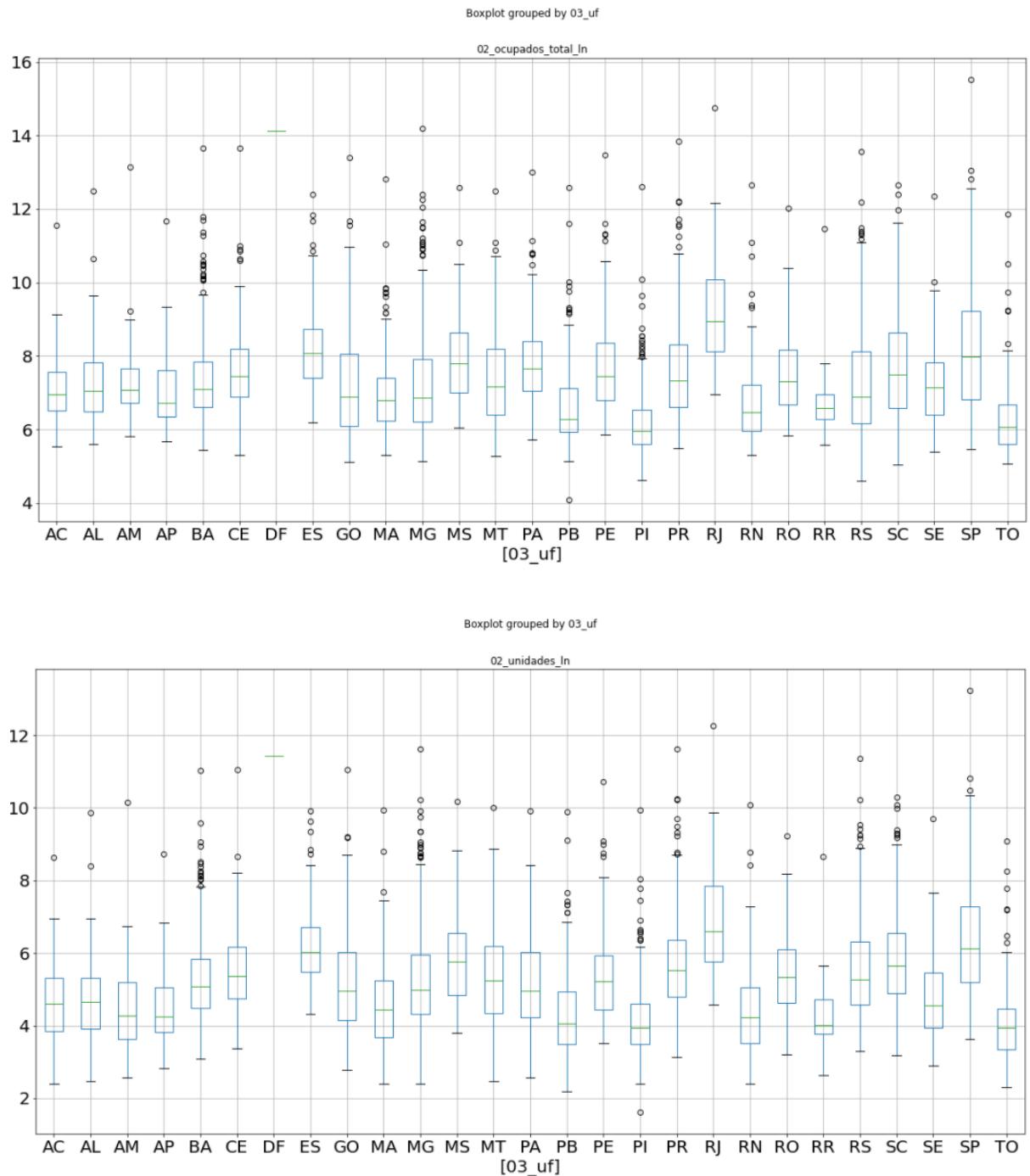


Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

A leitura do terceiro e quarto gráficos considera o quanto as distribuições se aproximam do percentual máximo (100%). E aí é interessante notar que é no estado de São Paulo onde estão os piores números relativos à variável "02\_unidades\_atuantes\_%\_ln" e, no Rio Grande do Sul, em relação à variável "02\_ocupados\_assalariados\_%\_ln".

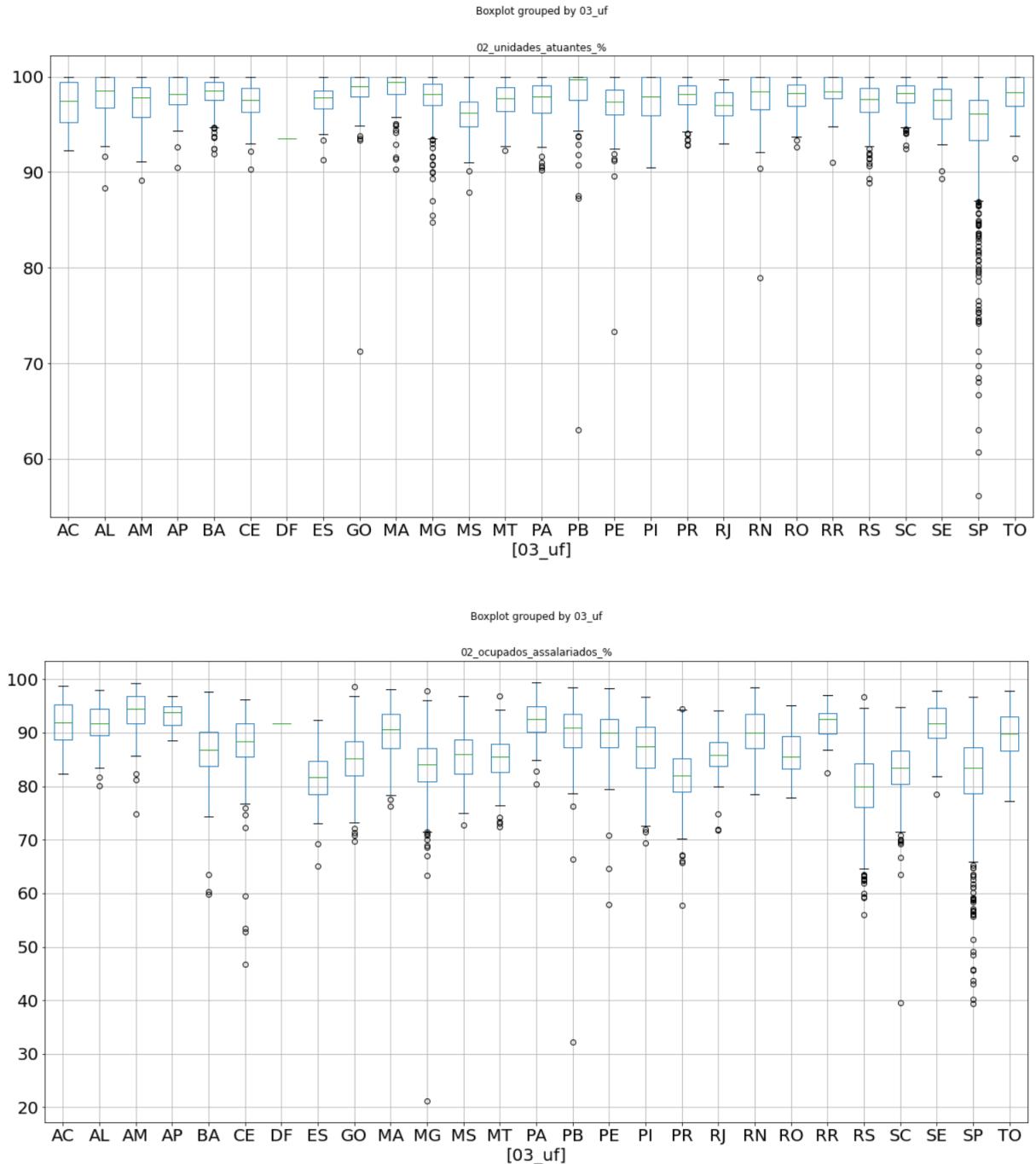


Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

### - **Boxplots por faixa de arrecadação**

Como discutido no título 3.5, criou-se a variável categórica faixa\_arrecadacao com o fim de contribuir com a busca de características comuns entre as ocorrências do *dataset*. Assim, o *dataset* foi classificado em 5 (cinco) faixas de arrecadação a partir do conteúdo da variável 01\_arrecadacao\_2018\_In.

Abaixo são exibidos *boxplots* que permitem a identificação de alguns aspectos socioeconômicos e demográficos interessantes.

No primeiro gráfico, que agrupa os valores da variável 02\_ocupados\_assalariados\_%, por faixa de arrecadação, vemos que as faixas de menor arrecadação (faixas 1 e 2) são as que apresentam maior variação do percentual de empregabilidade assalariada, havendo municípios com baixíssimo percentual de ocupados assalariados (próximo a 30%) e outros em que esse percentual aproxima-se da integralidade (100%). Já as faixas de maiores arrecadações essa variação é menor.

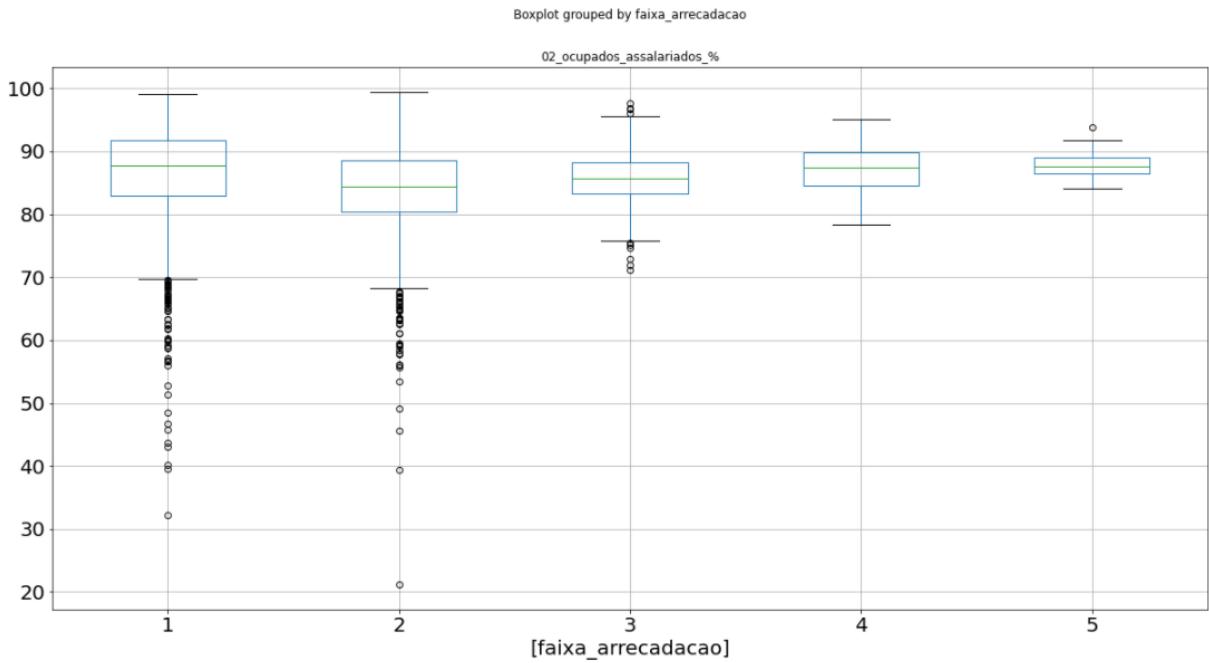


Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

O gráfico seguinte mostra que, a princípio, não obstante haja *outliers*, há relação direta entre a variável 02\_salário\_medio\_mensal\_em\_Reais (salário médio da população do município medido em Reais) e as faixas de arrecadação. De uma forma geral, quanto maior a média mensal dos salários maior a arrecadação do município (ressalvados os *outliers*, já que o maior valor dessa variável, R\$ 6.552,49, inclusive, é o município de Macambira-SE, que se encontra um na faixa 1).

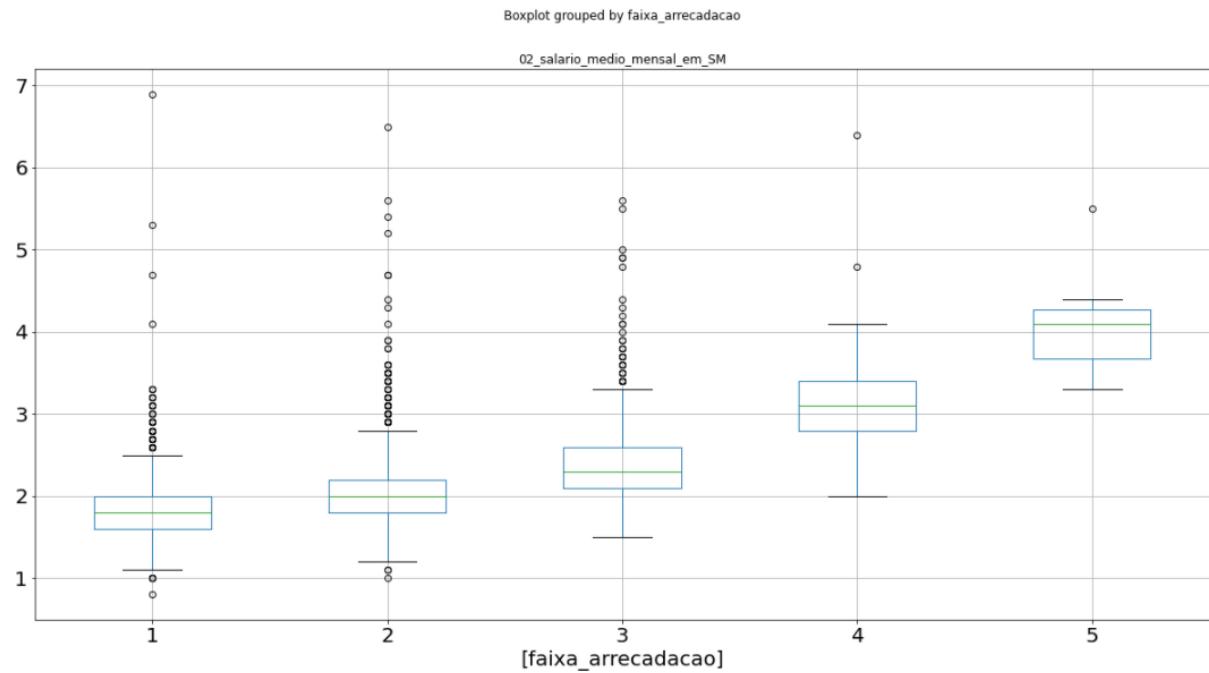
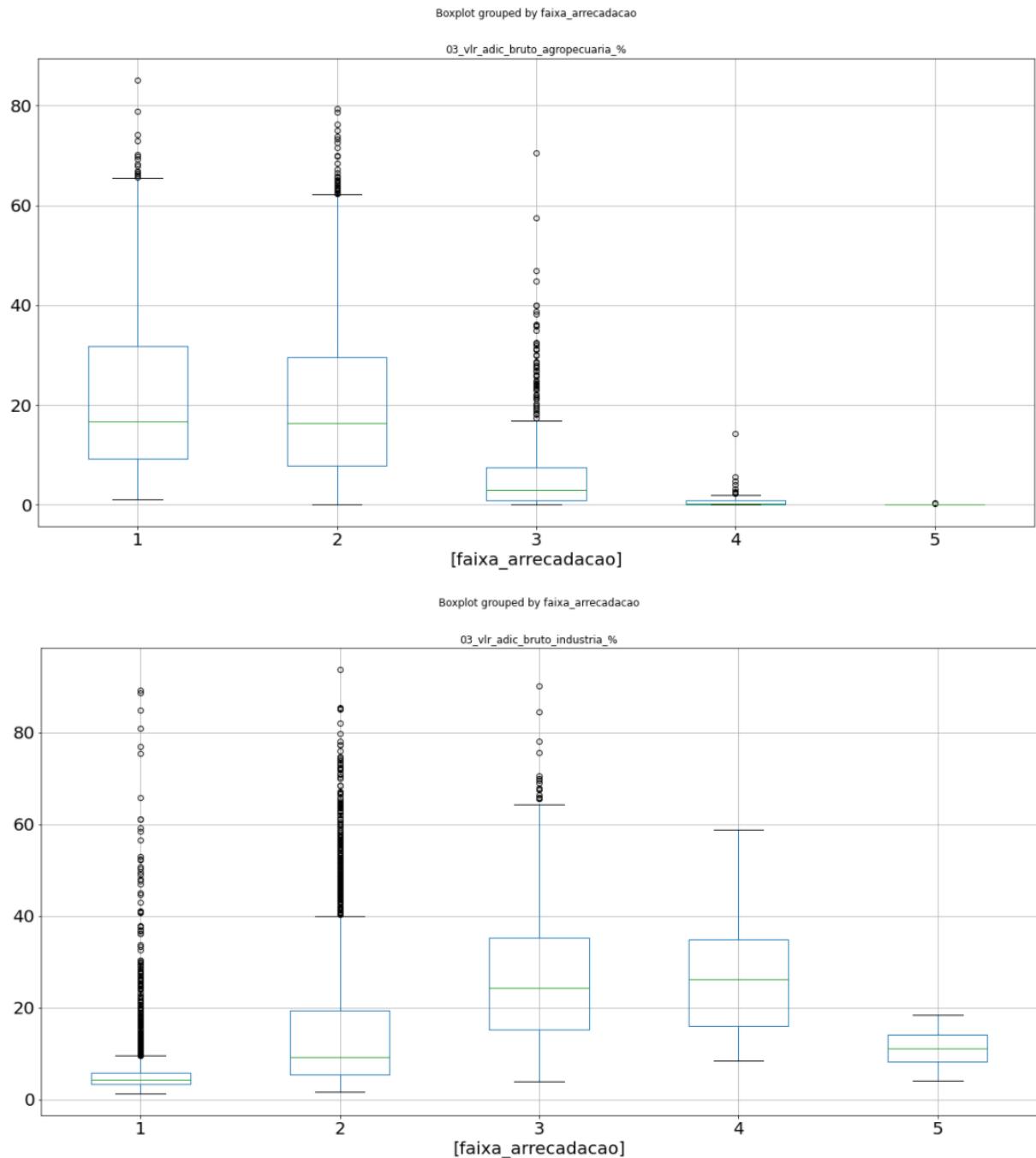


Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

O grupo de gráficos abaixo demonstra a participação relativa dos setores econômicos nas diversas faixas de arrecadação. Enquanto os municípios situados nas faixas 1 e 2 têm na agropecuária sua atividade mais relevante (primeiro gráfico), a atividade industrial é mais relevante nos municípios das faixas 3 e 4 (segundo gráfico). Já a participação relativa do setor de serviços (terceiro gráfico) é diretamente proporcional ao incremento nas faixas de arrecadação, ocorrendo o inverso em relação ao peso do setor administração pública (quarto gráfico). Especificamente em relação aos municípios da faixa 5, a atividade mais relevante é a de serviços.



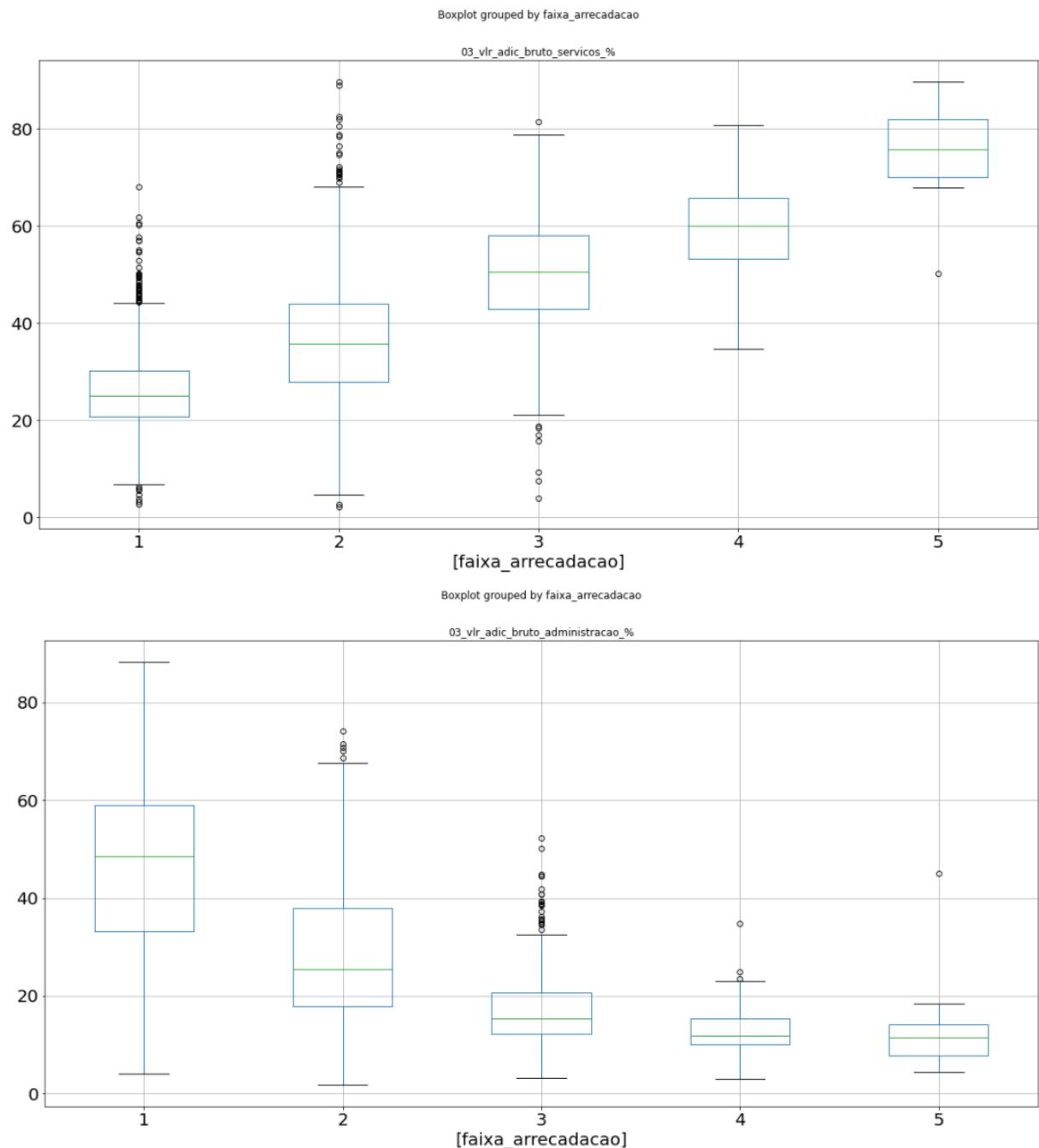
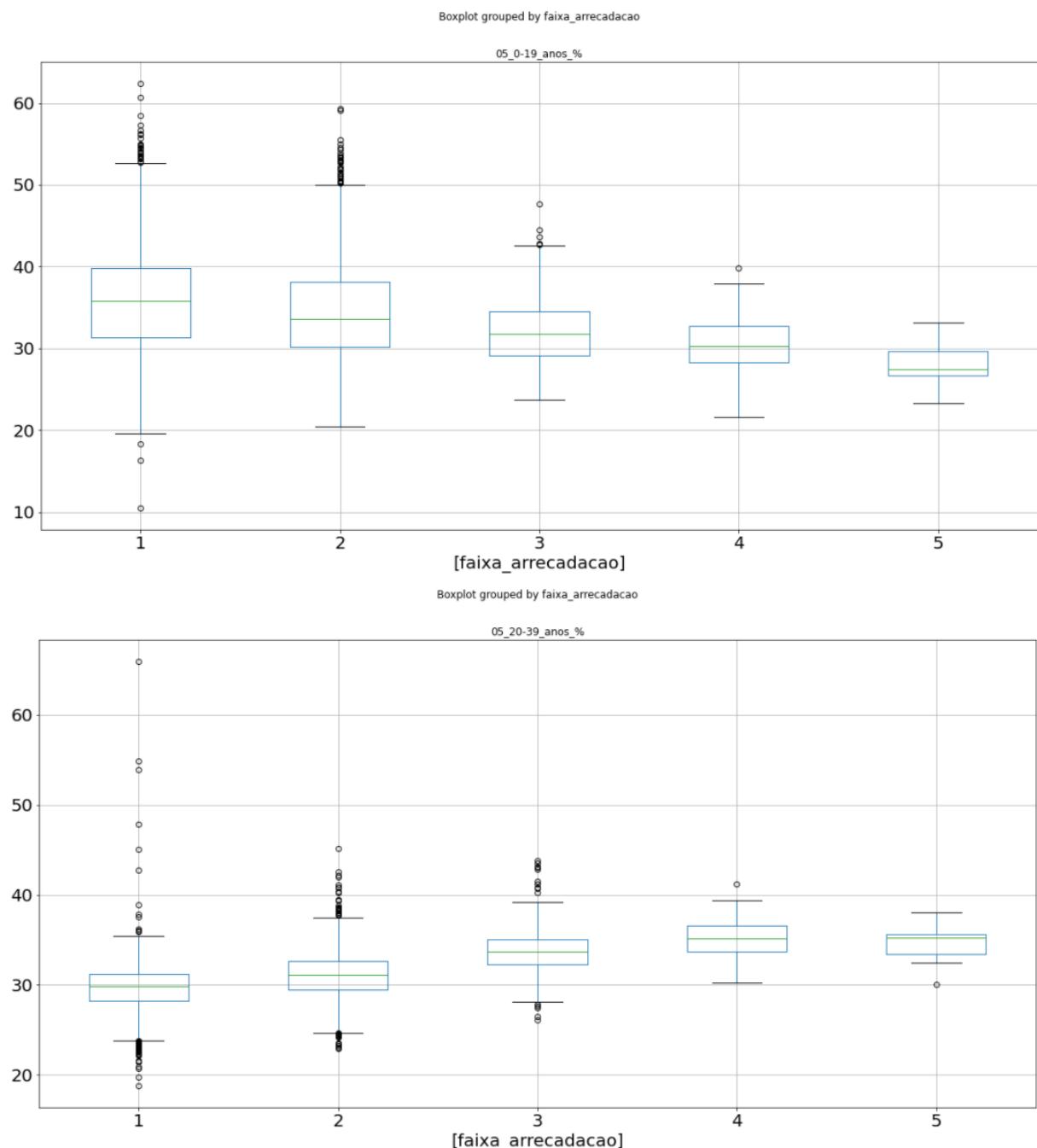


Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

O grupo de gráficos abaixo tem por foco a análise da relação entre a faixa de arrecadação e a idade da população. Enquanto o primeiro gráfico exibe uma relação inversamente proporcional entre o percentual da população com idade inferior a 20 anos e as faixas de arrecadação (quanto menor a proporção de pessoas nessa idade, maior a faixa de arrecadação), observa-se tendência contrária quando se examina a relação entre o percentual de população com idade entre 40 e 59 anos e a faixa (terceiro gráfico).

Já não se nota, porém, uma tendência tão evidente para os estratos entre 20-39 anos (segundo gráfico) e superiores a 60 anos da população (quarto gráfico).



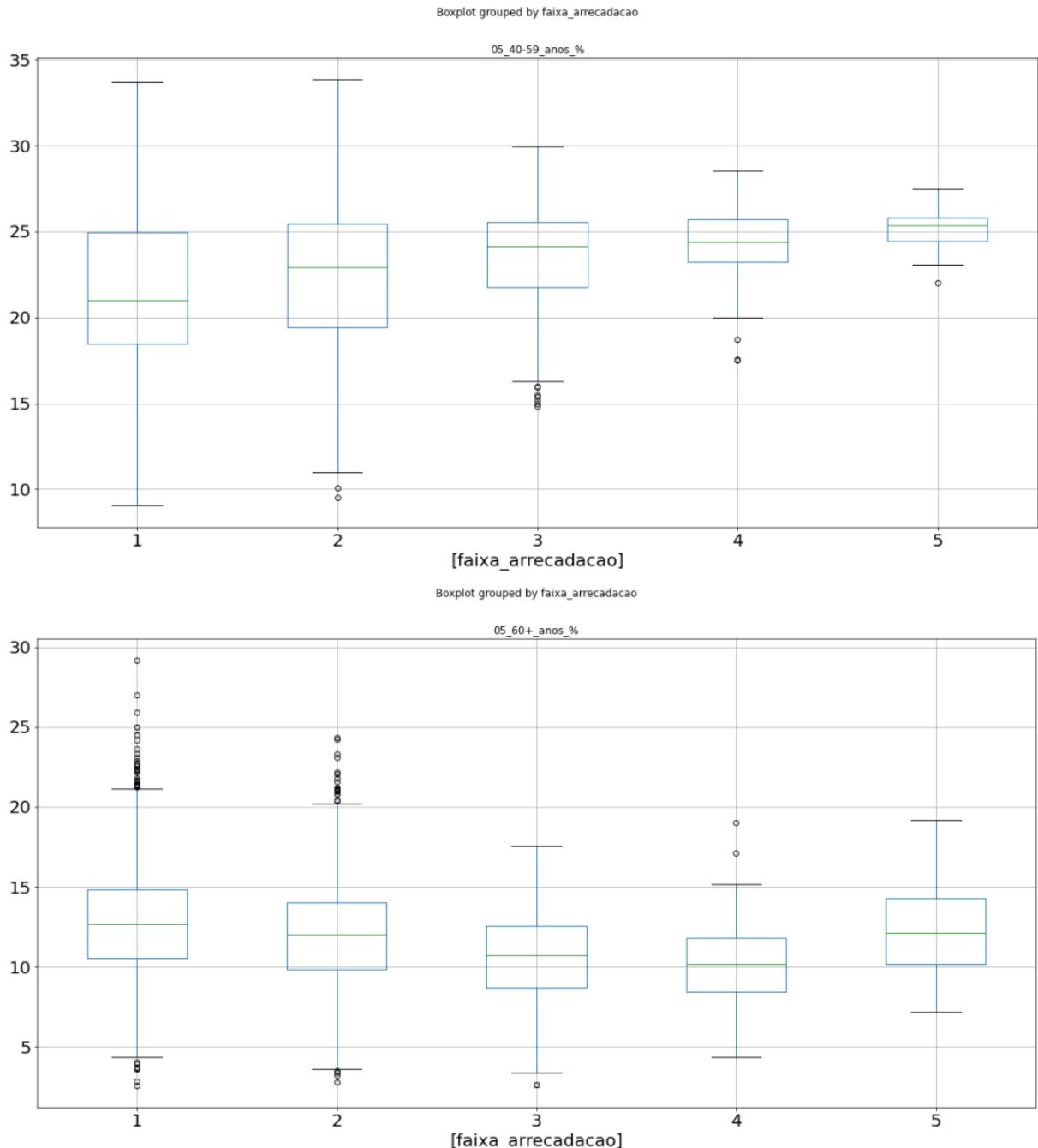
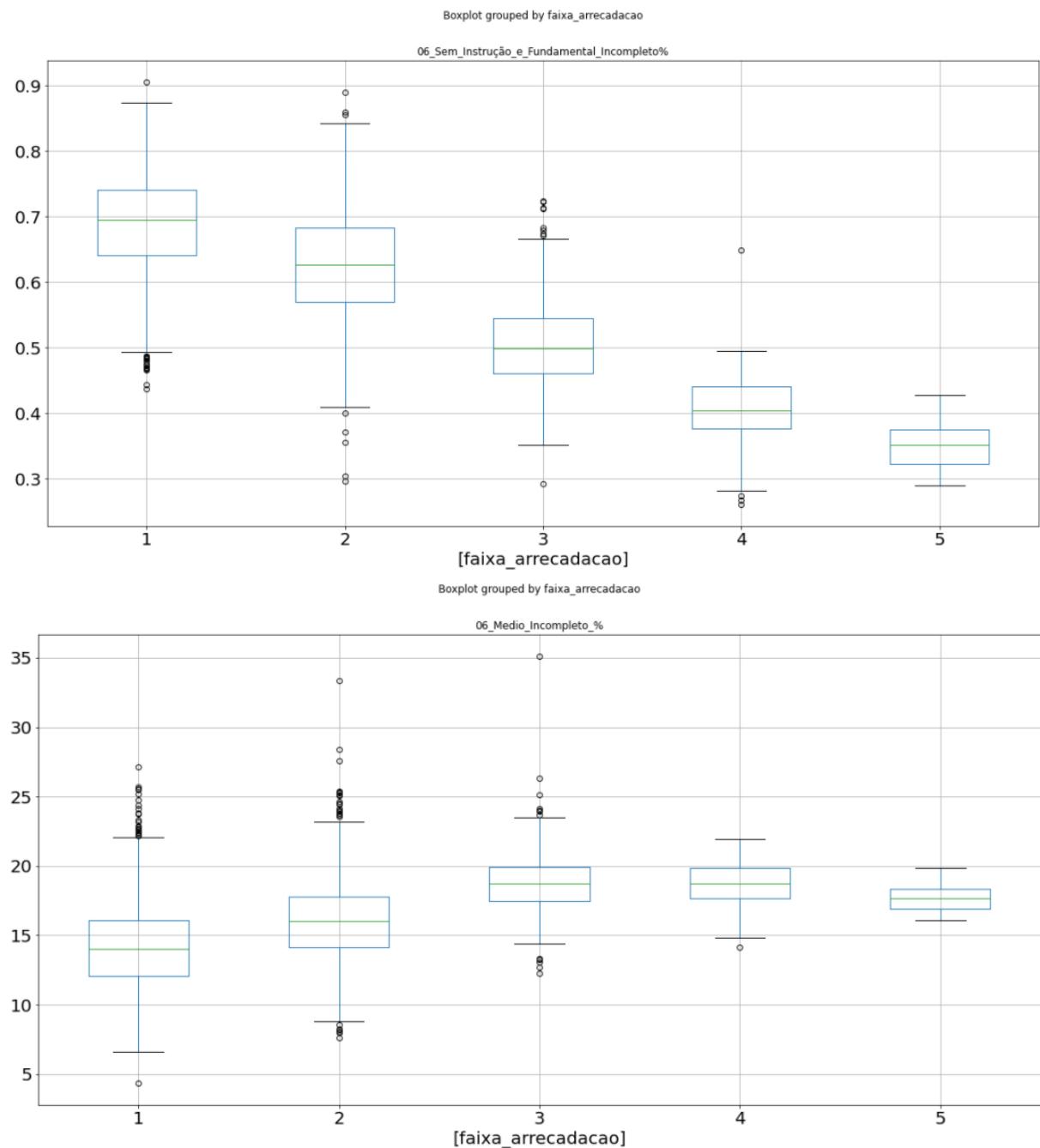


Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

O olhar agora volta-se à relação entre o nível de instrução da população e a faixa de arrecadação do município. O grupo de gráficos abaixo claramente demonstram que quanto menor a arrecadação do município, menor o nível de instrução e vice-versa. O primeiro gráfico exibe a participação proporcional da população sem instrução ou com ensino fundamental incompleto; o segundo, o estrato da população com ensino fundamental completo ou ensino médio incompleto; o terceiro, a parcela da população com ensino médio completo ou

superior completo; e o quarto gráfico, o percentual de municípios com ensino superior completo.



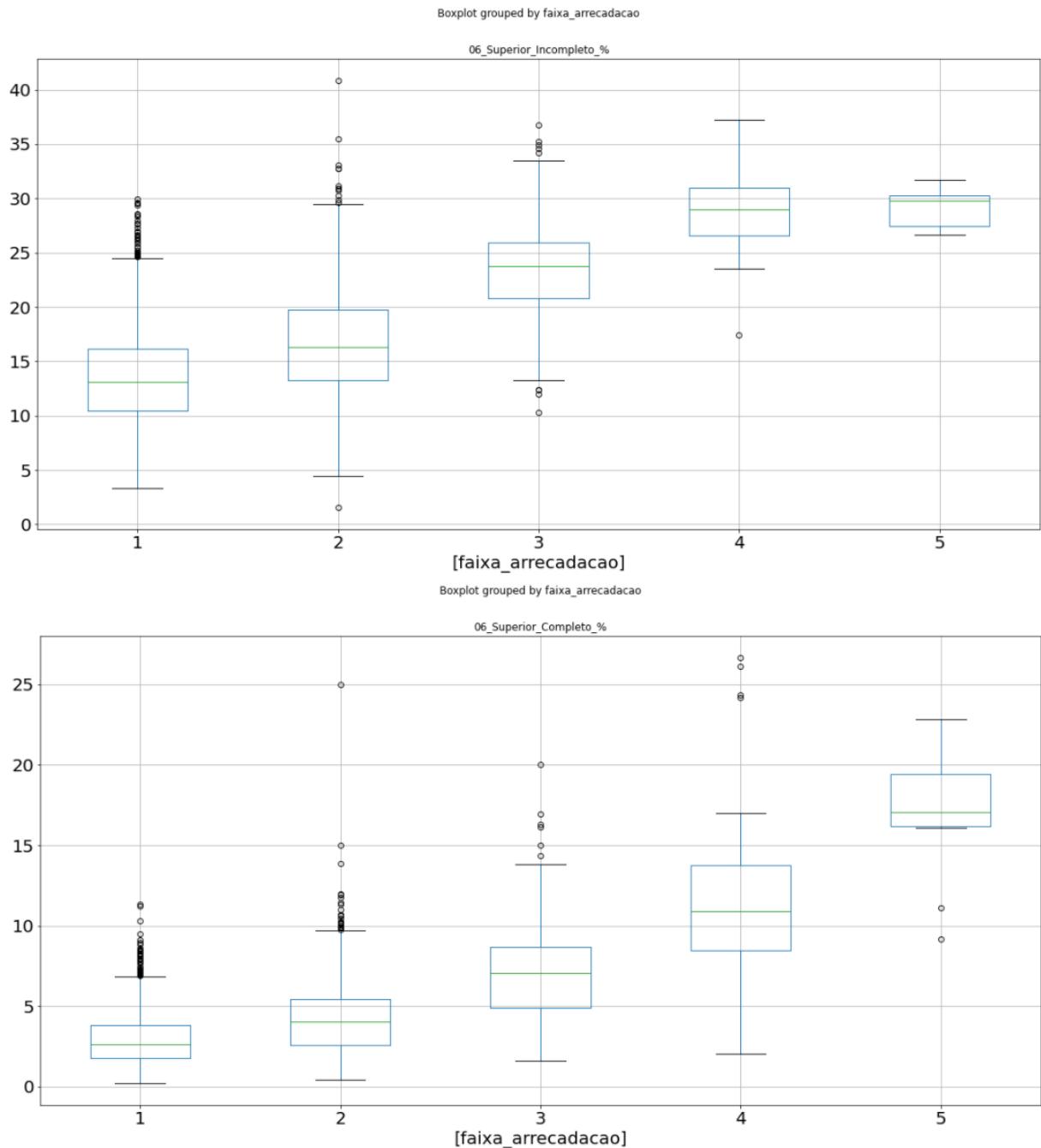


Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

#### 4.4. Análise das correlações e tratamento de multicolinearidades.

- Impressão de Matrizes de Gráficos de Dispersão para análise dos relacionamentos entre as variáveis

Matriz de Gráficos de Dispersão é um instrumento útil para a análise visual dos relacionamentos mútuos entre variáveis numéricas (inclusive aquelas que

resultaram da transformação das variáveis categóricas). Em vez de examinar, isoladamente, um grande número de gráficos (um para cada combinação de uma dupla de variáveis), a matriz permite examinar as relações entre as diversas variáveis em uma única visão.

A matriz é formada por um conjunto de gráficos de dispersão. Para cada variável há tanto uma linha e uma coluna. A variável é exibida no eixo vertical para cada linha, e no eixo horizontal para cada coluna. Desse modo, as combinações cruzadas para todas as variáveis são mostradas nessas duas possíveis orientações.

Como exemplo, seguem abaixo os gráficos com o relacionamento existente entre o *label* (na base logarítmica natural) e as variáveis da base ***ii. Tabela 1685 - Unidades locais, empresas e outras organizações atuantes, pessoal ocupado total, pessoal ocupado assalariado, pessoal assalariado médio, salários e outras remunerações e salário médio mensal (tabela1685.csv)***.

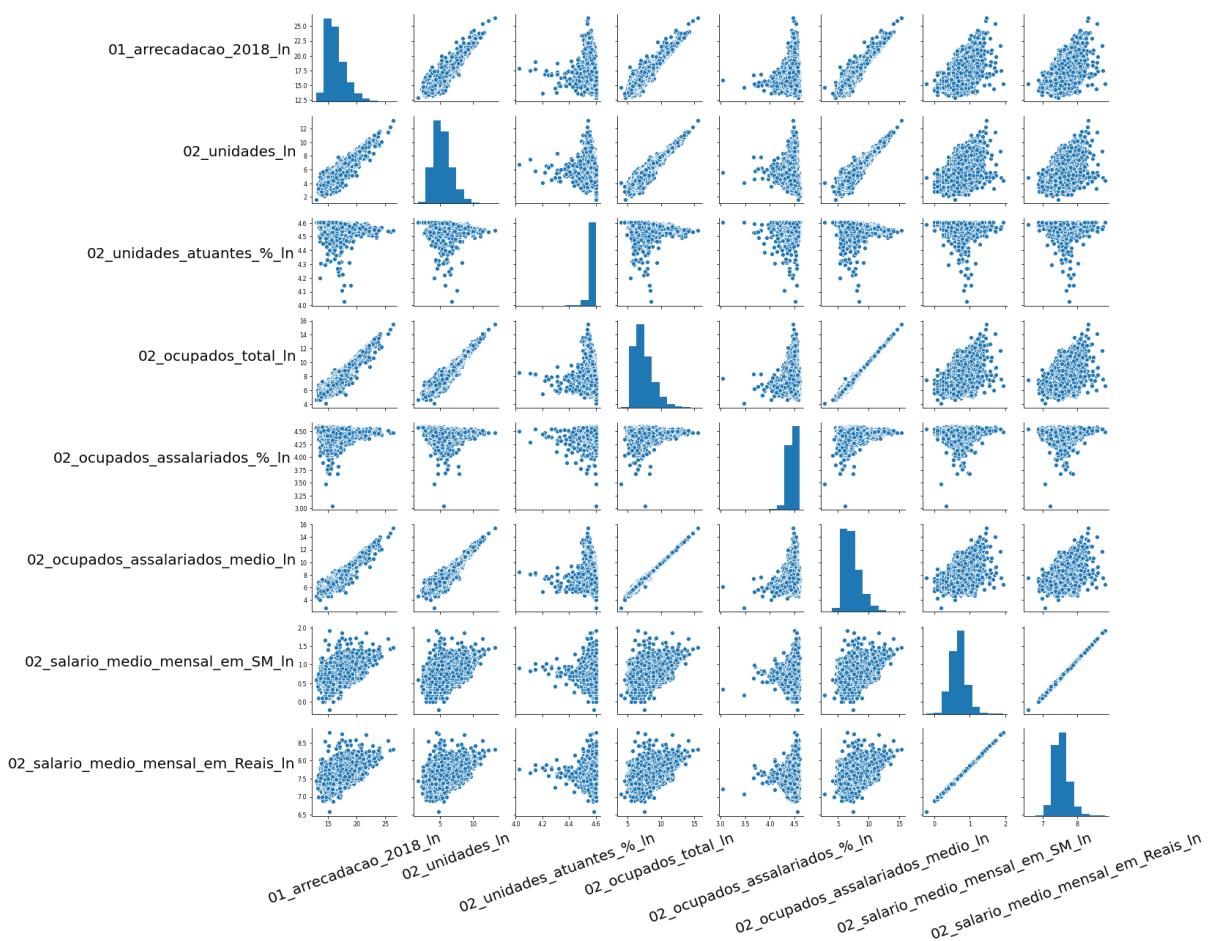


Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

A partir da matriz acima, tem-se que:

- é possível constatar, com os gráficos da primeira linha, que as variáveis 02\_unidades\_In, 02\_ocupados\_total\_In, 02\_ocupados\_assalariados\_medio\_In (segunda, quarta e sexta colunas, respectivamente), guardam evidente colinearidade com a variável de interesse, de modo que poderão, a princípio, contribuir com a predição desta;

- o relacionamento entre as variáveis 02\_ocupados\_total\_In e 02\_ocupados\_assalariados\_medio\_In (interseção entre a quarta linha e a sexta coluna) denota uma altíssima colinearidade, o que pode indicar que seriam redundantes a determinar a exclusão de uma delas sem prejuízo para o modelo de predição, tornando-o mais simples;

- da mesma forma, há forte relação entre as variáveis 02\_salario\_medio\_mensal\_em\_SM\_In e 02\_salario\_medio\_mensal\_em\_Reais\_In (interseção entre a sétima linha e a oitava coluna), o que, igualmente, determinará a escolha por apenas uma delas para a simplificação, sem prejuízos, ao modelo de predição.

Diante dessas constatações, é fundamental proceder a uma análise mais detida das correlações entre as variáveis de modo a identificar possíveis multicolinearidades e evitar "a maldição da dimensionalidade".

A maldição da dimensionalidade diz que a quantidade de dados de que você precisa, para alcançar o conhecimento desejado, impacta exponencialmente o número de atributos necessários. Com isso, o desempenho do classificador tende a se degradar a partir de um determinado número de atributos, mesmo que sejam úteis. (LENINE, Fabio. Como selecionar atributos para resolver a maldição da dimensionalidade. Documento eletrônico. Disponível em <<https://medium.com/@fabiolenine/como-selecionar-atributos-para-resolver-a-maldi%C3%A7%C3%A3o-da-dimensionalidade-5c810bc8449f#>>. Acesso em 1 abr 2021.)

Uma das causas desse efeito é a multicolinearidade, que ocorre quando o modelo inclui vários fatores correlacionados não apenas à sua variável de resposta, mas também entre as próprias variáveis independentes. Em outras palavras, resulta quando você tem fatores que são, em certa medida, redundantes.

A multicolinearidade faz aumentar os erros padrão dos coeficientes, e esse aumento significa que os coeficientes para algumas variáveis independentes podem não ser significativamente diferentes de 0. Em outras palavras, ao superinflacionar os erros padrão, a multicolinearidade torna algumas variáveis estatisticamente insignificantes quando deveriam ser significativas. Sem multicolinearidade (e, portanto, com erros padrão mais baixos), esses coeficientes poderiam ser significativos.

Assim, remover do modelo as preditoras que são altamente correlacionadas é uma estratégia para lidar com a multicolinearidade. Como eles fornecem informações redundantes, a remoção de um dos fatores altamente correlacionados geralmente não reduz drasticamente o resultado do modelo.

### - Análise das correlações

Passa-se, então, a examinar as correlações das diversas variáveis com a variável dependente: 01\_arrecadacao\_2018.

```
In [37]: correlacoes=df_transformado[colunas_a_manter].corr().abs()
print (df_transformado[colunas_a_manter].shape)
correlacoes=correlacoes['01_arrecadacao_2018'].sort_values(ascending=False)
print("Correlações com '01_arrecadacao_2018'")
df_correlacoes = pd.DataFrame(correlacoes)
df_correlacoes
```

Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

O resultado completo do processamento da célula acima pode ser consultado no **Apêndice V – Correlações entre as variáveis independentes e o *label***. Uma visualização sumarizada desse resultado, porém, é oferecida pelo histograma abaixo.

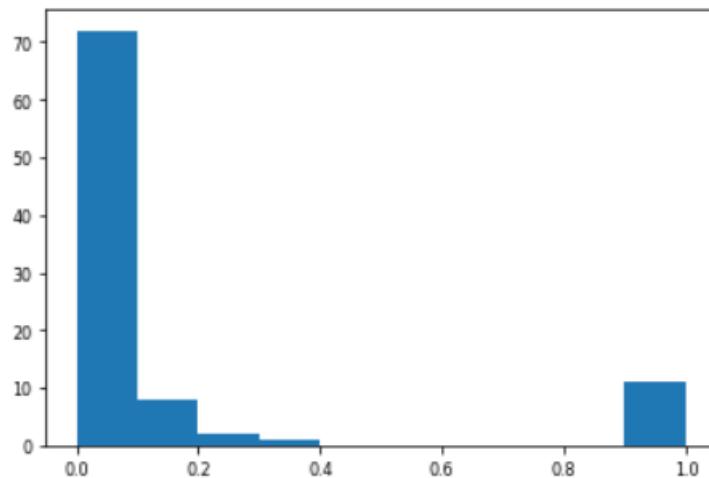


Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

Vemos, acima, que:

- há um pequeno número de variáveis (10, no total) com correlação superior a 0.9 com nosso *label*;

Variável independente	Correlação com o <i>label</i>
03_pib	0,974
03_impostos_sobre_produtos	0,974
03_vlr_adic_bruto_total	0,968
02_ocupados_assalariados_medio	0,943
02_ocupados_total	0,942
02_unidades	0,930
07_total_rendimentos	0,920
06_total_instrução	0,920
05_total_residentes	0,918
04_populacao_residente_2018	0,917

- outras 11, possuem correlação entre 0.10 e 0.32;

Variável independente	Correlação com o <i>label</i>
03_uf_DF	0,312
07_20SM+_%	0,258
07_Ate_20SM_%	0,234
06_Superior_Completo_%	0,183
02_salario_medio_mensal_em_Reais	0,175
02_salario_medio_mensal_em_SM	0,175
07_Ate_10SM_%	0,160
06_Sem_Instrução_e_Fundamental_Incompleto%	0,122
03_vlr_adic_bruto_servicos_%	0,119
03_atividade_principal_segunda_Comércio e reparação de veículos automotores e motocicletas	0,111
06_Superior_Incompleto_%	0,104

- na faixa de correlações entre 0.01 e 0.1, há 31 variáveis.

- há 41 variáveis com correlação de valor inferior a 0.01.

Se, porém, analisarmos as correlações entre as variáveis na versão convertida para a base logarítmica natural, a associação entre elas fica ainda mais evidente. Passam a existir 6 (seis) variáveis com correlação acima de 0.9; 19 (dezenove) variáveis com correlação entre 0.5 e 0.89; outras 19 (dezenove) com correlação entre 0.2 e 0.49. O resultado completo pode ser consultado no **Apêndice V – Correlações entre as variáveis independentes e o *label*.**

	01_arrecadacao_2018_ln
01_arrecadacao_2018_ln	1.000000
03_impostos_sobre_produtos_ln	0.956657
02_ocupados_total_ln	0.954628
02_ocupados_assalariados_medio_ln	0.950757
03_pib_ln	0.945561
03_vlr_adic_bruto_total_ln	0.941368
02_unidades_ln	0.928730
07_total_rendimentos_ln	0.816781
06_total_instrucao_ln	0.816781
04_populacao_residente_2018_ln	0.810196
05_total_residentes_ln	0.804971
06_Sem_Instrucao_e_Fundamental_Incompleto%_ln	0.725350
07_Ate_1SM_%_ln	0.646870
03_vlr_adic_bruto_administracao_%_ln	0.641945
03_vlr_adic_bruto_industria_%_ln	0.623916
03_vlr_adic_bruto_servicos_%_ln	0.618521
06_Superior_Incompleto_%_ln	0.597164
06_Superior_Completo_%_ln	0.572580
07_Ate_5SM_%_ln	0.572464
03_atividade_principal_primeira_Demais serviços	0.572182
03_atividade_principal_primeira_Administração, ...	0.561534
03_pib_per_capita_ln	0.556917
02_salario_medio_mensal_em_Reais_ln	0.544382
02_salario_medio_mensal_em_SM_ln	0.543254
07_Ate_3SM_%_ln	0.543168
07_Ate_10SM_%_ln	0.538032
05_20-39_anos_%_ln	0.496667
06_Medio_Incompleto_%_ln	0.484070
07_Ate_25M_%_ln	0.462500

Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

## - Identificação de multicolinearidades

Abaixo, será iniciada a investigação das multicolinearidades.

	01_arrecadacao_2018	02_unidades	02_unidades_atuantes_%	02_ocupados_total	02_ocupados_assalariados_%
01_arrecadacao_2018	1.000000	0.930497	-0.049776	0.942168	0.01833C
02_unidades	0.930497	1.000000	-0.079235	0.990256	0.007482
02_unidades_atuantes_%	-0.049776	-0.079235	1.000000	-0.080499	0.113011
02_ocupados_total	0.942168	0.990256	-0.080499	1.000000	0.02595E
02_ocupados_assalariados_%	0.018330	0.007482	0.113011	0.025958	1.00000C
02_ocupados_assalariados_medio	0.942503	0.987375	-0.081272	0.999781	0.028624
02_salario_medio_mensal_em_SM	0.174743	0.208032	-0.200304	0.218894	-0.054992
02_salario_medio_mensal_em_Reais	0.175183	0.208084	-0.200430	0.218904	-0.055422
03_vlr_adic_bruto_total	0.967767	0.967074	-0.078775	0.975217	0.028947
03_vlr_adic_bruto_agropecuaria_%	-0.057152	-0.097345	0.032978	-0.096977	-0.174904

Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

O quadro acima, mostra as diferentes correlações cruzadas entre as diversas variáveis, que podem ser consultadas de forma completa no **Apêndice IV – Matriz de correlações.**

Passa-se a selecionar aquelas que possuem indícios de multicolinearidade, ou seja, elevada correlação com outras variáveis independentes, além daquela já vista em relação ao label: 01\_arrecadacao\_2018.

```
In [40]: correlacoes = correlacoes.melt(id_vars='indice').copy()
#O comando abaixo, então, seleciona as altas correlações (valores absolutos superiores a ,95)
altas_correlacoes = correlacoes[(abs(correlacoes['value'])>=.95) \ 
    & (correlacoes['indice']!=correlacoes['variable']) \ 
    & (correlacoes['indice']!= '01_arrecadacao_2018') \ 
    & (correlacoes['indice']!= '01_arrecadacao_2018_ln') \ 
    & (correlacoes['variable']!= '01_arrecadacao_2018') \ 
    & (correlacoes['variable']!= '01_arrecadacao_2018_ln')].sort_values(by='indice')
print (altas_correlacoes.shape)
altas_correlacoes
```

(78, 3)

	indice	variable	value
104	02_ocupados_assalariados_medio	02_unidades	0.987375
2777	02_ocupados_assalariados_medio	07_total_rendimentos	0.981769
2183	02_ocupados_assalariados_medio	06_total_instrucao	0.981769
1688	02_ocupados_assalariados_medio	05_total_residentes	0.980877
1589	02_ocupados_assalariados_medio	04_populacao_residente_2018	0.980553
302	02_ocupados_assalariados_medio	02_ocupados_total	0.999781
1391	02_ocupados_assalariados_medio	03_pib	0.975664
797	02_ocupados_assalariados_medio	03_vlr_adic_bruto_total	0.975296
102	02_ocupados_total	02_unidades	0.990256
2775	02_ocupados_total	07_total_rendimentos	0.981046
2181	02_ocupados_total	06_total_instrucao	0.981046

Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

Se considerada elevada uma correlação de valor acima de 0.95, então haveria dois grupos com multicolinearidade. De todas essas variáveis, foram mantidas apenas duas (aqueelas, em cada um dos grupos, com maior correlação

com o label: 03\_pib e 02\_salario\_medio\_mensal\_em\_Reais) e excluídas dez (em itálico, abaixo).

Grupo 1	Grupo 2
<b>03_pib,</b> <i>02_ocupados_assalariados_medio,</i> <i>02_ocupados_total,</i> <i>02_unidades,</i> <i>03_vlr_adic_bruto_total,</i> <i>03_impostos_sobre_produtos,</i> <i>04_populacao_residente_2018,</i> <i>05_total_residentes,</i> <i>06_total_instrução,</i> <i>07_total_rendimentos</i>	<b>02_salario_medio_mensal_em_Reais,</b> <i>02_salario_medio_mensal_em_SM</i>

Na matriz de gráficos abaixo, essas relações mútuas entre as variáveis independentes ficam bastante evidentes (utilizamos para a impressão os valores convertidos em base logarítmica para permitir uma melhor comparação dos valores).

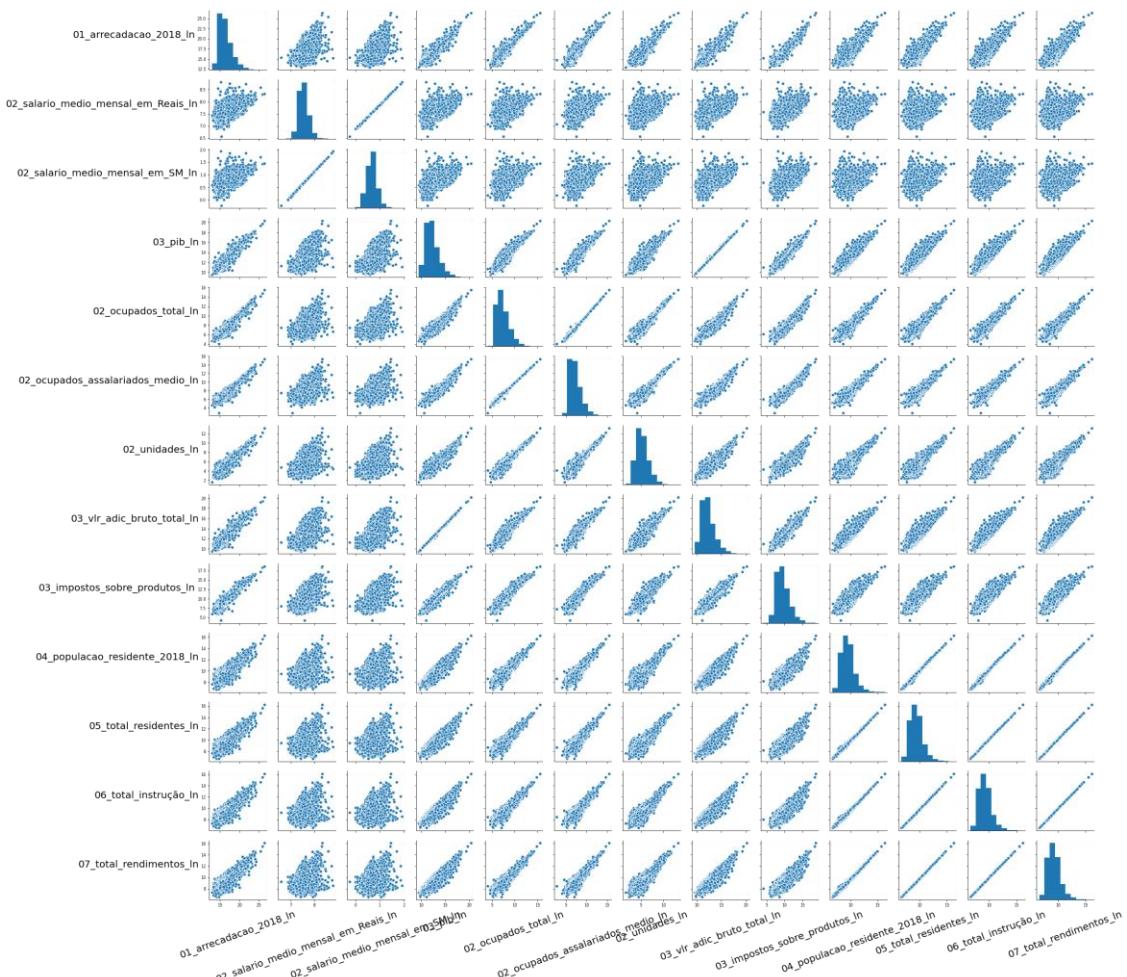


Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

Os gráficos da primeira linha mostram a significativa correlação entre as variáveis do quadro acima e o *label* 01\_arrecadacao\_2018 (convertido para a base logarítmica). Já nas linhas seguintes, nota-se que nas interseções intragrupos há colinearidade explícita, denotada por uma distribuição que praticamente coincide com uma linha reta, a exemplo das correlações:

- entre 02\_salario\_medio\_mensal\_em\_Reais\_In (segunda linha) e 02\_salario\_medio\_mensal\_em\_SM\_In (terceira coluna);
- entre 03\_pib\_In (quarta linha) e 03\_vlr\_adic\_bruto\_total\_In (oitava coluna);
- entre 02\_ocupados\_total\_In (quinta linha) e 02\_ocupados\_assalariados\_medio\_In (sexta coluna);
- entre 04\_populacao\_residente\_2018 (décima linha) e 03\_total\_residentes\_In, 04\_total\_instrucao\_In e 07\_total\_rendimentos\_In (colunas onze a treze, respectivamente).

A Matriz de Correlações também pode ser exibida na forma de mapa de calor, por meio da qual igualmente é possível confirmar essas multicolinearidades.

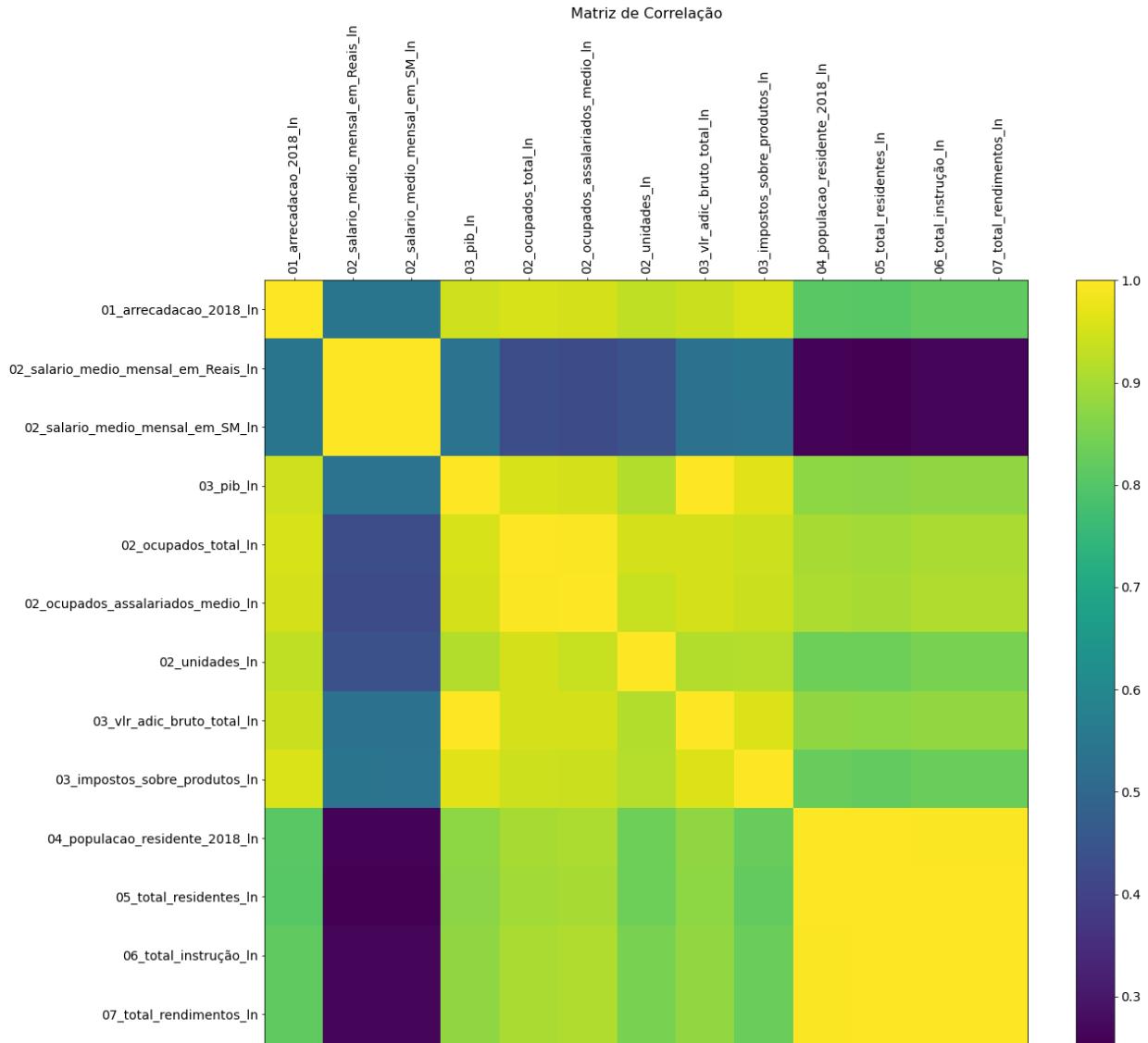


Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

Se desconsideras as multicolinearidades identificadas acima, as variáveis com as 20 maiores correlações com o *label* (variável dependente ou de interesse) passarão a ser:

Variável	Correlação com o <i>label</i>
01_arrecadacao_2018	1.000000
03_pib	0.974050
03_uf_DF	0.312061
07_20SM+_%	0.258040
07_Ate_20SM_%	0.234498
06_Superior_Completo_%	0.183179
02_salario_medio_mensal_em_Reais	0.175183
07_Ate_10SM_%	0.159924

06_Sem_Instrução_e_Fundamental_Incompleto%	0.122233
03_vlr_adic_bruto_servicos_%	0.118773
03_atividade_principal_segunda_Comércio e reparação de veículos automotores e motocicletas	0.110595
06_Superior_Incompleto_%	0.103661
07_Ate_1SM_%	0.097726
07_Ate_5SM_%	0.092199
03_pib_per_capita	0.070945
05_20-39_anos_%	0.065520
03_atividade_principal_primeira_Demais serviços	0.060559
07_Ate_3SM_%	0.060010
03_vlr_adic_bruto_agropecuaria_%	0.057152
03_uf_RJ	0.054291
03_vlr_adic_bruto_administracao_%	0.052273

## - Identificação das variáveis com baixa correlação

A sequência de comandos abaixo identifica os atributos com baixa correlação. Foram consideradas como baixas as correlações de valor inferior a 0,01, em relação ao o *label*. Tais variáveis (42 no total) serão, então, excluídas do dataset de modo a se evitar a já mencionada “maldição da dimensionalidade” (vide título 4.4, acima)

```
In [43]: correlacoes=df_transformado[colunas_a_manter].corr().abs()
print (df_transformado[colunas_a_manter].shape)
correlacoes=pd.DataFrame(correlacoes['01_arrecadacao_2018'].sort_values(ascending=False))
baixas_correlacoes = correlacoes.index[correlacoes['01_arrecadacao_2018']<=.01]
print(baixas_correlacoes.shape)
baixas_correlacoes

(5565, 99)
(42,)

Out[43]: Index(['03_uf_BA',
   '03_atividade_principal_segunda_Pecuária, inclusive apoio à pecuária',
   '03_uf_PI', '03_uf_PB',
   '03_atividade_principal_terceira_Produção florestal, pesca e aquicultura',
   '03_uf_MA', '03_uf_RN', '03_uf_TO', '03_uf_GO', '03_uf_RS',
   '03_atividade_principal_primeira_Pecuária, inclusive apoio à pecuária',
   '03_atividade_principal_primeira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação',
   '03_atividade_principal_terceira_Indústrias de transformação',
   '03_atividade_principal_segunda_Indústrias extractivas', '03_uf_AL',
   '03_uf_MT', '03_uf_PA', 'faixa_arrecadacao_3', '03_uf_CE',
   '03_atividade_principal_segunda_Produção florestal, pesca e aquicultura',
   '03_uf_SE', '03_uf_PR', '03_atividade_principal_terceira_Construção',
   '03_uf_PE',
   '03_atividade_principal_primeira_Produção florestal, pesca e aquicultura',
   '03_atividade_principal_terceira_Indústrias extractivas', '03_uf_RO',
   '03_uf_MS',
   '03_atividade_principal_terceira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação',
   '03_atividade_principal_primeira_Comércio e reparação de veículos automotores e motocicletas',
   '03_atividade_principal_segunda_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação',
   '03_uf_AC', '03_atividade_principal_primeira_Indústrias extractivas',
   '03_uf_SC', '03_atividade_principal_segunda_Construção', '03_uf_AP',
   '03_uf_RR', '03_atividade_principal_primeira_Construção', '03_uf_ES',
   '03_atividade_principal_primeira_Indústrias de transformação',
   '03_atividade_principal_segunda_Administração, defesa, educação e saúde públicas e seguridade social',
   '03_uf_AM'],
  dtype='object')
```

Figura. Vide Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb.

**- Correlação não implica necessariamente causalidade.**

Importante, porém, ressaltar que correlação não implica necessariamente causalidade. Isso quer dizer que a associação entre dois eventos não denota, necessariamente, uma relação de causa e efeito, ou seja, que um dos eventos tenha causado a ocorrência do outro.

Um exemplo palpável que cabe bem à situação seria a seguinte afirmativa: em países do hemisfério norte nota-se que as pessoas tendem a gastar mais em compras no frio. Apesar do gasto em compras estar correlacionado negativamente com a temperatura (- temperatura, + compras), não quer dizer que o frio cause um aumento nas vendas. Uma explicação mais plausível para esse constatado é que datas festivas — como o Natal, por exemplo — coincidem com épocas de frio nos países situados ao norte da Linha do Equador. (MAIAPOLO, Felipe. Correlação não implica em Causalidade. Documento eletrônico. Disponível em <<https://medium.com/@felipemaiapolo/correla%C3%A7%C3%A3o-implica-em-causalidade-8459179ad1bc#>>. Acesso em 1 abr 2021.)

A correlação, como investigado neste capítulo, pode, no entanto, ser um bom ponto inicial para um estudo mais aprofundado sobre efetivas causas.

## **5. Criação e avaliação de Modelos de *Machine Learning***

Neste capítulo serão avaliados modelos de aprendizado de máquina (machine learning), operacionalizados com a aplicação de algoritmos para Regressão. Na sequência, será feita uma análise de resíduos e, por fim, será feita uma análise comparada entre o melhor e o segundo melhor modelos. Os detalhes das implementações aqui mencionadas podem ser consultados no **Apêndice IX – Ntbk 04. Aplicação dos modelos de machine learning - Regressão.ipynb**.

Coletados, transformados e analisados os dados (como apresentado nos capítulos anteriores), pode-se agora criar modelos de aprendizado de máquina de modo a encontrar aquele que proporcione o melhor resultado, lembrando que o objetivo deste projeto é explorar atributos econômicos, sociais, demográficos e geopolíticos disponibilizados de maneira pública pelo IBGE (variáveis independentes) e avaliar sua capacidade de predizer a arrecadação dos municípios brasileiros para o ano de 2018 (variável dependente ou label).

01\_arrecadacao\_2018), informação essa também pública e disponibilizada pela Receita Federal do Brasil.

Para a identificação do melhor modelo foram implementadas uma série de ações e experimentos, a seguir enumerados:

1. Importação do *dataset* que contém os dados de interesse;
2. Criação de três *dataframes* de estudo;
3. Separação das bases de treinamento e de teste;
4. Definição de uma *baseline* (valor de referência) para avaliação dos modelos;
5. Aplicação dos algoritmos e comparação da capacidade de previsão em relação a cada um dos *dataframes* de estudo;
6. Análise dos resíduos, inclusive comparando a performance entre os dois modelos de melhor resultado e analisando os impactos da exclusão dos municípios de maior arrecadação para o algoritmo ElasticNet;
7. Aprimoramento do modelo mais promissor por meio da otimização de hiperparâmetros e uso de *cross validation*.

## 5.1. Ações preliminares

### - Importação do *dataset* com os dados de interesse.

A base utilizada para a realização dos experimentos foi “**ntbk 03 – dataset consolidado - maiores correlacoes.csv**”. Detalhes sobre a criação dessa base pode ser obtida no **Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb**.

O *dataset* possui 5565 linhas e 45 variáveis. Com exceção de ‘municipio-uf’, todas as demais variáveis já estão em formatos numéricos. Embora numérica, a variável ‘faixa\_arrecadacao’, criada neste projeto para classificação das arrecadações, não foi utilizada para o modelo, vez que sua criação exige conhecer previamente o valor das arrecadações o que, obviamente, não estará disponível em uma eventual predição de futuras arrecadações – ela, porém, será útil adiante para a análise dos resíduos.

```
In [13]: df_integral=pd.read_csv('ntbk 03 - dataset consolidado - maiores correlacoes.csv')

print(df_integral.shape)
print (df_integral.dtypes)
df_integral.head()

(5565, 45)

municipio_uf          object
faixa_arrecadacao     int64
01_arrecadacao_2018   float64
02_unidades_atuantes_ float64
02_ocupados_assalariados_ float64
02_salario_medio_mensal_em_Reais float64
03_uf_DF              int64
03_uf_MG              int64
03_uf_RJ              int64
03_uf_SP              int64
03_vlr_adic_bruto_agropecuaria_ float64
03_vlr_adic_bruto_industria_    float64
03_vlr_adic_bruto_servicos_    float64
03_vlr_adic_bruto_administracao_ float64
03_pib                float64
03_pib_per_capita      float64
03_atividade_principal_primeira_Administração, defesa, educação e saúde públicas e seguridade social int64
03_atividade_principal_primeira_Agricultura, inclusive apoio à agricultura e a pós colheita int64
03_atividade_principal_primeira_Demais serviços int64
03_atividade_principal_segunda_Agricultura, inclusive apoio à agricultura e a pós colheita int64
03_atividade_principal_segunda_Comércio e reparação de veículos automotores e motocicletas int64
03_atividade_principal_segunda_Demais serviços int64
03_atividade_principal_segunda_Indústrias de transformação int64
03_atividade_principal_terceira_Administração, defesa, educação e saúde públicas e seguridade social int64
03_atividade_principal_terceira_Agricultura, inclusive apoio à agricultura e a pós colheita int64
03_atividade_principal_terceira_Comércio e reparação de veículos automotores e motocicletas int64
03_atividade_principal_terceira_Demais serviços int64
03_atividade_principal_terceira_Pecuária, inclusive apoio à pecuária int64
05_0-19_anos_%         float64
05_20-39_anos_%       float64
05_40-59_anos_%       float64
05_60+anos_%          float64
06_Sem_Instrucao_e_Fundamental_Incompleto% float64
06_Medio_Incompleto_% float64
06_Superior_Incompleto_% float64
06_Superior_Completo_% float64
06_Ideterminado_%    float64
07_Ate_1SM_%          float64
07_Ate_2SM_%          float64
07_Ate_3SM_%          float64
07_Ate_5SM_%          float64
07_Ate_10SM_%         float64
07_Ate_20SM_%         float64
07_20SM+_%           float64
07_Sem_Rendimento_%  float64
dtype: object
```

Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

### - Criação dos *dataframes* de estudo.

Foram criados três *dataframes* de estudo, derivados do *dataframe* **df\_integral** (instanciado com o *dataset* “*ntbk 03 - dataset consolidado - maiores Correlacoes.csv*” – vide sequência de comandos da figura acima). O primeiro (**df\_original**) manterá os dados em sua forma original (excluídas as variáveis ‘municipio-uf’ e ‘faixa\_arrecadacao’). Os dois seguintes, serão resultado da conversão das variáveis dependentes: um para a base logarítmica natural (**df\_log**) e, outro, utilizando a técnica MinMax (**df\_MinMax**). A variável dependente (label: *01\_arrecadacao\_2018*) será mantida em sua forma original, por ser esse o objetivo declarado para este projeto.

### - df\_original

Contém os dados originais da base (excluído o conteúdo das variáveis ‘municipio-uf’ e ‘faixa\_arrecadacao’), ou seja, sem nenhuma conversão.

	01_arrecadacao_2018	02_unidades_atuantes_%	02_ocupados_assalariados_%	02_salario_medio_mensal_em_Reais	03_uf_DF	03_uf_MG	03_uf_RJ
count	5.565000e+03	5565.000000	5565.000000	5565.000000	5565.000000	5565.000000	5565.000000
mean	2.494351e+08	97.442228	85.305250	1926.828422	0.000180	0.153279	0.016532
std	4.865354e+09	2.952571	6.761261	465.926999	0.013405	0.360289	0.127521
min	4.165360e+05	56.103286	21.124207	719.780000	0.000000	0.000000	0.000000
25%	3.775316e+06	96.551724	81.526104	1617.660000	0.000000	0.000000	0.000000
50%	9.175717e+06	97.983015	85.862069	1842.640000	0.000000	0.000000	0.000000
75%	3.150111e+07	99.206349	89.880263	2122.500000	0.000000	0.000000	0.000000
max	2.804407e+11	100.000000	99.489796	6552.490000	1.000000	1.000000	1.000000

Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

### - df\_log

Neste *dataframe*, as 42 variáveis independentes remanescentes no df\_original foram convertidas para a base logarítmica natural, mantida sem conversão a variável dependente (nossa label: 01\_arrecadacao\_2018).

	01_arrecadacao_2018	02_unidades_atuantes_%_In	02_ocupados_assalariados_%_In	02_salario_medio_mensal_em_Reais_In	03_uf_DF_In	03_uf_MG_In	03_uf_RJ_In
count	5.565000e+03	5565.000000	5565.000000	5565.000000	5565.000000	5565.000000	5565.000000
mean	2.494351e+08	4.578749	4.442744	7.538907	-12.997664	-11.007367	-11.007367
std	4.865354e+09	0.032886	0.086424	0.215903	0.174265	4.683757	4.683757
min	4.165360e+05	4.027194	3.050420	6.578946	-13.000000	-13.000000	-13.000000
25%	3.775316e+06	4.570079	4.400923	7.388736	-13.000000	-13.000000	-13.000000
50%	9.175717e+06	4.584794	4.452742	7.518955	-13.000000	-13.000000	-13.000000
75%	3.150111e+07	4.597202	4.498478	7.660350	-13.000000	-13.000000	-13.000000
max	2.804407e+11	4.605170	4.600055	8.787600	0.000000	0.000000	0.000000

Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

Abaixo a função definida para a conversão para a base logarítmica.

```
In [2]: # Converte variáveis numéricas para a base Logarítmica
# Pode ser passado por parâmetro uma única variável, ou mesmo o dataframe inteiro.

def converte_para_ln(df,cols=''):
    if cols == '':
        cols = df.columns.tolist()
    contador = 0
    for col in cols:

        if df[col].dtype in [np.int64, np.int32, np.float64]:
            nova_coluna = col + "_ln"
            # A linha abaixo calcula o log de cada valor das colunas numéricas
            # O abs para x é porque o np.log retorna erro para valores negativos, usaremos, assim, o valor do módulo.
            # A função log retorna um erro (-inf) quando o x = 0
            # Assim, convencionou-se que, quando o valor for zero, a função retornará -13,82 (que é o ln de 0,000001).
            df[nova_coluna]=df[col].apply(lambda x: np.log(abs(x)) if x!= 0 else -13,82).astype(float)
            contador = contador + 1
    print ('Colunas criadas:',contador)
    return('Concluído')
```

Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

### - df\_MinMax

Neste dataset, as 42 variáveis independentes remanescentes no df\_original foram convertidas utilizando-se a técnica Mínimo-Máximo, mantida sem conversão a variável dependente (nossa label: 01\_arrecadacao\_2018).

Por essa técnica, as variáveis são transformadas para valores entre 0 (zero) e 1 (um).

```
In [16]: df_MinMax = df_original.copy()

#Seleciona as colunas numéricas...
colunas_a_converter = df_MinMax.dtypes[df_MinMax.dtypes != "object"].index.tolist()
#... exceto a primeira: '01_arrecadacao_2018'.
colunas_a_converter.pop(0)

# Chama a função que fará a conversão
converte_para_MinMax(df_MinMax, colunas_a_converter)

# Apaga as colunas originais
df_MinMax = df_MinMax.drop(colunas_a_converter, axis=1)
print (df_MinMax.shape)
df_MinMax.describe()

Colunas criadas: 42
(5565, 44)
```

	01_arrecadacao_2018	02_unidades_atuantes_%_minmax	02_ocupados_assalariados_%_minmax	02_salario_medio_mensal_em_Reais_minmax	03_uf_DF_minmax
count	5.565000e+03	5565.000000	5565.000000	5565.000000	5565.000000
mean	2.494351e+08	0.941732	0.818995	0.206945	0.00
std	4.865354e+09	0.067262	0.086278	0.079882	0.01
min	4.165360e+05	0.000000	0.000000	0.000000	0.00
25%	3.775316e+06	0.921446	0.770771	0.153939	0.00
50%	9.175717e+06	0.954052	0.826101	0.192511	0.00
75%	3.150111e+07	0.981920	0.877376	0.240492	0.00
max	2.804407e+11	1.000000	1.000000	1.000000	1.00

Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

A conversão ocorre de acordo com a fórmula abaixo, onde:

- **X**: é o valor a ser transformado; e
- **Xmin** e **Xmax**: apontam para a amplitude da variável.

$$X_{transformado} = \frac{(X - X_{min})}{X_{max} - X_{min}} \times (X_{max} - X_{min}) + X_{min}$$

Abaixo a função definida para esse procedimento. Note que, para isso, utilizamos a classe *MinMaxScaler* do pacote *sklearn.preprocessing*.

```
In [3]: # Converte variáveis numéricas usando a técnica MinMax
from sklearn.preprocessing import MinMaxScaler

def converte_para_MinMax(df,cols):
    contador = 0
    for col in cols:
        #define um novo nome para as colunas transformadas
        nova_coluna = col + "_minmax"

        # transforma os dados para o formato exigido pelo MinMaxScaler() e procede à transformação.
        dados_a_transformar = df[col].values.reshape(-1, 1)
        scaler = MinMaxScaler()
        dados_transformados = scaler.fit_transform(dados_a_transformar)

        # restabelece o formato DataFrame e atribui o valor à nova coluna criada
        df[nova_coluna] = DataFrame(dados_transformados)
        #print (col, nova_coluna, " - Min", scaler.data_min_, " - Máx", scaler.data_max_, " - Scaler", scaler.scale_)

        contador = contador + 1
    print ('Colunas criadas:',contador)
    return('Concluído')
```

Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

## - Separação dos *dataframes* em base de treinamento e de teste.

Para a aplicação inicial dos modelos, foram separados os *dataframes* em base treinamento e base teste. A distribuição foi de 70% e 30% entre as bases de treinamento e teste.

Como se sabe, a variável dependente (a variável a ser predita) será '01\_arrecadacao\_2018'.

- **df\_original**:

```
In [17]: #Identificar a coluna que contém o Label (Y)
Y_orig=df_original['01_arrecadacao_2018']

#X será as demais colunas. Dado X tenho Y.
X_orig=df_original.drop(['municipio-uf','01_arrecadacao_2018'],axis=1)

x_orig_train, x_orig_test, y_orig_train, y_orig_test = train_test_split(X_orig,Y_orig,test_size=0.3, random_state=17)
```

Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

### - df\_log:

```
In [19]: #Identificar a coluna que contém o Label (Y)
y_log=df_log['01_arrecadacao_2018']

#X será as demais colunas. Dado X tenho Y.
x_log=df_log.drop(['municipio-uf','01_arrecadacao_2018'],axis=1)

x_log_train, x_log_test, y_log_train, y_log_test = train_test_split(x_log,y_log,test_size=0.3, random_state=17)
```

Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

### - df\_MinMax:

```
In [18]: #Identificar a coluna que contém o Label (Y)
y_minmax=df_MinMax['01_arrecadacao_2018']

#X será as demais colunas. Dado X tenho Y.
X_minmax=df_MinMax.drop(['municipio-uf','01_arrecadacao_2018'],axis=1)

x_minmax_train, x_minmax_test, y_minmax_train, y_minmax_test = train_test_split(X_minmax,Y_minmax,test_size=0.3, random_state=17)
```

Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

Mais adiante, porém, quando do ajuste dos hiperparâmetros para o melhor modelo encontrado, não mais será feita uma única separação, fixa, do *dataframe* em apenas duas bases (treinamento e teste), mas será utilizada a técnica *cross validation*, com o método K-Fold.

No K-Fold, o conjunto de dados é dividido em grupos (normalmente entre 5 e 10), em seguida um dos grupos é escolhido de maneira aleatória um para ser o grupo de validação ou teste. O restante dos grupos se junta, para treinar o modelo. Após treinado, o modelo é aplicado no grupo de validação e os resultados são comparados e a taxa de erro calculada. Esse método se repete até que todos os grupos tenham sido considerados como grupo de validação e todos tenham um percentual de erro, no fim se calcula o erro médio do modelo. (VAZ, Arthur Lamblet. Cross Validation de maneira didática. Documento eletrônico. Disponível em < Fonte: <https://medium.com/data-hackers/crossvalidation-de-maneira-did%C3%A1tica-79c9b080a6ec> >. Acesso em 1 abr 2021.)

## - Definição da métrica e de uma *baseline* para avaliação dos resultados.

### - Métricas

A qualidade do modelo será aferida a partir da métrica MAE (Mean Absolute Error), embora também seja feita referência as métricas RMSE (Root Mean Square Error) e R<sup>2</sup>:

- MAE calcula o "erro absoluto médio" dos erros entre valores observados (reais) e previsões (hipóteses) – para essa métrica, quanto menor o valor obtido, melhor;

$$\text{MAE} = \frac{1}{n} \sum_{j=1}^n |y_j - \hat{y}_j|$$

Para o cálculo utilizaremos a métrica *mean\_absolute\_error*, disponível no módulo *sklearn.metrics*.

- RMSE é a medida que calcula "a raiz quadrática média dos erros" entre valores observados (reais) e previsões (hipóteses), ou, é a raiz quadrada do erro médio quadrado (*mean squared error*) – para essa métrica, quanto menor o valor obtido, melhor;

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2}$$

Para o cálculo do RMSE foi definida a função “*rmse(y\_true, y\_pred)*”, abaixo, que calcula a raiz quadrada do valor da métrica *mean\_squared\_error*, disponível no módulo *sklearn.metrics*.

```
In [4]: # Calcula a métrica Root Mean Squared Error
def rmse(y_true, y_pred):
    return mean_squared_error(y_true,y_pred)**0,5
```

Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

- O R<sup>2</sup>, ou Coeficiente de Determinação, é uma métrica que visa expressar a quantidade da variância dos dados que é explicada pelo modelo construído. Em outras palavras, essa medida calcula qual a porcentagem da variância que pode ser explicada pelo modelo de regressão e, portanto, nos diz o quanto próximas as medidas reais estão do nosso modelo. Esta métrica gera valores entre 0 e 1 e quanto maior o valor obtido maior o grau de explicação para a variância obtida.

$$R^2 = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

Já o R<sup>2</sup> é calculado a partir do método “.score” para a classe *LinearRegression*

### **- Valor de referência (baseline) para avaliação dos modelos.**

Para essa apuração, utilizou-se a função para o cálculo do MAE, vista no item imediatamente acima.

Como baseline, vamos atribuir indistintamente ao y\_pred (valor predito pelo modelo) o valor R\$ 249.435.087,56 (que é a média para a variável "01\_arrecadacao\_2018").

```
In [20]: #rmse(y_test, np.zeros(y_test.shape))
baseline_orig = mean_absolute_error(y_orig_test, np.full(y_orig_test.shape, 249435087.56, dtype=float))
print ('Baseline para o modelo utilizando os dados originais', baseline_orig)

#rmse(y_test, np.zeros(y_test.shape))
baseline_minmax = mean_absolute_error(y_minmax_test, np.full(y_minmax_test.shape, 249435087.56, dtype=float))
print ('Baseline para o modelo utilizando os dados transformados - MinMax', baseline_minmax)

#rmse(y_test, np.zeros(y_test.shape))
baseline_log = mean_absolute_error(y_log_test, np.full(y_log_test.shape, 249435087.56, dtype=float))
print ('Baseline para o modelo utilizando os dados transformados - Log Natural', baseline_log)

Baseline para o modelo utilizando os dados originais 504698969.2189041
Baseline para o modelo utilizando os dados transformados - MinMax 504698969.2189041
Baseline para o modelo utilizando os dados transformados - Log Natural 504698969.2189041
```

Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

Desse modo, para superarem um modelo que considera a média da distribuição total como previsão indistinta e, assim, terem alguma validade, nossos modelos deverão ter um MAE inferior a **R\$ 504.698.969,21**.

## 5.2. Aplicação dos algoritmos de machine learning

A seguir avaliaremos a capacidade de predição dos seguintes modelos de aprendizado supervisionado de máquina, aplicados sobre nossas três bases de estudo:

- Regressão Linear:

Em estatística, Regressão Linear é uma equação para se estimar o valor de uma variável  $y$ , dados os valores de outras variáveis independentes. Ou seja, busca-se uma relação entre as variáveis que possa ser representada como uma função; (CompPET, publicado por ALVES, Salomão. A matemática por trás de um dos principais algoritmos de Machine Learning. Documento eletrônico. Disponível em <<https://comppet.github.io/PrintF/regressao-linear-a-matematica-por-tras-de-um-dos-principais-algoritmos-de-machine-learning/>>. Acesso em 1 abr 2021.)

- Lasso, Ridge e ElasticNet:

Dentro da regressão linear, Ridge e Lasso são formas de regularizar a função resultante de modo a obter uma melhor relação entre os erros *bias* e *variance*. Regularização é a inserção de um *bias* ao modelo e tem por efeito desencorajar o ajuste excessivo dos dados a fim de diminuir sua variância.

Lasso inclui um termo de regularização chamado L1. Nesse caso, a penalização é calculada a partir dos módulos dos coeficientes da função. Além de diminuir a variância do modelo, essa regularização, quando há múltiplas *features* altamente correlacionadas, seleciona apenas uma dessas *features* e zera os coeficientes das outras, de forma a minimizar a penalização L1. Desse modo, dizemos que esse modelo realiza *feature selection* automaticamente, gerando vários coeficientes com peso zero, ou seja, que são ignorados pelo modelo. Isso facilita a interpretação do modelo.

Ridge inclui um termo de regularização chamado L2. Nesse caso, a penalização consiste nos quadrados dos coeficientes, ao invés de seus módulos. Como a penalização L2 é desproporcionalmente maior para coeficientes maiores, a regularização Ridge faz com que *features* correlacionadas tenham coeficientes parecidos.

ElasticNet trata exatamente de combinar os termos de regularização de L1 e L2. (DEVAY, Andre. Modelos de Predição | Regressão de Ridge e Lasso - Um olhar para as parcelas L1 e L2 (Ridge e Lasso). Documento eletrônico. Disponível

em <<https://medium.com/turing-talks/turing-talks-20-regress%C3%A3o-de-ridge-e-lasso-a0fc467b5629>>. Acesso em 1 abr 2021.)

- Random Forest: é um algoritmo de aprendizagem supervisionada que cria uma “floresta” de um modo aleatório. A “floresta” é uma combinação (*ensemble*) de árvores de decisão, na maioria dos casos treinados com o método de *bagging*. A ideia principal do método de *bagging* é que a combinação dos modelos de aprendizado o que aumenta o resultado geral. Dizendo de outro modo, o algoritmo cria várias árvores de decisão e as combina para obter uma predição com maior acurácia e mais estável. (COSTA DA SILVA, Josenildo. Aprendendo em uma Floresta Aleatória - Veja a floresta e não as árvores . Documento eletrônico. Disponível em <<https://medium.com/machina-sapiens/o-algoritmo-da-floresta-aleat%C3%B3ria-3545f6babdf8#>>. Acesso em 1 abr 2021.)

Neste primeiro momento, os algoritmos serão aplicados sem a definição de hiperparâmetros.

Criou-se a função abaixo. Por ela, apura-se o resultado da aplicação dos modelos tanto nas bases de treinamento como nas de teste. Cria também um dicionário com esses resultados que alimentará um *dataframe* para comparações.

```
In [6]: #aplica modelos de machine Learning

def aplicar_modelo_ML(df, modelo_definido, x_train, y_train, x_test, y_test):
    print('Dataframe:',df)
    modelo = modelo_definido
    print("Modelo:",modelo)

    modelo.fit(x_train,y_train)
    y_pred = modelo.predict(x_train)
    rmse_train=rmse(y_train,y_pred)
    mae_train = mean_absolute_error(y_train,y_pred)
    print("MAE na base de treinamento: ",mae_train)
    print("RMSE na base de treinamento: ",rmse_train)
    print("R² na base de treinamento: {:.2f}".format(modelo.score(x_train, y_train)))

    modelo.fit(x_train,y_train)
    y_pred = modelo.predict(x_test)
    rmse_test=rmse(y_test,y_pred)
    mae_test = mean_absolute_error(y_test,y_pred)
    print("MAE na base de teste: ",mae_test)
    print("RMSE na base de teste: ",rmse_test)
    print("R² na base de teste: {:.2f}".format(modelo.score(x_test, y_test)))
    print("\n")

    y_test_ln = np.log(np.abs(np.where(y_test==0,0.00001,y_test)))
    y_pred_ln = np.log(np.abs(np.where(y_pred==0,0.00001,y_pred)))

    #imprime_scatter_plot(np.log(y_test),np.log(y_pred))
    imprime_scatter_plot(y_test_ln,y_pred_ln)

    resultados = {'Dataframe': df,
                  'Modelo': [str(modelo)],
                  'MAE - Base Treinamento':mae_train,
                  'MAE - Base Teste':mae_test,
                  'RMSE - Base Treinamento': rmse_train,
                  'RMSE - Base Teste': rmse_test,
                  'R2 (Acurácia) - Base Treinamento': modelo.score(x_train, y_train),
                  'R2 (Acurácia) - Base Teste': modelo.score(x_test, y_test)}

    return resultados
```

Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

## - Regressão Linear

Será avaliada, aqui, a capacidade de predição do algoritmo ***sklearn.linear\_model.LinearRegression()***.

```
In [21]: modelo = LinearRegression()

#Aplicação LinearRegression() no df_original
resultado = aplicar_modelo_ML('df_original', modelo, x_orig_train, y_orig_train, x_orig_test, y_orig_test)
df_resultados_modelos = pd.DataFrame(resultado)

#Aplicação LinearRegression() no df_MinMax
resultado = aplicar_modelo_ML('df_MinMax', modelo, x_minmax_train, y_minmax_train, x_minmax_test, y_log_test)
df_resultados_modelos.append(resultado, ignore_index=True)

#Aplicação LinearRegression() no df_log
resultado = aplicar_modelo_ML('df_log', modelo, x_log_train, y_log_train, x_log_test, y_log_test)
df_resultados_modelos.append(resultado, ignore_index=True)

#Aponta o melhor modelo.
print ('O melhor resultado foi com o modelo', df_resultados_modelos['Modelo'][df_resultados_modelos['MAE - Base Teste'] == min(df_resultados_modelos['MAE - Base Teste'])])
print ('Para o dataframe', df_resultados_modelos['Dataframe'][df_resultados_modelos['MAE - Base Teste'] == min(df_resultados_modelos['MAE - Base Teste'])])
print ('MAE', min(df_resultados_modelos['MAE - Base Teste']))

df_resultados_modelos.round(2)
```

Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

Como resultado do processamento da célula acima, nota-se que o melhor modelo foi o obtido com a aplicação do modelo ao dataset **df\_MinMax**, com um **MAE de R\$ 343.549.603,80**, calculado em relação à **base de testes** (vide Apêndice IX – Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb).

	Dataframe	Modelo	MAE - Base Treinamento	MAE - Base Teste	RMSE - Base Treinamento	RMSE - Base Teste	R2 (Acurácia) - Base Treinamento	R2 (Acurácia) - Base Teste
0	df_original	LinearRegression()	2.817382e+08	3.454691e+08	7.176948e+08	1.593124e+09	0.98	0.91
1	df_MinMax	[LinearRegression()]	2.817446e+08	3.435496e+08	7.176948e+08	1.566404e+09	0.98	0.92
2	df_log	[LinearRegression()]	8.104379e+08	9.095102e+08	4.432094e+09	5.123286e+09	0.08	0.10

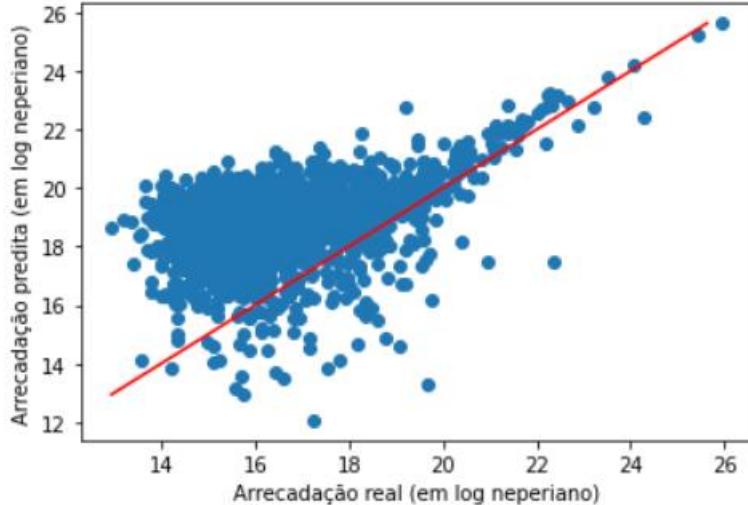


Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

Esse MAE, embora ainda elevado, já é inferior à *baseline* calculada (que foi de mais de R\$ 504,69 mi – vide capítulo anterior.).

## - Lasso

Será avaliada, aqui, a capacidade de predição do algoritmo ***sklearn.linear\_model.Lasso()***.

```
In [23]: # O parâmetro max_iter foi definido abaixo por conta de um aviso exibido pelo python no momento da execução
modelo = Lasso()

#Aplicação Lasso() no df_original
resultado = aplicar_modelo_ML('df_original', modelo, x_orig_train, y_orig_train, x_orig_test, y_orig_test)
df_resultados_modelos = df_resultados_modelos.append(resultado,ignore_index=True)

#Aplicação Lasso() no df_MinMax
resultado = aplicar_modelo_ML('df_MinMax', modelo, x_minmax_train, y_minmax_train, x_minmax_test, y_log_test)
df_resultados_modelos = df_resultados_modelos.append(resultado,ignore_index=True)

#Aplicação Lasso() no df_Log
resultado = aplicar_modelo_ML('df_log', modelo, x_log_train, y_log_train, x_log_test, y_log_test)
df_resultados_modelos = df_resultados_modelos.append(resultado,ignore_index=True)

#Aponta o melhor modelo.
print ('O melhor resultado foi com o modelo',df_resultados_modelos['Modelo'][df_resultados_modelos['MAE - Base Teste'] == min(df_resultados_modelos['MAE - Base Teste'])])
print ('Para o Dataframe',df_resultados_modelos['Dataframe'][df_resultados_modelos['MAE - Base Teste'] == min(df_resultados_modelos['MAE - Base Teste'])])
print('MAE',min(df_resultados_modelos['MAE - Base Teste']))

df_resultados_modelos
```

Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

Como resultado do processamento da célula acima, nota-se que o melhor modelo foi obtido com a aplicação do modelo ao dataset **df\_MinMax**, com um **MAE de R\$ 343.182.047,33**, calculado em relação à **base de testes** (vide Apêndice IX – Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb).

Out[23]:	Dataframe	Modelo	MAE - Base Treinamento	MAE - Base Teste	RMSE - Base Treinamento	RMSE - Base Teste	R2 (Acurácia) - Base Treinamento	R2 (Acurácia) - Base Teste
6	df_original	[Lasso()]	2.817792e+08	3.431821e+08	7.177482e+08	1.566357e+09	0.975843	0.915741
7	df_MinMax	[Lasso()]	2.817792e+08	3.431820e+08	7.177482e+08	1.566357e+09	0.975843	0.915741
8	df_log	[Lasso()]	8.104379e+08	9.095101e+08	4.432094e+09	5.123286e+09	0.078884	0.098576

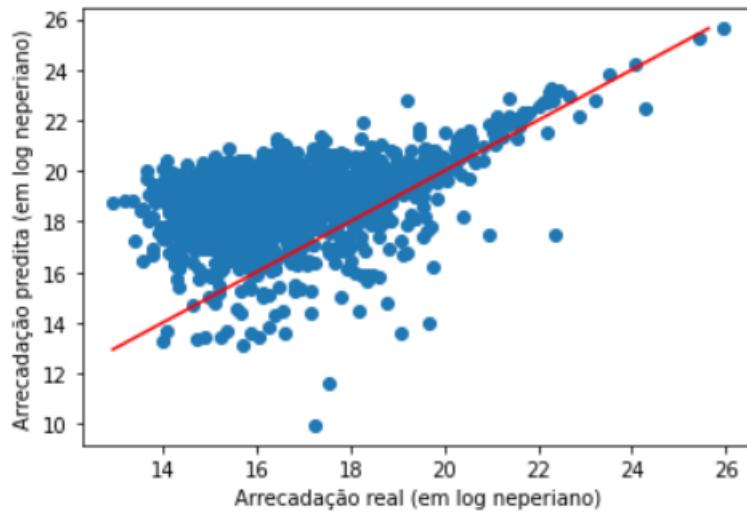


Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

O MAE obtido com a aplicação do *Lasso()* - de R\$ 343.182.047,33 - é ligeiramente inferior ao MAE do algoritmo *LinearRegression()* - R\$ 343.549.603,80, fazendo com o *Lasso()* seja o algoritmo de melhor resultado até o momento.

## - Ridge

Será avaliada, aqui, a capacidade de predição do algoritmo *sklearn.linear\_model.Ridge()*.

```
In [22]: modelo = Ridge()

#Aplicação Ridge() no df_original
resultado = aplicar_modelo_ML('df_original', modelo, x_orig_train, y_orig_train, x_orig_test, y_orig_test)
df_resultados_modelos = df_resultados_modelos.append(resultado, ignore_index=True)

#Aplicação Ridge() no df_MinMax
resultado = aplicar_modelo_ML('df_MinMax', modelo, x_minmax_train, y_minmax_train, x_minmax_test, y_log_test)
df_resultados_modelos = df_resultados_modelos.append(resultado, ignore_index=True)

#Aplicação Ridge() no df_Log
resultado = aplicar_modelo_ML('df_Log', modelo, x_log_train, y_log_train, x_log_test, y_log_test)
df_resultados_modelos = df_resultados_modelos.append(resultado, ignore_index=True)

#Aponta o melhor modelo.
print ('O melhor resultado foi com o modelo', df_resultados_modelos['Modelo'][df_resultados_modelos['MAE - Base Teste'] == min(df_resultados_modelos['MAE'])])
print ('Para o dataframe', df_resultados_modelos['Dataframe'][df_resultados_modelos['MAE - Base Teste'] == min(df_resultados_modelos['MAE'])])
print ('MAE', min(df_resultados_modelos['MAE - Base Teste'])))
```

Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

Como resultado do processamento da célula acima, nota-se que o melhor modelo foi o obtido com a aplicação do modelo ao dataset **df\_original**, com um

**MAE de R\$ 343.151.529,80**, calculado em relação à base de testes (vide Apêndice IX – Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb).

Out[22]:

	Dataframe	Modelo	MAE - Base Treinamento	MAE - Base Teste	RMSE - Base Treinamento	RMSE - Base Teste	R2 (Acurácia) - Base Treinamento	R2 (Acurácia) - Base Teste
3	df_original	[Ridge()]	2.816667e+08	3.431515e+08	7.177380e+08	1.566276e+09	0.975844	0.915750
4	df_MinMax	[Ridge()]	3.436276e+08	4.282319e+08	2.299322e+09	3.152693e+09	0.752089	0.658653
5	df_log	[Ridge()]	8.080235e+08	9.079830e+08	4.432133e+09	5.123834e+09	0.078868	0.098383

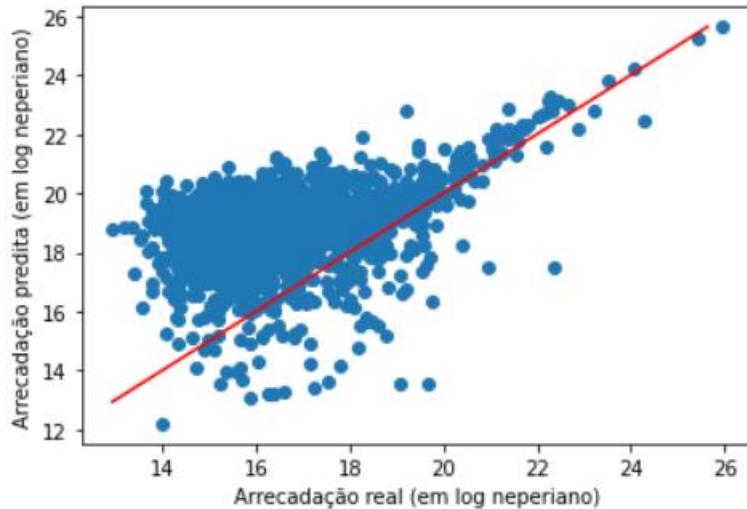


Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

O MAE obtido com a aplicação do *Ridge()* - de R\$ 343.151.529,80 - é discretamente inferior ao MAE do modelo *Lasso()* - R\$ 343.182.047,33, fazendo com que passe a ser o de melhor resultado até o momento.

### - ElasticNet

Será avaliada, aqui, a capacidade de predição do algoritmo ***sklearn.linear\_model.ElasticNet()***.

```
In [24]: modelo = ElasticNet()

#Aplicação ElasticNet() no df_original
resultado = aplicar_modelo_ML('df_original', modelo, x_orig_train, y_orig_train, x_orig_test, y_orig_test)
df_resultados_modelos = df_resultados_modelos.append(resultado, ignore_index=True)

#Aplicação ElasticNet() no df_MinMax
resultado = aplicar_modelo_ML('df_MinMax', modelo, x_minmax_train, y_minmax_train, x_minmax_test, y_log_test)
df_resultados_modelos = df_resultados_modelos.append(resultado, ignore_index=True)

#Aplicação ElasticNet() no df_Log
resultado = aplicar_modelo_ML('df_Log', modelo, x_log_train, y_log_train, x_log_test, y_log_test)
df_resultados_modelos = df_resultados_modelos.append(resultado, ignore_index=True)

#Aponta o melhor modelo.
print ('O melhor resultado foi com o modelo',df_resultados_modelos['Modelo'][df_resultados_modelos['MAE - Base Teste'] == min(df_resultados_modelos['MAE - Base Teste'])])
print ('Para o Dataframe',df_resultados_modelos['Dataframe'][df_resultados_modelos['MAE - Base Teste'] == min(df_resultados_modelos['MAE - Base Teste'])])
print ('MAE',min(df_resultados_modelos['MAE - Base Teste'])))

df_resultados_modelos
```

Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

Como resultado do processamento da célula acima, nota-se que o melhor modelo foi o obtido com a aplicação do modelo ao dataset **df\_original**, com um **MAE de R\$ 336.341.664,68**, calculado em relação à **base de testes** (vide Apêndice IX – Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb).

Out[24]:	Dataframe	Modelo	MAE - Base Treinamento	MAE - Base Teste	RMSE - Base Treinamento	RMSE - Base Teste	R2 (Acurácia) - Base Treinamento	R2 (Acurácia) - Base Teste
9	df_original	[ElasticNet()]	2.781357e+08	3.363417e+08	7.284709e+08	1.559539e+09	0.975116	0.916473
10	df_MinMax	[ElasticNet()]	3.167394e+08	4.377077e+08	4.590456e+09	5.370983e+09	0.011884	0.009306
11	df_log	[ElasticNet()]	5.628429e+08	6.833164e+08	4.467609e+09	5.245745e+09	0.064063	0.054968

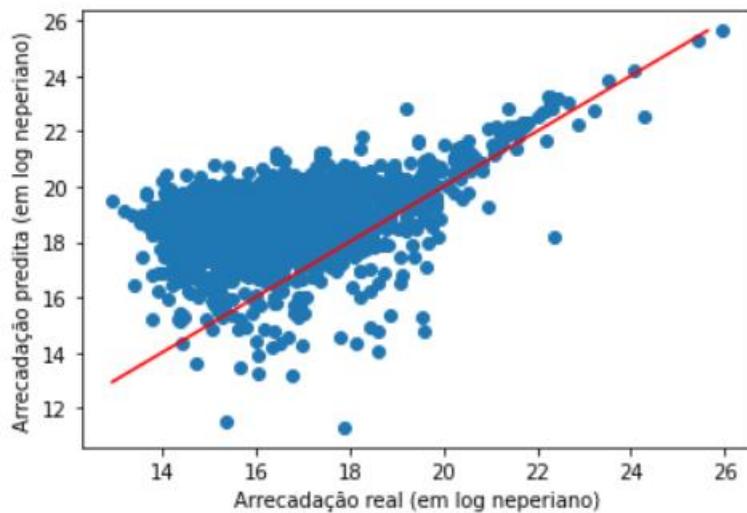


Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

O MAE obtido com a aplicação do *ElasticNet()* - de R\$ 336.341.664,68 - é inferior ao MAE do modelo *Ridge()* - R\$ 343.151.529,80, fazendo com que o primeiro suceda este último como o de melhor resultado até o momento.

## - Random Forest

Será avaliada, aqui, a capacidade de predição do algoritmo `sklearn.ensemble.RandomForestRegressor()`.

```
In [25]: modelo = RandomForestRegressor()

#Aplicação RandomForestRegressor() no df_original
resultado = aplicar_modelo_ML('df_original', modelo, x_orig_train, y_orig_train, x_orig_test, y_orig_test)
df_resultados_modelos = df_resultados_modelos.append(resultado, ignore_index=True)

#Aplicação RandomForestRegressor() no df_MinMax
resultado = aplicar_modelo_ML('df_MinMax', modelo, x_minmax_train, y_minmax_train, x_minmax_test, y_log_test)
df_resultados_modelos = df_resultados_modelos.append(resultado, ignore_index=True)

#Aplicação RandomForestRegressor() no df_Log
resultado = aplicar_modelo_ML('df_log', modelo, x_log_train, y_log_train, x_log_test, y_log_test)
df_resultados_modelos = df_resultados_modelos.append(resultado, ignore_index=True)

#Aponta o melhor modelo.
print ('O melhor resultado foi com o modelo', df_resultados_modelos['Modelo'][df_resultados_modelos['MAE - Base Teste'] == min(df_resultados_modelos['MAE - Base Teste'])])
print ('Para o dataframe', df_resultados_modelos['Dataframe'][df_resultados_modelos['MAE - Base Teste'] == min(df_resultados_modelos['MAE - Base Teste'])])
print('MAE', min(df_resultados_modelos['MAE - Base Teste']))

df_resultados_modelos
```

Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

Como resultado do processamento da célula acima, nota-se que o melhor modelo foi o obtido com a aplicação do modelo ao dataset **df\_log**, com um **MAE de R\$ 150.471.027,94**, calculado em relação à **base de testes** (vide Apêndice IX – Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb).

Out[25]:	Dataframe	Modelo	MAE - Base Treinamento	MAE - Base Teste	RMSE - Base Treinamento	RMSE - Base Teste	R2 (Acurácia) - Base Treinamento	R2 (Acurácia) - Base Teste
12	df_original	[RandomForestRegressor()]	5.274192e+07	2.590430e+08	1.637213e+09	4.904329e+09	0.897742	0.173978
13	df_MinMax	[RandomForestRegressor()]	4.962830e+07	2.723002e+08	1.277936e+09	4.963558e+09	0.847931	0.153907
14	df_log	[RandomForestRegressor()]	5.125396e+07	1.409964e+08	1.440971e+09	1.780061e+09	0.881810	0.891182

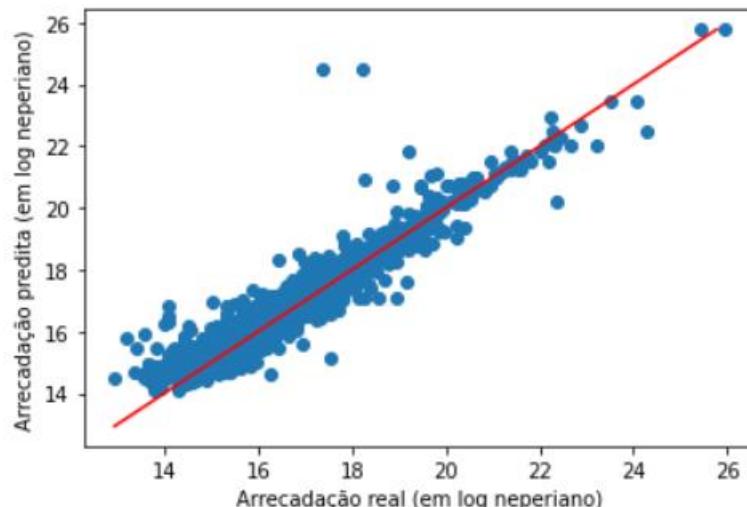


Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

O MAE obtido com a aplicação do *RandomForestRegressor()* - de R\$ 150.471.027,94 – é significativamente inferior ao MAE do modelo *ElasticNet()* - R\$ 336.341.664,68, fazendo com que aquele modelo passe a ser o de melhor resultado.

Inclusive, se comparados os gráficos de dispersão que representam a arrecadação real e a arrecadação predita é possível claramente observar a maior convergência dos valores gerados pelo *RandomForestRegressor()* em relação aos demais algoritmos, especialmente no tocante às menores arrecadações.

### - Melhor modelo identificado para este primeiro experimento

Concluídos esses primeiros experimentos em que aplicados os cinco algoritmos acima, sem qualquer definição de hiperparâmetros, aos três *dataframes* de estudo (*df\_original*, *df\_log* e *df\_MinMax*), têm-se que **o melhor resultado apurado em relação à base de testes foi com a aplicação do algoritmo *RandomForestRegressor()* sobre o *dataframe df\_log*, com MAE de R\$ 150.471.027,94.**

```
In [26]: print ('O melhor resultado foi com o modelo:',df_resultados_modelos['Modelo'][df_resultados_modelos['MAE - Base Teste']] == min(df_resultados_modelos['MAE - Base Teste']))
print ('Para o dataframe:',df_resultados_modelos['Dataframe'][df_resultados_modelos['MAE - Base Teste']] == min(df_resultados_modelos['MAE - Base Teste']))
print('MAE:',min(df_resultados_modelos['MAE - Base Teste']))
print('RMSE:',min(df_resultados_modelos['RMSE - Base Teste']))
print('R²:',max(df_resultados_modelos['R2 (Acurácia) - Base Teste']))
df_resultados_modelos
```

O melhor resultado foi com o modelo: [RandomForestRegressor()]
Para o dataframe: df\_log
MAE: 128759274.8120163
RMSE: 1559539289.6383786
R²: 0.9164734130787983

	Dataframe	Modelo	MAE - Base Treinamento	MAE - Base Teste	RMSE - Base Treinamento	RMSE - Base Teste	R2 (Acurácia) - Base Treinamento	R2 (Acurácia) - Base Teste
0	df_original	LinearRegression()	2.817382e+08	3.454691e+08	7.176948e+08	1.593124e+09	0.975847	0.912837
1	df_MinMax	[LinearRegression()]	2.817446e+08	3.435496e+08	7.176948e+08	1.566404e+09	0.975847	0.915736
2	df_log	[LinearRegression()]	8.104379e+08	9.095102e+08	4.432094e+09	5.123286e+09	0.078884	0.098576
3	df_original	[Ridge()]	2.816667e+08	3.431515e+08	7.177380e+08	1.566276e+09	0.975844	0.915750
4	df_MinMax	[Ridge()]	3.436276e+08	4.282319e+08	2.299322e+09	3.152693e+09	0.752089	0.658653
5	df_log	[Ridge()]	8.080235e+08	9.079830e+08	4.432133e+09	5.123834e+09	0.078868	0.098383
6	df_original	[Lasso()]	2.817792e+08	3.431821e+08	7.177482e+08	1.566357e+09	0.975843	0.915741
7	df_MinMax	[Lasso()]	2.817792e+08	3.431820e+08	7.177482e+08	1.566357e+09	0.975843	0.915741
8	df_log	[Lasso()]	8.104379e+08	9.095101e+08	4.432094e+09	5.123286e+09	0.078884	0.098576
9	df_original	[ElasticNet()]	2.781357e+08	3.363417e+08	7.284709e+08	1.559539e+09	0.975116	0.916473
10	df_MinMax	[ElasticNet()]	3.167394e+08	4.377077e+08	4.590456e+09	5.370983e+09	0.011884	0.009306
11	df_log	[ElasticNet()]	5.628429e+08	6.833164e+08	4.467609e+09	5.245745e+09	0.064063	0.054968
12	df_original	[RandomForestRegressor()]	5.274702e+07	2.512679e+08	1.541059e+09	4.835928e+09	0.830121	0.196859
13	df_MinMax	[RandomForestRegressor()]	5.201617e+07	2.471127e+08	1.533677e+09	4.804878e+09	0.860340	0.207139
14	df_log	[RandomForestRegressor()]	4.935219e+07	1.287593e+08	1.545915e+09	1.571449e+09	0.825057	0.915193

Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

### 5.3. Análise dos resíduos

Passa-se à análise dos resíduos, que são a diferença entre o valor da arrecadação real para 2018 e os valores preditos pelos modelos.

#### - Resíduos gerados pelo melhor modelo (*RandomForestRegressor* e *df\_log*)

Para se avaliar o erro, primeiro será reaplicado o melhor algoritmo (*RandomForestRegressor*) ao *dataframe df\_log* integral (não mais dividido entre base de treinamento e de teste).

Será criada uma variável ('predicao\_melhor\_modelo') com os valores previstos apontados pelo modelo e uma variável ('residuos') com a diferença entre o valor original e o valor previsto.

```
In [27]: # Isolar as features e o label
Y=df_log['01_arrecadacao_2018']
X=df_log.drop(['municipio-uf','01_arrecadacao_2018'],axis=1)

#reaplicar o modelo com os melhores hiperparâmetros
modelo = RandomForestRegressor().fit(X,Y)
y_pred = modelo.predict(X)

#cria coluna com as previsões
df_log['predicao_melhor_modelo'] = y_pred

#cria coluna auxiliar
df_log['UF']=df_log['municipio-uf'][-2:]

#df_experimento['faixa_arrecadacao']=df_transformado['faixa_arrecadacao']
df_log['faixa_arrecadacao']=df_integral['faixa_arrecadacao'].astype(str)

#df_experimento['faixa_arrecadacao'] = df_experimento.apply(lambda row:faixa_arrecadacao(row), axis = 1)

# Calculando o RMSE-total (base logarítmica)
mae_total = mean_absolute_error(Y,y_pred)
print("MAE-total", mae_total)

# Calcular os resíduos
df_log['residuos'] =abs(df_log['01_arrecadacao_2018'] - df_log['predicao_melhor_modelo'])

MAE-total 42892364.82286113
```

Figura. Vide Apêndice IX– Ntblk 04. Criação dos modelos de machine learning-Regressão.ipynb

Vê-se que o MAE do modelo aplicado à base completa foi de **R\$ 41.443.789,69**.

Para permitir uma melhor visualização dos resultados, as variáveis '01\_arrecadacao\_2018', 'residuos' e 'predicao\_melhor\_modelo' serão convertidas para a base logarítmica natural.

Feita essa conversão, o gráfico abaixo mostra a distribuição dos resíduos considerando uma curva ascendente da arrecadação real (variável 01\_arrecadacao\_2018) e o histograma dos resíduos.

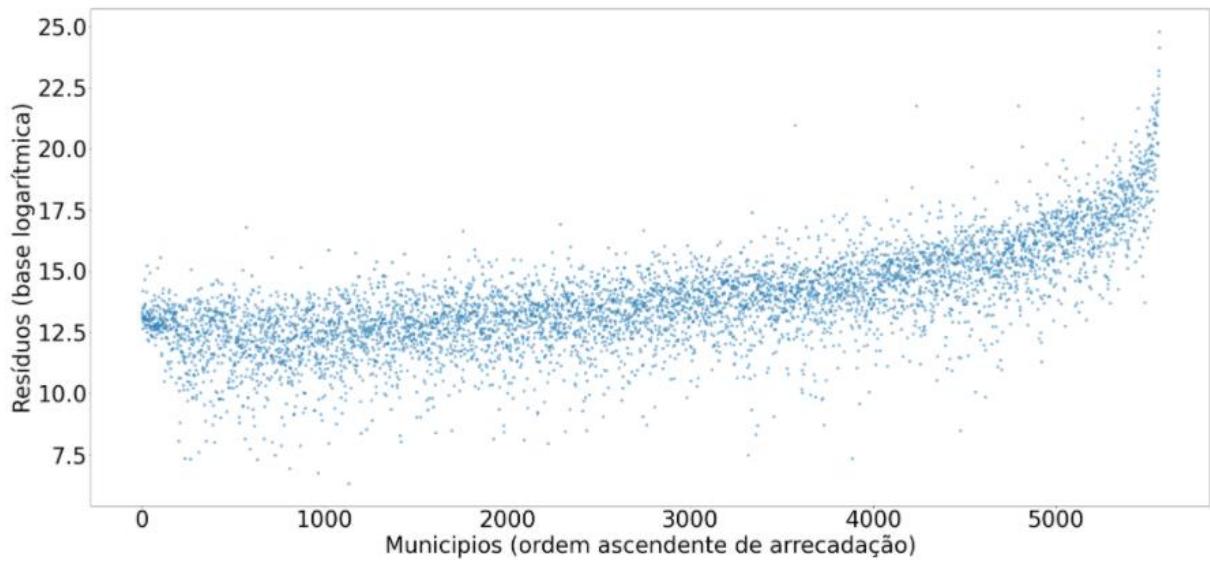


Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning–Regressão.ipynb

Abaixo, seguem o *bloxpot* e o histograma dos resíduos.

```
In [31]: plotstats(df_log, 'residuos_ln')
print(df_log['residuos'].describe().apply(lambda x: '%.6f' % x))
print("\n",df_log['residuos_ln'].describe().apply(lambda x: '%.6f' % x))

count      5565.000000
mean     41443789.693370
std      928719272.901685
min      554.452200
25%    283028.850200
50%    826400.210000
75%   3116983.301500
max   58645034880.819336
Name: residuos, dtype: object

count      5565.000000
mean     13.841317
std      2.075045
min      6.317981
25%    12.553304
50%    13.624834
75%    14.952376
max    24.794769
Name: residuos_ln, dtype: object
```

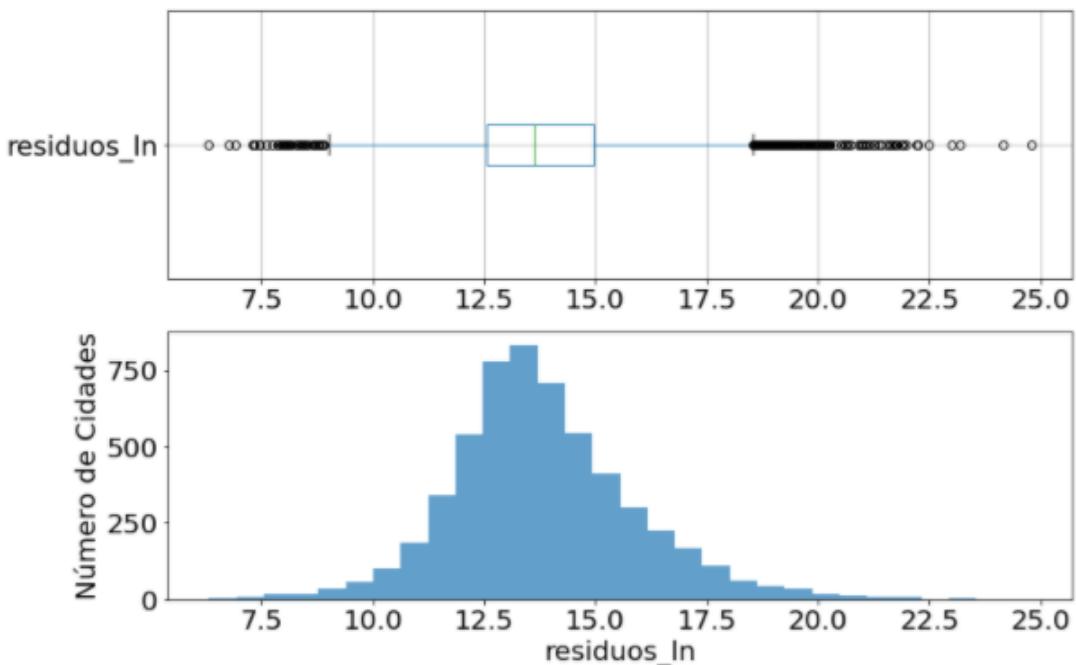


Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

A análise do *boxplot* e do *histograma* acima demonstram que a distribuição dos resíduos se aproxima da curva normal e se concentra praticamente no intervalo entre R\$ 22 mil e R\$ 270 mi (10 e 17.5, em valores logarítmicos).

O *boxplot* abaixo exibe os valores percentuais dos resíduos em relação ao valor da arrecadação real (o valor máximo para o eixo y foi limitado em 200% para ser possível visualizar os resultados).

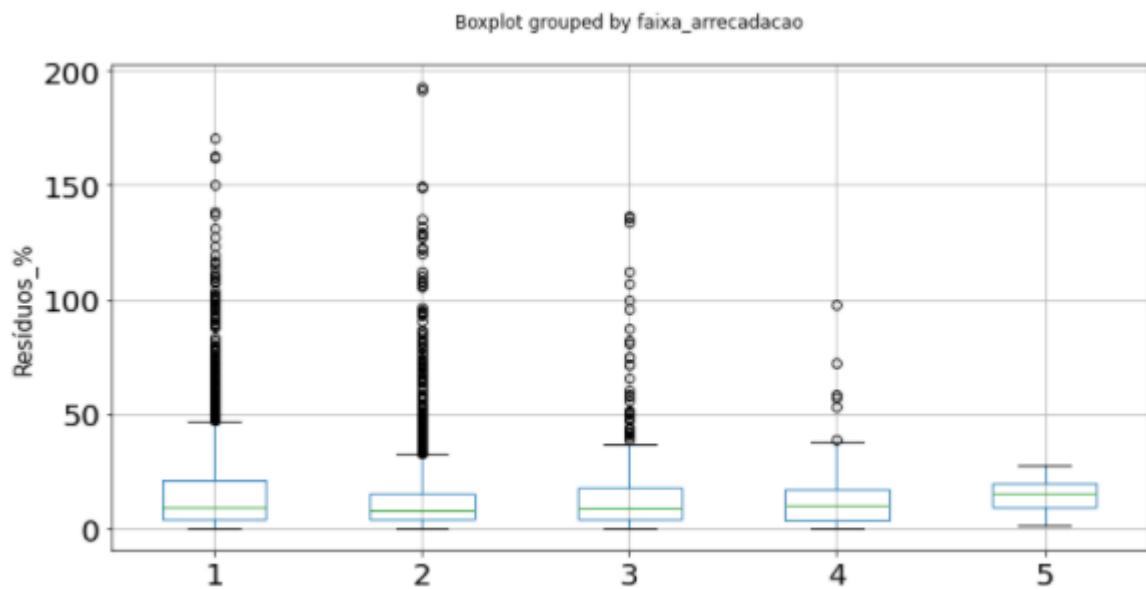


Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

Esses resíduos, embora não possam ser considerados desprezíveis, demonstram que o modelo teve boa eficácia, visto que o valor médio para os resíduos é de cerca de R\$ 41 mi (ou 13,83 em base logarítmica) e o valor médio das arrecadações é de cerca de R\$ 249 mi (vide capítulo 4.2), ou seja o erro médio representa cerca de, apenas, 16,44% do valor da arrecadação média.

Por fim, os Gráficos de Linhas abaixo permitem comparar a evolução da arrecadação (em base logarítmica) - linha azul - e a capacidade de predição do modelo - linha vermelha – para cada faixa de arrecadação.

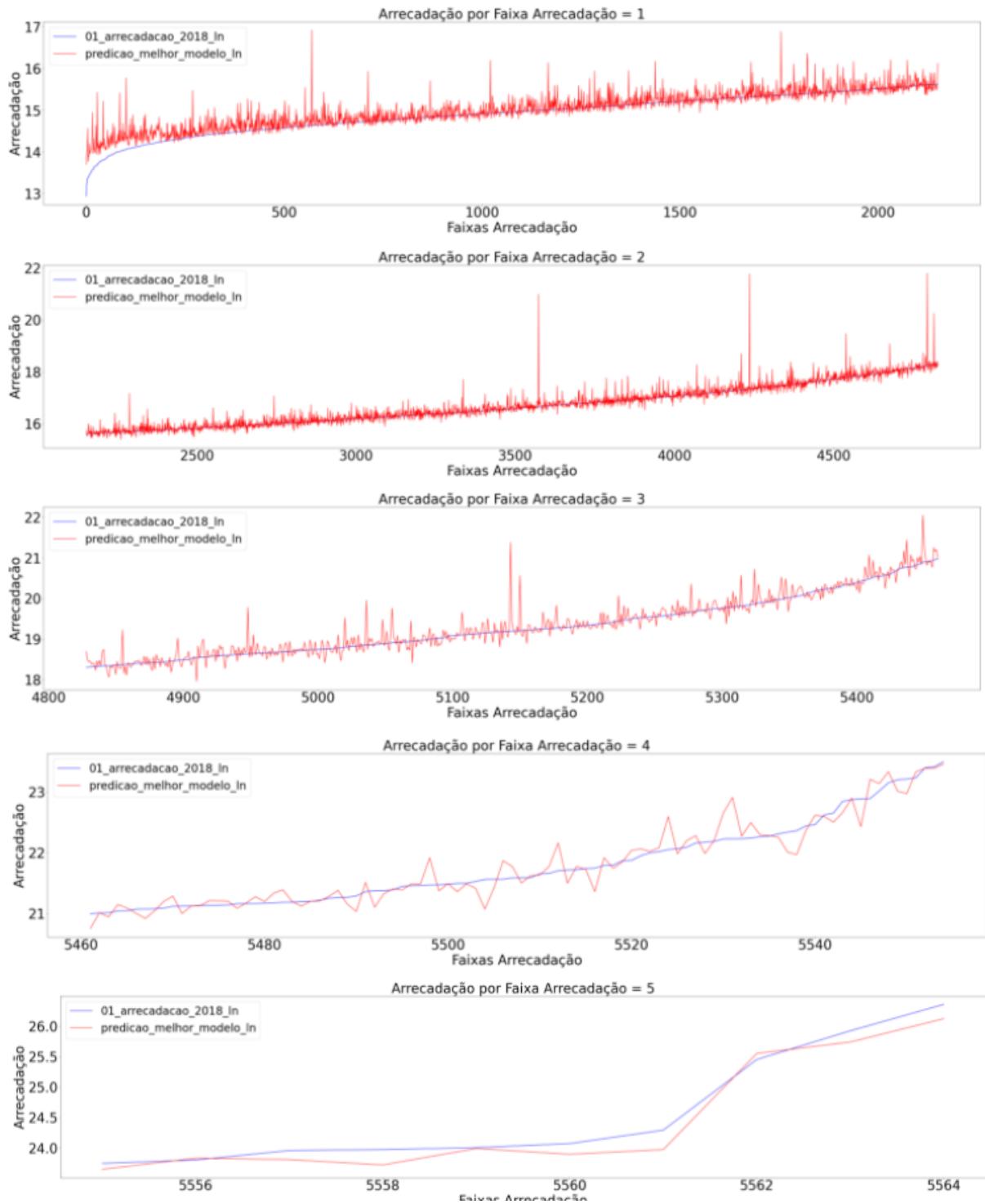


Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

Interessante observar que, não obstante haja variações, inclusive alguns picos relevantes, especialmente nas faixas 1 a 3, a linha vermelha acompanha a linha azul em sua tendência para a grande parte dos casos.

## - Comparação com o segundo melhor modelo (*ElasticNet* e *df\_original*)

O modelo com o segundo melhor resultado foi com o algoritmo *ElasticNet()*. Para a comparação dos resíduos, esse algoritmo será reaplicado ao *dataframe df\_original* (integralmente, não mais dividido entre base de treinamento e de teste), pois essa foi a melhor combinação (vide título *ElasticNet* do capítulo 5.2).

Vê-se que o MAE do modelo aplicado à base completa foi de **R\$ 303.356.514,04**.

```
In [35]: # Isolar as features e o Label
Y=df_original['01_arrecadacao_2018']
X=df_original.drop(['municipio-uf','01_arrecadacao_2018'],axis=1)

#reaplicar o modelo com os melhores hiperparâmetros
#modelo = ElasticNet(alpha= 1000.0, fit_intercept= False, l1_ratio= 0.99, max_iter = 10000).fit(X,Y)
modelo = ElasticNet().fit(X,Y)
y_pred = modelo.predict(X)

#cria coluna com as predições
df_original['predicao_melhor_modelo'] = y_pred

#cria coluna auxiliar
df_original['UF']=df_original['municipio-uf'][-2:]

#df_experimento['faixa_arrecadacao']=df_transformado['faixa_arrecadacao']
df_original['faixa_arrecadacao']=df_integral['faixa_arrecadacao'].astype(str)

#df_experimento['faixa_arrecadacao'] = df_experimento.apply(lambda row:faixa_arrecadacao(row), axis = 1)

# Calculando o RMSE-total (base Logarítmica)
mae_total = mean_absolute_error(Y,y_pred)
print("MAE-total", mae_total)

# Calcular os resíduos
#df_original['residuos'] = df_original['01_arrecadacao_2018'] - df_original['predicao_melhor_modelo']
df_original['residuos'] =abs(df_original['01_arrecadacao_2018'] - df_original['predicao_melhor_modelo'])

#Cria a coluna 'indice_ordenado'
df_original.sort_values(by='01_arrecadacao_2018', ascending=True, inplace = True)
df_original['indice_ordenado']=np.arange(0,df_original.shape[0])
df_original.sort_index(ascending=True, inplace = True)

colunas = ['01_arrecadacao_2018','residuos','predicao_melhor_modelo']
converte_para_ln(df_original,colunas)
df_original[['01_arrecadacao_2018_ln','residuos_ln','predicao_melhor_modelo_ln']]
```

MAE-total 303356514.0401894

Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

Visualizando os Gráficos de Linhas que compararam a evolução da arrecadação (em base logarítmica) – linha azul – e a capacidade de predição do modelo – linha vermelha, para cada faixa de arrecadação, vemos que diferentemente do resultado obtido para o modelo *RandomForestRegressor()*, as predições somente se aproximam dos valores reais de arrecadação para as faixas de valores maiores (faixas 3 a 5), apontando que o modelo penaliza mais severamente os maiores resíduos e assim busca reduzi-los ao máximo. Considerando que as faixas de valores menores, porém, possuem o maior número de municípios, isso ajuda a explicar o desempenho inferior deste modelo.

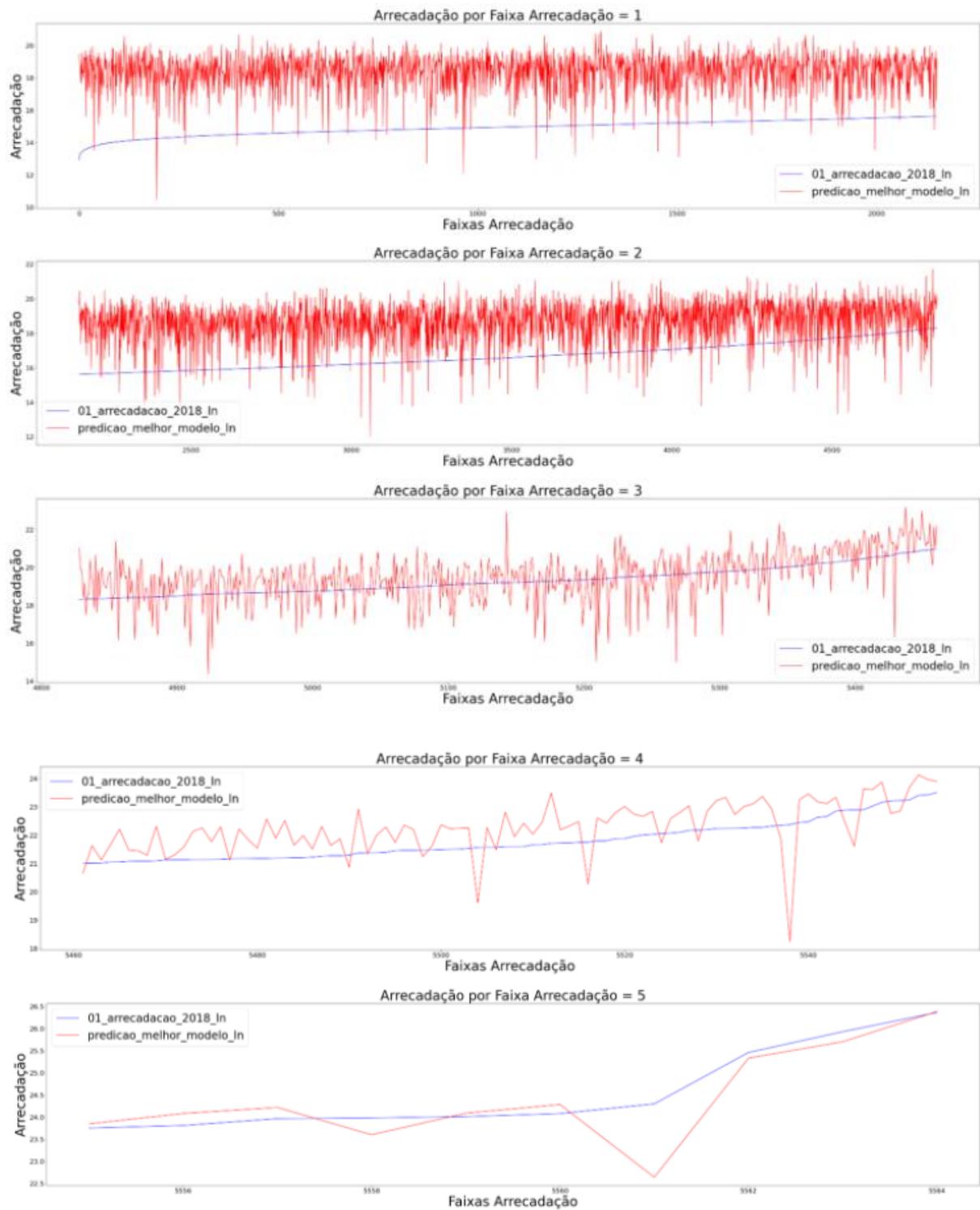


Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

Mas, o que ocorreria se as maiores arrecadações fossem excluídas deste modelo?

## - Resultado da exclusão das maiores arrecadações para o *ElasticNet*

Viu-se, no capítulo 4.2., que a distribuição das arrecadações (variável 01\_arrecadacao\_2018) indica a existência de 158 outliers: municípios com arrecadação superior a R\$ 757.532.350.87.

Foi mencionado que esses 158 municípios foram mantidos no modelo porque, embora suas arrecadações de fato destoem dos demais, não são incorretas e ajudam a caracterizar o universo sob análise, além de também ser de nosso interesse predizer a arrecadação para eles.

Também foi mencionado que esses valores, no entanto, influenciam os resultados do modelo, que precisa moldar-se ao conjunto de valores em análise. Será aqui avaliada, então, essa influência.

Ao reaplicar o algoritmo ElasticNet (que foi o segundo melhor modelo e para o qual a manutenção dos 158 municípios tiveram importante impacto) a uma nova base, sem as maiores arrecadações, tem-se que o MAE passa a ser de tão-somente **20.695.422,23** - uma redução extraordinária quando comparado com o modelo sobre base que mantinha os *outliers* (que fora de **303.356.514,04** – vide título imediatamente anterior) e melhor, inclusive, que o resultado alcançado com a aplicação do algoritmo RandomForestRegressor, calculado em **41.443.789,69** (vide título *Resíduos gerados pelo melhor modelo (RandomForestRegressor e df\_log)*, deste mesmo capítulo 5.3).

```
In [38]: # Isolar as features e o Label
Y=df_original['01_arrecadacao_2018']
X=df_original.drop(['municipio-uf','01_arrecadacao_2018'],axis=1)

#reaplicar o modelo com os melhores hiperparâmetros
modelo = ElasticNet().fit(X,Y)
y_pred = modelo.predict(X)

#cria coluna com as previsões
df_original['predicao_melhor_modelo'] = y_pred

#cria coluna auxiliar
df_original['UF']=df_original['municipio-uf'][-2:]

#df_experimento['faixa_arrecadacao']=df_transformado['faixa_arrecadacao']
df_original['faixa_arrecadacao']=df_integral['faixa_arrecadacao'].astype(str)

#df_experimento['faixa_arrecadacao'] = df_experimento.apply(lambda row:faixa_arrecadacao(row), axis = 1)

# Calculando o RMSE-total (base Logarítmica)
mae_total = mean_absolute_error(Y,y_pred)
print("MAE-total", mae_total)

# Calcular os resíduos
df_original['residuos'] =abs(df_original['01_arrecadacao_2018'] - df_original['predicao_melhor_modelo'])

MAE-total 20695422.2336082
```

Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

É visualizando e comparando, porém, os gráficos de linha dos resultados da base sem os *outliers* com o da base completa que uma diferença relevante se

apresenta. Os valores preditos (linhas vermelhas) agora sobrepõem-se aos da arrecadação real (linha azul) já nas primeiras faixas de arrecadação, quando no gráfico da base completa (visto acima) essa sobreposição somente ocorria para as faixas de arrecadação mais elevadas. Observação: não há elementos nos últimos dois gráficos visto que não há mais elementos nas faixas 4 e 5 após a exclusão dos *outliers*.

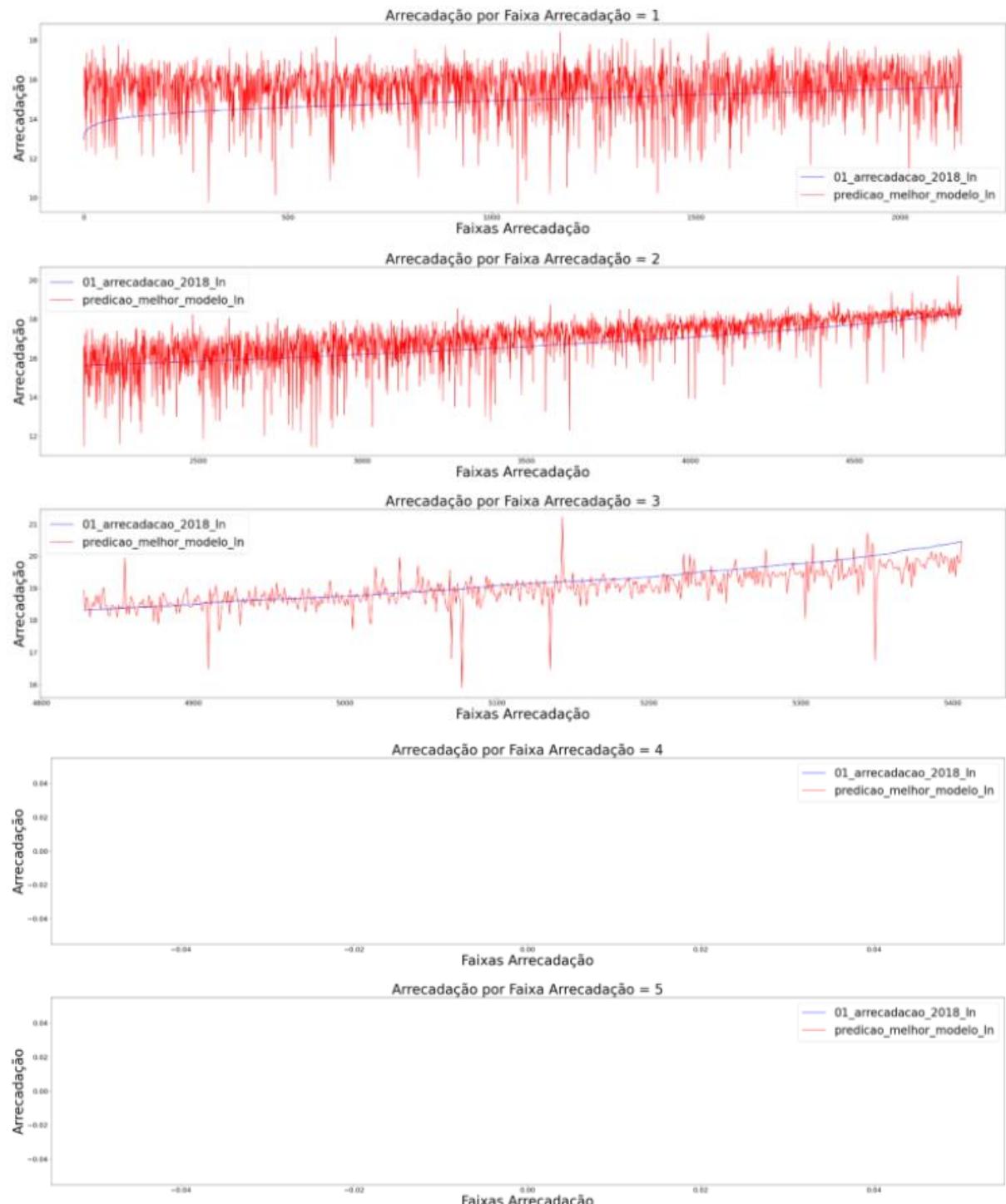


Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

Ou seja, com a exclusão das 158 maiores arrecadações, há uma sensível melhora na capacidade de predição das arrecadações de menor valor, pelo algoritmo ElasticNet.

Diante desse resultado, uma estratégia que considere a divisão do *dataset* entre municípios com altíssima arrecadação e os demais, merece ser considerada para uma predição mais precisa para esses dois estratos.

#### 5.4. Otimização de hiperparâmetros e *cross validation*

Avaliada a melhor combinação entre os *dataframes* de estudo e os algoritmos, passa-se, agora, a buscar o aprimoramento do resultado por meio da otimização dos hiperparâmetros e do uso da *cross validation*.

Este estudo seguirá, então, tendo por base a combinação de melhor resultado (algoritmo *RandomForestRegressor* e *dataframe df\_log*) e considerando o *dataset* completo (sem a exclusão dos municípios com maiores arrecadações).

No aprendizado de máquina, a otimização ou ajuste de hiperparâmetro é a ação de escolher um conjunto de configurações ideais para um algoritmo de aprendizado. Cada algoritmo possui um conjunto de parâmetros que podem ser definidos de modo personalizado com vistas a otimizar sua performance ante à base de dados em estudo.

No próximo experimento, será buscada a combinação ideal considerando os seis hiperparâmetros abaixo.

```
#Número de árvores da Random Forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 100)]

# Número de variáveis a considerar a cada split
max_features = ['auto', 'sqrt']

# Número máximo de níveis na árvore
max_depth = [int(x) for x in np.linspace(10, 110, num = 20)]
max_depth.append(None)

# Número mínimo de exemplos para fazer o split do nó
min_samples_split = [2, 5, 10]

# Número mínimo de exemplos para cada nó-folha
min_samples_leaf = [1, 2, 4]

# Método de seleção de exemplos para treinar cada árvore
bootstrap = [True, False]
```

Figura. Vide Apêndice IX– Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

Para tanto, será utilizada a classe *Random Search Cross Validation*, da biblioteca Scikit Learn: *sklearn.model\_selection.RandomizedSearchCV()*.

Esse algoritmo promove avaliações a partir de combinações aleatórias de valores para os hiperparâmetros indicados e retorna a melhor delas.

Para otimizar a identificação dos melhores hiperparâmetros (como adiantado no título *Separação dos datasets em base de treinamento e testes*, do capítulo 5.1) em vez de se fazer uma única separação, fixa, do dataset em apenas duas bases (treinamento e teste), o RandomizedSearchCV utiliza a técnica *cross validation*.

```
In [24]: # Busca pelos melhores hiperparâmetros usando RandomizedSearchCV

# define o modelo
modelo = RandomForestRegressor(random_state=12)

# Número de árvores da Random Forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 100)]

# Número de variáveis a considerar a cada split
max_features = ['auto', 'sqrt']

# Número máximo de níveis na árvore
max_depth = [int(x) for x in np.linspace(10, 110, num = 20)]
max_depth.append(None)

# Número mínimo de exemplos para fazer o split do nó
min_samples_split = [2, 5, 10]

# Número mínimo de exemplos para cada nó-folha
min_samples_leaf = [1, 2, 4]

# Método de seleção de exemplos para treinar cada árvore
bootstrap = [True, False]

# Cria o random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}

pprint(random_grid)

# Procura randômica por parâmetros, usando cross validation com 3 folds.
# Busca em 20 diferentes combinações, e use todas as variáveis independentes
modelo_randomizado = RandomizedSearchCV(estimator = modelo, error_score=mean_absolute_error, param_distributions = random_grid, n_iter = 20, cv = 3, verbose = 1, random_state = 42, n_jobs = -1)

# Roda o modelo de busca randômica.
x_train = x_log_train
y_train = y_log_train
modelo_randomizado.fit(x_train, y_train)

print('MAE: %.3f' % modelo_randomizado.best_score_)
print('Config: %s' % modelo_randomizado.best_params_)
```

Executada a célula acima (cerca de 17 minutos em um computador pessoal), a melhor combinação de valores identificada para os hyperparâmetros indicados foi:

```
Config:{'n_estimators': 563,
        'min_samples_split': 2,
        'min_samples_leaf': 1,
        'max_features': 'auto',
        'max_depth': 88,
        'bootstrap': False}
```

Será, então, reaplicado o algoritmo RandomForestRegressor (com os hiperparâmetros apontados) sobre o *dataframe* df\_log integral (sem a separação entre base de treinamento e de teste) de modo a comparar os resultados com o do experimento descrito no título *Resíduos gerados pelo melhor modelo*, deste mesmo capítulo.

```
In [33]: modelo = RandomForestRegressor(n_estimators= 563, min_samples_split= 2, min_samples_leaf= 1,
                                         max_features= 'auto', max_depth= 88, bootstrap = False)

# Isolar as features e o label
Y=df_log['01_arrecadacao_2018']
X=df_log.drop(['municipio-uf','01_arrecadacao_2018'],axis=1)

#reaplicar o modelo com os melhores hiperparâmetros
modelo = RandomForestRegressor(n_estimators= 654, min_samples_split= 5, min_samples_leaf= 1,
                                         max_features= 'auto', max_depth= 88, bootstrap = False).fit(X,Y)
y_pred = modelo.predict(X)

#cria coluna com as predições
df_log['predicao_melhor_modelo'] = y_pred

#cria coluna auxiliar
df_log['UF']=df_log['municipio-uf'][-2:]

#df_experimento['faixa_arrecadacao']=df_transformado['faixa_arrecadacao']
df_log['faixa_arrecadacao']=df_integral['faixa_arrecadacao'].astype(str)

#df_experimento['faixa_arrecadacao'] = df_experimento.apply(lambda row:faixa_arrecadacao(row), axis = 1)

# Calculando o RMSE-total (base logarítmica)
mae_total = mean_absolute_error(Y,y_pred)
print("MAE-total", mae_total)

# Calcular os resíduos
df_log['residuos'] = abs(df_log['01_arrecadacao_2018'] - df_log['predicao_melhor_modelo'])

MAE-total 41072249.73690276
```

Vê-se, assim, que promover o ajuste dos hiperparâmetros implicou em uma discreta melhora em relação ao modelo que não teve os hiperparâmetros definidos:

- MAE do modelo sem definição de hiperparâmetros: R\$ 41.443.789,69;
- MAE do modelo com ajuste dos hiperparâmetros: R\$ 41.072.249,73

## 6. Apresentação dos Resultados

De maneira objetiva, apresentam-se abaixo as conclusões que se extraem em relação aos dois questionamentos que nortearam este estudo.

**i. Foi possível constatar correlação entre a arrecadação de tributos federais por municípios e fatores sociais, econômicos, demográficos e geopolíticos a eles relacionados.**

A existência dessas associações foi inicialmente identificada no capítulo 4.3. *Análise visual das variáveis numéricas* quando se observou comportamentos correspondentes, por exemplo, entre as variáveis

161602\_salario\_medio\_mensal\_em\_Reais e o *label*, ou entre 03\_vlr\_adic\_bruto\_servicos e o *label*, ou entre 03\_vlr\_adic\_bruto\_administracao e o *label*, evidenciados pelo fato de que os valores dessas variáveis independentes aumentavam (se positivamente correlacionadas) ou diminuíam (se negativamente correlacionadas) à medida em que avançavam as faixas de arrecadação.

Essas correlações foram posteriormente comprovadas no capítulo 4.4. *Análise das correlações e tratamento de multicolinearidade*, quando numericamente calculadas.

É interessante notar que entre as 20 variáveis com maiores correlações apuradas em relação ao *label* há exemplos para cada um dos fatores de interesse mencionados:

- há duas variáveis de cunho geopolítico: 03\_uf\_DF e 03\_uf\_RJ;
- outras seis variáveis são de cunho econômico: 03\_pib, 03\_vlr\_adic\_bruto\_servicos\_%, 03\_atividade\_principal\_segunda\_Comércio e reparação de veículos automotores e motocicletas, 03\_atividade\_principal\_primeira\_Demais serviços, 03\_vlr\_adic\_bruto\_agropecuaria\_% e 03\_vlr\_adic\_bruto\_administracao\_%;
- são oito as de caráter socioeconômicos: 07\_20SM+\_%, 07\_Ate\_20SM\_%, 02\_salario\_medio\_mensal\_em\_Reais, 07\_Ate\_10SM\_%, 07\_Ate\_1SM\_%, 07\_Ate\_5SM\_%, 03\_pib\_per\_capita e 07\_Ate\_3SM\_%;
- encontram-se três relacionadas a fatores sociais: 06\_Superior\_Completo\_%, 06\_Sem\_Instrução\_e\_Fundamental\_Incompleto% e 06\_Superior\_Incompleto\_%; e
- uma última voltada a aspecto demográfico: 05\_20-39\_anos\_%.

Embora não seja possível afirmar que esses fatores e variáveis sejam causa direta do montante arrecadado no âmbito dos municípios, restou inegável a existência de associações entre eles, de modo que, conhecendo um, é possível apontar tendências para o outro.

ii. E, sim, diante da existência de correlação entre a variável dependente (01\_arrecadacao\_2018) e as variáveis ligadas aos fatores acima enumerados **foi possível prever, com um erro médio absoluto (MAE) de R\$ 41.072.249,73** (calculado sobre a base integral e após ajustes de hiperparâmetros— vide capítulo 5.4.), a

**arrecadação para cada um dos municípios brasileiros tão-somente a partir de dados públicos relacionados àqueles fatores.**

Esse resultado pode ser considerado promissor quando comparado com nossa baseline (calculada em mais de R\$ 504,69 mi, vide capítulo 5.1.). Além disso, tendo em vista que o **valor médio para os resíduos é de cerca de R\$ 41 mi e o valor médio das arrecadações é de cerca de R\$ 249 mi** (vide capítulo 4.2), **esse MAE representa cerca de, apenas, 16,44% do valor da arrecadação média.**

Para a identificação da melhor opção foram implementados, inicialmente, 15 modelos que combinavam os algoritmos RandomForestRegressor, LinearRegression, Lasso, Ridge e ElasticNet com os três *dataframes* de estudo criados (*df\_original*, *df\_log* e *df\_MinMax*). Esses modelos foram, então, inicialmente aplicados à base de treinamento (que compreendia 70% das ocorrências) e avaliados a partir da base de testes (os 30% restantes). Os resultados de cada um dos experimentos podem ser consultados no título *Melhor modelo identificado para este primeiro experimento* do capítulo 5.2.

Constatado que o melhor resultado foi o obtido com a combinação entre o algoritmo RandomForestRegressor e o *dataframe* de estudo *df\_log* (no qual as variáveis independentes foram convertidas para a base logarítmica – vide título *Criação dos dataframes de estudo* do capítulo 5.1), esse modelo foi então aplicado à base integral (não mais separada em base de treinamento e testes) o que resultou em um MAE de R\$ 41.443.789,69.

Ao compararmos os resíduos gerados pelo melhor modelo (algoritmo *RandomForestRegressor* sobre o *dataframe* *df\_log*) com aqueles gerados pelo segundo melhor modelo (algoritmo *ElasticNet* sobre o *dataframe* *df\_original*), percebe-se claramente a vantagem do primeiro que acompanha, de maneira bastante próxima, a tendência da arrecadação geral, quando, no que concerne ao segundo, as previsões somente se aproximam dos valores reais de arrecadação para as faixas de valores maiores (faixas 3 a 5). Há, porém, uma significativa melhora na capacidade de predição do algoritmo *ElasticNet* quando são excluídas as 158 maiores arrecadações – vide título *Análise dos outliers*, do capítulo 4.2., e o contido no capítulo 5.3.

Em busca de aprimorar o resultado, promoveu-se à calibragem dos hiperparâmetros do algoritmo *RandomForestRegressor()* – vide capítulo 5.4. Os ajustes nos hiperparâmetros implicaram em uma discreta melhora em relação ao modelo anterior, para o qual foram mantidos os hiperparâmetros padrão do algoritmo. O MAE após a otimização dos hiperparâmetros passou a ser R\$ 41.072.249,73.

Dadas as constatações acima e considerando que a arrecadação no âmbito dos municípios nada mais é do que o conjunto de arrecadações individuais de seus moradores e das unidades empresariais/industriais/públicas neles instaladas, o presente estudo permite apontar, de maneira especial, duas importantes linhas para futuras investigação pelas Administrações Tributárias, no seu mister de garantir a conformidade individual para o alcance da arrecadação global:

- aprofundar o exame das razões pelas quais as arrecadações previstas foram significativamente distintas das arrecadações reais para alguns municípios, para compreender quais fatores específicos conduziram a essa discrepância;
- voltar olhos à conformidade individual: provado haver correlação entre as variáveis sob estudo e a arrecadação para o conjunto de moradores/unidades empresariais, examinar se essas mesmas variáveis mantêm a correspondência com a arrecadação individual e, à medida do possível, acrescentar variáveis não disponíveis publicamente em razão do sigilo fiscal.

## 7. Links

Para visualizar o vídeo com uma apresentação geral sobre o projeto acesse:  
<https://youtu.be/lvos6ISrnAM>.

As bases de dados utilizadas no projeto e os jupyter notebooks criados podem ser consultados em: <https://github.com/ALT-Andre/TCC-PUCMG>

## REFERÊNCIAS

BANCO MUNDIAL. Risk-based tax audits: approaches and country experiences. Washington, DC: The Banco Mundial, 2011.

BRASIL. Secretaria da Receita Federal do Brasil. Coordenação-Geral de Auditoria Interna e Gestão de Riscos (RFB/AUDIT). Guia teórico-metodológico: gestão de riscos de conformidade tributária e aduaneira. Brasília/DF – 2016. (RFB/AUDIT – Guia)

BRASIL. Secretaria da Receita Federal do Brasil. Coordenação-Geral de Auditoria Interna e Gestão de Riscos (RFB/AUDIT). Estudo sobre a situação atual da RFB em relação à gestão de riscos de conformidade tributária e aduaneira. Brasília/DF – 2016. (RFB/AUDIT – Estudo)

INSTITUTO BRASILEIRO DE GOVERNANÇA CORPORATIVA (IBGC). Guia de Orientação para Gerenciamento de Riscos Corporativos. São Paulo, SP – 2007.

ORGANIZAÇÃO PARA A COOPERAÇÃO E DESENVOLVIMENTO ECONÔMICO (OCDE). Compliance risk management: managing and improving tax compliance. Paris: OCDE, Centre for Tax Policy and Administration, 2004.

TRAPP, Adriana Cristina Garcia e CORRAR, Luiz J. Avaliação e gerenciamento do risco operacional no Brasil: análise de caso de uma instituição financeira de grande porte. Rev. contab. finanç. vol.16 no.37 São Paulo Jan./Apr. 2005.

SECRETARIADO DO TADAT. Tax Administration Diagnostic Assessment Tool: field guide. Washington, DC: SECRETARIADO DO TADAT, 2019. Disponível em: <<https://www.tadat.org/fieldGuide>>. Acesso em: 31 mar 2021

UNIÃO EUROPEIA. European Commission. Compliance risk management guide for tax administrations. Brussels: EC, Fiscais Risk Management Platform Group, 2010.

## APÊNDICES

**Apêndice I – Resumo do projeto (Canvas)**

**Apêndice II – Detalhes adicionais para a coleta dos *datasets* originais**

**Apêndice III – Resumo das transformações nos *datasets* originais**

**Apêndice IV – Matriz de correlações**

**Apêndice V – Correlações entre as variáveis independentes e o *label***

**Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb**

**Apêndice VII – Ntbk 02. Criação do *dataset* consolidado - merges e saneamentos.ipynb**

**Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb**

**Apêndice IX – Ntbk 04. Criação dos modelos de machine learning-Rregressão.ipynb**

## Apêndice I – Resumo do projeto (Canvas)

<b>Título:</b> PREDIÇÃO DA ARRECADAÇÃO DE TRIBUTOS FEDERAIS, POR MUNICÍPIO BRASILEIRO, A PARTIR DE INFORMAÇÕES SOCIAIS, ECONÔMICAS, DEMOGRÁFICAS E GEOPOLÍTICAS.		
<b>1. Problema proposto:</b> Que problemas se está tentando resolver? Quais os principais pontos a serem tratados?  Avaliar a capacidade de predição da arrecadação de tributos federais, por municípios brasileiros, a partir de dados públicos, divulgados pela Receita Federal do Brasil – RFB e pelo Instituto Brasileiro de Geografia e Estatística (IBGE), relacionados a aspectos sociais, econômicos, demográficos e geopolíticos.	<b>2. Resultados/Previsões:</b> Que previsões você está tentando fazer? Identifique as variáveis preditoras a variável alvo.  - Há correlação entre fatores sociais, econômicos, demográficos e geopolíticos, e a arrecadação de tributos federais agregada por município?  - É possível prever a arrecadação para cada um dos municípios brasileiros tão-somente a partir de dados públicos relacionados a esses fatores?	<b>3. Coleta de dados:</b> Quais são as fontes para seus dados? Há dados suficientes? Você pode trabalhar os dados?  As informações utilizadas neste projeto foram coletadas dos sites do Governo Federal Brasileiro ( <a href="http://www.gov.br">www.gov.br</a> ) e do Instituto Brasileiro de Geografia e Estatística – IBGE ( <a href="http://www.ibge.gov.br">www.ibge.gov.br</a> ).
<b>4. Preparação dos dados:</b> O que você precisa fazer em seus dados de modo a aplicar seus modelos e obter os resultados?  - Descompactação de arquivos; - Visualização, análise e importação das bases; - Criação de variável padronizada: município-uf; - Manutenção, apenas, de informações relativas ao ano de interesse: 2018; - Renomeação de variáveis; - Enriquecimento de dados (criação de novas variáveis derivadas); - Remoção de variáveis redundantes, com valores nulos ou fixos; - Transformação de variáveis numéricas; - Transformação de variáveis categóricas; - Tratamento de multicolinearidades; - Integração das bases e saneamento de divergências.	<b>5. Criando o modelo:</b> Que modelos são mais apropriados diante dos resultados esperados?  Foram implementados 15 modelos de aprendizado supervisionado de máquina – Regressão que combinavam os algoritmos RandomForestRegressor, LinearRegression, Lasso, Ridge e ElasticNet com os três <i>dataframes</i> de estudo criados (df_original, df_log e df_MinMax).  Na sequência, promoveu-se à análise dos resíduos e à otimização dos hiperparâmetros.	<b>6. Avaliação do modelo:</b> Como você pode avaliar a performance do seu modelo?  Foram evidenciadas correlações entre a arrecadação por município e os fatores mencionados no quadro 2, acima. A previsão da arrecadação a partir deles também se mostrou eficaz.  A qualidade do modelo foi aferida a partir da métrica MAE (Mean Absolute Error).  O resultado obtido pelo melhor modelo foi um MAE de R\$ 41.006.322,99 (para o algoritmo RandomForestRegressor aplicado sobre o dataframe df_log – no qual as variáveis independentes foram convertidas para a base logarítmica natural). Esse valor pode ser considerado promissor, se considerada uma <i>baseline</i> calculada em R\$ 504,6 mi (a partir da média da variável dependente) e por representar, apenas, cerca de 16,44% da média da variável dependente.

## Apêndice II – Detalhes adicionais para a coleta dos datasets originais.

### 1. Arrecadação das Receitas Administradas pela RFB, por Município ([arrecadacao-da-receita-administrada-pela-rfb-por-municipio-2018.xlsx](https://www.gov.br/receitafederal/pt-br/acesso-a-informacao/dados-abertos/receitadata/arrecadacao/copy_of_arrecadacao-das-receitas-administradas-pela-rfb-por-municipio/arrecadacao-das-receitas-administradas-pela-rfb/arrecadacao-da-receita-administrada-pela-rfb-por-municipio-2018.xlsx))

- Detalhes para obtenção alternativa (não direta) dos dados:

Para esta base, nenhum detalhe adicional para obtenção dos dados é necessário, vez que o download do arquivo é feito diretamente a partir do link

[https://www.gov.br/receitafederal/pt-br/acesso-a-informacao/dados-abertos/receitadata/arrecadacao/copy\\_of\\_arrecadacao-das-receitas-administradas-pela-rfb-por-municipio/arrecadacao-das-receitas-administradas-pela-rfb/arrecadacao-da-receita-administrada-pela-rfb-por-municipio-2018.xlsx](https://www.gov.br/receitafederal/pt-br/acesso-a-informacao/dados-abertos/receitadata/arrecadacao/copy_of_arrecadacao-das-receitas-administradas-pela-rfb-por-municipio/arrecadacao-das-receitas-administradas-pela-rfb/arrecadacao-da-receita-administrada-pela-rfb-por-municipio-2018.xlsx).

O documento possui três planilhas, a de interesse é a denominada “Total” (que é a soma entre os valores pagos por meio de DARF e por GPS – que são tipos de guias para pagamento).

- Data da consulta: 19/3/2021.

### 2. Tabela 1685 - Unidades locais, empresas e outras organizações atuantes, pessoal ocupado total, pessoal ocupado assalariado, pessoal assalariado médio, salários e outras remunerações e salário médio mensal ([tabela1685.csv](https://sidra.ibge.gov.br/tabela/1685))

- Detalhes para obtenção alternativa (não direta) dos dados:

Acesse a página <https://sidra.ibge.gov.br/tabela/1685>.

The screenshot shows the SIDRA (Sistema IBGE de Recuperação Automática) interface. The URL in the browser is https://sidra.ibge.gov.br/tabela/1685. The main title is "Tabela 1685: Unidades locais, empresas e outras organizações atuantes, pessoal ocupado total, pessoal ocupado assalariado, pessoal assalariado médio, salários e outras remunerações e salário médio mensal". Below the title, there are two tabs: "Quadro" (selected) and "Cartograma". A message says "A seleção atual não possui erros." (The current selection does not have errors.). Under "Layout", it says "Layout: 8 tabelas [5.570 x 1] - 44.560 valores". A note says "Selecione e arraste uma dimensão para definir sua posição". A preview table shows the structure: "Unidades locais, empresas e outras organizações atuantes, pessoal ocupado total, pessoal ocupado assalariado, pessoal assalariado médio, salários e outras remunerações e salário médio mensal" (Variável (8)), "Ano (1)", and "Unidade Territorial (5570)". The bottom status bar shows the date 19/03/2021 and time 08:21.

No campo “Layout” ajuste a posição dos elementos da pesquisa de modo a estarem configurados na forma da figura abaixo.

CADASTRO CENTRAL DE EMPRESAS

Tabela 1685 - Unidades locais, empresas e outras organizações atuantes, pessoal ocupado total, pessoal ocupado assalariado, pessoal assalariado médio, salários e outras remunerações e salário médio mensal (Vide Notas)

Quadro Cartograma

Layout: 1 tabela [5.570 x 8] - 44.560 valores

Selecione e arraste uma dimensão para definir sua posição

Unidades locais, empresas e outras organizações atuantes, pessoal ocupado total, pessoal ocupado assalariado, pessoal assalariado médio, salários e outras remunerações e salário médio mensal		Variável (8)
Unidade Territorial (5570)	① Ano (1)	

Variável [8/8]

No campo “Variável”, selecionar todas as 8 (oito) opções; no campo “Ano”, selecionar “2018”; no campo “Unidade Territorial” selecionar “Município [5570/5570]”; clicar no botão “Download”;

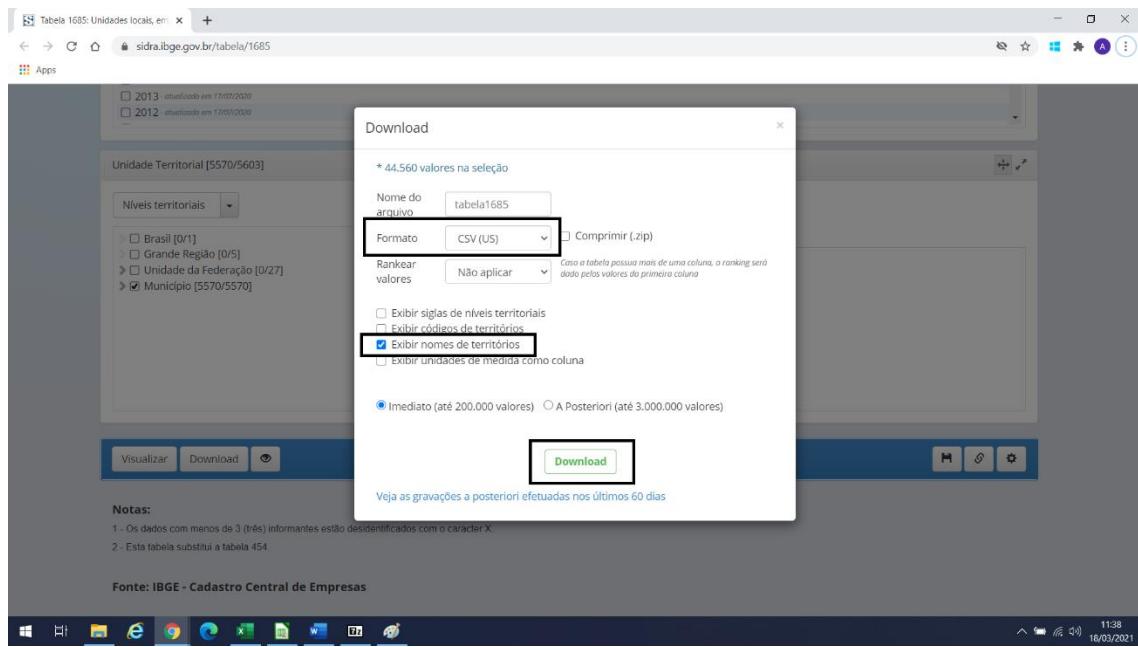
Variável (8/8)

Ano [1/1]

Unidade Territorial [5570/5570]

Visualizar Download

Na nova janela exibida: selecionar o formato “CSV (US)”; selecionar apenas a caixa de seleção “Exibir nomes de territórios”; e clicar em download.

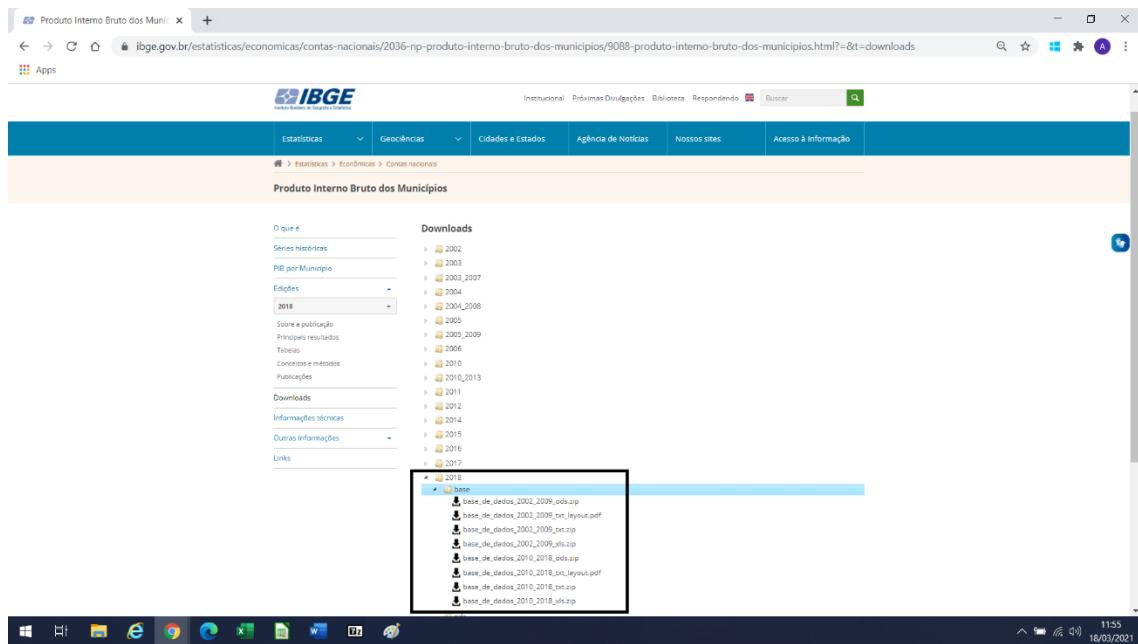


- Data da consulta: 19/3/2021;

### 3. Produto Interno Bruto dos Municípios (‘base\_de\_dados\_2010\_2018\_xls.zip’)

- Detalhes para obtenção alternativa (não direta) dos dados:

Aberta a página a partir do endereço acima: clicar sobre o ano “2018”; em seguida, clicar sobre “Base”; e, então, clicar sobre “base\_de\_dados\_2010\_2018.xls.zip”;



Após descompactar o arquivo baixado, será gerado o arquivo “PIB dos Municípios - base de dados 2010-2018.xls”.

- Data da consulta: 19/3/2021;

#### 4. Tabela 6579 - População residente estimada (*tabela6579.csv*)

- Detalhes para obtenção alternativa (não direta) dos dados:

Acesse a página <https://sidra.ibge.gov.br/tabela/6579>.

The screenshot shows the SIDRA (Sistema IBGE de Recuperação Automática) interface. The main title is 'ESTIMATIVAS DE POPULAÇÃO' and the specific table is 'Tabela 6579 - População residente estimada (Vide Notas)'. Below the title, there are two tabs: 'Quadro' (selected) and 'Cartograma'. A message says 'A seleção atual não possui erros.' (The current selection does not have errors.). The 'Layout' section shows a table structure with three columns: 'Variável (1)', 'Ano (1)', and 'Unidade Territorial (5570)'. At the bottom left, it says 'Variável [1/1]'. The bottom right corner of the screen shows the date and time: '19/03/2021 09:34'.

No campo “Layout” ajuste a posição dos elementos da pesquisa de modo a estarem configurados na forma da figura abaixo.

The screenshot shows the SIDRA (Sistema IBGE de Recuperação Automática) interface. At the top, there are navigation links for 'IBGE', 'BRASIL', 'Serviços', 'Simplifique!', 'Participe', 'Acesso à informação', 'Legislação', and 'Canais'. Below this is the SIDRA logo and a menu bar with 'PESQUISAS', 'ACERVO', 'TERRITÓRIO', 'CONTATO', 'AJUDA', and a search icon. The main content area is titled 'ESTIMATIVAS DE POPULAÇÃO' and 'Tabela 6579 - População residente estimada (Vide Notas)'. It displays two tabs: 'Quadro' (selected) and 'Cartograma'. A message says 'A seleção atual não possui erros.' Below this is a 'Layout' section showing '1 tabela [5.570 x 1] - 5.570 valores'. A tooltip says 'Selecione e arraste uma dimensão para definir sua posição'. A table structure is shown with columns: 'População residente estimada' (Variável), 'Unidade Territorial (5570)' (Ano), and 'Município [5570/5570]' (Variável). The bottom of the window shows a toolbar with icons for 'Visualizar', 'Download', and 'Imprimir'.

Aberta a página a partir do endereço acima: no campo "Variável", selecionar "População residente estimada (Pessoas)"; no campo "Ano", selecionar "2018"; no campo "Unidade Territorial" selecionar "Município [5570/5570]"; clicar no botão "Download";

This screenshot shows the detailed configuration of variables for the same table. The 'Variável' section has 'População residente estimada (Pessoas)' selected. The 'Ano' section has '2018' selected. The 'Unidade Territorial' section has 'Município [5570/5570]' selected. At the bottom, the 'Download' button is highlighted with a black box. The status bar at the bottom right shows the date as 18/03/2021 and the time as 13:38.

Na nova janela exibida: selecionar o formato "CSV (US)"; selecionar apenas a caixa de seleção "Exibir nomes de territórios"; e clicar em download.

- Data da consulta: 19/3/2021.

## 5. Tabela 200 - População residente, por sexo, situação e grupos de idade (*tabela200.csv*)

- Detalhes para obtenção alternativa (não direta) dos dados:

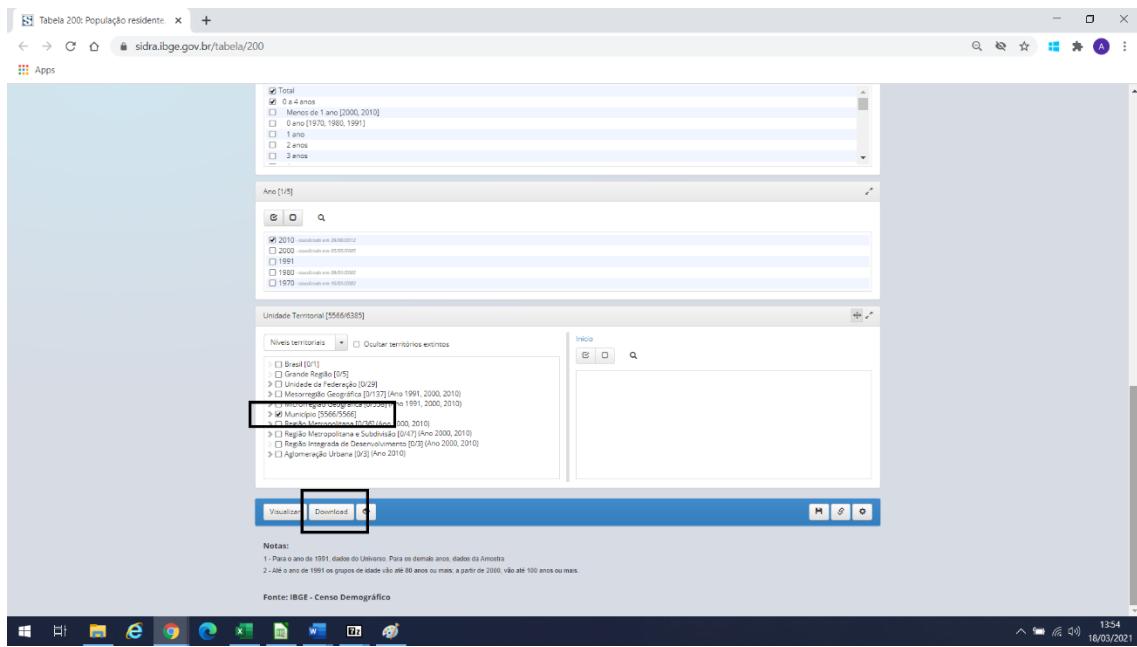
Acesse a página <https://sidra.ibge.gov.br/tabela/200>.

No campo “Layout” ajuste a posição dos elementos da pesquisa de modo a estarem configurados na forma da figura abaixo.

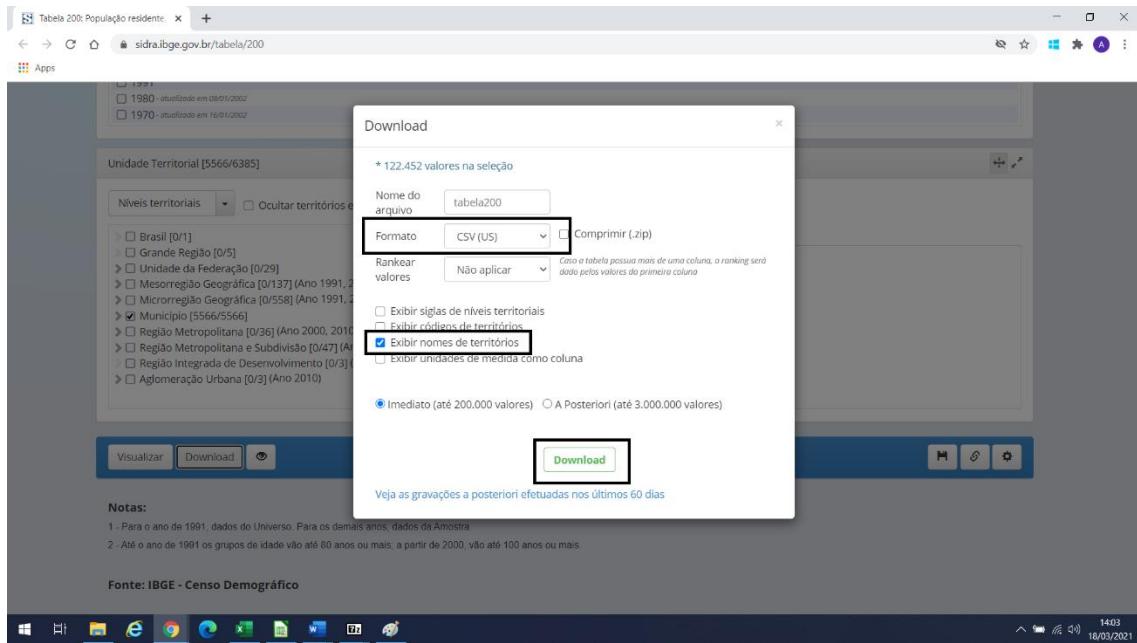
The screenshot shows the SIDRA (Sistema IBGE de Recuperação Automática) interface. At the top, there are navigation links for 'IBGE', 'SIDRA', 'Simplifique!', 'Participe', 'Acesso à informação', 'Legislação', and 'Canais'. Below this, the main title 'CENSO DEMOGRÁFICO' and the specific table 'Tabela 200 - População residente, por sexo, situação e grupos de idade - Amostra - Características Gerais da População (Vide Notas)' are displayed. A blue bar at the top of the table area contains 'Quadro' and 'Cartograma' buttons. The main content area shows a grid of variables: 'Variável (1)' (Population residente (Pessoas)), 'Unidade Territorial (5566)', 'Ano (1)', 'Situação do domicílio (1)', and 'Sexo (1)'. A message at the bottom left says 'A seleção atual não possui erros.' (The current selection does not have errors). The status bar at the bottom right shows the date '19/03/2021' and time '09:52'.

No campo "Variável", selecionar "População residente (Pessoas)"; no campo "Sexo", selecionar "Total"; no campo "Situação do domicílio", selecionar "Total"; no campo "Grupo de idade", selecionar "Total", "0 a 4 anos", "5 a 9 anos", "10 a 14 anos", "15 a 19 anos", "20 a 24 anos", "25 a 29 anos", "30 a 34 anos", "35 a 39 anos", "40 a 44 anos", "45 a 49 anos", "50 a 54 anos", "55 a 59 anos", "60 a 64 anos", "65 a 69 anos", "70 a 74 anos", "75 a 79 anos", "80 a 84 anos", "85 a 89 anos", "90 a 94 anos", "95 a 99 anos" e "100 anos ou mais"; no campo "Ano", selecionar "2010"; no campo "Unidade Territorial" selecionar "Município [5565/5565]"; clicar no botão "Download";

This screenshot shows the detailed variable selection interface for Table 200. It displays five panels: 'Variável (1/2)', 'Sexo (1/3)', 'Situação do domicílio (1/3)', 'Grupo de idade (22/50)', and 'Ano (1/3)'. The 'Variável' panel has 'População residente (Pessoas)' selected. The 'Sexo' panel has 'Total' selected. The 'Situação do domicílio' panel has 'Total' selected. The 'Grupo de idade' panel has 'Total' selected, with '0 a 4 anos' checked. The 'Ano' panel has '2010' selected. The status bar at the bottom right shows the date '16/03/2021' and time '13:53'.



Na nova janela exibida: selecionar o formato “CSV (US)”; selecionar apenas a caixa de seleção “Exibir nomes de territórios”; e clicar em download.



- Data da consulta: 19/3/2021.).

## 6. Tabela 1554 - Pessoas de 10 anos ou mais de idade, por nível de instrução (*tabela1554.csv*)

- Detalhes para obtenção alternativa (não direta) dos dados:

Acesse a página <https://sidra.ibge.gov.br/tabela/1554>.

The screenshot shows the SIDRA (Sistema IBGE de Recuperação Automática) interface. At the top, there's a navigation bar with links for Simplifique!, Participe, Acesso à Informação, Legislação, Canais, and Favoritos. Below the navigation bar, the main header reads "CENSO DEMOGRÁFICO" and "Tabela 1554 - Pessoas de 10 anos ou mais de idade, por nível de instrução - Resultados Gerais da Amostra (Vide Notas)". There are two tabs: "Quadro" (selected) and "Cartograma". A dropdown menu labeled "Quadros Públicos" is open. Below it, a message says "A seleção atual não possui erros." A "Layout" section indicates "Layout: 1 tabela [5.565 x 6] - 33.390 valores". A tooltip says "Selecione e arraste uma dimensão para definir sua posição". A table structure is shown: "Pessoas de 10 anos ou mais de idade, por nível de instrução - Resultados Gerais da Amostra" with columns "Variável (1)", "Ano (1)", and "Nível de instrução (6)". The first row contains "Unidade Territorial (5565)" and the second row contains "Ano (1)". The bottom status bar shows the date "19/03/2021" and time "09:56".

No campo “Layout” ajuste a posição dos elementos da pesquisa de modo a estarem configurados na forma da figura abaixo.

This screenshot shows the same SIDRA interface as the previous one, but with a different layout configuration. The table structure is now arranged as follows: "Pessoas de 10 anos ou mais de idade, por nível de instrução - Resultados Gerais da Amostra" with columns "Variável (1)", "Nível de instrução (6)", and "Ano (1)". The first row contains "Unidade Territorial (5565)" and the second row contains "Ano (1)". A tooltip at the bottom left says "Variável [1/2]". The status bar at the bottom right shows the date "19/03/2021" and time "09:57".

No campo “Variável”, selecionar a opção “Pessoas de 10 anos ou mais de idade”; no campo “Nível de Instrução”, selecionar todas as 6 (seis) opções ; no campo “Ano”, selecionar “2010”; no campo “Unidade Territorial” selecionar “Município [5565/5565]”.

The screenshot shows the search interface for Tabela 1554. The search parameters are set to 'Pessoas de 10 anos ou mais de idade (Pessoas)' and '2010'. The 'Download' button is highlighted.

Clicar no botão “Download” e, na nova janela exibida: selecionar o formato “CSV (US)”; selecionar apenas a caixa de seleção “Exibir nomes de territórios”; e clicar em download.

The screenshot shows the 'Download' dialog box for Tabela 1554. The 'Formato' is set to 'CSV (US)' and the 'Exibir nomes de territórios' checkbox is selected. The 'Download' button is highlighted.

- Data da consulta: 19/3/2021.

## 7. Tabela 2030 - Pessoas de 10 anos ou mais de idade, por classes de rendimento nominal mensal (*tabela2030.csv*)

- Detalhes para obtenção alternativa (não direta) dos dados:

Acesse a página <https://sidra.ibge.gov.br/tabela/2030>.

The screenshot shows the SIDRA (Sistema IBGE de Recuperação Automática) interface. At the top, there are links for 'IBGE', 'BRASIL', 'Serviços', 'Simplifique!', 'Participe', 'Acesso à informação', 'Legislação', and 'Canais'. Below this is the SIDRA logo and navigation links for 'PESQUISAS', 'ACERVO', 'TERRITÓRIO', 'CONTATO', and 'AJUDA'. A search bar is also present.

The main content area is titled 'CENSO DEMOGRÁFICO' and displays 'Tabela 2030 - Pessoas de 10 anos ou mais de idade, por classes de rendimento nominal mensal - Resultados Gerais da Amostra (Vide Notas)'. There are two tabs: 'Quadro' (selected) and 'Cartograma'. A message below says 'A seleção atual não possui erros.' (The current selection does not have errors.)

The 'Layout' section indicates 'Layout: 1 tabela [5.565 x 9] - 50.085 valores'. A note says 'Selecione e arraste uma dimensão para definir sua posição'. A table structure is shown:

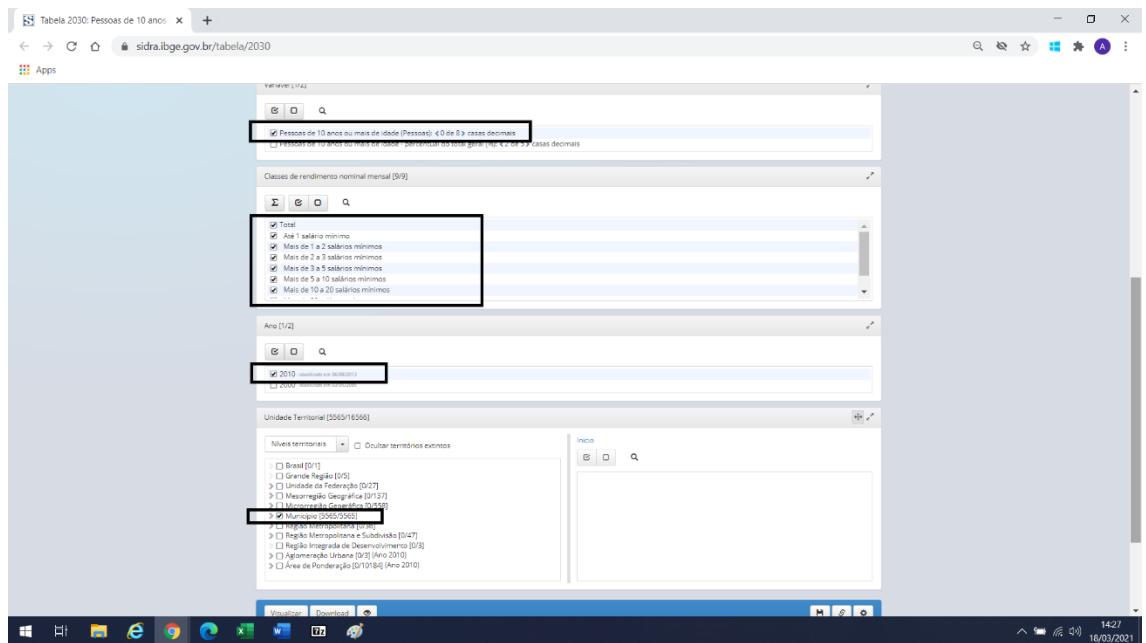
Pessoas de 10 anos ou mais de idade, por classes de rendimento nominal mensal - Resultados Gerais da Amostra	
Variável (1)	
Unidade Territorial (5565)	<input type="radio"/> Ano (1)
	Classes de rendimento nominal mensal (9)

No campo “Layout” ajuste a posição dos elementos da pesquisa de modo a estarem configurados na forma da figura abaixo.

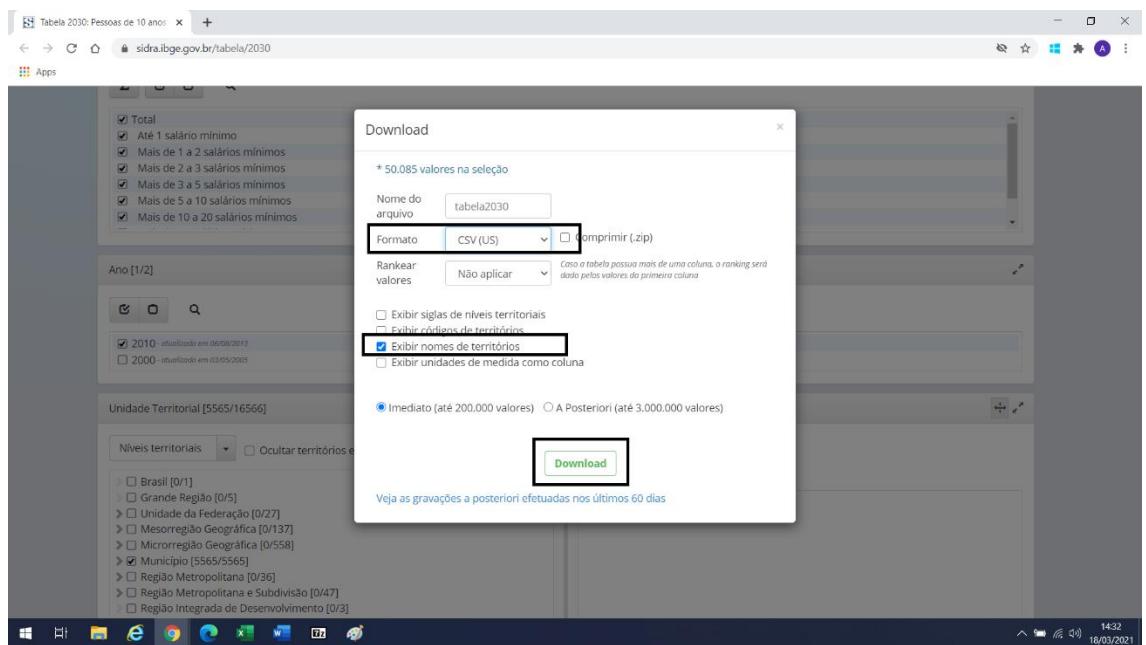
This screenshot shows the same SIDRA interface as the previous one, but with a different layout configuration for the table. The table structure is now:

Pessoas de 10 anos ou mais de idade, por classes de rendimento nominal mensal - Resultados Gerais da Amostra	
Variável (1)	
Unidade Territorial (5565)	<input type="radio"/> Ano (1)
	Classes de rendimento nominal mensal (9)

No campo “Variável”, selecionar a opção “Pessoas de 10 anos ou mais de idade”; no campo “Classes de rendimento nominal mensal”, selecionar todas as 9 (nove) opções ; no campo “Ano”, selecionar “2010”; no campo “Unidade Territorial” selecionar “Município [5565/5565]”.



Clicar no botão “Download” e, na nova janela exibida: selecionar o formato “CSV (US)”; selecionar apenas a caixa de seleção “Exibir nomes de territórios”; e clicar em download.



- Data da consulta: 19/3/2021.

## 8. Tabela Auxiliar: Códigos dos Municípios - 2020 (DTB\_2020\_v2.zip)

O download do arquivo é feito diretamente a partir do link

[https://geoftp.ibge.gov.br/organizacao\\_do\\_territorio/estrutura\\_territorial/divisao\\_territorial/2020/DTB\\_2020\\_v2.zip](https://geoftp.ibge.gov.br/organizacao_do_territorio/estrutura_territorial/divisao_territorial/2020/DTB_2020_v2.zip)

Trata-se de um arquivo compactado que agrega 6 (seis) documentos, sendo que a base de interesse é: "RELATORIO\_DTB\_BRASIL\_MUNICIPIO.xls"

- Data da consulta: 19/3/2021.

## Apêndice III – Resumo das transformações nos datasets originais

1. Arrecadação das Receitas Administradas pela RFB, por Município (arrecadacao-da-receita-administrada-pela-rfb-por-municipio-2018.xlsx)

Base de dados original:			Ação implementada sobre a variável original	Linha de código correspondente no Jupyter Notebook*	Nova base de dados gerada: Ntbk 01_01 - RFB Arrecadacao 2018.csv			
Variável	Ocorrências não nulas	Tipo			Variável	Ocorrências não nulas	Tipo	Descrição da variável
MUNICÍPIOS	5576 non-null	object	Excluída (deu origem a nova variável)	13	--	--	--	--
UF	5576 non-null	object	Excluída (deu origem a nova variável)	13	--	--	--	--
ARRECADAÇÃO	5576 non-null	float64	Renomeada	12	01_arrecadacao_2018	5576 non-null	float64	Valor da arrecadação em 2018.
--	--	--	Criação da nova variável (derivada de MUNICÍPIOS e UF)	10	municipio-uf	5576 non-null	object	Nome do município e Sigla do Estado da Federação

\* Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb

2. Tabela 1685 - Unidades locais, empresas e outras organizações atuantes, pessoal ocupado total, pessoal ocupado assalariado, pessoal assalariado médio, salários e outras remunerações e salário médio mensal (tabela1685.csv)

Base de dados original:			Ação implementada sobre a variável original	Linha de código correspondente no Jupyter Notebook*	Nova base de dados gerada: Ntbk 01_02 - IBGE Empresas 2018.csv			
Variável	Ocorrências não nulas	Tipo			Variável	Ocorrências não nulas	Tipo	Descrição da variável
Município	16710 non-null	object	Excluída (deu origem a nova variável)	24	--	--	--	--
Ano	16710 non-null	int64	Excluída (valor fixo em todo o dataset - 2018)	24	--	--	--	--
Número de unidades locais (Unidades)	16710 non-null	int64	Renomeada	22	02_unidades	5570 non-null	int64	Número de unidades locais (Unidades)
Número de empresas e outras organizações atuantes (Unidades)	16710 non-null	int64	Excluída (deu origem a nova variável)	24	--	--	--	--
Pessoal ocupado total (Pessoas)	16710 non-null	int64	Renomeada	22	02_ocupados_total	5570 non-null	int64	Pessoal ocupado total (Pessoas)
Pessoal ocupado assalariado (Pessoas)	16710 non-null	int64	Excluída (deu origem a nova variável)	24	--	--	--	--
Pessoal assalariado médio (Pessoas)	16710 non-null	float64	Renomeada	22	02_ocupados_assalariados_medio	5570 non-null	float64	Pessoal assalariado médio (Pessoas)
Salários e outras remunerações (Mil Reais)	16710 non-null	int64	Excluída (pode ser obtida a partir de operação com outras colunas)	24	--	--	--	--
Salário médio mensal (Salários mínimos)	16710 non-null	float64	Renomeada	22	02_salario_medio_mensal_em_SM	5570 non-null	float64	Salário médio mensal (Salários mínimos)
Salário médio mensal em reais (Reais)	16710 non-null	float64	Renomeada	22	02_salario_medio_mensal_em_Reais	5570 non-null	float64	Salário médio mensal em reais (Reais)
--	--	--	Criação da nova variável (derivada de Município)	20	municipio-uf	5570 non-null	object	Nome do município e Sigla do Estado da Federação
--	--	--	Criação da nova variável [Número de empresas e outras organizações atuantes (Unidades)]	24	02_unidades_atuantes_%	5570 non-null	float64	Número de empresas e outras organizações atuantes (Unidades) dividido pelo Número de unidades locais (Unidades)
--	--	--	Criação da nova variável [Pessoal ocupado assalariado (Pessoas)]	24	02_ocupados_assalariados_%	5570 non-null	float64	Pessoal ocupado assalariado (Pessoas) dividido pelo Pessoal ocupado total (Pessoas)

\* Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb

3. Produto Interno Bruto dos Municípios ('PIB dos Municípios - base de dados 2010-2018.xls')

Base de dados original:			Ação implementada sobre a variável original	Linha de código correspondente no Jupyter Notebook*	Nova base de dados gerada: Ntbk 01_03 - IBGE PIB 2018.csv			
Variável	Ocorrências não nulas	Tipo			Variável	Ocorrências não nulas	Tipo	Descrição da variável
Ano	50115 non-null	int64	Excluída (valor fixo em todo o dataset - 2018)	38	--	--	--	--
Código da Grande Região	50115 non-null	int64	Excluída a variável com o código e mantida a com o nome.	36	--	--	--	--
Nome da Grande Região	50115 non-null	object	Renomeada	37	03_grande_regiao	5570 non-null	object	Nome da Grande Região
Código da Unidade da Federação	50115 non-null	int64	Excluída a variável com o código e mantida a com o nome.	36	--	--	--	--
Sigla da Unidade da Federação	50115 non-null	object	Renomeada	37	03_uf	5570 non-null	object	Sigla da Unidade da Federação
Nome da Unidade da Federação	50115 non-null	object	Renomeada	37	03_nome_uf	5570 non-null	object	Nome da Unidade da Federação
Código do Município	50115 non-null	int64	Renomeada e alterado o tipo	37	03_cod_municipio	5570 non-null	object	Código do Município
Nome do Município	50115 non-null	object	Excluída (deu origem a nova variável)	38	--	--	--	--
Região Metropolitana	12489 non-null	object	Excluída a variável (com expressivos NaN)	36	--	--	--	--
Código da Mesorregião	50115 non-null	int64	Excluída a variável com o código e mantida a com o nome.	36	--	--	--	--
Nome da Mesorregião	50115 non-null	object	Renomeada	37	03_mesorregiao	5570 non-null	object	Nome da Mesorregião
Código da Microrregião	50115 non-null	int64	Excluída a variável com o código e mantida a com o nome.	36	--	--	--	--
Nome da Microrregião	50115 non-null	object	Renomeada	37	03_micoregiao	5570 non-null	object	Nome da Microrregião
Código da Região Geográfica Imediata	50115 non-null	int64	Excluída a variável com o código e mantida a com o nome.	36	--	--	--	--
Nome da Região Geográfica Imediata	50115 non-null	object	Renomeada	37	03_Regiao_imediata	5570 non-null	object	Nome da Região Geográfica Imediata
Município da Região Geográfica Imediata	50115 non-null	object	Excluída a variável (sem interesse ou com expressivos NaN)	36	--	--	--	--
Código da Região Geográfica Intermediária	50115 non-null	int64	Excluída a variável com o código e mantida a com o nome.	36	--	--	--	--
Nome da Região Geográfica Intermediária	50115 non-null	object	Renomeada	37	03_Regiao_intermediaria	5570 non-null	object	Nome da Região Geográfica Intermediária
Município da Região Geográfica Intermediária	50115 non-null	object	Excluída a variável (sem interesse ou com expressivos NaN)	36	--	--	--	--
Código Concentração Urbana	5931 non-null	float64	Excluída a variável (com expressivos NaN)	36	--	--	--	--
Nome Concentração Urbana	5931 non-null	object	Excluída a variável (com expressivos NaN)	36	--	--	--	--
Tipo Concentração Urbana	5931 non-null	object	Excluída a variável (com expressivos NaN)	36	--	--	--	--
Código Arranjo Populacional	8595 non-null	float64	Excluída a variável (com expressivos NaN)	36	--	--	--	--
Nome Arranjo Populacional	8595 non-null	object	Excluída a variável (com expressivos NaN)	36	--	--	--	--
Hierarquia Urbana	50115 non-null	object	Excluída a variável (sem interesse ou com expressivos NaN)	36	--	--	--	--
Hierarquia Urbana (principais categorias)	50115 non-null	object	Excluída a variável (sem interesse ou com expressivos NaN)	36	--	--	--	--
Código da Região Rural	50115 non-null	int64	Excluída a variável (sem interesse ou com expressivos NaN)	36	--	--	--	--
Nome da Região Rural	50115 non-null	object	Excluída a variável (sem interesse ou com expressivos NaN)	36	--	--	--	--
Região rural (segundo classificação do núcleo)	50115 non-null	object	Excluída a variável (sem interesse ou com expressivos NaN)	36	--	--	--	--
Amazônia Legal	50115 non-null	object	Excluída a variável (sem interesse ou com expressivos NaN)	36	--	--	--	--
Semiárido	50115 non-null	object	Excluída a variável (sem interesse ou com expressivos NaN)	36	--	--	--	--
Cidade-Região de São Paulo	50115 non-null	object	Excluída a variável (sem interesse ou com expressivos NaN)	36	--	--	--	--

Valor adicionado bruto da Agropecuária, a preços correntes (R\$ 1.000)	50115 non-null	float64	Excluída (deu origem a nova variável)	38	--	--	--	--
Valor adicionado bruto da Indústria, a preços correntes (R\$ 1.000)	50115 non-null	float64	Excluída (deu origem a nova variável)	38	--	--	--	--
Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R\$ 1.000)	50115 non-null	float64	Excluída (deu origem a nova variável)	38	--	--	--	--
Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R\$ 1.000)	50115 non-null	float64	Excluída (deu origem a nova variável)	38	--	--	--	--
Valor adicionado bruto total, a preços correntes (R\$ 1.000)	50115 non-null	float64	Renomeada	37	03_vlr_adic_bruto_total	5570 non-null	float64	Valor adicionado bruto total, a preços correntes (R\$ 1.000)
Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R\$ 1.000)	50115 non-null	float64	Renomeada	37	03_impostos_sobre_produtos	5570 non-null	float64	Impostos, líquidos de subsídios, sobre produtos, a preços correntes (R\$ 1.000)
Produto Interno Bruto, a preços correntes (R\$ 1.000)	50115 non-null	float64	Renomeada	37	03_pib	5570 non-null	float64	Produto Interno Bruto, a preços correntes (R\$ 1.000)
Produto Interno Bruto per capita, a preços correntes (R\$ 1,00)	50115 non-null	float64	Renomeada	37	03_pib_per_capita	5570 non-null	float64	Produto Interno Bruto per capita, a preços correntes (R\$ 1,00)
Atividade com maior valor adicionado bruto	50115 non-null	object	Renomeada	37	03_atividade_principal_primeira	5570 non-null	object	Atividade com maior valor adicionado bruto
Atividade com segundo maior valor adicionado bruto	50115 non-null	object	Renomeada	37	03_atividade_principal_segunda	5570 non-null	object	Atividade com segundo maior valor adicionado bruto
Atividade com terceiro maior valor adicionado bruto	50115 non-null	object	Renomeada	37	03_atividade_principal_terceira	5570 non-null	object	Atividade com terceiro maior valor adicionado bruto
--	--	--	Criação do nova variável (derivada de Nome do Município e Sigla da Unidade da Federação)	38	municipio-uf	5570 non-null	object	Nome do município e Sigla do Estado da Federação
--	--	--	Criação do nova variável (derivada do valor adicionado pelo respectivo setor em relação ao valor adicionado bruto total)	38	03_vlr_adic_bruto_agropecuaria_%	5570 non-null	float64	Valor adicionado bruto da Agropecuária, a preços correntes (R\$ 1.000)
--	--	--	Criação do nova variável (derivada do valor adicionado pelo respectivo setor em relação ao valor adicionado bruto total)	38	03_vlr_adic_bruto_industria_%	5570 non-null	float64	Valor adicionado bruto da Indústria, a preços correntes (R\$ 1.000)
--	--	--	Criação do nova variável (derivada do valor adicionado pelo respectivo setor em relação ao valor adicionado bruto total)	38	03_vlr_adic_bruto_servicos_%	5570 non-null	float64	Valor adicionado bruto dos Serviços, a preços correntes - exceto Administração, defesa, educação e saúde públicas e seguridade social (R\$ 1.000)
--	--	--	Criação do nova variável (derivada do valor adicionado pelo respectivo setor em relação ao valor adicionado bruto total)	38	03_vlr_adic_bruto_administracao_%	5570 non-null	float64	Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, a preços correntes (R\$ 1.000)

\* Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb

4. Tabela 6579 - População residente estimada (tabela6579.csv)

Base de dados original:			Ação implementada sobre a variável original	Linha de código correspondente no Jupyter Notebook*	Nova base de dados gerada: Ntbk 01_04 - IBGE População Residente 2018.csv			
4. Tabela 6579 - População residente estimada (tabela6579.csv)					Variável	Ocorrências não nulas	Tipo	Descrição da variável
Variável	Ocorrências não nulas	Tipo						
Município	5570 non-null	object	Excluída (deu origem a nova variável)	48	--	--	--	--
Ano	5570 non-null	int64	Excluída (valor fixo em todo o dataset - 2018)	48	--	--	--	--
População residente estimada (Pessoas)	5570 non-null	float64	Renomeada e tipo alterado	48	04_populacao_residente_2018	5570 non-null	int64	População residente estimada para 2018
--	--	--	Criação do nova variável (derivada de Município)	47	municipio-uf	5570 non-null	object	Nome do município e Sigla do Estado da Federação

\* Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb

5. Tabela 200 - População residente, por sexo, situação e grupos de idade (tabela200.csv)

Base de dados original:			Ação implementada sobre a variável original	Linha de código correspondente no Jupyter Notebook*	Nova base de dados gerada: Ntbk 01_05 - IBGE Censo 2010 Idades.csv			
5. Tabela 200 - População residente, por sexo, situação e grupos de idade (tabela200.csv)					Variável	Ocorrências não nulas	Tipo	Descrição da variável
Variável	Ocorrências não nulas	Tipo						
Município	5566 non-null	object	Excluída (deu origem a nova variável)	56	--	--	--	--
Ano	5566 non-null	int64	Excluída (valor fixo em todo o dataset - 2018)	56	--	--	--	--
Situação do domicílio	5566 non-null	object	Excluída (valor fixo em todo o dataset )	56	--	--	--	--
Sexo	5566 non-null	object	Excluída (valor fixo em todo o dataset )	56	--	--	--	--
Total	5566 non-null	object	Renomeada e tipo alterado	57	05_total_residentes	5565 non-null	float64	População residente em 2018
0 a 4 anos	5566 non-null	object	Excluída (deu origem a nova variável)	62	--	--	--	--
5 a 9 anos	5566 non-null	object	Excluída (deu origem a nova variável)	62	--	--	--	--
10 a 14 anos	5566 non-null	object	Excluída (deu origem a nova variável)	62	--	--	--	--
15 a 19 anos	5566 non-null	object	Excluída (deu origem a nova variável)	62	--	--	--	--
20 a 24 anos	5566 non-null	object	Excluída (deu origem a nova variável)	62	--	--	--	--
25 a 29 anos	5566 non-null	object	Excluída (deu origem a nova variável)	62	--	--	--	--
30 a 34 anos	5566 non-null	object	Excluída (deu origem a nova variável)	62	--	--	--	--
35 a 39 anos	5566 non-null	object	Excluída (deu origem a nova variável)	62	--	--	--	--
40 a 44 anos	5566 non-null	object	Excluída (deu origem a nova variável)	62	--	--	--	--
45 a 49 anos	5566 non-null	object	Excluída (deu origem a nova variável)	62	--	--	--	--
50 a 54 anos	5566 non-null	object	Excluída (deu origem a nova variável)	62	--	--	--	--
55 a 59 anos	5566 non-null	object	Excluída (deu origem a nova variável)	62	--	--	--	--
60 a 64 anos	5566 non-null	object	Excluída (deu origem a nova variável)	62	--	--	--	--
65 a 69 anos	5566 non-null	object	Excluída (deu origem a nova variável)	62	--	--	--	--
70 a 74 anos	5566 non-null	object	Excluída (deu origem a nova variável)	62	--	--	--	--
75 a 79 anos	5566 non-null	object	Excluída (deu origem a nova variável)	62	--	--	--	--
80 a 84 anos	5566 non-null	object	Excluída (deu origem a nova variável)	62	--	--	--	--
85 a 89 anos	5566 non-null	object	Excluída (deu origem a nova variável)	62	--	--	--	--
90 a 94 anos	5566 non-null	object	Excluída (deu origem a nova variável)	62	--	--	--	--
95 a 99 anos	5566 non-null	object	Excluída (deu origem a nova variável)	62	--	--	--	--
100 anos ou mais	5566 non-null	object	Excluída (deu origem a nova variável)	62	--	--	--	--
--	--	--	Criação do nova variável (derivada de Município)	54	municipio-uf	5565 non-null	object	Nome do município e Sigla do Estado da Federação
--	--	--	Criação do nova variável (derivada - respectivas faixas)	62	05_0-19_anos_%	5565 non-null	float64	Participação relativa da população com esta faixa etária em relação à população total
--	--	--	Criação do nova variável (derivada - respectivas faixas)	62	05_20-39_anos_%	5565 non-null	float64	Participação relativa da população com esta faixa etária em relação à população total
--	--	--	Criação do nova variável (derivada - respectivas faixas)	62	05_40-59_anos_%	5565 non-null	float64	Participação relativa da população com esta faixa etária em relação à população total
--	--	--	Criação do nova variável (derivada - respectivas faixas)	62	05_60+_anos_%	5565 non-null	float64	Participação relativa da população com esta faixa etária em relação à população total

\* Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb |

6. Tabela 1554 - Pessoas de 10 anos ou mais de idade, por nível de instrução (tabela1554.csv)

Base de dados original:			Ação implementada sobre a variável original	Linha de código correspondente no Jupyter Notebook*	Nova base de dados gerada: Ntbk 01_06 - IBGE Censo Instrucao.csv			
Variável	Ocorrências não nulas	Tipo			Variável	Ocorrências não nulas	Tipo	Descrição da variável
Município	5565 non-null	object	Excluída (deu origem a nova variável)	72	--	--	--	--
Ano	5565 non-null	int64	Excluída (valor fixo em todo o dataset - 2018)	72	--	--	--	--
Total	5565 non-null	int64	Renomeada	79	06_total_instrucao	5565 non-null	int64	População residente em 2018
Sem instrução e fundamental incompleto	5565 non-null	int64	Excluída (deu origem a nova variável)	77	--	--	--	--
Fundamental completo e médio incompleto	5565 non-null	int64	Excluída (deu origem a nova variável)	77	--	--	--	--
Médio completo e superior incompleto	5565 non-null	int64	Excluída (deu origem a nova variável)	77	--	--	--	--
Superior completo	5565 non-null	int64	Excluída (deu origem a nova variável)	77	--	--	--	--
Não determinado	5565 non-null	object	Excluída (deu origem a nova variável)	77	--	--	--	--
--	--	--	Criação do nova variável (derivada de Município)	70	municipio-uf	5565 non-null	object	Nome do município e Sigla do Estado da Federação
--	--	--	Criação do nova variável (derivada)	77	06_Sem_Instrucao_e_Fundamental_Incompleto%	5565 non-null	float64	Participação relativa da população com este nível educacional em relação à população total
--	--	--	Criação do nova variável (derivada)	77	06_Medio_Incompleto_%	5565 non-null	float64	Participação relativa da população com este nível educacional em relação à população total
--	--	--	Criação do nova variável (derivada)	77	06_Superior_Incompleto_%	5565 non-null	float64	Participação relativa da população com este nível educacional em relação à população total
--	--	--	Criação do nova variável (derivada)	77	06_Superior_Completo_%	5565 non-null	float64	Participação relativa da população com este nível educacional em relação à população total
--	--	--	Criação do nova variável (derivada)	77	06_Indeterminado_%	5565 non-null	float64	Participação relativa da população com este nível educacional em relação à população total

\* Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb |

7. Tabela 2030 - Pessoas de 10 anos ou mais de idade, por classes de rendimento nominal mensal (tabela2030.csv)

Base de dados original:			Ação implementada sobre a variável original	Linha de código correspondente no Jupyter Notebook*	Nova base de dados gerada: Ntbk 01_07 - IBGE Censo Rendimentos.csv			
Variável	Ocorrências não nulas	Tipo			Variável	Ocorrências não nulas	Tipo	Descrição da variável
Município	5565 non-null	object	Excluída (deu origem a nova variável)	87	--	--	--	--
Ano	5565 non-null	int64	Excluída (valor fixo em todo o dataset - 2018)	87	--	--	--	--
Total	5565 non-null	int64	Renomeada	94	07_total_rendimentos	5565 non-null	int64	População residente em 2018
Até 1 salário mínimo	5565 non-null	int64	Excluída (deu origem a nova variável)	92	--	--	--	--
Mais de 1 a 2 salários mínimos	5565 non-null	int64	Excluída (deu origem a nova variável)	92	--	--	--	--
Mais de 2 a 3 salários mínimos	5565 non-null	int64	Excluída (deu origem a nova variável)	92	--	--	--	--
Mais de 3 a 5 salários mínimos	5565 non-null	int64	Excluída (deu origem a nova variável)	92	--	--	--	--
Mais de 5 a 10 salários mínimos	5565 non-null	object	Excluída (deu origem a nova variável)	92	--	--	--	--
Mais de 10 a 20 salários mínimos	5565 non-null	object	Excluída (deu origem a nova variável)	92	--	--	--	--
Mais de 20 salários mínimos	5565 non-null	object	Excluída (deu origem a nova variável)	92	--	--	--	--
Sem rendimento	5565 non-null	int64	Excluída (deu origem a nova variável)	92	--	--	--	--
--	--	--	Criação do nova variável (derivada de Município)	85	municipio-uf	5565 non-null	object	Nome do município e Sigla do Estado da Federação
--	--	--	Criação do nova variável (derivada)	92	07_Ate_15M_%	5565 non-null	float64	Participação relativa da população com esta faixa de renda em relação à população total
--	--	--	Criação do nova variável (derivada)	92	07_Ate_25M_%	5565 non-null	float64	Participação relativa da população com esta faixa de renda em relação à população total
--	--	--	Criação do nova variável (derivada)	92	07_Ate_35M_%	5565 non-null	float64	Participação relativa da população com esta faixa de renda em relação à população total
--	--	--	Criação do nova variável (derivada)	92	07_Ate_55M_%	5565 non-null	float64	Participação relativa da população com esta faixa de renda em relação à população total
--	--	--	Criação do nova variável (derivada)	92	07_Ate_10SM_%	5565 non-null	float64	Participação relativa da população com esta faixa de renda em relação à população total
--	--	--	Criação do nova variável (derivada)	92	07_Ate_20SM_%	5565 non-null	float64	Participação relativa da população com esta faixa de renda em relação à população total
--	--	--	Criação do nova variável (derivada)	92	07_20SM+_%	5565 non-null	float64	Participação relativa da população com esta faixa de renda em relação à população total
--	--	--	Criação do nova variável (derivada)	92	07_Sem_Rendimento_%	5565 non-null	float64	Participação relativa da população com esta faixa de renda em relação à população total

\* Vide Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb |

## Apêndice IV – Matriz de correlações

	01_arrecadacao_2018	02_unidades	02_unidades_atuantes_%	02_ocupados_total	02_ocupados_assalariados_%	02_ocupados_iados_medio	02_salario_mensal_em_SM	02_salario_mensal_em_Reais	02_vlr_adic_bruto_total	03_vlr_adic_bruto_agropecuaria_%	03_vlr_adic_bruto_industria_%	03_vlr_adic_bruto_servicos_%	03_impostos_sobre_produtos_%	03_pib	
01_arrecadacao_2018	1,00	0,93	-0,05	0,94	0,02	0,94	0,17	0,18	0,97	-0,06	0,01	0,12	-0,05	0,97	0,97
02_unidades	0,93	1,00	-0,08	0,99	0,01	0,99	0,21	0,21	0,97	-0,10	0,04	0,19	-0,10	0,94	0,97
02_unidades_atuantes_%	-0,05	-0,08	1,00	-0,08	0,11	-0,08	-0,20	-0,20	-0,08	0,03	-0,11	-0,23	0,24	-0,07	-0,08
02_ocupados_total	0,94	0,99	-0,08	1,00	0,03	1,00	0,22	0,22	0,98	-0,10	0,04	0,19	-0,09	0,95	0,98
02_ocupados_assalariados_%	0,02	0,01	0,11	0,03	1,00	0,03	-0,05	-0,06	0,03	-0,17	0,06	-0,28	0,32	0,03	0,03
02_ocupados_assalariados_medio	0,94	0,99	-0,08	1,00	0,03	1,00	0,22	0,22	0,98	-0,10	0,04	0,18	-0,09	0,95	0,98
02_salario_medio_mensal_em_SM	0,17	0,21	-0,20	0,22	-0,05	0,22	1,00	1,00	0,24	-0,01	0,45	0,24	-0,54	0,21	0,24
02_salario_medio_mensal_em_Reais	0,18	0,21	-0,20	0,22	-0,06	0,22	1,00	1,00	0,24	-0,01	0,45	0,24	-0,55	0,21	0,24
03_vlr_adic_bruto_total	0,97	0,97	-0,08	0,98	0,03	0,98	0,24	0,24	1,00	-0,10	0,07	0,18	-0,10	0,96	1,00
03_vlr_adic_bruto_agropecuaria_%	-0,06	-0,10	0,03	-0,10	-0,17	-0,10	-0,01	-0,01	-0,10	1,00	-0,35	-0,45	-0,25	-0,08	-0,10
03_vlr_adic_bruto_industria_%	0,01	0,04	-0,11	0,04	0,06	0,04	0,45	0,45	0,07	-0,35	1,00	0,08	-0,56	0,05	0,06
03_vlr_adic_bruto_servicos_%	0,12	0,19	-0,23	0,19	-0,28	0,18	0,24	0,24	0,18	-0,45	0,08	1,00	-0,44	0,16	0,17
03_vlr_adic_bruto_administracao_%	-0,05	-0,10	0,24	-0,09	0,32	-0,09	-0,54	-0,55	-0,10	-0,25	-0,56	-0,44	1,00	-0,09	-0,10
03_impostos_sobre_produtos	0,97	0,94	-0,07	0,95	0,03	0,95	0,21	0,21	0,96	-0,08	0,05	0,16	-0,09	1,00	0,98
03_pib	0,97	0,97	-0,08	0,98	0,03	0,98	0,24	0,24	1,00	-0,10	0,06	0,17	-0,10	0,98	1,00
03_pib_per_capita	0,07	0,08	-0,14	0,08	-0,04	0,08	0,56	0,56	0,13	0,09	0,55	0,13	-0,61	0,11	0,13
04_populacao_residente_2018	0,92	0,96	-0,10	0,98	0,04	0,98	0,23	0,23	0,96	-0,13	0,06	0,22	-0,10	0,94	0,96
05_total_residentes	0,92	0,96	-0,10	0,98	0,04	0,98	0,23	0,23	0,96	-0,13	0,05	0,22	-0,10	0,94	0,96
05_0-19_anos_%	-0,04	-0,07	0,17	-0,06	0,51	-0,06	-0,28	-0,28	-0,06	-0,04	-0,23	-0,40	0,53	-0,05	-0,06
05_20-39_anos_%	0,07	0,11	-0,17	0,12	0,18	0,12	0,22	0,22	0,12	-0,32	0,31	0,31	-0,21	0,10	0,12
05_40-59_anos_%	0,03	0,05	-0,10	0,04	-0,53	0,03	0,29	0,29	0,03	0,19	0,18	0,31	-0,56	0,03	0,03
05_60+anos_%	-0,01	-0,03	-0,03	-0,03	-0,43	-0,04	-0,04	-0,04	-0,04	0,12	-0,09	0,08	-0,09	-0,03	-0,04
06_total_instrucao	0,92	0,96	-0,10	0,98	0,04	0,98	0,23	0,23	0,96	-0,13	0,05	0,22	-0,10	0,95	0,96
06_Sem_Instrucao_e_Fundamental_Incompleto%	-0,12	-0,20	0,27	-0,19	0,25	-0,19	-0,51	-0,51	-0,20	0,18	-0,42	-0,63	0,66	-0,17	-0,19
06_Medio_Incompleto_%	0,04	0,08	-0,18	0,07	-0,23	0,07	0,39	0,39	0,08	0,00	0,33	0,41	-0,58	0,07	0,08
06_Superior_Incompleto_%	0,10	0,17	-0,26	0,17	-0,19	0,17	0,46	0,46	0,18	-0,25	0,40	0,59	-0,56	0,15	0,17
06_Superior_Completo_%	0,18	0,26	-0,26	0,26	-0,28	0,26	0,46	0,46	0,25	-0,13	0,29	0,60	-0,58	0,21	0,25
06_Indeterminado_%	0,03	0,04	-0,01	0,04	0,07	0,04	0,08	0,08	0,05	-0,03	0,05	0,05	-0,05	0,04	0,05
07_total rendimentos	0,92	0,96	-0,10	0,98	0,04	0,98	0,23	0,23	0,96	-0,13	0,05	0,22	-0,10	0,95	0,96
07_Ate_1SM %	-0,10	-0,15	0,25	-0,14	0,16	-0,14	-0,58	-0,58	-0,16	0,03	-0,44	-0,40	0,63	-0,13	-0,15
07_Ate_2SM %	0,03	0,07	-0,21	0,06	-0,41	0,05	0,44	0,44	0,06	0,10	0,39	0,42	-0,73	0,05	0,06
07_Ate_3SM %	0,06	0,11	-0,25	0,10	-0,37	0,10	0,55	0,55	0,11	0,02	0,44	0,45	-0,72	0,09	0,10
07_Ate_5SM %	0,09	0,15	-0,24	0,14	-0,36	0,14	0,58	0,58	0,15	-0,01	0,41	0,50	-0,71	0,13	0,14
07_Ate_10SM %	0,16	0,23	-0,21	0,23	-0,31	0,23	0,58	0,58	0,23	-0,06	0,38	0,55	-0,68	0,20	0,23
07_Ate_20SM %	0,23	0,31	-0,19	0,31	-0,24	0,31	0,53	0,53	0,31	-0,07	0,30	0,51	-0,57	0,26	0,30
07_20SM+ %	0,26	0,31	-0,15	0,32	-0,20	0,32	0,44	0,44	0,32	-0,04	0,23	0,42	-0,47	0,27	0,31
07_Sem_Rendimento_%	-0,02	-0,04	0,13	-0,03	0,52	-0,03	-0,28	-0,28	-0,03	-0,13	-0,25	-0,39	0,60	-0,02	-0,03
03_uf_AC	0,00	0,00	-0,01	0,00	0,06	0,00	-0,01	-0,01	0,00	0,01	-0,04	-0,05	0,07	0,00	0,00
03_uf_AL	-0,01	-0,01	0,03	-0,01	0,13	-0,01	-0,07	-0,08	-0,01	0,06	-0,07	-0,08	0,06	-0,01	-0,01
03_uf_AM	0,00	0,00	-0,01	0,00	0,13	0,00	-0,05	-0,05	0,00	0,04	-0,05	-0,11	0,09	0,00	0,00
03_uf_AP	0,00	0,00	0,00	0,00	0,06	0,00	0,03	0,03	0,00	-0,04	0,00	-0,06	0,08	0,00	0,00
03_uf_BA	-0,01	-0,01	0,09	-0,01	0,05	-0,01	-0,09	-0,09	-0,01	-0,10	-0,04	0,01	0,11	-0,01	-0,01
03_uf_CE	0,00	0,00	0,00	0,00	0,06	0,00	-0,14	-0,14	-0,01	-0,06	-0,05	-0,03	0,11	-0,01	-0,01
03_uf_DF	0,31	0,14	-0,02	0,19	0,01	0,20	0,10	0,10	0,31	-0,02	-0,01	0,02	0,01	0,17	0,28
03_uf_ES	0,00	0,01	0,00	0,00	-0,06	0,00	-0,01	-0,01	0,00	-0,04	0,05	0,06	-0,05	0,00	0,00
03_uf_GO	-0,01	-0,01	0,08	-0,01	-0,01	-0,01	0,01	0,01	-0,01	0,11	0,02	0,00	-0,11	-0,01	-0,01
03_uf_MA	-0,01	-0,01	0,09	-0,01	0,15	-0,01	-0,06	-0,06	-0,01	-0,03	-0,09	-0,06	0,15	-0,01	-0,01
03_uf_MG	-0,01	-0,01	0,06	-0,01	-0,10	-0,01	-0,18	-0,18	-0,02	-0,11	0,00	0,09	0,03	-0,02	-0,02
03_uf_MS	0,00	0,00	-0,06	0,00	0,00	0,00	0,06	0,06	0,00	0,11	0,01	-0,04	-0,07	0,00	0,00
03_uf_MT	-0,01	-0,01	0,01	-0,01	0,00	-0,01	0,12	0,12	0,00	0,13	-0,02	-0,05	-0,07	-0,01	0,00
03_uf_PA	-0,01	-0,01	0,00	0,00	0,17	0,00	0,00	0,00	0,00	0,06	-0,01	-0,11	0,04	-0,01	0,00
03_uf_PB	-0,01	-0,02	0,06	-0,01	0,14	-0,01	-0,14	-0,14	-0,02	-0,12	-0,09	-0,11	0,27	-0,01	-0,02
03_uf_PE	0,00	0,00	-0,02	0,00	0,12	0,00	-0,11	-0,11	0,00	-0,10	-0,04	-0,03	0,14	0,00	0,00

## Apêndice IV – Matriz de correlações

	01_arreca dacao_201 8	02_unidad es	02_unidad es_atuante s %	02_ocupad os_total	02_ocupad os_assalar iados %	02_ocupad os_assalar iados_med io	02_salario mensal_em Reais	02_salario medio_m ensal_em SM	02_vlr_adi c_bruto_tot al	03_vlr_adi c_bruto_ag ropecuaria %	03_vlr_adi c_bruto_in dustria %	03_vlr_adi c_bruto_se rvicos %	03_vlr_adi c_bruto_ad ministraca o %	03_impost os_sobre_p rodutos	03_pib
03_nf_PI	-0,01	-0,02	0,00	-0,02	0,05	-0,02	-0,09	-0,09	-0,02	-0,08	-0,10	-0,15	0,26	-0,01	-0,02
03_nf_PR	0,00	0,00	0,05	0,00	-0,14	0,00	0,08	0,08	0,00	0,19	0,03	0,05	-0,23	0,00	0,00
03_nf_RJ	0,05	0,05	-0,02	0,05	0,01	0,06	0,06	0,06	0,08	-0,12	0,06	0,08	-0,01	0,07	0,08
03_nf_RN	-0,01	-0,01	0,03	-0,01	0,12	-0,01	-0,08	-0,08	-0,01	-0,12	0,00	-0,09	0,18	-0,01	-0,01
03_nf_RO	0,00	0,00	0,01	0,00	0,01	0,00	-0,01	-0,01	0,00	0,09	-0,03	-0,07	0,00	0,00	0,00
03_nf_RR	0,00	0,00	0,01	0,00	0,05	0,00	-0,03	-0,03	0,00	-0,01	-0,03	-0,07	0,08	0,00	0,00
03_nf_RS	-0,01	0,00	0,00	-0,01	-0,26	-0,01	0,22	0,22	-0,01	0,23	0,04	0,02	-0,25	-0,01	-0,01
03_nf_SC	0,00	0,00	0,05	0,00	-0,08	0,00	0,11	0,11	-0,01	0,04	0,15	0,05	-0,19	0,00	0,00
03_nf_SE	0,00	-0,01	-0,01	0,00	0,10	0,00	0,05	0,05	-0,01	-0,07	0,00	-0,04	0,09	-0,01	-0,01
03_nf_SP	0,05	0,06	-0,38	0,05	-0,18	0,05	0,28	0,28	0,07	-0,08	0,15	0,32	-0,29	0,06	0,07
03_nf_TO	-0,01	-0,01	0,04	-0,01	0,11	-0,01	-0,11	-0,10	-0,01	0,06	-0,07	-0,13	0,10	-0,01	-0,01
03_atividade_principal_primeira_Administração, defesa, educação e saúde públicas e segurança social	-0,04	-0,09	0,22	-0,08	0,26	-0,08	-0,46	-0,46	-0,08	-0,15	-0,46	-0,43	0,83	-0,07	-0,08
03_atividade_principal_primeira_Agricultura, inclusive apoio à agricultura e a pós colheita	-0,02	-0,03	-0,01	-0,03	-0,16	-0,03	0,12	0,12	-0,03	0,64	-0,16	-0,19	-0,29	-0,02	-0,03
03_atividade_principal_primeira_Comércio e reparação de veículos automotores e motocicletas	0,00	0,00	-0,03	0,00	0,02	0,00	0,07	0,07	0,01	-0,05	-0,01	0,21	-0,10	0,02	0,01
03_atividade_principal_primeira_Construção	0,00	0,00	0,01	0,00	0,03	0,00	0,04	0,04	0,00	-0,02	0,08	-0,03	-0,02	0,00	0,00
03_atividade_principal_primeira_Demais serviços	0,06	0,13	-0,20	0,11	-0,24	0,11	0,23	0,23	0,10	-0,25	0,18	0,70	-0,46	0,09	0,10
03_atividade_principal_primeira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	-0,01	-0,01	-0,01	-0,01	0,01	-0,01	0,10	0,10	0,00	-0,08	0,45	-0,17	-0,16	-0,01	0,00
03_atividade_principal_primeira_Indústrias de transformação	0,00	0,00	-0,06	0,00	0,08	0,00	0,23	0,23	0,01	-0,15	0,51	-0,03	-0,26	0,02	0,02
03_atividade_principal_primeira_Indústrias extractivas	0,00	0,00	0,01	0,00	0,07	0,00	0,17	0,17	0,02	-0,10	0,31	-0,05	-0,12	0,00	0,02
03_atividade_principal_primeira_Pecuária, inclusive apoio à pecuária	-0,01	-0,01	0,02	-0,01	-0,04	-0,01	0,01	0,01	-0,01	0,23	-0,06	-0,12	-0,06	-0,01	-0,01
03_atividade_principal_primeira_Produção florestal, pesca e aquicultura	0,00	-0,01	-0,02	-0,01	0,03	-0,01	-0,01	-0,01	-0,01	0,18	-0,03	-0,10	-0,05	-0,01	-0,01
03_atividade_principal_segunda_Administração, defesa, educação e saúde públicas e segurança social	0,00	0,01	-0,06	0,01	-0,19	0,01	0,04	0,04	0,00	0,10	0,02	0,24	-0,28	0,00	0,00
03_atividade_principal_segunda_Agricultura, inclusive apoio à agricultura e a pós colheita	-0,01	-0,02	-0,07	-0,02	-0,03	-0,02	0,00	0,00	-0,02	0,26	-0,08	-0,11	-0,08	-0,02	-0,02
03_atividade_principal_segunda_Comércio e reparação de veículos automotores e motocicletas	0,11	0,17	-0,06	0,17	-0,03	0,17	0,17	0,17	0,15	-0,11	0,02	0,37	-0,20	0,13	0,15
03_atividade_principal_segunda_Construção	0,00	0,00	0,02	0,00	0,04	0,00	0,00	0,00	0,00	-0,03	0,07	-0,04	0,00	0,00	0,00
03_atividade_principal_segunda_Demais serviços	-0,04	-0,08	0,12	-0,07	0,16	-0,07	-0,21	-0,21	-0,06	-0,17	-0,14	-0,29	0,48	-0,05	-0,06
03_atividade_principal_segunda_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	0,00	0,00	0,00	0,00	0,01	0,00	0,01	0,01	0,00	-0,02	0,09	-0,04	-0,02	0,00	0,00
03_atividade_principal_segunda_Indústrias de transformação	0,01	0,05	-0,09	0,04	0,02	0,04	0,25	0,25	0,05	-0,17	0,34	0,21	-0,28	0,05	0,05
03_atividade_principal_segunda_Indústrias extractivas	0,01	0,01	0,01	0,01	0,03	0,01	0,04	0,04	0,01	-0,05	0,12	-0,01	-0,04	0,01	0,01
03_atividade_principal_segunda_Pecuária, inclusive apoio à pecuária	-0,01	-0,02	0,06	-0,02	0,01	-0,02	-0,03	-0,03	-0,02	0,20	-0,09	-0,17	0,03	-0,01	-0,02
03_atividade_principal_segunda_Produção florestal, pesca e aquicultura	0,00	-0,01	0,02	-0,01	0,04	-0,01	-0,04	-0,04	-0,01	0,09	-0,03	-0,11	0,03	-0,01	-0,01
03_atividade_principal_terceira_Administração, defesa, educação e saúde públicas e segurança social	0,03	0,07	-0,13	0,06	-0,13	0,06	0,31	0,31	0,07	0,06	0,37	0,17	-0,48	0,05	0,06
03_atividade_principal_terceira_Agricultura, inclusive apoio à agricultura e a pós colheita	-0,02	-0,04	-0,01	-0,04	-0,03	-0,04	-0,10	-0,11	-0,04	0,06	-0,15	-0,09	0,13	-0,03	-0,04
03_atividade_principal_terceira_Comércio e reparação de veículos automotores e motocicletas	0,03	0,04	-0,06	0,04	-0,06	0,04	-0,02	-0,01	0,04	-0,33	-0,10	0,41	0,05	0,04	0,04
03_atividade_principal_terceira_Construção	0,00	0,00	0,02	0,00	0,01	0,00	-0,02	-0,02	-0,01	-0,08	0,03	0,01	0,04	-0,01	-0,01
03_atividade_principal_terceira_Demais serviços	-0,02	-0,04	0,07	-0,04	0,07	-0,04	0,00	0,00	-0,04	0,44	-0,05	-0,38	-0,05	-0,03	-0,04
03_atividade_principal_terceira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	0,00	0,00	0,00	0,00	0,04	0,00	-0,01	-0,01	0,00	-0,07	0,03	-0,03	0,05	0,00	0,00
03_atividade_principal_terceira_Indústrias de transformação	0,01	0,01	-0,03	0,01	0,02	0,01	0,04	0,04	0,01	-0,15	0,17	0,14	-0,12	0,02	0,01
03_atividade_principal_terceira_Indústrias extractivas	0,00	-0,01	0,01	0,00	0,04	0,00	0,02	0,02	0,00	-0,07	0,08	-0,02	0,01	0,00	0,00
03_atividade_principal_terceira_Pecuária, inclusive apoio à pecuária	-0,02	-0,04	0,14	-0,04	0,10	-0,04	-0,20	-0,20	-0,04	-0,05	-0,21	-0,22	0,38	-0,03	-0,04
03_atividade_principal_terceira_Produção florestal, pesca e aquicultura	-0,01	-0,02	0,05	-0,02	0,08	-0,02	-0,09	-0,09	-0,02	0,01	-0,09	-0,10	0,15	-0,01	-0,02

## Apêndice IV – Matriz de correlações

	03_pib_per_capita	04_populacao_residente_2018	05_total_residentes	05_0-19_anos_%	05_20-39_anos_%	05_40-59_anos_%	05_60+_anos_%	06_total_instrucao	06_Sem_Instrucao_e_Fundamental_Incompleto%	06_Medio_Incompleto_%	06_Superior_Incompleto_%	06_Superior_Completo_%	06_Indeterminado_%	07_total_rendimentos	07_Ate_1SM_%
01_arrecadacao_2018	0,07	0,92	0,92	-0,04	0,07	0,03	-0,01	0,92	-0,12	0,04	0,10	0,18	0,03	0,92	-0,10
02_unidades	0,08	0,96	0,96	-0,07	0,11	0,05	-0,03	0,96	-0,20	0,08	0,17	0,26	0,04	0,96	-0,15
02_unidades_atuantes_%	-0,14	-0,10	-0,10	0,17	-0,17	-0,10	-0,03	-0,10	0,27	-0,18	-0,26	-0,26	-0,01	-0,10	0,25
02_ocupados_total	0,08	0,98	0,98	-0,06	0,12	0,04	-0,03	0,98	-0,19	0,07	0,17	0,26	0,04	0,98	-0,14
02_ocupados_assalariados_%	-0,04	0,04	0,04	0,51	0,18	-0,53	-0,43	0,04	0,25	-0,23	-0,19	-0,28	0,07	0,04	0,16
02_ocupados_assalariados_medio	0,08	0,98	0,98	-0,06	0,12	0,03	-0,04	0,98	-0,19	0,07	0,17	0,26	0,04	0,98	-0,14
02_salario_medio_mensal_em_SM	0,56	0,23	0,23	-0,28	0,22	0,29	-0,04	0,23	-0,51	0,39	0,46	0,46	0,08	0,23	-0,58
02_salario_medio_mensal_em_Reais	0,56	0,23	0,23	-0,28	0,22	0,29	-0,04	0,23	-0,51	0,39	0,46	0,46	0,08	0,23	-0,58
03_vlr_adic_bruto_total	0,13	0,96	0,96	-0,06	0,12	0,03	-0,04	0,96	-0,20	0,08	0,18	0,25	0,05	0,96	-0,16
03_vlr_adic_bruto_agropecuaria_%	0,09	-0,13	-0,13	-0,04	-0,32	0,19	0,12	-0,13	0,18	0,00	-0,25	-0,13	-0,03	-0,13	0,03
03_vlr_adic_bruto_industria_%	0,55	0,06	0,05	-0,23	0,31	0,18	-0,09	0,05	-0,42	0,33	0,40	0,29	0,05	0,05	-0,44
03_vlr_adic_bruto_servicos_%	0,13	0,22	0,22	-0,40	0,31	0,31	0,08	0,22	-0,63	0,41	0,59	0,60	0,05	0,22	-0,40
03_vlr_adic_bruto_administracao_%	-0,61	-0,10	-0,10	0,53	-0,21	-0,56	-0,09	-0,10	0,66	-0,58	-0,56	-0,58	-0,05	-0,10	0,63
03_impostos_sobre_produtos	0,11	0,94	0,94	-0,05	0,10	0,03	-0,03	0,95	-0,17	0,07	0,15	0,21	0,04	0,95	-0,13
03_pib	0,13	0,96	0,96	-0,06	0,12	0,03	-0,04	0,96	-0,19	0,08	0,17	0,25	0,05	0,96	-0,15
03_pib_per_capita	1,00	0,07	0,07	-0,29	0,16	0,31	0,00	0,07	-0,41	0,34	0,35	0,38	0,05	0,07	-0,47
04_populacao_residente_2018	0,07	1,00	1,00	-0,05	0,16	0,01	-0,07	1,00	-0,22	0,09	0,21	0,26	0,06	1,00	-0,16
05_total_residentes	0,07	1,00	1,00	-0,05	0,15	0,02	-0,07	1,00	-0,22	0,09	0,21	0,26	0,05	1,00	-0,16
05_0-19_anos_%	-0,29	-0,05	-0,05	1,00	0,04	-0,92	-0,73	-0,05	0,56	-0,40	-0,51	-0,55	0,11	-0,05	0,34
05_20-39_anos_%	0,16	0,16	0,15	0,04	1,00	-0,29	-0,61	0,15	-0,35	0,24	0,38	0,18	0,13	0,15	-0,33
05_40-59_anos_%	0,31	0,01	0,02	-0,92	-0,29	1,00	0,71	0,02	-0,51	0,42	0,43	0,53	-0,10	0,02	-0,37
05_60+_anos_%	0,00	-0,07	-0,07	-0,73	-0,61	0,71	1,00	-0,06	-0,08	0,00	0,06	0,20	-0,18	-0,06	0,12
06_total_instrucao	0,07	1,00	1,00	-0,05	0,15	0,02	-0,06	1,00	-0,22	0,09	0,20	0,26	0,05	1,00	-0,16
06_Sem_Instrucao_e_Fundamental_Incompleto%	-0,41	-0,22	-0,22	0,56	-0,35	-0,51	-0,08	-0,22	1,00	-0,77	-0,94	-0,82	-0,08	-0,22	0,73
06_Medio_Incompleto_%	0,34	0,09	0,09	-0,40	0,24	0,42	0,00	0,09	-0,77	1,00	0,58	0,47	0,01	0,09	-0,62
06_Superior_Incompleto_%	0,35	0,21	0,21	-0,51	0,38	0,43	0,06	0,20	-0,94	0,58	1,00	0,70	0,03	0,20	-0,63
06_Superior_Completo_%	0,38	0,26	0,26	-0,55	0,18	0,53	0,20	0,26	-0,82	0,47	0,70	1,00	0,04	0,26	-0,63
06_Indeterminado_%	0,05	0,06	0,05	0,11	0,13	-0,10	-0,18	0,05	-0,08	0,01	0,03	0,04	1,00	0,05	-0,11
07_total_rendimentos	0,07	1,00	1,00	-0,05	0,15	0,02	-0,06	1,00	-0,22	0,09	0,20	0,26	0,05	1,00	-0,16
07_Ate_1SM_%	-0,47	-0,16	-0,16	0,34	-0,33	-0,37	0,12	-0,16	0,73	-0,62	-0,63	-0,63	-0,11	-0,16	1,00
07_Ate_2SM_%	0,43	0,05	0,05	-0,67	0,13	0,70	0,25	0,05	-0,70	0,64	0,60	0,59	-0,01	0,05	-0,77
07_Ate_3SM_%	0,48	0,10	0,09	-0,64	0,21	0,64	0,19	0,10	-0,76	0,64	0,66	0,67	0,01	0,10	-0,84
07_Ate_5SM_%	0,48	0,14	0,14	-0,63	0,18	0,64	0,19	0,14	-0,80	0,63	0,70	0,75	0,02	0,14	-0,83
07_Ate_10SM_%	0,48	0,24	0,23	-0,56	0,20	0,57	0,14	0,23	-0,80	0,57	0,70	0,84	0,03	0,23	-0,77
07_Ate_20SM_%	0,41	0,31	0,31	-0,46	0,18	0,46	0,11	0,31	-0,70	0,44	0,60	0,80	0,04	0,31	-0,63
07_20SM+_%	0,35	0,32	0,31	-0,37	0,16	0,37	0,08	0,31	-0,57	0,35	0,48	0,70	0,05	0,31	-0,52
07_Sem_Rendimento_%	-0,31	-0,01	-0,01	0,80	0,08	-0,81	-0,54	-0,02	0,54	-0,46	-0,46	-0,52	0,12	-0,02	0,33
03_nf_AC	-0,03	0,00	0,00	0,15	-0,01	-0,12	-0,11	0,00	0,04	-0,04	-0,03	-0,03	0,04	0,00	-0,03
03_nf_AL	-0,06	0,00	0,00	0,17	0,00	-0,16	-0,12	0,00	0,16	-0,15	-0,13	-0,12	-0,03	0,00	0,08
03_nf_AM	-0,05	0,01	0,01	0,28	-0,02	-0,23	-0,20	0,01	0,07	-0,07	-0,06	-0,07	0,04	0,01	-0,05
03_nf_AP	-0,01	0,00	0,00	0,13	0,01	-0,10	-0,11	0,00	-0,02	0,04	0,02	-0,02	0,00	0,00	-0,04
03_nf_BA	-0,13	0,00	0,00	0,12	0,06	-0,20	-0,04	0,00	0,20	-0,26	-0,09	-0,24	0,03	0,00	0,27
03_nf_CE	-0,10	0,01	0,01	0,10	-0,02	-0,14	0,00	0,01	0,05	0,02	-0,04	-0,11	0,00	0,01	0,19
03_nf_DF	0,03	0,18	0,17	-0,01	0,03	0,00	-0,02	0,16	-0,04	0,00	0,03	0,07	0,01	0,16	-0,03
03_nf_ES	0,03	0,01	0,01	-0,05	0,07	0,04	-0,02	0,01	-0,04	0,02	0,04	0,05	-0,02	0,01	0,01
03_nf_GO	0,04	-0,01	-0,01	-0,09	0,07	0,10	-0,02	-0,01	-0,08	0,09	0,08	0,04	0,02	-0,01	-0,04
03_nf_MA	-0,11	0,00	0,00	0,31	-0,01	-0,30	-0,19	-0,01	0,14	-0,06	-0,14	-0,17	0,01	-0,01	0,10
03_nf_MG	-0,06	-0,02	-0,02	-0,14	-0,07	0,15	0,13	-0,02	0,05	-0,05	-0,09	0,04	0,06	-0,02	0,15
03_nf_MS	0,09	0,00	0,00	0,02	0,04	0,00	-0,07	0,00	-0,02	0,01	0,00	0,08	0,02	0,00	-0,07
03_nf_MT	0,10	-0,01	-0,01	0,04	0,13	-0,02	-0,17	-0,01	-0,06	0,07	0,02	0,04	0,23	-0,01	-0,12
03_nf_PA	-0,04	0,02	0,01	0,27	0,08	-0,25	-0,25	0,01	0,11	-0,03	-0,12	-0,14	-0,01	0,01	0,01
03_nf_PB	-0,11	-0,02	-0,02	0,06	-0,02	-0,13	0,07	-0,02	0,19	-0,23	-0,13	-0,14	-0,08	-0,02	0,22
03_nf_PE	-0,08	0,01	0,01	0,10	0,06	-0,15	-0,05	0,01	0,12	-0,15	-0,08	-0,11	0,01	0,01	0,14

## Apêndice IV – Matriz de correlações

	03_pib_per_capita	04_população_residente_2018	05_total_residentes	05_0-19_anos_%	05_20-39_anos_%	05_40-59_anos_%	05_60+_anos_%	06_total_instrução	06_Sem_Instrução_e_Fundamental_Incompleto%	06_Medio_Incompleto%	06_Superior_Incompleto%	06_Superior_Completo%	06_Indeterminado_%	07_total_rendimentos	07_Ate_1SM_%
03_nf_PI	-0,10	-0,02	-0,02	0,12	-0,02	-0,14	-0,03	-0,02	0,21	-0,18	-0,21	-0,12	-0,09	-0,02	0,15
03_nf_PR	0,10	-0,01	-0,01	-0,10	-0,10	0,17	0,06	-0,01	-0,12	0,12	0,09	0,12	-0,02	-0,01	-0,10
03_nf_RJ	0,07	0,09	0,09	-0,09	0,03	0,10	0,02	0,09	-0,15	0,08	0,08	0,17	0,10	-0,02	0,09
03_nf_RN	-0,06	-0,01	-0,01	0,04	0,05	-0,10	0,01	-0,01	0,08	-0,11	-0,03	-0,10	-0,04	-0,01	0,14
03_nf_RO	0,00	0,00	0,00	0,06	0,06	-0,03	-0,12	0,00	0,02	0,01	-0,04	-0,02	0,02	0,00	-0,03
03_nf_RR	-0,01	0,00	0,00	0,12	-0,03	-0,08	-0,10	0,00	0,01	-0,02	-0,01	-0,02	0,08	0,00	-0,04
03_nf_RS	0,19	-0,02	-0,02	-0,34	-0,33	0,43	0,38	-0,02	-0,11	0,23	0,03	0,08	-0,09	-0,02	-0,17
03_nf_SC	0,10	-0,01	-0,02	-0,13	-0,03	0,21	0,02	-0,01	-0,15	0,17	0,10	0,14	-0,05	-0,01	-0,25
03_nf_SE	-0,05	0,00	0,00	0,09	0,06	-0,10	-0,09	0,00	0,07	-0,08	-0,05	-0,06	-0,02	0,00	0,10
03_nf_SP	0,14	0,05	0,05	-0,29	0,18	0,20	0,12	0,05	-0,42	0,27	0,41	0,37	0,03	0,05	-0,44
03_nf_TO	-0,03	-0,02	-0,02	0,15	-0,03	-0,11	-0,11	-0,02	0,00	-0,03	0,00	0,01	0,01	-0,02	0,04
03_atividade_principal_primeira_Administração, defesa, educação e saúde públicas e segurança social	-0,49	-0,09	-0,09	0,46	-0,18	-0,48	-0,09	-0,09	0,60	-0,50	-0,52	-0,54	-0,04	-0,09	0,59
03_atividade_principal_primeira_Agricultura, inclusive apoio à agricultura e a pós colheita	0,20	-0,05	-0,04	-0,09	-0,14	0,17	0,08	-0,04	0,00	0,09	-0,05	0,00	-0,01	-0,04	-0,07
03_atividade_principal_primeira_Comércio e reparação de veículos automotores e motocicletas	0,16	0,00	0,00	-0,02	0,05	0,02	-0,02	0,00	-0,06	0,05	0,05	0,03	0,04	0,00	-0,07
03_atividade_principal_primeira_Construção	0,01	0,00	0,00	0,01	0,03	-0,01	-0,02	0,00	-0,02	0,01	0,02	0,00	0,01	0,00	0,00
03_atividade_principal_primeira_Demais serviços	0,12	0,14	0,14	-0,37	0,23	0,33	0,07	0,14	-0,56	0,38	0,52	0,55	0,03	0,14	-0,44
03_atividade_principal_primeira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	0,21	-0,01	-0,01	0,00	0,00	0,01	-0,01	-0,01	-0,01	0,02	0,01	0,00	0,00	-0,01	-0,03
03_atividade_principal_primeira_Indústrias de transformação	0,27	0,00	0,00	-0,12	0,14	0,10	-0,03	0,00	-0,17	0,18	0,16	0,08	0,00	0,00	-0,24
03_atividade_principal_primeira_Indústrias extractivas	0,24	0,01	0,00	0,00	0,06	-0,01	-0,05	0,00	-0,06	0,04	0,07	0,03	0,02	0,00	-0,05
03_atividade_principal_primeira_Pecuária, inclusive apoio à pecuária	0,01	-0,02	-0,02	-0,07	-0,06	0,10	0,05	-0,02	0,01	0,02	-0,03	-0,01	0,00	-0,02	-0,03
03_atividade_principal_primeira_Produção florestal, pesca e aquicultura	0,02	-0,01	-0,01	0,05	0,00	-0,03	-0,05	-0,01	0,02	-0,01	-0,03	-0,01	0,01	-0,01	0,01
03_atividade_principal_segunda_Administração, defesa, educação e saúde públicas e segurança social	-0,01	0,02	0,02	-0,18	0,02	0,20	0,07	0,02	-0,19	0,17	0,16	0,17	0,00	0,02	-0,13
03_atividade_principal_segunda_Agricultura, inclusive apoio à agricultura e a pós colheita	0,01	-0,03	-0,03	0,02	-0,09	0,03	0,01	-0,03	0,03	0,00	-0,05	0,00	0,00	-0,03	-0,01
03_atividade_principal_segunda_Comércio e reparação de veículos automotores e motocicletas	0,14	0,18	0,18	-0,11	0,13	0,09	-0,02	0,18	-0,26	0,17	0,23	0,28	0,02	0,18	-0,21
03_atividade_principal_segunda_Construção	0,03	-0,01	-0,01	0,01	0,00	-0,02	0,00	-0,01	0,02	-0,02	-0,01	-0,02	0,00	-0,01	0,03
03_atividade_principal_segunda_Demais serviços	-0,13	-0,08	-0,08	0,24	-0,09	-0,28	-0,01	-0,08	0,37	-0,30	-0,31	-0,37	-0,02	-0,08	0,36
03_atividade_principal_segunda_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	0,01	0,00	0,00	0,01	0,03	-0,02	-0,03	0,00	0,00	0,01	0,00	0,00	0,00	0,00	-0,02
03_atividade_principal_segunda_Indústrias de transformação	0,17	0,06	0,05	-0,17	0,21	0,14	-0,04	0,05	-0,33	0,22	0,31	0,30	0,03	0,05	-0,35
03_atividade_principal_segunda_Indústrias extractivas	0,02	0,00	0,00	0,00	0,03	0,00	-0,02	0,00	-0,03	0,00	0,03	0,03	0,00	0,00	-0,02
03_atividade_principal_segunda_Pecuária, inclusive apoio à pecuária	-0,03	-0,03	-0,03	0,03	-0,09	0,03	-0,02	-0,02	0,07	-0,02	-0,08	-0,06	0,01	-0,02	0,01
03_atividade_principal_segunda_Produção florestal, pesca e aquicultura	0,00	-0,01	-0,01	0,09	-0,01	-0,07	-0,07	-0,01	0,05	-0,01	-0,06	-0,05	-0,02	-0,01	0,02
03_atividade_principal_terceira_Administração, defesa, educação e saúde públicas e segurança social	0,34	0,06	0,06	-0,26	0,12	0,27	0,05	0,06	-0,35	0,29	0,31	0,30	0,02	0,06	-0,38
03_atividade_principal_terceira_Agricultura, inclusive apoio à agricultura e a pós colheita	-0,12	-0,04	-0,04	0,05	-0,06	-0,05	0,03	-0,04	0,13	-0,10	-0,12	-0,10	-0,03	-0,04	0,11
03_atividade_principal_terceira_Comércio e reparação de veículos automotores e motocicletas	-0,05	0,07	0,07	-0,01	0,12	-0,05	-0,03	0,07	-0,11	0,04	0,12	0,11	0,00	0,07	0,01
03_atividade_principal_terceira_Construção	-0,02	-0,01	-0,01	0,01	0,01	-0,02	0,00	-0,01	0,01	-0,02	-0,01	-0,01	-0,01	-0,01	0,04
03_atividade_principal_terceira_Demais serviços	0,00	-0,05	-0,05	0,13	-0,12	-0,05	-0,07	-0,05	0,16	-0,05	-0,19	-0,16	-0,01	-0,05	0,06
03_atividade_principal_terceira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	-0,01	0,00	0,00	0,01	0,02	-0,02	0,00	0,00	-0,01	-0,01	0,03	0,00	0,01	0,00	0,02
03_atividade_principal_terceira_Indústrias de transformação	0,06	0,01	0,01	-0,09	0,12	0,06	-0,01	0,01	-0,14	0,10	0,13	0,09	0,04	0,01	-0,10
03_atividade_principal_terceira_Indústrias extractivas	-0,02	-0,01	-0,01	0,00	0,02	-0,02	-0,01	-0,01	0,01	-0,03	0,00	-0,02	0,01	-0,01	0,02
03_atividade_principal_terceira_Pecuária, inclusive apoio à pecuária	-0,18	-0,05	-0,05	0,10	-0,16	-0,09	0,08	-0,05	0,25	-0,22	-0,22	-0,19	-0,01	-0,05	0,25
03_atividade_principal_terceira_Produção florestal, pesca e aquicultura	-0,09	-0,02	-0,02	0,15	-0,03	-0,14	-0,07	-0,02	0,12	-0,08	-0,11	-0,10	0,00	-0,02	0,08

## Apêndice IV – Matriz de correlações

	07_Ate_2S M_%	07_Ate_3S M_%	07_Ate_5S M_%	07_Ate_10 SM_%	07_Ate_20 SM_%	07_20SM+ %	07_Sem_R endimento %	03_nf_AC	03_nf_AL	03_nf_AM	03_nf_AP	03_nf_BA	03_nf_CE	03_nf_DF	03_nf_ES
01_arrecadacao_2018	0,03	0,06	0,09	0,16	0,23	0,26	-0,02	0,00	-0,01	0,00	0,00	-0,01	0,00	0,31	0,00
02_unidades	0,07	0,11	0,15	0,23	0,31	0,31	-0,04	0,00	-0,01	0,00	0,00	-0,01	0,00	0,14	0,01
02_unidades_atuantes_%	-0,21	-0,25	-0,24	-0,21	-0,19	-0,15	0,13	-0,01	0,03	-0,01	0,00	0,09	0,00	-0,02	0,00
02_ocupados_total	0,06	0,10	0,14	0,23	0,31	0,32	-0,03	0,00	-0,01	0,00	0,00	-0,01	0,00	0,19	0,00
02_ocupados_assalariados_%	-0,41	-0,37	-0,36	-0,31	-0,24	-0,20	0,52	0,06	0,13	0,13	0,06	0,05	0,06	0,01	-0,06
02_ocupados_assalariados_medio	0,05	0,10	0,14	0,23	0,31	0,32	-0,03	0,00	-0,01	0,00	0,00	-0,01	0,00	0,20	0,00
02_salario_medio_mensal_em_SM	0,44	0,55	0,58	0,58	0,53	0,44	-0,28	-0,01	-0,07	-0,05	0,03	-0,09	-0,14	0,10	-0,01
02_salario_medio_mensal_em_Reais	0,44	0,55	0,58	0,58	0,53	0,44	-0,28	-0,01	-0,08	-0,05	0,03	-0,09	-0,14	0,10	-0,01
03_vlr_adic_bruto_total	0,06	0,11	0,15	0,23	0,31	0,32	-0,03	0,00	-0,01	0,00	0,00	-0,01	0,01	0,31	0,00
03_vlr_adic_bruto_agropecuaria_%	0,10	0,02	-0,01	-0,06	-0,07	-0,04	-0,13	0,01	0,06	0,04	-0,04	-0,10	-0,06	-0,02	-0,04
03_vlr_adic_bruto_industria_%	0,39	0,44	0,41	0,38	0,30	0,23	-0,25	-0,04	-0,07	-0,05	0,00	-0,04	-0,05	-0,01	0,05
03_vlr_adic_bruto_servicos_%	0,42	0,45	0,50	0,55	0,51	0,42	-0,39	-0,05	-0,08	-0,11	-0,06	0,01	-0,03	0,02	0,06
03_vlr_adic_bruto_administracao_%	-0,73	-0,72	-0,71	-0,68	-0,57	-0,47	0,60	0,07	0,06	0,09	0,08	0,11	0,11	0,01	-0,05
03_impostos_sobre_produtos	0,05	0,09	0,13	0,20	0,26	0,27	-0,02	0,00	-0,01	0,00	0,00	-0,01	-0,01	0,17	0,00
03_pib	0,06	0,10	0,14	0,23	0,30	0,31	-0,03	0,00	-0,01	0,00	0,00	-0,01	-0,01	0,28	0,00
03_pib_per_capita	0,43	0,48	0,48	0,48	0,41	0,35	-0,31	-0,03	-0,06	-0,05	-0,01	-0,13	-0,10	0,03	0,03
04_populacao_residente_2018	0,05	0,10	0,14	0,24	0,31	0,32	-0,01	0,00	0,00	0,01	0,00	0,00	0,01	0,18	0,01
05_total_residentes	0,05	0,09	0,14	0,23	0,31	0,31	-0,01	0,00	0,00	0,01	0,00	0,00	0,01	0,17	0,01
05_0-19_anos_%	-0,67	-0,64	-0,63	-0,56	-0,46	-0,37	0,80	0,15	0,17	0,28	0,13	0,12	0,10	-0,01	-0,05
05_20-39_anos_%	0,13	0,21	0,18	0,20	0,18	0,16	0,08	-0,01	0,00	-0,02	0,01	0,06	-0,02	0,03	0,07
05_40-59_anos_%	0,70	0,64	0,64	0,57	0,46	0,37	-0,81	-0,12	-0,16	-0,23	-0,10	-0,20	-0,14	0,00	0,04
05_60+anos_%	0,25	0,19	0,19	0,14	0,11	0,08	-0,54	-0,11	-0,12	-0,20	-0,11	-0,04	0,00	-0,02	-0,02
06_total_instrucao	0,05	0,10	0,14	0,23	0,31	0,31	-0,02	0,00	0,00	0,01	0,00	0,00	0,01	0,16	0,01
06_Sem_Instrucao_e_Fundamental_Incompleto%	-0,70	-0,76	-0,80	-0,80	-0,70	-0,57	0,54	0,04	0,16	0,07	-0,02	0,20	0,05	-0,04	-0,04
06_Medio_Incompleto_%	0,64	0,64	0,63	0,57	0,44	0,35	-0,46	-0,04	-0,15	-0,07	0,04	-0,26	0,02	0,00	0,02
06_Superior_Incompleto_%	0,60	0,66	0,70	0,70	0,60	0,48	-0,46	-0,03	-0,13	-0,06	0,02	-0,09	-0,04	0,03	0,04
06_Superior_Completo_%	0,59	0,67	0,75	0,84	0,80	0,70	-0,52	-0,03	-0,12	-0,07	-0,02	-0,24	-0,11	0,07	0,05
06_Indeterminado_%	-0,01	0,01	0,02	0,03	0,04	0,05	0,12	0,04	-0,03	0,04	0,00	0,03	0,00	0,01	-0,02
07_total rendimentos	0,05	0,10	0,14	0,23	0,31	0,31	-0,02	0,00	0,00	0,01	0,00	0,00	0,01	0,16	0,01
07_Ate_1SM %	-0,77	-0,84	-0,83	-0,77	-0,63	-0,52	0,33	-0,03	0,08	-0,05	-0,04	0,27	0,19	-0,03	0,01
07_Ate_2SM %	1,00	0,89	0,83	0,71	0,53	0,44	-0,81	-0,04	-0,15	-0,10	-0,03	-0,27	-0,20	0,00	0,03
07_Ate_3SM %	0,89	1,00	0,93	0,82	0,64	0,52	-0,72	-0,04	-0,13	-0,09	-0,02	-0,25	-0,18	0,01	0,01
07_Ate_5SM %	0,83	0,93	1,00	0,90	0,74	0,61	-0,69	-0,03	-0,13	-0,08	0,00	-0,24	-0,18	0,02	0,03
07_Ate_10SM %	0,71	0,82	0,90	1,00	0,86	0,73	-0,60	-0,01	-0,11	-0,06	0,02	-0,21	-0,16	0,06	0,05
07_Ate_20SM %	0,53	0,64	0,74	0,86	1,00	0,80	-0,47	-0,02	-0,08	-0,05	0,01	-0,16	-0,12	0,13	0,05
07_20SM+ %	0,44	0,52	0,61	0,73	0,80	1,00	-0,40	-0,01	-0,07	-0,03	0,01	-0,13	-0,10	0,16	0,03
07_Sem_Rendimento %	-0,81	-0,72	-0,69	-0,60	-0,47	-0,40	1,00	0,10	0,16	0,22	0,08	0,16	0,13	-0,01	-0,06
03_nf_AC	-0,04	-0,04	-0,03	-0,01	-0,02	-0,01	0,10	1,00	-0,01	0,00	-0,02	-0,01	0,00	-0,01	-0,01
03_nf_AL	-0,15	-0,13	-0,13	-0,11	-0,08	-0,07	0,16	-0,01	1,00	-0,01	-0,01	-0,04	-0,03	0,00	-0,02
03_nf_AM	-0,10	-0,09	-0,08	-0,06	-0,05	-0,03	0,22	-0,01	-0,01	1,00	-0,01	-0,03	-0,02	0,00	-0,01
03_nf_AP	-0,03	-0,02	0,00	0,02	0,01	0,01	0,08	0,00	-0,01	-0,01	1,00	-0,02	-0,01	0,00	-0,01
03_nf_BA	-0,27	-0,25	-0,24	-0,21	-0,16	-0,13	0,16	-0,02	-0,04	-0,03	-0,02	1,00	-0,05	0,00	-0,03
03_nf_CE	-0,20	-0,18	-0,18	-0,16	-0,12	-0,10	0,13	-0,01	-0,03	-0,02	-0,01	-0,05	1,00	0,00	-0,02
03_nf_DF	0,00	0,01	0,02	0,06	0,13	0,16	-0,01	0,00	0,00	0,00	0,00	0,00	0,00	1,00	0,00
03_nf_ES	0,03	0,01	0,03	0,05	0,05	0,03	-0,06	-0,01	-0,02	-0,01	-0,01	-0,03	-0,02	0,00	1,00
03_nf_GO	0,07	0,06	0,05	0,06	0,05	0,06	-0,08	-0,01	-0,03	-0,02	-0,01	-0,06	-0,04	0,00	-0,03
03_nf_MA	-0,22	-0,20	-0,19	-0,17	-0,13	-0,11	0,26	-0,01	-0,03	-0,02	-0,01	-0,06	-0,04	0,00	-0,02
03_nf_MG	-0,01	-0,07	-0,08	-0,07	-0,04	-0,02	-0,10	-0,03	-0,06	-0,05	-0,02	-0,12	-0,08	-0,01	-0,05
03_nf_MS	0,06	0,04	0,05	0,06	0,05	0,07	-0,02	-0,01	-0,02	-0,01	-0,01	-0,03	-0,02	0,00	-0,01
03_nf_MT	0,05	0,08	0,07	0,07	0,05	0,04	0,03	-0,01	-0,02	-0,02	-0,01	-0,05	-0,03	0,00	-0,02
03_nf_PA	-0,13	-0,11	-0,10	-0,09	-0,06	-0,05	0,20	-0,01	-0,02	-0,02	-0,01	-0,05	-0,03	0,00	-0,02
03_nf_PB	-0,20	-0,19	-0,19	-0,18	-0,13	-0,11	0,11	-0,01	-0,03	-0,02	-0,01	-0,06	-0,04	0,00	-0,02
03_nf_PE	-0,16	-0,15	-0,15	-0,13	-0,10	-0,09	0,13	-0,01	-0,03	-0,02	-0,01	-0,05	-0,03	0,00	-0,02

## Apêndice IV – Matriz de correlações

	07_Ate_2S M_%	07_Ate_3S M_%	07_Ate_5S M_%	07_Ate_10 SM_%	07_Ate_20 SM_%	07_20SM+ %	07_Sem_R endimento %	03_nf_AC	03_nf_AL	03_nf_AM	03_nf_AP	03_nf_BA	03_nf_CE	03_nf_DF	03_nf_ES
03_nf_PI	-0,22	-0,18	-0,20	-0,18	-0,14	-0,12	0,20	-0,01	-0,03	-0,02	-0,01	-0,06	-0,04	0,00	-0,02
03_nf_PR	0,22	0,12	0,10	0,11	0,06	0,06	-0,19	-0,02	-0,04	-0,03	-0,01	-0,08	-0,05	0,00	-0,03
03_nf_RJ	0,05	0,07	0,09	0,12	0,12	0,06	-0,02	-0,01	-0,02	-0,01	-0,01	-0,04	-0,02	0,00	-0,02
03_nf_RN	-0,15	-0,14	-0,13	-0,13	-0,10	-0,08	0,10	-0,01	-0,02	-0,02	-0,01	-0,05	-0,03	0,00	-0,02
03_nf_RO	0,00	0,01	0,02	0,02	0,00	0,01	0,02	-0,01	-0,01	-0,01	-0,01	-0,03	-0,02	0,00	-0,01
03_nf_RR	-0,04	-0,03	-0,02	-0,01	-0,01	-0,01	0,10	0,00	-0,01	-0,01	0,00	-0,01	-0,01	0,00	-0,01
03_nf_RS	0,31	0,30	0,31	0,27	0,22	0,18	-0,36	-0,02	-0,04	-0,03	-0,02	-0,09	-0,06	0,00	-0,04
03_nf_SC	0,30	0,32	0,32	0,24	0,17	0,13	-0,26	-0,01	-0,03	-0,03	-0,01	-0,07	-0,04	0,00	-0,03
03_nf_SE	-0,11	-0,10	-0,09	-0,07	-0,06	-0,06	0,06	-0,01	-0,02	-0,01	-0,01	-0,03	-0,02	0,00	-0,01
03_nf_SP	0,39	0,41	0,38	0,29	0,21	0,16	-0,20	-0,02	-0,05	-0,04	-0,02	-0,10	-0,07	0,00	-0,04
03_nf_TO	-0,09	-0,09	-0,08	-0,04	-0,05	-0,02	0,10	-0,01	-0,02	-0,02	-0,01	-0,05	-0,03	0,00	-0,02
03_atividade_principal_primeira_Administração, defesa, educação e saúde públicas e segurança social	-0,64	-0,64	-0,63	-0,61	-0,51	-0,42	0,50	0,05	0,06	0,07	0,05	0,14	0,12	0,01	-0,01
03_atividade_principal_primeira_Agricultura, inclusive apoio à agricultura e a pós colheita	0,14	0,11	0,08	0,06	0,05	0,04	-0,13	-0,02	0,03	-0,02	-0,02	-0,07	-0,05	0,00	-0,03
03_atividade_principal_primeira_Comércio e reparação de veículos automotores e motocicletas	0,06	0,07	0,07	0,05	0,04	0,05	-0,03	0,03	-0,01	-0,01	0,00	0,01	-0,02	0,00	-0,01
03_atividade_principal_primeira_Construção	-0,01	0,00	-0,01	-0,01	0,00	-0,01	0,00	0,00	0,00	0,00	0,00	0,01	0,02	0,00	0,00
03_atividade_principal_primeira_Demais serviços	0,45	0,47	0,50	0,52	0,45	0,37	-0,37	-0,03	-0,06	-0,06	-0,03	-0,08	-0,07	-0,01	0,03
03_atividade_principal_primeira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	0,01	0,03	0,03	0,03	0,02	0,02	0,00	-0,01	-0,02	-0,02	0,02	-0,01	0,00	0,00	-0,01
03_atividade_principal_primeira_Indústrias de transformação	0,26	0,26	0,21	0,16	0,09	0,06	-0,17	-0,01	-0,03	-0,02	-0,01	-0,03	-0,02	0,00	-0,01
03_atividade_principal_primeira_Indústrias extractivas	0,00	0,02	0,03	0,03	0,05	0,03	0,02	-0,01	-0,01	0,02	-0,01	0,01	-0,02	0,00	0,06
03_atividade_principal_primeira_Pecuária, inclusive apoio à pecuária	0,06	0,06	0,06	0,03	0,04	0,05	-0,07	-0,01	-0,02	-0,01	-0,01	-0,03	-0,02	0,00	0,00
03_atividade_principal_primeira_Produção florestal, pesca e aquicultura	-0,01	-0,02	-0,03	-0,02	-0,02	-0,02	0,02	0,00	0,02	0,06	0,00	-0,02	-0,01	0,00	-0,01
03_atividade_principal_segunda_Administração, defesa, educação e saúde públicas e segurança social	0,20	0,17	0,17	0,17	0,14	0,11	-0,20	-0,02	-0,01	-0,03	-0,02	-0,04	-0,05	-0,01	0,02
03_atividade_principal_segunda_Agricultura, inclusive apoio à agricultura e a pós colheita	0,03	0,01	0,01	0,00	-0,01	-0,01	-0,02	0,00	0,07	0,04	-0,02	-0,05	-0,02	0,00	0,00
03_atividade_principal_segunda_Comércio e reparação de veículos automotores e motocicletas	0,16	0,19	0,23	0,27	0,25	0,23	-0,11	-0,01	-0,03	-0,02	-0,01	-0,03	-0,02	0,00	0,00
03_atividade_principal_segunda_Construção	-0,02	-0,02	-0,02	-0,02	-0,02	-0,02	0,01	0,00	-0,01	0,00	0,00	0,02	-0,01	0,00	-0,01
03_atividade_principal_segunda_Demais serviços	-0,37	-0,37	-0,39	-0,39	-0,33	-0,27	0,28	0,00	0,00	0,01	0,02	0,11	0,08	0,01	-0,01
03_atividade_principal_segunda_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	0,00	0,00	0,01	0,01	0,00	0,00	0,02	0,00	-0,01	-0,01	0,05	-0,01	0,02	0,00	-0,01
03_atividade_principal_segunda_Indústrias de transformação	0,28	0,33	0,34	0,31	0,26	0,21	-0,17	-0,02	-0,03	0,00	-0,06	-0,03	0,00	-0,06	-0,01
03_atividade_principal_segunda_Indústrias extractivas	-0,02	0,00	0,01	0,03	0,06	0,04	0,02	0,00	-0,01	-0,01	0,00	0,00	0,00	0,00	0,01
03_atividade_principal_segunda_Pecuária, inclusive apoio à pecuária	-0,01	-0,01	0,00	0,00	-0,02	-0,01	0,01	0,08	-0,03	-0,01	0,01	-0,01	-0,03	0,00	-0,02
03_atividade_principal_segunda_Produção florestal, pesca e aquicultura	-0,05	-0,05	-0,04	-0,05	-0,04	-0,04	0,06	-0,01	0,03	0,10	0,03	-0,01	-0,02	0,00	-0,01
03_atividade_principal_terceira_Administração, defesa, educação e saúde públicas e segurança social	0,41	0,40	0,37	0,34	0,27	0,21	-0,29	-0,03	-0,04	-0,04	-0,03	-0,08	-0,07	-0,01	0,00
03_atividade_principal_terceira_Agricultura, inclusive apoio à agricultura e a pós colheita	-0,08	-0,09	-0,11	-0,14	-0,12	-0,09	0,05	0,03	0,03	0,04	-0,01	0,00	0,08	-0,01	0,02
03_atividade_principal_terceira_Comércio e reparação de veículos automotores e motocicletas	-0,06	-0,01	0,03	0,09	0,11	0,09	0,02	-0,02	-0,02	-0,05	0,00	0,06	-0,01	0,03	0,02
03_atividade_principal_terceira_Construção	-0,03	-0,03	-0,02	-0,02	-0,02	-0,02	0,01	-0,01	0,00	-0,01	0,02	0,02	0,00	0,02	0,00
03_atividade_principal_terceira_Demais serviços	-0,06	-0,08	-0,08	-0,09	-0,09	-0,08	0,07	0,04	0,07	0,06	0,02	-0,04	-0,02	-0,01	-0,02
03_atividade_principal_terceira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	-0,03	-0,02	-0,02	-0,01	-0,01	-0,01	0,02	-0,01	0,00	-0,01	0,03	0,00	0,00	0,00	0,02
03_atividade_principal_terceira_Indústrias de transformação	0,12	0,11	0,10	0,09	0,07	0,04	-0,08	-0,01	-0,03	-0,03	-0,01	-0,03	0,00	0,00	0,02
03_atividade_principal_terceira_Indústrias extractivas	-0,03	-0,02	-0,02	-0,02	-0,02	-0,02	0,04	-0,01	0,01	0,00	-0,01	0,01	-0,02	0,00	0,03
03_atividade_principal_terceira_Pecuária, inclusive apoio à pecuária	-0,24	-0,24	-0,24	-0,23	-0,19	-0,15	0,15	0,01	-0,02	-0,04	-0,02	0,08	0,01	-0,01	-0,04
03_atividade_principal_terceira_Produção florestal, pesca e aquicultura	-0,12	-0,12	-0,12	-0,11	-0,10	-0,08	0,13	-0,01	0,00	0,14	0,05	-0,01	0,02	0,00	-0,02

## Apêndice IV – Matriz de correlações

	03_uf_GO	03_uf_MA	03_uf_MG	03_uf_MS	03_uf_MT	03_uf_PA	03_uf_PB	03_uf_PE	03_uf_PI	03_uf_PR	03_uf_RJ	03_uf_RN	03_uf_RO	03_uf_RR	03_uf_RS
01_arrecadacao_2018	-0,01	-0,01	-0,01	0,00	-0,01	-0,01	-0,01	0,00	-0,01	0,00	0,05	-0,01	0,00	0,00	-0,01
02_unidades	-0,01	-0,01	-0,01	0,00	-0,01	-0,01	-0,02	0,00	-0,02	0,00	0,05	-0,01	0,00	0,00	0,00
02_unidades_atuantes_%	0,08	0,09	0,06	-0,06	0,01	0,00	0,06	-0,02	0,00	0,05	-0,02	0,03	0,01	0,01	0,00
02_ocupados_total	-0,01	-0,01	-0,01	0,00	-0,01	0,00	-0,01	0,00	-0,02	0,00	0,05	-0,01	0,00	0,00	-0,01
02_ocupados_assalariados_%	-0,01	0,15	-0,10	0,00	0,00	0,17	0,14	0,12	0,05	-0,14	0,01	0,12	0,01	0,05	-0,26
02_ocupados_assalariados_medio	-0,01	-0,01	-0,01	0,00	-0,01	0,00	-0,01	0,00	-0,02	0,00	0,06	-0,01	0,00	0,00	-0,01
02_salario_medio_mensal_em_SM	0,01	-0,06	-0,18	0,06	0,12	0,00	-0,14	-0,11	-0,09	0,08	0,06	-0,08	-0,01	-0,03	0,22
02_salario_medio_mensal_em_Reais	0,01	-0,06	-0,18	0,06	0,12	0,00	-0,14	-0,11	-0,09	0,08	0,06	-0,08	-0,01	-0,03	0,22
03_vlr_adic_bruto_total	-0,01	-0,01	-0,02	0,00	0,00	0,00	-0,02	0,00	-0,02	0,00	0,08	-0,01	0,00	0,00	-0,01
03_vlr_adic_bruto_agropecuaria_%	0,11	-0,03	-0,11	0,11	0,13	0,06	-0,12	-0,10	-0,08	0,19	-0,12	-0,12	0,09	-0,01	0,23
03_vlr_adic_bruto_industria_%	0,02	-0,09	0,00	0,01	-0,02	-0,01	-0,09	-0,04	-0,10	0,03	0,06	0,00	-0,03	-0,03	0,04
03_vlr_adic_bruto_servicos_%	0,00	-0,06	0,09	-0,04	-0,05	-0,11	-0,11	-0,03	-0,15	0,05	0,08	-0,09	-0,07	-0,07	0,02
03_vlr_adic_bruto_administracao_%	-0,11	0,15	0,03	-0,07	-0,07	0,04	0,27	0,14	0,26	-0,23	-0,01	0,18	0,00	0,08	-0,25
03_impostos_sobre_produtos	-0,01	-0,01	-0,02	0,00	-0,01	-0,01	-0,01	0,00	-0,01	0,00	0,07	-0,01	0,00	0,00	-0,01
03_pib	-0,01	-0,01	-0,02	0,00	0,00	0,00	-0,02	0,00	-0,02	0,00	0,08	-0,01	0,00	0,00	-0,01
03_pib_per_capita	0,04	-0,11	-0,06	0,09	0,10	-0,04	-0,11	-0,08	-0,10	0,10	0,07	-0,06	0,00	-0,01	0,19
04_populacao_residente_2018	-0,01	0,00	-0,02	0,00	-0,01	0,02	-0,02	0,01	-0,02	-0,01	0,09	-0,01	0,00	0,00	-0,02
05_total_residentes	-0,01	0,00	-0,02	0,00	-0,01	0,01	-0,02	0,01	-0,02	-0,01	0,09	-0,01	0,00	0,00	-0,02
05_0-19_anos_%	-0,09	0,31	-0,14	0,02	0,04	0,27	0,06	0,10	0,12	-0,10	-0,09	0,04	0,06	0,12	-0,34
05_20-39_anos_%	0,07	-0,01	-0,07	0,04	0,13	0,08	-0,02	0,06	-0,02	-0,10	0,03	0,05	0,06	-0,03	-0,33
05_40-59_anos_%	0,10	-0,30	0,15	0,00	-0,02	-0,25	-0,13	-0,15	-0,14	0,17	0,10	-0,10	-0,03	-0,08	0,43
05_60+anos_%	-0,02	-0,19	0,13	-0,07	-0,17	-0,25	0,07	-0,05	-0,03	0,06	0,02	0,01	-0,12	-0,10	0,38
06_total_instrucao	-0,01	-0,01	-0,02	0,00	-0,01	0,01	-0,02	0,01	-0,02	-0,01	0,09	-0,01	0,00	0,00	-0,02
06_Sem_Instrucao_e_Fundamental_Incompleto%	-0,08	0,14	0,05	-0,02	-0,06	0,11	0,19	0,12	0,21	-0,12	-0,15	0,08	0,02	0,01	-0,11
06_Medio_Incompleto_%	0,09	-0,06	-0,05	0,01	0,07	-0,03	-0,23	-0,15	-0,18	0,12	0,08	-0,11	0,01	-0,02	0,23
06_Superior_Incompleto_%	0,08	-0,14	-0,09	0,00	0,02	-0,12	-0,13	-0,08	-0,21	0,09	0,17	-0,03	-0,04	-0,01	0,03
06_Superior_Completo_%	0,04	-0,17	0,04	0,08	0,04	-0,14	-0,14	-0,11	-0,12	0,12	0,10	-0,10	-0,02	-0,02	0,08
06_Indeterminado_%	0,02	0,01	0,06	0,02	0,23	-0,01	-0,08	0,01	-0,09	-0,02	-0,02	-0,04	0,02	0,08	-0,09
07_total rendimentos	-0,01	-0,01	-0,02	0,00	-0,01	0,01	-0,02	0,01	-0,02	-0,01	0,09	-0,01	0,00	0,00	-0,02
07_Ate_1SM %	-0,04	0,10	0,15	-0,07	-0,12	0,01	0,22	0,14	0,15	-0,10	-0,10	0,14	-0,03	-0,04	-0,17
07_Ate_2SM %	0,07	-0,22	-0,01	0,06	0,05	-0,13	-0,20	-0,16	-0,22	0,22	0,05	-0,15	0,00	-0,04	0,31
07_Ate_3SM %	0,06	-0,20	-0,07	0,04	0,08	-0,11	-0,19	-0,15	-0,18	0,12	0,07	-0,14	0,01	-0,03	0,30
07_Ate_5SM %	0,05	-0,19	-0,08	0,05	0,07	-0,10	-0,19	-0,15	-0,20	0,10	0,09	-0,13	0,02	-0,02	0,31
07_Ate_10SM %	0,06	-0,17	-0,07	0,06	0,07	-0,09	-0,18	-0,13	-0,18	0,11	0,12	-0,13	0,02	-0,01	0,27
07_Ate_20SM %	0,05	-0,13	-0,04	0,05	0,05	-0,06	-0,13	-0,10	-0,14	0,06	0,12	-0,10	0,00	-0,01	0,22
07_20SM+ %	0,06	-0,11	-0,02	0,07	0,04	-0,05	-0,11	-0,09	-0,12	0,06	0,06	-0,08	0,01	-0,01	0,18
07_Sem_Rendimento %	-0,08	0,26	-0,10	-0,02	0,03	0,20	0,11	0,13	0,20	-0,19	-0,02	0,10	0,02	0,10	-0,36
03_uf_AC	-0,01	-0,01	-0,03	-0,01	-0,01	-0,01	-0,01	-0,01	-0,01	-0,02	-0,01	-0,01	-0,01	0,00	-0,02
03_uf_AL	-0,03	-0,03	-0,06	-0,02	-0,02	-0,02	-0,03	-0,03	-0,03	-0,04	-0,02	-0,02	-0,01	-0,04	-0,04
03_uf_AM	-0,02	-0,02	-0,05	-0,01	-0,02	-0,02	-0,02	-0,02	-0,02	-0,03	-0,01	-0,02	-0,01	-0,03	-0,03
03_uf_AP	-0,01	-0,01	-0,02	-0,01	-0,01	-0,01	-0,01	-0,01	-0,01	-0,01	-0,01	-0,01	-0,01	-0,01	-0,02
03_uf_BA	-0,06	-0,06	-0,12	-0,03	-0,05	-0,05	-0,06	-0,05	-0,06	-0,08	-0,04	-0,05	-0,03	-0,01	-0,09
03_uf_CE	-0,04	-0,04	-0,08	-0,02	-0,03	-0,03	-0,04	-0,03	-0,04	-0,05	-0,02	-0,03	-0,02	-0,01	-0,06
03_uf_DF	0,00	0,00	-0,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
03_uf_ES	-0,03	-0,02	-0,05	-0,01	-0,02	-0,02	-0,02	-0,02	-0,02	-0,03	-0,02	-0,02	-0,01	-0,01	-0,04
03_uf_GO	1,00	-0,04	-0,09	-0,03	-0,03	-0,03	-0,04	-0,04	-0,04	-0,06	-0,03	-0,04	-0,02	-0,01	-0,07
03_uf_MA	-0,04	1,00	-0,09	-0,02	-0,03	-0,03	-0,04	-0,04	-0,04	-0,06	-0,03	-0,04	-0,02	-0,01	-0,06
03_uf_MG	-0,09	-0,09	1,00	-0,05	-0,07	-0,07	-0,09	-0,08	-0,09	-0,12	-0,06	-0,07	-0,04	-0,02	-0,13
03_uf_MS	-0,03	-0,02	-0,05	1,00	-0,02	-0,02	-0,02	-0,02	-0,02	-0,02	-0,03	-0,02	-0,02	-0,01	-0,04
03_uf_MT	-0,03	-0,03	-0,07	-0,02	1,00	-0,03	-0,03	-0,03	-0,03	-0,03	-0,04	-0,02	-0,02	-0,01	-0,05
03_uf_PA	-0,03	-0,03	-0,07	-0,02	-0,03	1,00	-0,03	-0,03	-0,03	-0,03	-0,05	-0,02	-0,03	-0,01	-0,05
03_uf_PB	-0,04	-0,04	-0,09	-0,02	-0,03	-0,03	1,00	-0,04	-0,04	-0,06	-0,03	-0,04	-0,02	-0,01	-0,06
03_uf_PE	-0,04	-0,04	-0,08	-0,02	-0,03	-0,03	-0,04	1,00	-0,04	-0,05	-0,02	-0,03	-0,02	-0,01	-0,06

## Apêndice IV – Matriz de correlações

	03_uf_GO	03_uf_MA	03_uf_MG	03_uf_MS	03_uf_MT	03_uf_PA	03_uf_PB	03_uf_PE	03_uf_PI	03_uf_PR	03_uf_RJ	03_uf_RN	03_uf_RO	03_uf_RR	03_uf_RS
03_uf_PI	-0,04	-0,04	-0,09	-0,02	-0,03	-0,03	-0,04	-0,04	1,00	-0,06	-0,03	-0,04	-0,02	-0,01	-0,06
03_uf_PR	-0,06	-0,06	-0,12	-0,03	-0,04	-0,05	-0,06	-0,05	-0,06	1,00	-0,04	-0,05	-0,03	-0,01	-0,09
03_uf_RJ	-0,03	-0,03	-0,06	-0,02	-0,02	-0,02	-0,02	-0,03	-0,02	-0,03	-0,04	1,00	-0,02	-0,01	-0,04
03_uf_RN	-0,04	-0,04	-0,07	-0,02	-0,03	-0,03	-0,04	-0,03	-0,04	-0,05	-0,02	1,00	-0,02	-0,01	-0,06
03_uf_RO	-0,02	-0,02	-0,04	-0,01	-0,02	-0,02	-0,02	-0,02	-0,02	-0,03	-0,01	-0,02	1,00	-0,01	-0,03
03_uf_RR	-0,01	-0,01	-0,02	-0,01	-0,01	-0,01	-0,01	-0,01	-0,01	-0,01	-0,01	-0,01	-0,01	1,00	-0,02
03_uf_RS	-0,07	-0,06	-0,13	-0,04	-0,05	-0,05	-0,06	-0,06	-0,06	-0,09	-0,04	-0,06	-0,03	-0,02	1,00
03_uf_SC	-0,05	-0,05	-0,10	-0,03	-0,04	-0,04	-0,05	-0,04	-0,05	-0,07	-0,03	-0,04	-0,02	-0,01	-0,07
03_uf_SE	-0,03	-0,02	-0,05	-0,01	-0,02	-0,02	-0,02	-0,02	-0,02	-0,03	-0,02	-0,02	-0,01	-0,01	-0,04
03_uf_SP	-0,08	-0,07	-0,15	-0,04	-0,06	-0,06	-0,07	-0,07	-0,07	-0,10	-0,05	-0,06	-0,04	-0,02	-0,11
03_uf_TO	-0,03	-0,03	-0,07	-0,02	-0,03	-0,03	-0,03	-0,03	-0,03	-0,04	-0,02	-0,03	-0,02	-0,01	-0,05
03_atividade_principal_primeira_Administração, defesa, educação e saúde públicas e segurança social	-0,08	0,15	0,04	-0,05	-0,03	0,07	0,18	0,11	0,17	-0,20	-0,01	0,12	0,02	0,05	-0,22
03_atividade_principal_primeira_Agricultura, inclusive apoio à agricultura e a pós colheita	0,02	-0,05	-0,10	0,06	0,13	-0,03	-0,07	-0,06	-0,04	0,18	-0,05	-0,06	-0,03	-0,02	0,24
03_atividade_principal_primeira_Comércio e reparação de veículos automotores e motocicletas	0,00	-0,01	-0,01	-0,01	0,02	-0,01	-0,01	-0,02	-0,02	0,00	-0,01	-0,02	-0,01	0,00	0,02
03_atividade_principal_primeira_Construção	0,02	-0,01	-0,01	0,00	-0,01	-0,01	-0,01	0,02	-0,01	-0,01	0,00	-0,01	0,00	0,00	0,01
03_atividade_principal_primeira_Demais serviços	0,04	-0,10	0,04	-0,01	-0,05	-0,06	-0,11	-0,07	-0,12	0,10	0,03	-0,09	-0,02	-0,03	0,03
03_atividade_principal_primeira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	0,03	-0,01	0,00	0,00	0,01	0,00	-0,02	0,00	0,00	0,01	0,00	0,03	-0,01	-0,01	0,01
03_atividade_principal_primeira_Indústrias de transformação	-0,01	-0,03	-0,02	0,01	-0,01	-0,02	-0,04	-0,02	-0,04	0,01	-0,03	-0,02	-0,02	-0,01	0,09
03_atividade_principal_primeira_Indústrias extractivas	0,01	-0,01	0,04	-0,01	-0,02	0,06	-0,02	-0,02	-0,01	-0,03	0,12	0,01	-0,01	-0,01	-0,03
03_atividade_principal_primeira_Pecuária, inclusive apoio à pecuária	0,12	-0,03	-0,04	0,02	-0,02	0,03	-0,03	0,01	-0,03	0,01	-0,02	0,12	-0,01	0,03	
03_atividade_principal_primeira_Produção florestal, pesca e aquicultura	-0,02	-0,02	0,03	0,11	0,00	0,00	-0,02	-0,01	-0,02	-0,01	0,03	0,04	0,00	-0,02	
03_atividade_principal_segunda_Administração, defesa, educação e saúde públicas e segurança social	0,05	-0,07	0,05	0,03	0,00	-0,01	-0,09	-0,04	-0,09	0,05	0,01	-0,06	0,03	-0,03	0,07
03_atividade_principal_segunda_Agricultura, inclusive apoio à agricultura e a pós colheita	0,01	-0,02	-0,06	0,02	0,01	0,05	-0,03	-0,03	-0,03	0,07	-0,03	-0,04	-0,01	0,02	0,05
03_atividade_principal_segunda_Comércio e reparação de veículos automotores e motocicletas	-0,01	-0,02	-0,04	-0,01	0,06	0,00	-0,03	-0,02	-0,04	0,03	-0,02	-0,03	-0,02	-0,01	0,06
03_atividade_principal_segunda_Construção	-0,01	0,01	-0,01	-0,01	-0,01	0,02	0,01	-0,01	-0,01	-0,01	-0,01	0,07	0,00	0,00	-0,01
03_atividade_principal_segunda_Demais serviços	-0,07	0,11	0,06	-0,04	-0,04	-0,02	0,15	0,07	0,15	-0,10	0,01	0,09	-0,07	0,01	-0,11
03_atividade_principal_segunda_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	0,03	-0,01	0,00	-0,01	0,04	-0,01	-0,01	0,02	0,00	-0,02	0,01	0,01	0,02	0,00	-0,02
03_atividade_principal_segunda_Indústrias de transformação	0,00	-0,05	-0,05	-0,01	-0,02	-0,03	-0,05	-0,02	-0,05	0,04	0,02	-0,03	-0,02	-0,01	0,00
03_atividade_principal_segunda_Indústrias extractivas	-0,02	0,00	0,01	-0,01	-0,01	0,01	0,00	-0,01	-0,01	-0,02	0,05	0,06	-0,01	0,00	-0,02
03_atividade_principal_segunda_Pecuária, inclusive apoio à pecuária	0,08	0,01	-0,08	0,05	0,06	0,01	-0,04	0,00	-0,03	-0,02	-0,03	-0,03	0,20	0,01	0,00
03_atividade_principal_segunda_Produção florestal, pesca e aquicultura	-0,02	-0,02	0,01	0,02	-0,01	0,09	0,00	-0,02	-0,02	-0,02	-0,01	0,03	-0,01	0,03	0,01
03_atividade_principal_terceira_Administração, defesa, educação e saúde públicas e segurança social	0,02	-0,08	-0,06	0,01	0,00	-0,05	-0,09	-0,07	-0,08	0,15	0,01	-0,07	-0,03	-0,03	0,12
03_atividade_principal_terceira_Agricultura, inclusive apoio à agricultura e a pós colheita	-0,04	0,02	0,01	-0,01	-0,03	0,02	-0,03	0,00	0,04	0,00	-0,03	-0,04	-0,03	-0,02	-0,04
03_atividade_principal_terceira_Comércio e reparação de veículos automotores e motocicletas	-0,05	0,04	0,00	-0,02	-0,01	-0,05	0,08	0,07	0,05	-0,05	0,05	0,02	-0,01	0,01	-0,02
03_atividade_principal_terceira_Construção	-0,01	-0,02	0,04	-0,01	-0,02	0,00	-0,01	-0,01	0,02	-0,03	0,05	0,01	-0,01	-0,01	-0,02
03_atividade_principal_terceira_Demais serviços	0,03	-0,01	-0,11	0,05	0,10	0,10	-0,06	0,00	-0,05	0,02	-0,04	-0,02	0,10	0,02	0,08
03_atividade_principal_terceira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	0,00	-0,02	0,01	0,00	0,00	-0,02	-0,01	0,00	-0,01	-0,03	0,02	0,13	0,01	0,00	-0,02
03_atividade_principal_terceira_Indústrias de transformação	0,02	-0,05	0,07	0,00	-0,02	-0,03	-0,02	0,01	-0,05	-0,01	0,03	0,02	-0,02	-0,01	-0,03
03_atividade_principal_terceira_Indústrias extractivas	-0,01	-0,01	0,06	-0,01	-0,02	-0,01	0,00	-0,01	0,01	-0,02	0,03	0,03	-0,01	-0,01	-0,03
03_atividade_principal_terceira_Pecuária, inclusive apoio à pecuária	0,06	0,04	0,05	-0,01	-0,02	-0,02	0,12	0,01	0,06	-0,07	-0,04	0,04	0,01	0,02	-0,09
03_atividade_principal_terceira_Produção florestal, pesca e aquicultura	-0,04	0,08	0,08	-0,02	-0,03	0,06	-0,01	-0,02	0,00	-0,02	-0,02	0,02	-0,02	-0,02	0,03

## Apêndice IV – Matriz de correlações

	03_nf_SC	03_nf_SE	03_nf_SP	03_nf_TO	03_atividade_principal_primeira_Administração, defesa, educação e saúde públicas e segurança social	03_atividade_principal_primeira_Agriculura, inclusive apoio à agricultura e a pós colheita	03_atividade_principal_primeira_Comércio e reparação de veículos automotores e motocicletas	03_atividade_principal_primeira_Construção	03_atividade_principal_primeira_Demais serviços	03_atividade_principal_primeira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação
01_arrecadacao_2018	0,00	0,00	0,05	-0,01	-0,04	-0,02	0,00	0,00	0,06	-0,01
02_unidades	0,00	-0,01	0,06	-0,01	-0,09	-0,03	0,00	0,00	0,13	-0,01
02_unidades_atuantes_%	0,05	-0,01	-0,38	0,04	0,22	-0,01	-0,03	0,01	-0,20	-0,01
02_ocupados_total	0,00	0,00	0,05	-0,01	-0,08	-0,03	0,00	0,00	0,11	-0,01
02_ocupados_assalariados_%	-0,08	0,10	-0,18	0,11	0,26	-0,16	0,02	0,03	-0,24	0,01
02_ocupados_assalariados_medio	0,00	0,00	0,05	-0,01	-0,08	-0,03	0,00	0,00	0,11	-0,01
02_salario_medio_mensal_em_SM	0,11	0,05	0,28	-0,11	-0,46	0,12	0,07	0,04	0,23	0,10
02_salario_medio_mensal_em_Reais	0,11	0,05	0,28	-0,10	-0,46	0,12	0,07	0,04	0,23	0,10
03_vlr_adic_bruto_total	-0,01	-0,01	0,07	-0,01	-0,08	-0,03	0,01	0,00	0,10	0,00
03_vlr_adic_bruto_agropecuaria_%	0,04	-0,07	-0,08	0,06	-0,15	0,64	-0,05	-0,02	-0,25	-0,08
03_vlr_adic_bruto_industria_%	0,15	0,00	0,15	-0,07	-0,46	-0,16	-0,01	0,08	0,18	0,45
03_vlr_adic_bruto_servicos_%	0,05	-0,04	0,32	-0,13	-0,43	-0,19	0,21	-0,03	0,70	-0,17
03_vlr_adic_bruto_administracao_%	-0,19	0,09	-0,29	0,10	0,83	-0,29	-0,10	-0,02	-0,46	-0,16
03_impostos_sobre_produtos	0,00	-0,01	0,06	-0,01	-0,07	-0,02	0,02	0,00	0,09	-0,01
03_pib	0,00	-0,01	0,07	-0,01	-0,08	-0,03	0,01	0,00	0,10	0,00
03_pib_per_capita	0,10	-0,05	0,14	-0,03	-0,49	0,20	0,16	0,01	0,12	0,21
04_populacao_residente_2018	-0,01	0,00	0,05	-0,02	-0,09	-0,05	0,00	0,00	0,14	-0,01
05_total_residentes	-0,02	0,00	0,05	-0,02	-0,09	-0,04	0,00	0,00	0,14	-0,01
05_0-19_anos_%	-0,13	0,09	-0,29	0,15	0,46	-0,09	-0,02	0,01	-0,37	0,00
05_20-39_anos_%	-0,03	0,06	0,18	-0,03	-0,18	-0,14	0,05	0,03	0,23	0,00
05_40-59_anos_%	0,21	-0,10	0,20	-0,11	-0,48	0,17	0,02	-0,01	0,33	0,01
05_60+anos_%	0,02	-0,09	0,12	-0,11	-0,09	0,08	-0,02	-0,02	0,07	-0,01
06_total_instrucao	-0,01	0,00	0,05	-0,02	-0,09	-0,04	0,00	0,00	0,14	-0,01
06_Sem_Instrucao_e_Fundamental_Incompleto%	-0,15	0,07	-0,42	0,00	0,60	0,00	-0,06	-0,02	-0,56	-0,01
06_Medio_Incompleto_%	0,17	-0,08	0,27	-0,03	-0,50	0,09	0,05	0,01	0,38	0,02
06_Superior_Incompleto_%	0,10	-0,05	0,41	0,00	-0,52	-0,05	0,05	0,02	0,52	0,01
06_Superior_Completo_%	0,14	-0,06	0,37	0,01	-0,54	0,00	0,03	0,00	0,55	0,00
06_Indeterminado_%	-0,05	-0,02	0,03	0,01	-0,04	-0,01	0,04	0,01	0,03	0,00
07_total rendimentos	-0,01	0,00	0,05	-0,02	-0,09	-0,04	0,00	0,00	0,14	-0,01
07_Ate_1SM %	-0,25	0,10	-0,44	0,04	0,59	-0,07	-0,07	0,00	-0,44	-0,03
07_Ate_2SM %	0,30	-0,11	0,39	-0,09	-0,64	0,14	0,06	-0,01	0,45	0,01
07_Ate_3SM %	0,32	-0,10	0,41	-0,09	-0,64	0,11	0,07	0,00	0,47	0,03
07_Ate_5SM %	0,32	-0,09	0,38	-0,08	-0,63	0,08	0,07	-0,01	0,50	0,03
07_Ate_10SM %	0,24	-0,07	0,29	-0,04	-0,61	0,06	0,05	-0,01	0,52	0,03
07_Ate_20SM %	0,17	-0,06	0,21	-0,05	-0,51	0,05	0,04	0,00	0,45	0,02
07_20SM+ %	0,13	-0,06	0,16	-0,02	-0,42	0,04	0,05	-0,01	0,37	0,02
07_Sem_Rendimento_%	-0,26	0,06	-0,20	0,10	0,50	-0,13	-0,03	0,00	-0,37	0,00
03_nf_AC	-0,01	-0,01	-0,02	-0,01	0,05	-0,02	0,03	0,00	-0,03	-0,01
03_nf_AL	-0,03	-0,02	-0,05	-0,02	0,06	0,03	-0,01	0,00	-0,06	-0,02
03_nf_AM	-0,03	-0,01	-0,04	-0,02	0,07	-0,02	-0,01	0,00	-0,06	-0,02
03_nf_AP	-0,01	-0,01	-0,02	-0,01	0,05	-0,02	0,00	0,00	-0,03	0,02
03_nf_BA	-0,07	-0,03	-0,10	-0,05	0,14	-0,07	0,01	0,01	-0,08	-0,01
03_nf_CE	-0,04	-0,02	-0,07	-0,03	0,12	-0,05	-0,02	0,02	-0,07	0,00
03_nf_DF	0,00	0,00	0,00	0,00	0,01	0,00	0,00	0,00	-0,01	0,00
03_nf_ES	-0,03	-0,01	-0,04	-0,02	-0,01	-0,03	-0,01	0,00	0,03	-0,01
03_nf_GO	-0,05	-0,03	-0,08	-0,03	-0,08	0,02	0,00	0,02	0,04	0,03
03_nf_MA	-0,05	-0,02	-0,07	-0,03	0,15	-0,05	-0,01	-0,01	-0,10	-0,01
03_nf_MG	-0,10	-0,05	-0,15	-0,07	0,04	-0,10	-0,01	-0,01	0,04	0,00
03_nf_MS	-0,03	-0,01	-0,04	-0,02	-0,05	0,06	-0,01	0,00	-0,01	0,00
03_nf_MT	-0,04	-0,02	-0,06	-0,03	-0,03	0,13	0,02	-0,01	-0,05	0,01
03_nf_PA	-0,04	-0,02	-0,06	-0,03	0,07	-0,03	-0,01	-0,01	-0,06	0,00
03_nf_PB	-0,05	-0,02	-0,07	-0,03	0,18	-0,07	-0,01	-0,01	-0,11	-0,02
03_nf_PE	-0,04	-0,02	-0,07	-0,03	0,11	-0,06	-0,02	0,02	-0,07	0,00

## Apêndice IV – Matriz de correlações

	03_uf_SC	03_uf_SE	03_uf_SP	03_uf_TO	03_atividade_principal_primeira_Administração, defesa, educação e saúde públicas e segurança social	03_atividade_principal_primeira_Agricultura, inclusive apoio à agricultura e a pós colheita	03_atividade_principal_primeira_Comércio e reparação de veículos automotores e motocicletas	03_atividade_principal_primeira_Construção	03_atividade_principal_primeira_Demais serviços	03_atividade_principal_primeira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação
03_uf_PI	-0,05	-0,02	-0,07	-0,03	0,17	-0,04	-0,02	-0,01	-0,12	0,00
03_uf_PR	-0,07	-0,03	-0,10	-0,04	-0,20	0,18	0,00	-0,01	0,10	0,01
03_uf_RJ	-0,03	-0,02	-0,05	-0,02	-0,01	-0,05	-0,01	0,00	0,03	0,00
03_uf_RN	-0,04	-0,02	-0,06	-0,03	0,12	-0,06	-0,02	-0,01	-0,09	0,03
03_uf_RO	-0,02	-0,01	-0,04	-0,02	0,02	-0,03	-0,01	0,00	-0,02	-0,01
03_uf_RR	-0,01	-0,01	-0,02	-0,01	0,05	-0,02	0,00	0,00	-0,03	-0,01
03_uf_RS	-0,07	-0,04	-0,11	-0,05	-0,22	0,24	0,02	0,01	0,03	0,01
03_uf_SC	1,00	-0,03	-0,09	-0,04	-0,15	-0,02	0,01	-0,01	0,09	0,01
03_uf_SE	-0,03	1,00	-0,04	-0,02	0,08	-0,04	-0,01	0,00	-0,06	-0,01
03_uf_SP	-0,09	-0,04	1,00	-0,06	-0,29	-0,01	0,04	-0,01	0,32	-0,02
03_uf_TO	-0,04	-0,02	-0,06	1,00	0,08	-0,01	0,02	0,03	-0,09	0,00
03_atividade_principal_primeira_Administração, defesa, educação e saúde públicas e segurança social	-0,15	0,08	-0,29	0,08	1,00	-0,35	-0,09	-0,03	-0,62	-0,14
03_atividade_principal_primeira_Agricultura, inclusive apoio à agricultura e a pós colheita	-0,02	-0,04	-0,01	-0,01	-0,35	1,00	-0,03	-0,01	-0,22	-0,05
03_atividade_principal_primeira_Comércio e reparação de veículos automotores e motocicletas	0,01	-0,01	0,04	0,02	-0,09	-0,03	1,00	0,00	-0,06	-0,01
03_atividade_principal_primeira_Construção	-0,01	0,00	-0,01	0,03	-0,03	-0,01	0,00	1,00	-0,02	0,00
03_atividade_principal_primeira_Demais serviços	0,09	-0,06	0,32	-0,09	-0,62	-0,22	-0,06	-0,02	1,00	-0,09
03_atividade_principal_primeira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	0,01	-0,01	-0,02	0,00	-0,14	-0,05	-0,01	0,00	-0,09	1,00
03_atividade_principal_primeira_Indústrias de transformação	0,15	-0,01	0,05	-0,03	-0,22	-0,08	-0,02	-0,01	-0,14	-0,03
03_atividade_principal_primeira_Indústrias extractivas	-0,02	0,03	-0,02	-0,01	-0,11	-0,04	-0,01	0,00	-0,07	-0,02
03_atividade_principal_primeira_Pecuária, inclusive apoio à pecuária	0,06	-0,01	-0,04	0,01	-0,13	-0,05	-0,01	0,00	-0,08	-0,02
03_atividade_principal_primeira_Produção florestal, pesca e aquicultura	0,00	-0,01	-0,03	0,02	-0,08	-0,03	-0,01	0,00	-0,05	-0,01
03_atividade_principal_segunda_Administração, defesa, educação e saúde públicas e segurança social	0,01	-0,04	0,08	-0,01	-0,55	0,14	-0,04	0,03	0,48	0,07
03_atividade_principal_segunda_Agricultura, inclusive apoio à agricultura e a pós colheita	0,01	-0,01	0,03	0,04	0,03	-0,10	-0,01	-0,01	0,05	0,03
03_atividade_principal_segunda_Comércio e reparação de veículos automotores e motocicletas	0,02	-0,02	0,10	-0,01	-0,17	-0,02	-0,02	-0,01	0,25	-0,03
03_atividade_principal_segunda_Construção	0,01	0,00	-0,02	-0,01	0,03	-0,02	0,00	0,00	-0,03	0,02
03_atividade_principal_segunda_Demais serviços	-0,09	0,07	-0,19	-0,04	0,57	0,00	0,06	-0,01	-0,67	-0,07
03_atividade_principal_segunda_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	0,00	-0,01	-0,01	-0,01	0,01	-0,01	-0,01	0,00	0,00	-0,01
03_atividade_principal_segunda_Indústrias de transformação	0,10	-0,02	0,22	-0,04	-0,24	-0,07	-0,02	-0,01	0,37	-0,01
03_atividade_principal_segunda_Indústrias extractivas	0,01	0,04	-0,02	0,01	-0,02	-0,02	-0,01	0,00	0,03	0,05
03_atividade_principal_segunda_Pecuária, inclusive apoio à pecuária	0,05	-0,02	-0,06	0,15	0,13	-0,03	-0,01	-0,01	-0,09	-0,01
03_atividade_principal_segunda_Produção florestal, pesca e aquicultura	0,00	-0,01	-0,03	0,00	0,07	-0,03	-0,01	0,00	-0,05	0,04
03_atividade_principal_terceira_Administração, defesa, educação e saúde públicas e segurança social	0,12	-0,04	0,20	-0,06	-0,47	0,21	0,06	0,01	0,18	0,04
03_atividade_principal_terceira_Agricultura, inclusive apoio à agricultura e a pós colheita	-0,06	0,03	0,05	-0,03	0,15	-0,16	-0,01	-0,01	-0,01	-0,03
03_atividade_principal_terceira_Comércio e reparação de veículos automotores e motocicletas	-0,04	0,01	-0,03	-0,03	0,00	-0,11	-0,05	-0,02	0,17	-0,06
03_atividade_principal_terceira_Construção	-0,01	0,04	-0,02	0,00	0,03	-0,03	-0,01	0,00	0,00	0,00
03_atividade_principal_terceira_Demais serviços	0,02	-0,01	-0,11	0,10	0,01	0,26	0,00	0,05	-0,27	0,12
03_atividade_principal_terceira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	-0,02	0,02	-0,01	-0,02	0,03	-0,03	-0,01	0,00	-0,01	-0,01
03_atividade_principal_terceira_Indústrias de transformação	0,03	0,01	0,05	-0,03	-0,09	-0,06	0,06	-0,01	0,18	-0,01
03_atividade_principal_terceira_Indústrias extractivas	-0,02	0,03	-0,02	-0,02	0,02	-0,03	-0,01	0,00	0,01	0,00
03_atividade_principal_terceira_Pecuária, inclusive apoio à pecuária	-0,03	0,00	-0,13	0,07	0,33	-0,12	-0,03	-0,01	-0,19	-0,04
03_atividade_principal_terceira_Produção florestal, pesca e aquicultura	-0,03	-0,02	-0,05	-0,02	0,14	-0,06	-0,02	-0,01	-0,08	-0,03

## Apêndice IV – Matriz de correlações

	03_atividade_principal_primeira_Indústria_s de transformação	03_atividade_principal_primeira_Indústria_s extrativas	03_atividade_principal_primeira_Pecuária, inclusive apoio à pecuária	03_atividade_principal_primeira_Produção florestal, pesca e aquicultura	03_atividade_principal_segunda_Administração, defesa, educação e saúde públicas e segurança social	03_atividade_principal_segunda_Agricultura, inclusive apoio à agricultura e a pós colheita	03_atividade_principal_segunda_Comércio e reparação de veículos automotores e motocicletas	03_atividade_principal_segunda_Construção	03_atividade_principal_segunda_Demais serviços
01_arrecadacao_2018	0,00	0,00	-0,01	0,00	0,00	-0,01	0,11	0,00	-0,04
02_unidades	0,00	0,00	-0,01	-0,01	0,01	-0,02	0,17	0,00	-0,08
02_unidades_atuantes_%	-0,06	0,01	0,02	-0,02	-0,06	-0,07	-0,06	0,02	0,12
02_ocupados_total	0,00	0,00	-0,01	-0,01	0,01	-0,02	0,17	0,00	-0,07
02_ocupados_assalariados_%	0,08	0,07	-0,04	0,03	-0,19	-0,03	-0,03	0,04	0,16
02_ocupados_assalariados_medio	0,00	0,00	-0,01	-0,01	0,01	-0,02	0,17	0,00	-0,07
02_salario_medio_mensal_em_SM	0,23	0,17	0,01	-0,01	0,04	0,00	0,17	0,00	-0,21
02_salario_medio_mensal_em_Reais	0,23	0,17	0,01	-0,01	0,04	0,00	0,17	0,00	-0,21
03_vlr_adic_bruto_total	0,01	0,02	-0,01	-0,01	0,00	-0,02	0,15	0,00	-0,06
03_vlr_adic_bruto_agropecuaria_%	-0,15	-0,10	0,23	0,18	0,10	0,26	-0,11	-0,03	-0,17
03_vlr_adic_bruto_industria_%	0,51	0,31	-0,06	-0,03	0,02	-0,08	0,02	0,07	-0,14
03_vlr_adic_bruto_servicos_%	-0,03	-0,05	-0,12	-0,10	0,24	-0,11	0,37	-0,04	-0,29
03_vlr_adic_bruto_administracao_%	-0,26	-0,12	-0,06	-0,05	-0,28	-0,08	-0,20	0,00	0,48
03_impostos_sobre_produtos	0,02	0,00	-0,01	-0,01	0,00	-0,02	0,13	0,00	-0,05
03_pib	0,02	0,02	-0,01	-0,01	0,00	-0,02	0,15	0,00	-0,06
03_pib_per_capita	0,27	0,24	0,01	0,02	-0,01	0,01	0,14	0,03	-0,13
04_populacao_residente_2018	0,00	0,01	-0,02	-0,01	0,02	-0,03	0,18	-0,01	-0,08
05_total_residentes	0,00	0,00	-0,02	-0,01	0,02	-0,03	0,18	-0,01	-0,08
05_0-19_anos_%	-0,12	0,00	-0,07	0,05	-0,18	0,02	-0,11	0,01	0,24
05_20-39_anos_%	0,14	0,06	-0,06	0,00	0,02	-0,09	0,13	0,00	-0,09
05_40-59_anos_%	0,10	-0,01	0,10	-0,03	0,20	0,03	0,09	-0,02	-0,28
05_60+anos_%	-0,03	-0,05	0,05	-0,05	0,07	0,01	-0,02	0,00	-0,01
06_total_instrucao	0,00	0,00	-0,02	-0,01	0,02	-0,03	0,18	-0,01	-0,08
06_Sem_Instrucao_e_Fundamental_Incompleto%	-0,17	-0,06	0,01	0,02	-0,19	0,03	-0,26	0,02	0,37
06_Medio_Incompleto_%	0,18	0,04	0,02	-0,01	0,17	0,00	0,17	-0,02	-0,30
06_Superior_Incompleto_%	0,16	0,07	-0,03	-0,03	0,16	-0,05	0,23	-0,01	-0,31
06_Superior_Completo_%	0,08	0,03	-0,01	-0,01	0,17	0,00	0,28	-0,02	-0,37
06_Indeterminado_%	0,00	0,02	0,00	0,01	0,00	0,00	0,02	0,00	-0,02
07_total_rendimentos	0,00	0,00	-0,02	-0,01	0,02	-0,03	0,18	-0,01	-0,08
07_Ate_1SM %	-0,24	-0,05	-0,03	0,01	-0,13	-0,01	-0,21	0,03	0,36
07_Ate_2SM %	0,26	0,00	0,06	-0,01	0,20	0,03	0,16	-0,02	-0,37
07_Ate_3SM %	0,26	0,02	0,06	-0,02	0,17	0,01	0,19	-0,02	-0,37
07_Ate_5SM %	0,21	0,03	0,06	-0,03	0,17	0,01	0,23	-0,02	-0,39
07_Ate_10SM %	0,16	0,03	0,03	-0,02	0,17	0,00	0,27	-0,02	-0,39
07_Ate_20SM %	0,09	0,05	0,04	-0,02	0,14	-0,01	0,25	-0,02	-0,33
07_20SM+ %	0,06	0,03	0,05	-0,02	0,11	-0,01	0,23	-0,02	-0,27
07_Sem_Rendimento_%	-0,17	0,02	-0,07	0,02	-0,20	-0,02	-0,11	0,01	0,28
03_nf_AC	-0,01	-0,01	-0,01	0,00	-0,02	0,00	-0,01	0,00	0,00
03_nf_AL	-0,03	-0,01	-0,02	0,02	-0,01	0,07	-0,03	-0,01	0,00
03_nf_AM	-0,02	0,02	-0,01	0,06	-0,03	0,04	-0,02	0,00	0,01
03_nf_AP	-0,01	-0,01	-0,01	0,00	-0,02	-0,02	-0,01	0,00	0,02
03_nf_BA	-0,03	0,01	-0,03	-0,02	-0,04	-0,05	-0,03	0,02	0,11
03_nf_CE	-0,02	-0,02	-0,02	-0,01	-0,05	-0,02	-0,02	-0,01	0,08
03_nf_DF	0,00	0,00	0,00	0,00	-0,01	0,00	0,00	0,00	0,01
03_nf_ES	-0,01	0,06	0,00	-0,01	0,02	0,00	0,00	-0,01	-0,01
03_nf_GO	-0,01	0,01	0,12	-0,02	0,05	0,01	-0,01	-0,01	-0,07
03_nf_MA	-0,03	-0,01	-0,03	-0,02	-0,07	-0,02	-0,02	0,01	0,11
03_nf_MG	-0,02	0,04	-0,04	0,03	0,05	-0,06	-0,04	-0,01	0,06
03_nf_MS	0,01	-0,01	0,02	0,11	0,03	0,02	-0,01	-0,01	-0,04
03_nf_MT	-0,01	-0,02	-0,02	0,00	0,00	0,01	0,06	-0,01	-0,04
03_nf_PA	-0,02	0,06	0,03	0,00	-0,01	0,05	0,00	0,02	-0,02
03_nf_PB	-0,04	-0,02	-0,03	-0,02	-0,09	-0,03	-0,03	0,01	0,15
03_nf_PE	-0,02	-0,02	0,01	-0,01	-0,04	-0,03	-0,02	-0,01	0,07

## **Apêndice IV – Matriz de correlações**

	03_atividade_principal_primeira_Indústrias de transformação	03_atividade_principal_primeira_Indústrias extrativas	03_atividade_principal_primeira_Pecuária, inclusive apoio à pecuária	03_atividade_principal_primeira_Produção florestal, pesca e aquicultura	03_atividade_principal_segunda_Administração, defesa, educação e saúde públicas e segurança social	03_atividade_principal_segunda_Agricultura, inclusive apoio à agricultura e a pós colheita	03_atividade_principal_segunda_Construção	03_atividade_principal_segunda_Demais serviços	03_atividade_principal_segunda_Demais serviços
03_nf_PI	-0,04	-0,01	-0,03	-0,02	-0,09	-0,03	-0,04	-0,01	0,15
03_nf_PR	0,01	-0,03	0,01	-0,01	0,05	0,07	0,03	-0,01	-0,10
03_nf_RJ	-0,03	0,12	-0,02	-0,01	0,01	-0,03	-0,02	-0,01	0,01
03_nf_RN	-0,02	0,01	-0,02	0,03	-0,06	-0,04	-0,03	0,07	0,09
03_nf_RO	-0,02	-0,01	0,12	0,04	0,03	-0,01	-0,02	0,00	-0,07
03_nf_RR	-0,01	-0,01	-0,01	0,00	-0,03	0,02	-0,01	0,00	0,01
03_nf_RS	0,09	-0,03	0,03	-0,02	0,07	0,05	0,06	-0,01	-0,11
03_nf_SC	0,15	-0,02	0,06	0,00	0,01	0,01	0,02	0,01	-0,09
03_nf_SE	-0,01	0,03	-0,01	-0,01	-0,04	-0,01	-0,02	0,00	0,07
03_nf_SP	0,05	-0,02	-0,04	-0,03	0,08	0,03	0,10	-0,02	-0,19
03_nf_TO	-0,03	-0,01	0,01	0,02	-0,01	0,04	-0,01	-0,01	-0,04
03_atividade_principal_primeira_Administração, defesa, educação e saúde públicas e segurança social	-0,22	-0,11	-0,13	-0,08	-0,55	0,03	-0,17	0,03	0,57
03_atividade_principal_primeira_Agricultura, inclusive apoio à agricultura e a pós colheita	-0,08	-0,04	-0,05	-0,03	0,14	-0,10	-0,02	-0,02	0,00
03_atividade_principal_primeira_Comércio e reparação de veículos automotores e motocicletas	-0,02	-0,01	-0,01	-0,01	-0,04	-0,01	-0,02	0,00	0,06
03_atividade_principal_primeira_Construção	-0,01	0,00	0,00	0,00	0,03	-0,01	-0,01	0,00	-0,01
03_atividade_principal_primeira_Demais serviços	-0,14	-0,07	-0,08	-0,05	0,48	0,05	0,25	-0,03	-0,67
03_atividade_principal_primeira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	-0,03	-0,02	-0,02	-0,01	0,07	0,03	-0,03	0,02	-0,07
03_atividade_principal_primeira_Indústrias de transformação	1,00	-0,02	-0,03	-0,02	-0,09	-0,04	-0,03	0,01	0,16
03_atividade_principal_primeira_Indústrias extractivas	-0,02	1,00	-0,01	-0,01	-0,02	-0,03	-0,02	0,00	0,07
03_atividade_principal_primeira_Pecuária, inclusive apoio à pecuária	-0,03	-0,01	1,00	-0,01	0,15	0,00	-0,02	-0,01	-0,10
03_atividade_principal_primeira_Produção florestal, pesca e aquicultura	-0,02	-0,01	-0,01	1,00	0,06	0,01	-0,01	0,00	-0,06
03_atividade_principal_segunda_Administração, defesa, educação e saúde públicas e segurança social	-0,09	-0,02	0,15	0,06	1,00	-0,16	-0,11	-0,02	-0,59
03_atividade_principal_segunda_Agricultura, inclusive apoio à agricultura e a pós colheita	-0,04	-0,03	0,00	0,01	-0,16	1,00	-0,06	-0,01	-0,31
03_atividade_principal_segunda_Comércio e reparação de veículos automotores e motocicletas	-0,03	-0,02	-0,02	-0,01	-0,11	-0,06	1,00	-0,01	-0,20
03_atividade_principal_segunda_Construção	0,01	0,00	-0,01	0,00	-0,02	-0,01	-0,01	1,00	-0,05
03_atividade_principal_segunda_Demais serviços	0,16	0,07	-0,10	-0,06	-0,59	-0,31	-0,20	-0,05	1,00
03_atividade_principal_segunda_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	-0,01	-0,01	0,01	0,00	-0,04	-0,02	-0,01	0,00	-0,07
03_atividade_principal_segunda_Indústrias de transformação	-0,06	-0,03	-0,02	-0,01	-0,15	-0,08	-0,05	-0,01	-0,28
03_atividade_principal_segunda_Indústrias extractivas	-0,02	-0,01	-0,01	0,03	-0,04	-0,02	-0,01	0,00	-0,07
03_atividade_principal_segunda_Pecuária, inclusive apoio à pecuária	-0,03	-0,02	-0,02	0,05	-0,11	-0,06	-0,04	-0,01	-0,21
03_atividade_principal_segunda_Produção florestal, pesca e aquicultura	-0,02	-0,01	-0,01	-0,01	-0,05	-0,03	-0,02	0,00	-0,10
03_atividade_principal_terceira_Administração, defesa, educação e saúde públicas e segurança social	0,28	0,16	0,00	0,03	-0,27	0,08	0,21	-0,01	-0,02
03_atividade_principal_terceira_Agricultura, inclusive apoio à agricultura e a pós colheita	-0,05	-0,04	-0,01	-0,01	0,02	-0,13	-0,05	-0,01	0,13
03_atividade_principal_terceira_Comércio e reparação de veículos automotores e motocicletas	-0,07	-0,06	-0,07	-0,04	0,13	-0,11	-0,10	-0,02	0,03
03_atividade_principal_terceira_Construção	-0,01	0,00	-0,01	0,01	0,02	-0,03	-0,01	0,00	0,03
03_atividade_principal_terceira_Demais serviços	-0,03	-0,01	0,17	0,06	0,19	0,30	-0,02	0,07	-0,46
03_atividade_principal_terceira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	0,01	-0,01	-0,01	0,02	-0,01	-0,02	-0,02	0,00	0,05
03_atividade_principal_terceira_Indústrias de transformação	-0,06	-0,02	-0,02	-0,01	0,14	-0,05	0,10	-0,01	-0,07
03_atividade_principal_terceira_Indústrias extractivas	0,00	-0,01	-0,01	-0,01	0,03	-0,03	-0,02	0,00	0,03
03_atividade_principal_terceira_Pecuária, inclusive apoio à pecuária	-0,08	-0,04	-0,05	-0,03	-0,15	-0,07	-0,08	-0,02	0,29
03_atividade_principal_terceira_Produção florestal, pesca e aquicultura	-0,03	-0,01	-0,02	-0,01	-0,06	-0,03	-0,03	-0,01	0,12

## Apêndice IV – Matriz de correlações

	03_atividade_principal_segunda_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	03_atividade_principal_segunda_Indústrias de transformação	03_atividade_principal_segunda_Indústrias extractivas	03_atividade_principal_segunda_Pecuária, inclusive apoio à pecuária	03_atividade_principal_segunda_Produção florestal, pesca e aquicultura	03_atividade_principal_terceira_Administração, defesa, educação e saúde públicas e segurança social	03_atividade_principal_terceira_Agricultura, inclusive apoio à agricultura e a pós colheita	03_atividade_principal_terceira_Agronegócio e reparação de veículos automotores e motocicletas	03_atividade_principal_terceira_Comércio e Construção
01_arrecadacao_2018	0,00	0,01	0,01	-0,01	0,00	0,03	-0,02	0,03	0,00
02_unidades	0,00	0,05	0,01	-0,02	-0,01	0,07	-0,04	0,04	0,00
02_unidades_atuantes_%	0,00	-0,09	0,01	0,06	0,02	-0,13	-0,01	-0,06	0,02
02_ocupados_total	0,00	0,04	0,01	-0,02	-0,01	0,06	-0,04	0,04	0,00
02_ocupados_assalariados_%	0,01	0,02	0,03	0,01	0,04	-0,13	-0,03	-0,06	0,01
02_ocupados_assalariados_medio	0,00	0,04	0,01	-0,02	-0,01	0,06	-0,04	0,04	0,00
02_salario_medio_mensal_em_SM	0,01	0,25	0,04	-0,03	-0,04	0,31	-0,10	-0,02	-0,02
02_salario_medio_mensal_em_Reais	0,01	0,25	0,04	-0,03	-0,04	0,31	-0,11	-0,01	-0,02
03_vlr_adic_bruto_total	0,00	0,05	0,01	-0,02	-0,01	0,07	-0,04	0,04	-0,01
03_vlr_adic_bruto_agropecuaria_%	-0,02	-0,17	-0,05	0,20	0,09	0,06	0,06	-0,33	-0,08
03_vlr_adic_bruto_industria_%	0,09	0,34	0,12	-0,09	-0,03	0,37	-0,15	-0,10	0,03
03_vlr_adic_bruto_servicos_%	-0,04	0,21	-0,01	-0,17	-0,11	0,17	-0,09	0,41	0,01
03_vlr_adic_bruto_administracao_%	-0,02	-0,28	-0,04	0,03	0,03	-0,48	0,13	0,05	0,04
03_impostos_sobre_produtos	0,00	0,05	0,01	-0,01	-0,01	0,05	-0,03	0,04	-0,01
03_pib	0,00	0,05	0,01	-0,02	-0,01	0,06	-0,04	0,04	-0,01
03_pib_per_capita	0,01	0,17	0,02	-0,03	0,00	0,34	-0,12	-0,05	-0,02
04_populacao_residente_2018	0,00	0,06	0,00	-0,03	-0,01	0,06	-0,04	0,07	-0,01
05_total_residentes	0,00	0,05	0,00	-0,03	-0,01	0,06	-0,04	0,07	-0,01
05_0-19_anos_%	0,01	-0,17	0,00	0,03	0,09	-0,26	0,05	-0,01	0,01
05_20-39_anos_%	0,03	0,21	0,03	-0,09	-0,01	0,12	-0,06	0,12	0,01
05_40-59_anos_%	-0,02	0,14	0,00	0,03	-0,07	0,27	-0,05	-0,05	-0,02
05_60+anos_%	-0,03	-0,04	-0,02	-0,02	-0,07	0,05	0,03	-0,03	0,00
06_total_instrucao	0,00	0,05	0,00	-0,02	-0,01	0,06	-0,04	0,07	-0,01
06_Sem_Instrucao_e_Fundamental_Incompleto%	0,00	-0,33	-0,03	0,07	0,05	-0,35	0,13	-0,11	0,01
06_Medio_Incompleto_%	0,01	0,22	0,00	-0,02	-0,01	0,29	-0,10	0,04	-0,02
06_Superior_Incompleto_%	0,00	0,31	0,03	-0,08	-0,06	0,31	-0,12	0,12	-0,01
06_Superior_Completo_%	0,00	0,30	0,03	-0,06	-0,05	0,30	-0,10	0,11	-0,01
06_Indeterminado_%	0,00	0,03	0,00	0,01	-0,02	0,02	-0,03	0,00	-0,01
07_total rendimentos	0,00	0,05	0,00	-0,02	-0,01	0,06	-0,04	0,07	-0,01
07_Ate_1SM %	-0,02	-0,35	-0,02	0,01	0,02	-0,38	0,11	0,01	0,04
07_Ate_2SM %	0,00	0,28	-0,02	-0,01	-0,05	0,41	-0,08	-0,06	-0,03
07_Ate_3SM %	0,00	0,33	0,00	-0,01	-0,05	0,40	-0,09	-0,01	-0,03
07_Ate_5SM %	0,01	0,34	0,01	0,00	-0,04	0,37	-0,11	0,03	-0,02
07_Ate_10SM %	0,01	0,31	0,03	0,00	-0,05	0,34	-0,14	0,09	-0,02
07_Ate_20SM %	0,00	0,26	0,06	-0,02	-0,04	0,27	-0,12	0,11	-0,02
07_20SM+ %	0,00	0,21	0,04	-0,01	-0,04	0,21	-0,09	0,09	-0,02
07_Sem_Rendimento_%	0,02	-0,17	0,02	0,01	0,06	-0,29	0,05	0,02	0,01
03_nf_AC	0,00	-0,02	0,00	0,08	-0,01	-0,03	0,03	-0,02	-0,01
03_nf_AL	-0,01	-0,03	-0,01	-0,03	0,03	-0,04	0,03	-0,02	0,00
03_nf_AM	-0,01	-0,03	-0,01	-0,01	0,10	-0,04	0,04	-0,05	-0,01
03_nf_AP	0,05	0,00	0,00	0,01	0,03	-0,03	-0,01	0,00	0,02
03_nf_BA	-0,01	-0,06	0,00	-0,01	-0,01	-0,08	0,00	0,06	0,02
03_nf_CE	0,02	-0,03	0,00	-0,03	-0,02	-0,07	0,08	-0,01	0,02
03_nf_DF	0,00	0,00	0,00	0,00	0,00	-0,01	-0,01	0,03	0,00
03_nf_ES	-0,01	-0,01	0,01	-0,02	-0,01	0,00	0,02	0,02	0,00
03_nf_GO	0,03	0,00	-0,02	0,08	-0,02	0,02	-0,04	-0,05	-0,01
03_nf_MA	-0,01	-0,05	0,00	0,01	-0,02	-0,08	0,02	0,04	-0,02
03_nf_MG	0,00	-0,05	0,01	-0,08	0,01	-0,06	0,01	0,00	0,04
03_nf_MS	-0,01	-0,01	-0,01	0,05	0,02	0,01	-0,01	-0,02	-0,01
03_nf_MT	0,04	-0,02	-0,01	0,06	-0,01	0,00	-0,03	-0,01	-0,02
03_nf_PA	-0,01	-0,03	0,01	0,01	0,09	-0,05	0,02	-0,05	0,00
03_nf_PB	-0,01	-0,05	0,00	-0,04	0,00	-0,09	-0,03	0,08	-0,01
03_nf_PE	0,02	-0,02	-0,01	0,00	-0,02	-0,07	0,00	0,07	-0,01

## Apêndice IV – Matriz de correlações

	03_atividade_principal_segunda_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	03_atividade_principal_segunda_Indústrias de transformação	03_atividade_principal_segunda_Indústrias extractivas	03_atividade_principal_segunda_Pecuária, inclusive apoio à pecuária	03_atividade_principal_segunda_Produção florestal, pesca e aquicultura	03_atividade_principal_terceira_Administração, defesa, educação e saúde públicas e segurança social	03_atividade_principal_terceira_Agricultura, inclusive apoio à agricultura e a pós colheita	03_atividade_principal_terceira_Comércio e reparação de veículos automotores e motocicletas	03_atividade_principal_terceira_Construção	03_atividade_principal_terceira_Demais serviços
03_nf_PI	0,00	-0,05	-0,01	-0,03	-0,02	-0,08	0,04	0,05	0,02	
03_nf_PR	-0,02	0,04	-0,02	-0,02	-0,02	0,15	0,00	-0,05	-0,03	
03_nf_RJ	0,01	0,02	0,05	-0,03	-0,01	0,01	-0,03	0,05	0,05	
03_nf_RN	0,01	-0,03	0,06	-0,03	0,03	-0,07	-0,04	0,02	0,01	
03_nf_RO	0,02	-0,02	-0,01	0,20	-0,01	-0,03	-0,03	-0,03	-0,01	-0,01
03_nf_RR	0,00	-0,01	0,00	0,01	0,03	-0,03	-0,02	0,01	-0,01	-0,01
03_nf_RS	-0,02	0,00	-0,02	0,00	0,01	0,12	-0,04	-0,02	-0,02	-0,02
03_nf_SC	0,00	0,10	0,01	0,05	0,00	0,12	-0,06	-0,04	-0,01	-0,01
03_nf_SE	-0,01	-0,02	0,04	-0,02	-0,01	-0,04	0,03	0,01	0,04	
03_nf_SP	-0,01	0,22	-0,02	-0,06	-0,03	0,20	0,05	-0,03	-0,02	
03_nf_TO	-0,01	-0,04	0,01	0,15	0,00	-0,06	-0,03	-0,03	0,00	
03_atividade_principal_primeira_Administração, defesa, educação e saúde públicas e segurança social	0,01	-0,24	-0,02	0,13	0,07	-0,47	0,15	0,00	0,03	
03_atividade_principal_primeira_Agricultura, inclusive apoio à agricultura e a pós colheita	-0,01	-0,07	-0,02	-0,03	-0,03	0,21	-0,16	-0,11	-0,03	
03_atividade_principal_primeira_Comércio e reparação de veículos automotores e motocicletas	-0,01	-0,02	-0,01	-0,01	-0,01	0,06	-0,01	-0,05	-0,01	
03_atividade_principal_primeira_Construção	0,00	-0,01	0,00	-0,01	0,00	0,01	-0,01	-0,02	0,00	
03_atividade_principal_primeira_Demais serviços	0,00	0,37	0,03	-0,09	-0,05	0,18	-0,01	0,17	0,00	
03_atividade_principal_primeira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	-0,01	-0,01	0,05	-0,01	0,04	0,04	-0,03	-0,06	0,00	
03_atividade_principal_primeira_Indústrias de transformação	-0,01	-0,06	-0,02	-0,03	-0,02	0,28	-0,05	-0,07	-0,01	
03_atividade_principal_primeira_Indústrias extractivas	-0,01	-0,03	-0,01	-0,02	-0,01	0,16	-0,04	-0,06	0,00	
03_atividade_principal_primeira_Pecuária, inclusive apoio à pecuária	0,01	-0,02	-0,01	-0,02	-0,01	0,00	-0,01	-0,07	-0,01	
03_atividade_principal_primeira_Produção florestal, pesca e aquicultura	0,00	-0,01	0,03	0,05	-0,01	0,03	-0,01	-0,04	0,01	
03_atividade_principal_segunda_Administração, defesa, educação e saúde públicas e segurança social	-0,04	-0,15	-0,04	-0,11	-0,05	-0,27	0,02	0,13	0,02	
03_atividade_principal_segunda_Agricultura, inclusive apoio à agricultura e a pós colheita	-0,02	-0,08	-0,02	-0,06	-0,03	0,08	-0,13	-0,11	-0,03	
03_atividade_principal_segunda_Comércio e reparação de veículos automotores e motocicletas	-0,01	-0,05	-0,01	-0,04	-0,02	0,21	-0,05	-0,10	-0,01	
03_atividade_principal_segunda_Construção	0,00	-0,01	0,00	-0,01	0,00	-0,01	-0,01	-0,02	0,00	
03_atividade_principal_segunda_Demais serviços	-0,07	-0,28	-0,07	-0,21	-0,10	-0,02	0,13	0,03	0,03	
03_atividade_principal_segunda_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	1,00	-0,02	0,00	-0,01	-0,01	0,00	-0,01	-0,03	-0,01	
03_atividade_principal_segunda_Indústrias de transformação	-0,02	1,00	-0,02	-0,05	-0,02	0,28	-0,07	0,02	-0,03	
03_atividade_principal_segunda_Indústrias extractivas	0,00	-0,02	1,00	-0,01	-0,01	0,05	-0,03	-0,03	0,02	
03_atividade_principal_segunda_Pecuária, inclusive apoio à pecuária	-0,01	-0,05	-0,01	1,00	-0,02	-0,03	-0,05	-0,10	-0,02	
03_atividade_principal_segunda_Produção florestal, pesca e aquicultura	-0,01	-0,02	-0,01	-0,02	1,00	-0,02	-0,02	-0,05	-0,01	
03_atividade_principal_terceira_Administração, defesa, educação e saúde públicas e segurança social	0,00	0,28	0,05	-0,03	-0,02	1,00	-0,21	-0,25	-0,05	
03_atividade_principal_terceira_Agricultura, inclusive apoio à agricultura e a pós colheita	-0,01	-0,07	-0,03	-0,05	-0,02	-0,21	1,00	-0,23	-0,05	
03_atividade_principal_terceira_Comércio e reparação de veículos automotores e motocicletas	-0,03	0,02	-0,03	-0,10	-0,05	-0,25	-0,23	1,00	-0,06	
03_atividade_principal_terceira_Construção	-0,01	-0,03	0,02	-0,02	-0,01	-0,05	-0,05	-0,06	1,00	
03_atividade_principal_terceira_Demais serviços	0,07	-0,06	0,05	0,34	0,17	-0,21	-0,19	-0,23	-0,05	
03_atividade_principal_terceira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	-0,01	-0,03	-0,01	-0,02	-0,01	-0,05	-0,04	-0,05	-0,01	
03_atividade_principal_terceira_Indústrias de transformação	0,02	-0,07	-0,01	-0,05	-0,02	-0,13	-0,12	-0,14	-0,03	
03_atividade_principal_terceira_Indústrias extractivas	-0,01	-0,03	-0,01	-0,02	-0,01	-0,05	-0,05	-0,06	-0,01	
03_atividade_principal_terceira_Pecuária, inclusive apoio à pecuária	-0,03	-0,10	-0,03	-0,08	-0,04	-0,20	-0,18	-0,22	-0,05	
03_atividade_principal_terceira_Produção florestal, pesca e aquicultura	-0,01	-0,05	-0,01	-0,02	-0,02	-0,09	-0,08	-0,09	-0,02	

## Apêndice IV – Matriz de correlações

	03_atividade_principal_terceira_Demais serviços	03_atividade_principal_terceira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	03_atividade_principal_terceira_Indústria	03_atividade_principal_terceira_Indústrias extrativas	03_atividade_principal_terceira_Pecuária, inclusive apoio à pecuária	03_atividade_principal_terceira_Produção florestal, pesca e aquicultura
01_arrecadacao_2018	-0,02	0,00	0,01	0,00	-0,02	-0,01
02_unidades	-0,04	0,00	0,01	-0,01	-0,04	-0,02
02_unidades_atuantes_%	0,07	0,00	-0,03	0,01	0,14	0,05
02_ocupados_total	-0,04	0,00	0,01	0,00	-0,04	-0,02
02_ocupados_assalariados_%	0,07	0,04	0,02	0,04	0,10	0,08
02_ocupados_assalariados_medio	-0,04	0,00	0,01	0,00	-0,04	-0,02
02_salario_medio_mensal_em_SM	0,00	-0,01	0,04	0,02	-0,20	-0,09
02_salario_medio_mensal_em_Reais	0,00	-0,01	0,04	0,02	-0,20	-0,09
03_vlr_adic_bruto_total	-0,04	0,00	0,01	0,00	-0,04	-0,02
03_vlr_adic_bruto_agropecuaria_%	0,44	-0,07	-0,15	-0,07	-0,05	0,01
03_vlr_adic_bruto_industria_%	-0,05	0,03	0,17	0,08	-0,21	-0,09
03_vlr_adic_bruto_servicos_%	-0,38	-0,03	0,14	-0,02	-0,22	-0,10
03_vlr_adic_bruto_administracao_%	-0,05	0,05	-0,12	0,01	0,38	0,15
03_impostos_sobre_produtos	-0,03	0,00	0,02	0,00	-0,03	-0,01
03_pib	-0,04	0,00	0,01	0,00	-0,04	-0,02
03_pib_per_capita	0,00	-0,01	0,06	-0,02	-0,18	-0,09
04_populacao_residente_2018	-0,05	0,00	0,01	-0,01	-0,05	-0,02
05_total_residentes	-0,05	0,00	0,01	-0,01	-0,05	-0,02
05_0-19_anos_%	0,13	0,01	-0,09	0,00	0,10	0,15
05_20-39_anos_%	-0,12	0,02	0,12	0,02	-0,16	-0,03
05_40-59_anos_%	-0,05	-0,02	0,06	-0,02	-0,09	-0,14
05_60+anos_%	-0,07	0,00	-0,01	-0,01	0,08	-0,07
06_total_instrucao	-0,05	0,00	0,01	-0,01	-0,05	-0,02
06_Sem_Instrucao_e_Fundamental_Incompleto%	0,16	-0,01	-0,14	0,01	0,25	0,12
06_Medio_Incompleto_%	-0,05	-0,01	0,10	-0,03	-0,22	-0,08
06_Superior_Incompleto_%	-0,19	0,03	0,13	0,00	-0,22	-0,11
06_Superior_Completo_%	-0,16	0,00	0,09	-0,02	-0,19	-0,10
06_Indeterminado_%	-0,01	0,01	0,04	0,01	-0,01	0,00
07_total_rendimentos	-0,05	0,00	0,01	-0,01	-0,05	-0,02
07_Ate_1SM %	0,06	0,02	-0,10	0,02	0,25	0,08
07_Ate_2SM %	-0,06	-0,03	0,12	-0,03	-0,24	-0,12
07_Ate_3SM %	-0,08	-0,02	0,11	-0,02	-0,24	-0,12
07_Ate_5SM %	-0,08	-0,02	0,10	-0,02	-0,24	-0,12
07_Ate_10SM %	-0,09	-0,01	0,09	-0,02	-0,23	-0,11
07_Ate_20SM %	-0,09	-0,01	0,07	-0,02	-0,19	-0,10
07_20SM+ %	-0,08	-0,01	0,04	-0,02	-0,15	-0,08
07_Sem_Rendimento_%	0,07	0,02	-0,08	0,04	0,15	0,13
03_nf_AC	0,04	-0,01	-0,01	-0,01	0,01	-0,01
03_nf_AL	0,07	0,00	-0,03	0,01	-0,02	0,00
03_nf_AM	0,06	-0,01	-0,03	0,00	-0,04	0,14
03_nf_AP	0,02	0,03	-0,01	-0,01	-0,02	0,05
03_nf_BA	-0,04	0,00	-0,03	0,01	0,08	-0,01
03_nf_CE	-0,02	0,00	0,00	-0,02	0,01	0,02
03_nf_DF	-0,01	0,00	0,00	0,00	-0,01	0,00
03_nf_ES	-0,02	0,02	0,02	0,03	-0,04	-0,02
03_nf_GO	0,03	0,00	0,02	-0,01	0,06	-0,04
03_nf_MA	-0,01	-0,02	-0,05	-0,01	0,04	0,08
03_nf_MG	-0,11	0,01	0,07	0,06	0,05	0,08
03_nf_MS	0,05	0,00	0,00	-0,01	-0,01	-0,02
03_nf_MT	0,10	0,00	-0,02	-0,02	-0,02	-0,03
03_nf_PA	0,10	-0,02	-0,03	-0,01	-0,02	0,06
03_nf_PB	-0,06	-0,01	-0,02	0,00	0,12	-0,01
03_nf_PE	0,00	0,00	0,01	-0,01	0,01	-0,02

## Apêndice IV – Matriz de correlações

	03_atividade_principal_terceira_Demais serviços	03_atividade_principal_eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	03_atividade_principal_terceira_Indústria de transformação	03_atividade_principal_terceira_Indústrias extrativas	03_atividade_principal_terceira_Pecuária, inclusive apoio à pecuária	03_atividade_principal_terceira_Produção florestal, pesca e aquicultura
03_nf_PI	-0,05	-0,01	-0,05	0,01	0,06	0,00
03_nf_PR	0,02	-0,03	-0,01	-0,02	-0,07	-0,02
03_nf_RJ	-0,04	0,02	0,03	0,03	-0,04	-0,02
03_nf_RN	-0,02	0,13	0,02	0,03	0,04	0,02
03_nf_RO	0,10	0,01	-0,02	-0,01	0,01	-0,02
03_nf_RR	0,02	0,00	-0,01	-0,01	0,02	0,03
03_nf_RS	0,08	-0,02	-0,03	-0,03	-0,09	-0,04
03_nf_SC	0,02	-0,02	0,03	-0,02	-0,03	-0,03
03_nf_SE	-0,01	0,02	0,01	0,03	0,00	-0,02
03_nf_SP	-0,11	-0,01	0,05	-0,02	-0,13	-0,05
03_nf_TO	0,10	-0,02	-0,03	-0,02	0,07	-0,02
03_atividade_principal_primeira_Administração, defesa, educação e saúde públicas e segurança social	0,01	0,03	-0,09	0,02	0,33	0,14
03_atividade_principal_primeira_Agricultura, inclusive apoio à agricultura e a pós colheita	0,26	-0,03	-0,06	-0,03	-0,12	-0,06
03_atividade_principal_primeira_Comércio e reparação de veículos automotores e motocicletas	0,00	-0,01	0,06	-0,01	-0,03	-0,02
03_atividade_principal_primeira_Construção	0,05	0,00	-0,01	0,00	-0,01	-0,01
03_atividade_principal_primeira_Demais serviços	-0,27	-0,01	0,18	0,01	-0,19	-0,08
03_atividade_principal_primeira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	0,12	-0,01	-0,01	0,00	-0,04	-0,03
03_atividade_principal_primeira_Indústrias de transformação	-0,03	0,01	-0,06	0,00	-0,08	-0,03
03_atividade_principal_primeira_Indústrias extractivas	-0,01	-0,01	-0,02	-0,01	-0,04	-0,01
03_atividade_principal_primeira_Pecuária, inclusive apoio à pecuária	0,17	-0,01	-0,02	-0,01	-0,05	-0,02
03_atividade_principal_primeira_Produção florestal, pesca e aquicultura	0,06	0,02	-0,01	-0,01	-0,03	-0,01
03_atividade_principal_segunda_Administração, defesa, educação e saúde públicas e segurança social	0,19	-0,01	0,14	0,03	-0,15	-0,06
03_atividade_principal_segunda_Agricultura, inclusive apoio à agricultura e a pós colheita	0,30	-0,02	-0,05	-0,03	-0,07	-0,03
03_atividade_principal_segunda_Comércio e reparação de veículos automotores e motocicletas	-0,02	-0,02	0,10	-0,02	-0,08	-0,03
03_atividade_principal_segunda_Construção	0,07	0,00	-0,01	0,00	-0,02	-0,01
03_atividade_principal_segunda_Demais serviços	-0,46	0,05	-0,07	0,03	0,29	0,12
03_atividade_principal_segunda_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	0,07	-0,01	0,02	-0,01	-0,03	-0,01
03_atividade_principal_segunda_Indústrias de transformação	-0,06	-0,03	-0,07	-0,03	-0,10	-0,05
03_atividade_principal_segunda_Indústrias extractivas	0,05	-0,01	-0,01	-0,01	-0,03	-0,01
03_atividade_principal_segunda_Pecuária, inclusive apoio à pecuária	0,34	-0,02	-0,05	-0,02	-0,08	-0,02
03_atividade_principal_segunda_Produção florestal, pesca e aquicultura	0,17	-0,01	-0,02	-0,01	-0,04	-0,02
03_atividade_principal_terceira_Administração, defesa, educação e saúde públicas e segurança social	-0,21	-0,05	-0,13	-0,05	-0,20	-0,09
03_atividade_principal_terceira_Agricultura, inclusive apoio à agricultura e a pós colheita	-0,19	-0,04	-0,12	-0,05	-0,18	-0,08
03_atividade_principal_terceira_Comércio e reparação de veículos automotores e motocicletas	-0,23	-0,05	-0,14	-0,06	-0,22	-0,09
03_atividade_principal_terceira_Construção	-0,05	-0,01	-0,03	-0,01	-0,05	-0,02
03_atividade_principal_terceira_Demais serviços	1,00	-0,04	-0,12	-0,05	-0,18	-0,08
03_atividade_principal_terceira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	-0,04	1,00	-0,03	-0,01	-0,04	-0,02
03_atividade_principal_terceira_Indústrias de transformação	-0,12	-0,03	1,00	-0,03	-0,11	-0,05
03_atividade_principal_terceira_Indústrias extractivas	-0,05	-0,01	-0,03	1,00	-0,04	-0,02
03_atividade_principal_terceira_Pecuária, inclusive apoio à pecuária	-0,18	-0,04	-0,11	-0,04	1,00	-0,07
03_atividade_principal_terceira_Produção florestal, pesca e aquicultura	-0,08	-0,02	-0,05	-0,02	-0,07	1,00

## Apêndice V – Correlações entre as variáveis independentes e o *label*

Variáveis originais	Correlações com 01_arrecadacao_2018	Variáveis transformadas	Correlações com 01_arrecadacao_2018_In
01_arrecadacao_2018	1,00	01_arrecadacao_2018_In	1,00
03_pib	0,97	03_impostos_sobre_produtos_In	0,96
03_impostos_sobre_produtos	0,97	02_ocupados_total_In	0,95
03_vlr_adic_bruto_total	0,97	02_ocupados_assalariados_medio_In	0,95
02_ocupados_assalariados_medio	0,94	03_pib_In	0,95
02_ocupados_total	0,94	03_vlr_adic_bruto_total_In	0,94
02_unidades	0,93	02_unidades_In	0,93
07_total_rendimentos	0,92	07_total_rendimentos_In	0,82
06_total_instrução	0,92	06_total_instrução_In	0,82
05_total_residentes	0,92	04_populacao_residente_2018_In	0,81
04_populacao_residente_2018	0,92	05_total_residentes_In	0,80
03_uf_DF	0,31	06_Sem_Instrução_e_Fundamental_Incompleto%_In	0,73
07_20SM_ %	0,26	07_Ate_1SM_ %_In	0,65
07_Ate_20SM_ %	0,23	03_vlr_adic_bruto_administracao_%_In	0,64
06_Superior_Completo_ %	0,18	03_vlr_adic_bruto_industria_%_In	0,62
02_salario_medio_mensal_em_Reais	0,18	03_vlr_adic_bruto_servicos_%_In	0,62
02_salario_medio_mensal_em_SM	0,17	06_Superior_Incompleto_%_In	0,60
07_Ate_10SM_ %	0,16	06_Superior_Completo_%_In	0,57
06_Sem_Instrução_e_Fundamental_Incompleto%	0,12	07_Ate_5SM_%_In	0,57
03_vlr_adic_bruto_servicos_%	0,12	03_atividade_principal_primeira_Demais serviços	0,57
03_atividade_principal_segunda_Comércio_e_reparação_de_veículos	0,11	03_atividade_principal_primeira_Administração, defesa, educação e	0,56
automotores_e_motocicletas		saúde públicas e segurança social	
06_Superior_Incompleto_%	0,10	03_pib_per_capita_In	0,56
07_Ate_1SM_ %	0,10	02_salario_medio_mensal_em_Reais_In	0,54
07_Ate_5SM_ %	0,09	02_salario_medio_mensal_em_SM_In	0,54
03_pib_per_capita	0,07	07_Ate_3SM_%_In	0,54
05_20-39_anos_%	0,07	07_Ate_10SM_%_In	0,54
03_atividade_principal_primeira_Demais serviços	0,06	05_20-39_anos_%_In	0,50
07_Ate_3SM_%	0,06	06_Medio_Incompleto_%_In	0,48
03_vlr_adic_bruto_agropecuaria_%	0,06	07_Ate_25M_%_In	0,47
03_uf_RJ	0,05	07_20SM_ %_In	0,44
03_vlr_adic_bruto_administracao_%	0,05	07_Ate_20SM_%_In	0,40
02_unidades_atuantes_%	0,05	03_atividade_principal_segunda_Indústrias de transformação	0,37
03_uf_SP	0,05	03_vlr_adic_bruto_agropecuaria_%	0,34
05_0-19_anos %	0,04	03_atividade_principal_segunda_Demais serviços	0,34
06_Medio_Incompleto_%	0,04	03_atividade_principal_terceira_Administração, defesa, educação e	0,32
03_atividade_principal_primeira_Administração, defesa, educação e	0,04	saúde públicas e segurança social	
saúde públicas e segurança social		06_Indeterminado_%_In	0,30
03_atividade_principal_segunda_Demais serviços	0,04	03_atividade_principal_terceira_Pecuária, inclusive apoio à pecuária	0,30
03_atividade_principal_terceira_Administração, defesa, educação e		03_atividade_principal_segunda_Comércio_e_reparação_de_veículos	
saúde públicas e segurança social	0,03	automotores_e_motocicletas	0,29
06_Indeterminado_%	0,03	02_unidades_atuantes_%_In	0,26
07_Ate_25M_%	0,03	05_0-19_anos_%_In	0,25
03_atividade_principal_terceira_Comércio_e_reparação_de_veículos		07_Sem_Rendimento_%_In	0,25
automotores_e_motocicletas	0,03	03_uf_SP	0,23
05_40-59_anos %	0,03	03_atividade_principal_terceira_Comércio_e_reparação_de_veículos	0,21
03_atividade_principal_terceira_Demais serviços	0,02	automotores_e_motocicletas	
03_atividade_principal_terceira_Agricultura, inclusive apoio à agricultura		03_atividade_principal_terceira_Demais serviços	0,21
e a pós colheita	0,02	05_40-59_anos_%_In	0,21
03_atividade_principal_terceira_Pecuária, inclusive apoio à pecuária	0,02	03_atividade_principal_primeira_Indústrias de transformação	0,20
02_ocupados_assalariados %	0,02	05_60+_anos_%_In	0,19
03_atividade_principal_primeira_Agricultura, inclusive apoio à agricultura		03_atividade_principal_terceira_Indústrias de transformação	0,18
e a pós colheita	0,02	03_uf_PI	0,17
07_Sem_Rendimento_%	0,02	03_atividade_principal_terceira_Agricultura, inclusive apoio à agricultura	0,15
03_vlr_adic_bruto_industria_%	0,01	03_atividade_principal_segunda_Administração, defesa, educação e	
03_atividade_principal_segunda_Indústrias de transformação	0,01	saúde públicas e segurança social	0,15
03_atividade_principal_segunda_Agricultura, inclusive apoio à agricultura		03_uf_RJ	0,14
e a pós colheita	0,01	03_uf_PB	0,12
03_uf_MG	0,01	03_atividade_principal_segunda_Pecuária, inclusive apoio à pecuária	0,11
05_60+_anos %	0,01	03_uf_MA	0,11
03_uf_BA	0,01	03_uf_TO	0,11
03_atividade_principal_segunda_Pecuária, inclusive apoio à pecuária	0,01	03_uf_SC	0,10
03_uf_PI	0,01	03_atividade_principal_primeira_Agricultura, inclusive apoio à agricultura	0,10
03_uf_PB	0,01	03_atividade_principal_terceira_Produção florestal, pesca e aquicultura	0,10
03_atividade_principal_terceira_Produção florestal, pesca e aquicultura	0,01	03_atividade_principal_primeira_Indústrias extrativas	0,08
03_uf_MA	0,01	03_uf_RN	0,08
03_uf_RN	0,01	03_uf_DF	0,07
03_uf_TO	0,01	03_uf_AL	0,07
03_uf_GO	0,01	03_uf_ES	0,07
03_uf_RS	0,01	03_atividade_principal_primeira_Comércio_e_reparação_de_veículos	
03_atividade_principal_primeira_Pecuária, inclusive apoio à pecuária	0,01	automotores_e_motocicletas	0,07
atividades_de_gestão_de_resíduos_e_descontaminação		03_atividade_principal_segunda_Agricultura, inclusive apoio à agricultura	
03_atividade_principal_terceira_Indústrias de transformação	0,01	e a pós colheita	0,06
03_atividade_principal_segunda_Indústrias extrativas	0,01	03_atividade_principal_primeira_Pecuária, inclusive apoio à pecuária	0,06
03_uf_AL	0,01	03_uf_BA	0,06
03_uf_MT	0,01	03_uf_MS	0,06
03_uf_PA	0,01	03_uf_PR	0,06
03_uf_CE	0,00	03_atividade_principal_segunda_Produção florestal, pesca e aquicultura	0,05
03_atividade_principal_segunda_Produção florestal, pesca e aquicultura	0,00	02_ocupados_assalariados %_In	0,04
03_uf_SE	0,00	03_uf_MG	0,04
03_uf_PR	0,00	03_atividade_principal_segunda_Indústrias extrativas	0,04
03_atividade_principal_terceira_Construção	0,00	03_uf_MT	0,03
03_uf_PE	0,00	03_uf_AM	0,03
03_atividade_principal_primeira_Produção florestal, pesca e aquicultura	0,00	03_uf_RS	0,03

## Apêndice V – Correlações entre as variáveis independentes e o *label*

Variáveis originais	Correlações com 01_arrecadacao_2018	Variáveis transformadas	Correlações com 01_arrecadacao_2018_In
03_atividade_principal_terceira_Indústrias extrativas	0,00	03_atividade_principal_primeira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	0,03
03_uf_RO	0,00	03_uf_AP	0,03
03_uf_MS	0,00	03_uf_RR	0,02
03_atividade_principal_terceira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	0,00	03_atividade_principal_segunda_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	0,02
03_atividade_principal_primeira_Comércio e reparação de veículos automotores e motocicletas	0,00	03_atividade_principal_segunda_Construção	0,02
03_atividade_principal_segunda_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	0,00	03_atividade_principal_terceira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação	0,01
03_uf_AC	0,00	03_uf_GO	0,01
03_atividade_principal_primeira_Indústrias extrativas	0,00	03_uf_SE	0,01
03_uf_SC	0,00	03_atividade_principal_terceira_Construção	0,01
03_atividade_principal_segunda_Construção	0,00	03_uf_PA	0,01
03_uf_AP	0,00	03_atividade_principal_primeira_Produção florestal, pesca e aquicultura	0,01
03_uf_RR	0,00	03_atividade_principal_primeira_Construção	0,01
03_atividade_principal_primeira_Construção	0,00	03_uf_RO	0,01
03_uf_ES	0,00	03_uf_PE	0,00
03_atividade_principal_primeira_Indústrias de transformação	0,00	03_uf_CE	0,00
03_atividade_principal_segunda_Administração, defesa, educação e saúde públicas e segurança social	0,00	03_atividade_principal_terceira_Indústrias extrativas	0,00
03_uf_AM	0,00	03_uf_AC	0,00

# Apêndice VI – Ntbk 01. Coleta dos dados e transformações preliminares.ipynb.

Este notebook tem por objetivo a coleta e transformações preliminares nos *datasets* que serão utilizadas no projeto, tais como:

- Descompactação de arquivos;
- Criação da variável padronizada, chamada “município-uf”;
- Manutenção, apenas, das informações relativas ao ano de interesse (2018);
- Renomeação de variáveis;
- Criação, em alguns casos, de novas colunas derivadas (enriquecimento dos dados);
- Remoção de variáveis redundantes e com valores nulos ou constantes;
- Reordenação de colunas.

Está estruturado da seguinte maneira:

1. Ações Preliminares
2. Coleta das bases de dados
3. Resumo das bases de dados criadas

## 1. Ações Preliminares

### - Inicialização da plataforma

A célula a seguir inicializa a plataforma, carregando as bibliotecas que serão relevantes para o trabalho em seguida.

In [1]:

```
import csv
import sys
import pandas as pd

from zipfile import ZipFile

pd.set_option('max_columns', 200)
```

### - Declaração de funções que serão utilizadas no notebook.

In [2]:

```
## Função utilizada para visualizar a estrutura do arquivo '.csv' a ser importado.

import csv
import chardet

def ler_csv(nome_arquivo):

    with open(nome_arquivo, 'rb') as rawdata:
        result = chardet.detect(rawdata.read(100000))
        encoding_arquivo = result['encoding']

    with open(nome_arquivo, newline='', encoding = encoding_arquivo) as f:
        reader = csv.reader(f)
        try:
            print(reader.line_num)
            for row in reader:
                print(reader.line_num, row)
        except csv.Error as e:
            sys.exit('file {}, line {}: {}'.format(filename, reader.line_num, e))
    return encoding_arquivo
```

In [3]:

```
## Função utilizada como preparação para transformar
## string (números no formato brasileiro: com pontos nos milhares e vírgula nas decimais) e

## Para transformar números no formato texto para float precisamos retirar os pontos;
## então substituir parêntesis por '-', para números negativos; e, por fim
## substituir vírgula por ponto para a separação das casas decimais.

def remover_pontos_finais(nome):
    nome = nome.replace('.','').replace(',','-').replace(')','').replace(',','.')
    return nome
```

In [4]:

```
## Função utilizada para remover caracteres especiais.

def remover_caracteres_especiais(nome):
    nome = nome.replace('Á','A').replace('Â','A').replace('Ã','A').replace('À','A')\
        .replace('É','E').replace('Ê','E').replace('Í','I').replace('Ó','O').replace('Ô','O')\
        .replace('Õ','O').replace('Ú','U').replace('Ü','U').replace('Ç','C')
    return nome
```

In [5]:

```
## Função para identificar se há valores faltantes, ou NaN em um dataframe

def identificar_faltantes(df):

    #Mostra o tamanho do meu dataset
    print("Shape do dataframe:",df.shape)

    #mostra em que coluna estão esses elementos faltantes
    colunas_faltantes = df.columns[df.isna().any()].tolist()
    print("Colunas que contêm valores faltantes:",colunas_faltantes)

    #traz o número de Linhas que restariam se eu tirasse os que seriam excluídas pelo dropn
    print("Número de linhas com valores faltantes:",df.shape[0]-df.dropna().shape[0],"\n")

return colunas_faltantes
```

In [6]:

```
from zipfile import ZipFile

def descompactar(nome_arquivo_compactado):
    # Lê o arquivo compactado e extrai o conteúdo
    with ZipFile (nome_arquivo_compactado, 'r') as zip:
        zip.extractall()
        zip.printdir()
    return "Concluído"
```

## 2. Coleta das bases de dados

### 2.1. Arrecadação das Receitas Administradas pela RFB, por Município (arrecadacao-da-receita-administrada-pela-rfb-por-municipio-2018.xlsx)

Para esta base, nenhum detalhe adicional para obtenção dos dados é necessário, vez que o download do arquivo é feito diretamente a partir do link [https://www.gov.br/receitafederal/pt-br/acesso-a-informacao/dados-abertos/receitadata/arrecadacao/copy\\_of\\_arrecadacao-das-receitas-administradas-pela-rfb-por-municipio/arrecadacao-das-receitas-administradas-pela-rfb/arrecadacao-da-receita-administrada-pela-rfb-por-municipio-2018.xlsx](https://www.gov.br/receitafederal/pt-br/acesso-a-informacao/dados-abertos/receitadata/arrecadacao/copy_of_arrecadacao-das-receitas-administradas-pela-rfb-por-municipio/arrecadacao-das-receitas-administradas-pela-rfb/arrecadacao-da-receita-administrada-pela-rfb-por-municipio-2018.xlsx) ([https://www.gov.br/receitafederal/pt-br/acesso-a-informacao/dados-abertos/receitadata/arrecadacao/copy\\_of\\_arrecadacao-das-receitas-administradas-pela-rfb-por-municipio/arrecadacao-das-receitas-administradas-pela-rfb/arrecadacao-da-receita-administrada-pela-rfb-por-municipio-2018.xlsx](https://www.gov.br/receitafederal/pt-br/acesso-a-informacao/dados-abertos/receitadata/arrecadacao/copy_of_arrecadacao-das-receitas-administradas-pela-rfb-por-municipio/arrecadacao-das-receitas-administradas-pela-rfb/arrecadacao-da-receita-administrada-pela-rfb-por-municipio-2018.xlsx)).

O documento possui três planilhas, a de interesse é a denominada 'TOTAL' (que agrupa os valores pagos tanto por meio de DARF como por GPS – que são tipos de guias para pagamento).

- Data da coleta: 19/3/2021;
- Possui metadados: não.

#### - Leitura do arquivo '.xls'.

Iniciamos com a visualização do arquivo '.xls' para verificarmos sua estrutura original e analisarmos as linhas que efetivamente deverão ser lidas.

O arquivo possui três planilhas, sendo a que é de nosso interesse é a de nome "TOTAL" (que agrupa pagamentos feitos tanto por GPS quanto por DARF).

In [7]:

```
nome_arquivo = 'arrecadacao-da-receita-administrada-pela-rfb-por-municipio-2018.xlsx'
planilha_interesse = 'TOTAL'
arrecadacao_2018 = pd.read_excel(nome_arquivo, sheet_name=planilha_interesse)
arrecadacao_2018
```

Out[7]:

		Unnamed: 0	Unnamed: 1	Unnamed: 2
0	ARRECADAÇÃO DAS RECEITAS ADMINISTRADAS PELA RF...		NaN	NaN
1		TOTAL	NaN	NaN
2		PERÍODO: 2018	NaN	NaN
3		UNIDADE: R\$ 1,00	NaN	NaN
4		MUNICÍPIOS	UF	ARRECADAÇÃO
...		...	...	...
5578		ZE DOCA	MA	1.326e+07
5579		ZERADO	-	1.0381e+10
5580		ZORTEA	SC	4.23176e+06
5581		{Ñ CLASS}	-	5.15863e+07
5582		TOTAL	-	1.39896e+12

5583 rows × 3 columns

Como se observa, acima, as primeiras linhas do arquivo são apenas informativas e não deverão ser carregadas (assim temos que configurar o parâmetro skiprows quando importarmos o csv).

Vemos também que, no fim do arquivo, há linhas que não nos interessam (totalizações, comentários, notas, etc). Assim, será necessário limitar a importação dos dados, configurando o parâmetro skipfooter (no caso).

Como utilizaremos o skipfooter, é necessário configurar o parâmetro engine='python'

Por fim, há apenas três colunas de interesse (as primeiras) de modo que vamos definir o parâmetro usecols.

In [8]:

```
arrecadacao_2018 = pd.read_excel(nome_arquivo, sheet_name=planilha_interesse, skiprows=5, skipfooter=1)
arrecadacao_2018
```

Out[8]:

	MUNICÍPIOS	UF	ARRECADAÇÃO
0	ABADIA DE GOIAS	GO	1.483209e+07
1	ABADIA DOS DOURADOS	MG	8.950170e+06
2	ABADIANIA	GO	1.779163e+07
3	ABAETE	MG	3.525459e+07
4	ABAETETUBA	PA	5.956002e+07
...	...	...	...
5571	ZABELE	PB	1.588994e+06
5572	ZACARIAS	SP	3.651978e+06
5573	ZE DOCA	MA	1.325997e+07
5574	ZERADO	-	1.038097e+10
5575	ZORTEA	SC	4.231760e+06

5576 rows × 3 columns

In [9]:

```
arrecadacao_2018.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5576 entries, 0 to 5575
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   MUNICÍPIOS    5576 non-null   object  
 1   UF           5576 non-null   object  
 2   ARRECADAÇÃO   5576 non-null   float64 
dtypes: float64(1), object(2)
memory usage: 130.8+ KB
```

## - Promovendo transformações nos dados

- Criar uma nova coluna, padronizada, chamada 'municipio-uf' que será a chave para joins/merges com outros dataframes;
- Renomear a coluna 'ARRECADAÇÃO' para '01\_arrecadacao\_2018' e transformá-la para 'float'
- Reordenar as colunas.

In [10]:

```
## A coluna chamada municipio-uf que será o resultado da
## concatenação do conteúdo, em maiúsculas ".str.upper()", das colunas MUNICIPIO e UF
arrecadacao_2018['municipio-uf']=arrecadacao_2018['MUNICÍPIOS'].str.upper()+' - '+arrecadacao_2018.head()
```

Out[10]:

	MUNICÍPIOS	UF	ARRECADAÇÃO	municipio-uf
0	ABADIA DE GOIAS	GO	1.483209e+07	ABADIA DE GOIAS-GO
1	ABADIA DOS DOURADOS	MG	8.950170e+06	ABADIA DOS DOURADOS-MG
2	ABADIANIA	GO	1.779163e+07	ABADIANIA-GO
3	ABAETE	MG	3.525459e+07	ABAETE-MG
4	ABAETETUBA	PA	5.956002e+07	ABAETETUBA-PA

In [11]:

```
print('Shape',arrecadacao_2018.shape)
print('Valores únicos',pd.Series(arrecadacao_2018['municipio-uf']).unique().shape)
# Veremos que não há duplicidades.
```

Shape (5576, 4)  
 Valores únicos (5576,)

In [12]:

```
# Renomeando a coluna ARRECADAÇÃO para "01_arrecadacao_2018".
arrecadacao_2018=arrecadacao_2018.rename({"ARRECADAÇÃO": "01_arrecadacao_2018"}, axis=1,)
```

In [13]:

```
## Vamos, por fim, manter apenas as variáveis de interesse, reordenando-as.
colunas_a_manter = ['municipio-uf', '01_arrecadacao_2018']
arrecadacao_2018=arrecadacao_2018[colunas_a_manter]
arrecadacao_2018
```

Out[13]:

	municipio-uf	01_arrecadacao_2018
0	ABADIA DE GOIAS-GO	1.483209e+07
1	ABADIA DOS DOURADOS-MG	8.950170e+06
2	ABADIANIA-GO	1.779163e+07
3	ABAETE-MG	3.525459e+07
4	ABAETETUBA-PA	5.956002e+07
...	...	...
5571	ZABELE-PB	1.588994e+06
5572	ZACARIAS-SP	3.651978e+06
5573	ZE DOCA-MA	1.325997e+07
5574	ZERADO--	1.038097e+10
5575	ZORTEA-SC	4.231760e+06

5576 rows × 2 columns

In [14]:

arrecadacao\_2018.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5576 entries, 0 to 5575
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   municipio-uf      5576 non-null   object  
 1   01_arrecadacao_2018 5576 non-null   float64 
dtypes: float64(1), object(1)
memory usage: 87.2+ KB
```

- Cria '.csv' com o resultado das transformações na base original.

In [15]:

arrecadacao\_2018.to\_csv('ntbk\_01\_01 - rfb\_arrecadacao\_2018.csv', index=False)

## 2.2. Tabela 1685 - Unidades locais, empresas e outras organizações atuantes, pessoal ocupado total, pessoal ocupado assalariado, pessoal assalariado médio, salários e outras remunerações e salário médio mensal (tabela1685.csv)

Descrição:

O Cadastro Central de Empresas - Cempre constitui um importante acervo de dados sobre o universo das empresas e outras organizações formais e suas respectivas unidades locais existentes no Brasil, reunindo informações cadastrais e econômicas oriundas de pesquisas anuais do IBGE, nas áreas de Indústria, Construção, Comércio e Serviços, e de registros administrativos do Ministério do Trabalho e Previdência Social, como a Relação Anual de Informações Sociais - RAIS.

Tabela 1685 - Unidades locais, empresas e outras organizações atuantes, pessoal ocupado total, pessoal ocupado assalariado, pessoal assalariado médio, salários e outras remunerações e salário médio mensal.

- Fonte: Site do Instituto Brasileiro de Geografia e Estatística na Internet (Sistema IBGE de Recuperação Automática - SIDRA);
- Endereço eletrônico: <https://sidra.ibge.gov.br/geratabela?format=us.csv&name=tabela1685.csv&terr=N&rank=-&query=t/1685/n6/all/v/all/p/last%201/d/v662%200,v1606%201,v5944%202,v10143%202/l,v,t%2Bp>  
(<https://sidra.ibge.gov.br/geratabela?format=us.csv&name=tabela1685.csv&terr=N&rank=-&query=t/1685/n6/all/v/all/p/last%201/d/v662%200,v1606%201,v5944%202,v10143%202/l,v,t%2Bp>)
- Data da coleta: 19/3/2021;
- Possui metadados: sim, mas sem descrição das variáveis  
(<https://metadados.ibge.gov.br/consulta/estatisticos/operacoes-estatisticas/CL>  
(<https://metadados.ibge.gov.br/consulta/estatisticos/operacoes-estatisticas/CL>)).

## - Leitura do arquivo '.csv'.

Iniciamos com a visualização do arquivo '.csv' para verificarmos sua estrutura original e analisarmos as linhas que efetivamente deverão ser lidas.

In [16]:

```
nome_arquivo = 'tabela1685.csv'
encoding_arquivo = ler_csv(nome_arquivo) #Chama a função para a leitura do arquivo '.csv'
[1] 'Município , Ano , Número de Unidades Locais (Unidades) , Número de
empresas e outras organizações atuantes (Unidades)', 'Pessoal ocupado tota
1 (Pessoas)', 'Pessoal ocupado assalariado (Pessoas)', 'Pessoal assalariad
o médio (Pessoas)', 'Salários e outras remunerações (Mil Reais)', 'Salário
médio mensal (Salários mínimos)', 'Salário médio mensal em reais (Reais)'
4 ['Alta Floresta D'Oeste (RO)', '2018', '521', '513', '3271', '2584', '25
57.86', '57902', '1.8', '1741.31']
5 ['Ariquemes (RO)', '2018', '2507', '2407', '19886', '16578', '16683.88',
'422650', '2.0', '1948.68']
6 ['Cabixi (RO)', '2018', '79', '77', '607', '535', '540.03', '11744', '1.
8', '1672.78']
7 ['Cacoal (RO)', '2018', '2230', '2139', '19942', '16815', '17093.05', '4
31101', '2.0', '1940.07']
8 ['Cerejeiras (RO)', '2018', '389', '377', '2482', '2044', '2076.75', '52
751', '2.0', '1953.89']
9 ['Colorado do Oeste (RO)', '2018', '307', '303', '2401', '2071', '2109.4
2', '46829', '1.8', '1707.68']
10 ['Corumbiara (RO)', '2018', '109', '109', '860', '715', '715.59', '1721
1', '1.9', '1850.09']
```

Como se observa, acima, as primeiras 2 linhas do arquivo são apenas informativas e não deverão ser carregadas (assim temos que configurar o parâmetro skiprows = 5 quando importarmos o csv).

Vemos também que, no fim do arquivo, há linhas que não nos interessam (totalizações, comentários, notas, etc). Assim, será necessário limitar a importação dos dados, configurando o parâmetro `skipfooter = 14` (no caso).

Como utilizaremos o `skipfooter`, é necessário configurar o parâmetro `engine='python'`

In [17]:

```
IBGE_Empresas=pd.read_csv(nome_arquivo, delimiter=',', engine='python', skiprows=2, skipfoot  
IBGE_Empresas.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 5570 entries, 0 to 5569  
Data columns (total 10 columns):  
 #   Column           Non-Null  
Count Dtype  
---  ---  
----  ----  
0    Município        5570 non-  
null  object  
1    Ano              5570 non-  
null  int64  
2    Número de unidades locais (Unidades) 5570 non-  
null  int64  
3    Número de empresas e outras organizações atuantes (Unidades) 5570 non-  
null  int64  
4    Pessoal ocupado total (Pessoas)       5570 non-  
null  int64  
5    Pessoal ocupado assalariado (Pessoas) 5570 non-  
null  int64  
6    Pessoal assalariado médio (Pessoas)   5570 non-  
null  float64  
7    Salários e outras remunerações (Mil Reais) 5570 non-  
null  int64  
8    Salário médio mensal (Salários mínimos) 5570 non-  
null  float64  
9    Salário médio mensal em reais (Reais)   5570 non-  
null  float64  
dtypes: float64(3), int64(6), object(1)  
memory usage: 435.3+ KB
```

In [18]:

IBGE\_Empresas

Out[18]:

	Município	Ano	Número de unidades locais (Unidades)	Número de empresas e outras organizações atuantes (Unidades)	Pessoal ocupado total (Pessoas)	Pessoal ocupado assalariado (Pessoas)	Pessoal assalariado médio (Pessoas)	Salário médio remunerado (M)
0	Alta Floresta D'Oeste (RO)	2018	521	513	3271	2584	2557.86	
1	Ariquemes (RO)	2018	2507	2407	19886	16578	16683.88	
2	Cabixi (RO)	2018	79	77	607	535	540.03	
3	Cacoal (RO)	2018	2230	2139	19942	16815	17093.05	
4	Cerejeiras (RO)	2018	389	377	2482	2044	2076.75	
...	...	...	...	...	...	...	...	...
5565	Vianópolis (GO)	2018	390	388	2317	1873	1920.39	
5566	Vicentinópolis (GO)	2018	200	199	1923	1673	1808.01	
5567	Vila Boa (GO)	2018	66	65	885	825	961.96	
5568	Vila Propício (GO)	2018	79	76	857	763	756.26	
5569	Brasília (DF)	2018	92298	86282	1352143	1240901	1242284.83	8

5570 rows × 10 columns

## - Promovendo transformações nos dados

- Selecionar apenas os dados de 2018
- Criar uma nova coluna, padronizada, chamada 'municipio-uf' que será a chave para joins/merges com outros dataframes;
- Atribuir nomes mais sucintos às variáveis;
- Criação de colunas adicionais;
- Reordenar as colunas.

In [19]:

```
# Vamos agora manter no dataframe exclusivamente o ano de 2018
```

```
IBGE_Empresas = IBGE_Empresas.loc[IBGE_Empresas['Ano'] == 2018]
print (IBGE_Empresas.shape)
IBGE_Empresas.head()
```

(5570, 10)

Out[19]:

	Município	Ano	Número de unidades locais (Unidades)	Número de empresas e outras organizações atuantes (Unidades)	Pessoal ocupado total (Pessoas)	Pessoal ocupado assalariado (Pessoas)	Pessoal assalariado médio (Pessoas)	Salários outros remunerações (Mil Reais)
0	Alta Floresta D'Oeste (RO)	2018	521	513	3271	2584	2557.86	5790
1	Ariquemes (RO)	2018	2507	2407	19886	16578	16683.88	42265
2	Cabixi (RO)	2018	79	77	607	535	540.03	1174
3	Cacoal (RO)	2018	2230	2139	19942	16815	17093.05	43110
4	Cerejeiras (RO)	2018	389	377	2482	2044	2076.75	5275

◀ ▶

In [20]:

```
# Criando a nova coluna 'municipio-uf'.
IBGE_Empresas['municipio-uf']=IBGE_Empresas['Município'].str[:-5].str.upper() \
+'-' +IBGE_Empresas['Município'].str[-3:-1].str.upper()

# Removendo caracteres especiais da coluna recém-criada.
IBGE_Empresas['municipio-uf'] = IBGE_Empresas['municipio-uf'].apply(remover_caracteres_especiais)

IBGE_Empresas.head()
```

Out[20]:

	Município	Ano	Número de unidades locais (Unidades)	Número de empresas e outras organizações atuantes (Unidades)	Pessoal ocupado total (Pessoas)	Pessoal assalariado (Pessoas)	Pessoal assalariado médio (Pessoas)	Salários outras remunerações (Mil Reais)
0	Alta Floresta D'Oeste (RO)	2018	521	513	3271	2584	2557.86	5790
1	Ariquemes (RO)	2018	2507	2407	19886	16578	16683.88	4226
2	Cabixi (RO)	2018	79	77	607	535	540.03	1174
3	Cacoal (RO)	2018	2230	2139	19942	16815	17093.05	43110
4	Cerejeiras (RO)	2018	389	377	2482	2044	2076.75	5275

In [21]:

```
print('Shape',IBGE_Empresas.shape)
print('Valores únicos',pd.Series(IBGE_Empresas['municipio-uf']).unique().shape)
# Veremos que não há duplicidades.
```

Shape (5570, 11)  
 Valores únicos (5570,)

In [22]:

```
# Vamos também renomear as colunas para nomes mais sucintos.
```

```
IBGE_Empresas.columns = ['02_municipio','02_ano','02_unidades','02_unidades_atuantes','02_o
```

In [23]:

```
# Chama a função para identificar se há valores faltantes ou NaN.
identificar_faltantes(IBGE_Empresas)
```

Shape do dataframe: (5570, 11)  
 Colunas que contêm valores faltantes: []  
 Número de linhas com valores faltantes: 0

Out[23]:

[]

In [24]:

```
## Vamos criar duas colunas com valores percentuais de algumas colunas
## (para que tragam uma informação em mesma base independentemente do tamanho do valor nominal)
## Em seguida apagar a coluna que deu origem para não influenciar indevidamente o modelo.

IBGE_Empresas['02_unidades_atuantes_%']=IBGE_Empresas['02_unidades_atuantes']/IBGE_Empresas
IBGE_Empresas = IBGE_Empresas.drop(['02_unidades_atuantes'],1)

## Eu iria usar a coluna 02_ocupados_assalariados_medio, em vez de 02_ocupados_assalariados
## naquela coluna que geram valores superiores a 100% quando comparadas com 02_ocupados_tot
## Então vamos usar 02_ocupados_assalariados para criar o %.
#IBGE_Empresas['02_ocupados_assalariados_medio_%']=IBGE_Empresas['02_ocupados_assalariados']
#IBGE_Empresas = IBGE_Empresas.drop(['02_ocupados_assalariados_medio'],1)

IBGE_Empresas['02_ocupados_assalariados_%']=IBGE_Empresas['02_ocupados_assalariados']/IBGE_Empresas
IBGE_Empresas = IBGE_Empresas.drop(['02_ocupados_assalariados'],1)

## Eu iria apagar, também, a coluna 02_ocupados_assalariados_médio porque utilizaremos 02_ocupados_tot
## Mas, como 02_ocupados_assalariados_médio é importante para o cálculo da coluna 02_total_assalariados
## ela será mantida
## IBGE_Empresas = IBGE_Empresas.drop(['02_ocupados_assalariados_medio'],1)

## Apagar a coluna 02_total_salarios_remuneracoes porque é resultado da operação com outras
## 02_total_salarios_remuneracoes = (02_salario_medio_mensal_em_Reais * 13) * 02_ocupados_assalariados
IBGE_Empresas = IBGE_Empresas.drop(['02_total_salarios_remuneracoes'],1)

##Apagar a coluna '02_municipio', redundante com municipio-uf
IBGE_Empresas = IBGE_Empresas.drop(['02_municipio'],1)

##Apagar a coluna '02_ano', valor fixo em todo o dataset (2018)
IBGE_Empresas = IBGE_Empresas.drop(['02_ano'],1)
```

In [25]:

```
## Por fim, reodenar as colunas.
cols= ['municipio-uf', '02_unidades','02_unidades_atuantes_%',
       '02_ocupados_total', '02_ocupados_assalariados_%', '02_ocupados_assalariados_medio',
       '02_salario_medio_mensal_em_SM',
       '02_salario_medio_mensal_em_Reais']

IBGE_Empresas = IBGE_Empresas[cols]
```

In [26]:

IBGE\_Empresas.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5570 entries, 0 to 5569
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   municipio-uf      5570 non-null   object  
 1   02_unidades        5570 non-null   int64   
 2   02_unidades_atuantes_% 5570 non-null   float64 
 3   02_ocupados_total  5570 non-null   int64   
 4   02_ocupados_assalariados_% 5570 non-null   float64 
 5   02_ocupados_assalariados_medio 5570 non-null   float64 
 6   02_salario_medio_mensal_em_SM 5570 non-null   float64 
 7   02_salario_medio_mensal_em_Reais 5570 non-null   float64 
dtypes: float64(5), int64(2), object(1)
memory usage: 391.6+ KB
```

In [27]:

IBGE\_Empresas.head()

Out[27]:

	municipio-uf	02_unidades	02_unidades_atuantes_%	02_ocupados_total	02_ocupados_assalariados_%
0	ALTA FLORESTA D'OESTE-RO	521	98.464491	3271	
1	ARIQUEMES-RO	2507	96.011169	19886	
2	CABIXI-RO	79	97.468354	607	
3	CACOAL-RO	2230	95.919283	19942	
4	CEREJEIRAS-RO	389	96.915167	2482	

- Cria '.csv' com o resultado das transformações na base original.

In [28]:

IBGE\_Empresas.to\_csv('ntbk\_01\_02 - ibge empresas 2018.csv', index=False)

## 2.3. Produto Interno Bruto dos Municípios

## (‘base\_de\_dados\_2010\_2018.xls.zip’)

Descrição:

Fornece estimativas do Produto Interno Bruto - PIB dos Municípios, a preços correntes, e do valor adicionado bruto da Agropecuária, da Indústria, dos Serviços e da Administração, saúde e educação públicas e seguridade social, a preços correntes, compatível com as metodologias das Contas Regionais e Nacionais do Brasil, sendo as estimativas obtidas comparáveis entre si.

Trata-se de uma arquivo compactado que contém nossa base de interesse:

**RELATORIO\_DTB\_BRASIL\_MUNICIPIO.xls**

- Fonte: Site do Instituto Brasileiro de Geografia e Estatística na Internet;
- Endereço eletrônico: <https://www.ibge.gov.br/estatisticas/economicas/contas-nacionais/2036-np-produto-interno-bruto-dos-municipios/9088-produto-interno-bruto-dos-municipios.html?=&t=downloads> (<https://www.ibge.gov.br/estatisticas/economicas/contas-nacionais/2036-np-produto-interno-bruto-dos-municipios/9088-produto-interno-bruto-dos-municipios.html?=&t=downloads>);
- Data da coleta: 19/3/2021;
- Possui metadados: sim, mas sem descrição das variáveis (<https://metadados.ibge.gov.br/consulta/estatisticos/operacoes-estatisticas/IO> (<https://metadados.ibge.gov.br/consulta/estatisticos/operacoes-estatisticas/IO>)).

### - Descompactação do arquivo original.

In [29]:

```
arquivo_a_descompactar = 'base_de_dados_2010_2018.xls.zip'
descompactar(arquivo_a_descompactar) ## Chama a função declarada no início do notebook
```

File Name	Modified	S
size		
PIB dos Municípios - base de dados 2010-2018.xls	2020-12-11 14:42:38	290
66752		

Out[29]:

'Concluído'

### - Leitura do arquivo '.xls'.

Iniciamos com a visualização do arquivo '.xls' para verificarmos sua estrutura original e analisarmos as linhas que efetivamente deverão ser lidas.

In [30]:

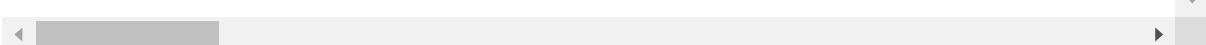
```
nome_arquivo = 'PIB dos Municípios - base de dados 2010-2018.xls'
IBGE_PIB=pd.read_excel(nome_arquivo)
IBGE_PIB
```

Out[30]:

Ano	Código da Grande Região	Nome da Grande Região	Código da Unidade da Federação	Sigla da Unidade da Federação	Nome da Unidade da Federação	Código do Município	Nome do Município
0	2010	1	Norte	11	RO	Rondônia	1100015 Alta Floresta D'Oeste
1	2010	1	Norte	11	RO	Rondônia	1100023 Ariquemes
2	2010	1	Norte	11	RO	Rondônia	1100031 Cabixi
3	2010	1	Norte	11	RO	Rondônia	1100049 Cacoal
4	2010	1	Norte	11	RO	Rondônia	1100056 Cerejeiras
...	...	...	...	...	...	...	...
50110	2018	5	Centro-oeste	52	GO	Goiás	5222005 Vianópolis

Ano	Código da Grande Região	Nome da Grande Região	Código da Unidade da Federação	Sigla da Unidade da Federação	Nome da Unidade da Federação	Código do Município	Nome do Município
50111	2018	5	Centro-oeste	52	GO	Goiás	5222054 Vicentinópolis
50112	2018	5	Centro-oeste	52	GO	Goiás	5222203 Vila Boa D
50113	2018	5	Centro-oeste	52	GO	Goiás	5222302 Vila Propício
50114	2018	5	Centro-oeste	53	DF	Distrito Federal	5300108 Brasília D

50115 rows × 43 columns



Como se observa, acima, não há linhas a excluir de modo que os parâmetros skiprows e skipfooter não precisam ser configurados. O dataframe IBGE\_PIB já contém os dados de interesse, portanto.

In [31]:

IBGE\_PIB.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50115 entries, 0 to 50114
Data columns (total 43 columns):
 #   Column
Non-Null Count Dtype
---  -----
0   Ano          int64
1   Código da Grande Região  int64
2   Nome da Grande Região   object
3   Código da Unidade da Federação int64
4   Sigla da Unidade da Federação object
5   Nome da Unidade da Federação  object
6   Código do Município    object
7   Nome do Município     int64
8   Região Metropolitana  object
9   Código da Mesorregião  int64
10  Nome da Mesorregião   object
11  Código da Microrregião int64
12  Nome da Microrregião  object
13  Código da Região Geográfica Imediata int64
14  Nome da Região Geográfica Imediata  object
15  Município da Região Geográfica Imediata object
16  Código da Região Geográfica Intermediária int64
17  Nome da Região Geográfica Intermediária object
18  Município da Região Geográfica Intermediária object
19  Código Concentração Urbana float64
20  Nome Concentração Urbana  object
21  Tipo Concentração Urbana  object
22  Código Arranjo Populacional float64
23  Nome Arranjo Populacional object
24  Hierarquia Urbana      object
50115 non-null object
```

```
25 Hierarquia Urbana (principais categorias)
50115 non-null object
26 Código da Região Rural
50115 non-null int64
27 Nome da Região Rural
50115 non-null object
28 Região rural (segundo classificação do núcleo)
50115 non-null object
29 Amazônia Legal
50115 non-null object
30 Semiárido
50115 non-null object
31 Cidade-Região de São Paulo
50115 non-null object
32 Valor adicionado bruto da Agropecuária,
a preços correntes
(R$ 1.000)
50115 non-null float64
33 Valor adicionado bruto da Indústria,
a preços correntes
(R$ 1.000)
50115 non-null float64
34 Valor adicionado bruto dos Serviços,
a preços correntes
- exceto Administração, defesa, educação e saúde públicas e seguridade social
1
(R$ 1.000) 50115 non-null float64
35 Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social,
a preços correntes
(R$ 1.000) 50115 non-null float64
36 Valor adicionado bruto total,
a preços correntes
(R$ 1.000)
50115 non-null float64
37 Impostos, líquidos de subsídios, sobre produtos,
a preços correntes
(R$ 1.000)
50115 non-null float64
38 Produto Interno Bruto,
a preços correntes
(R$ 1.000)
50115 non-null float64
39 Produto Interno Bruto per capita,
a preços correntes
(R$ 1,00)
50115 non-null float64
40 Atividade com maior valor adicionado bruto
50115 non-null object
41 Atividade com segundo maior valor adicionado bruto
50115 non-null object
42 Atividade com terceiro maior valor adicionado bruto
50115 non-null object
dtypes: float64(10), int64(9), object(24)
memory usage: 16.4+ MB
```

## - Promovendo transformações nos dados

- Selecionar apenas os dados de 2018

- Criar uma nova coluna, padronizada, chamada 'municipio-uf' que será a chave para joins/merges com outros dataframes;
- Excluir colunas que contêm com valores faltantes e de outras sem relevância;
- Atribuir nomes mais sucintos às variáveis;
- Criação de colunas adicionais;
- Reordenar as colunas.

In [32]:

```
# Vamos agora manter no dataframe exclusivamente o ano de 2018
```

```
IBGE_PIB = IBGE_PIB.loc[IBGE_PIB['Ano'] == 2018].copy()  
print (IBGE_PIB.shape)
```

(5570, 43)

In [33]:

```
# Criando a nova coluna 'municipio-uf'.
IBGE_PIB['municipio-uf']=IBGE_PIB['Nome do Município'].str.upper()+'-'+IBGE_PIB['Sigla da U

# Removendo caracteres especiais da coluna recém-criada.
IBGE_PIB['municipio-uf'] = IBGE_PIB['municipio-uf'].apply(remover_caracteres_especiais)

IBGE_PIB.head()
```

Out[33]:

Ano	Código da Grande Região	Nome da Grande Região	Código da Unidade da Federação	Sigla da Unidade da Federação	Nome da Unidade da Federação	Código do Município	Nome do Município	Metrop
44545	2018	1	Norte	11	RO	Rondônia	1100015	Alta Floresta D'Oeste
44546	2018	1	Norte	11	RO	Rondônia	1100023	Ariquemes
44547	2018	1	Norte	11	RO	Rondônia	1100031	Cabixi
44548	2018	1	Norte	11	RO	Rondônia	1100049	Cacoal
44549	2018	1	Norte	11	RO	Rondônia	1100056	Cerejeiras



In [34]:

```
print('Shape',IBGE_PIB.shape)
print('Valores únicos',pd.Series(IBGE_PIB['municipio-uf']).unique().shape)
# Veremos que não há duplicidades.
```

Shape (5570, 44)  
Valores únicos (5570,)

In [35]:

```
# Chama a função para identificar se há valores faltantes ou NaN.
colunas_faltantes = identificar_faltantes(IBGE_PIB)
```

Shape do dataframe: (5570, 44)  
Colunas que contêm valores faltantes: ['Região Metropolitana', 'Código Concen  
ntração Urbana', 'Nome Concentração Urbana', 'Tipo Concentração Urbana', 'Có  
digo Arranjo Populacional', 'Nome Arranjo Populacional']  
Número de linhas com valores faltantes: 5164

In [36]:

```
# Além de possuírem valores nulos, as colunas acima não agregam valor ao modelo e serão excluídas
IBGE_PIB = IBGE_PIB.drop(colunas_faltantes,1)

# Além delas, também serão excluídas as colunas que se referem a códigos para os quais os nomes são diferentes
# Ex. O Código de mesorregiao será excluído e mantido o Nome da Mesorregiao
# Apenas manterei, temporariamente, o código do município para futuro uso.
colunas_a_excluir=['Código da Grande Região','Código da Unidade da Federação',
                    'Código da Mesorregião','Código da Microrregião','Código da Região Geográfica Imediata',
                    'Município da Região Geográfica Imediata','Código da Região Geográfica Intermediária',
                    'Município da Região Geográfica Intermediária',
                    'Hierarquia Urbana','Hierarquia Urbana (principais categorias)',
                    'Código da Região Rural','Nome da Região Rural','Região rural (segundo classificação da IBGE)']
IBGE_PIB = IBGE_PIB.drop(colunas_a_excluir,1)
IBGE_PIB.columns
```

Out[36]:

```
Index(['Ano', 'Nome da Grande Região', 'Sigla da Unidade da Federação',
       'Nome da Unidade da Federação', 'Código do Município',
       'Nome do Município', 'Nome da Mesorregião', 'Nome da Microrregião',
       'Nome da Região Geográfica Imediata',
       'Nome da Região Geográfica Intermediária',
       'Valor adicionado bruto da Agropecuária, \na preços correntes\n(R$ 1.000)',

       'Valor adicionado bruto da Indústria,\nna preços correntes\n(R$ 1.000)',

       'Valor adicionado bruto dos Serviços,\nna preços correntes \n- exceto Administração, defesa, educação e saúde públicas e seguridade social\n(R$ 1.000)',

       'Valor adicionado bruto da Administração, defesa, educação e saúde públicas e seguridade social, \na preços correntes\n(R$ 1.000)',

       'Valor adicionado bruto total, \na preços correntes\n(R$ 1.000)',

       'Impostos, líquidos de subsídios, sobre produtos, \na preços correntes\n(R$ 1.000)',

       'Produto Interno Bruto, \na preços correntes\n(R$ 1.000)',

       'Produto Interno Bruto per capita, \na preços correntes\n(R$ 1,00)',

       'Atividade com maior valor adicionado bruto',
       'Atividade com segundo maior valor adicionado bruto',
       'Atividade com terceiro maior valor adicionado bruto', 'municipio-uf'],
      dtype='object')
```

In [37]:

```
# Vamos também renomear as colunas para nomes mais sucintos...

IBGE_PIB.columns=['03_ano','03_grande_regiao','03_uf','03_nome_uf','03_cod_municipio','03_m
    '03_mesorregiao','03_microregiao','03_regiao_imediata','03_regiao_intermediaria',
    '03_vlr_adic_bruto_agropecuaria','03_vlr_adic_bruto_industria','03_vlr_adic_bruto_ser
    '03_vlr_adic_bruto_administracao','03_vlr_adic_bruto_total','03_impostos_sobre_produto
    '03_pib','03_pib_per_capita','03_atividade_principal_primeira',
    '03_atividade_principal_segunda','03_atividade_principal_terceira', 'municipio-uf']

#... e agora reordene-as.

cols = ['municipio-uf','03_ano','03_grande_regiao','03_nome_uf','03_uf','03_cod_municipio',
    '03_mesorregiao','03_microregiao','03_regiao_imediata','03_regiao_intermediaria',
    '03_vlr_adic_bruto_agropecuaria','03_vlr_adic_bruto_industria','03_vlr_adic_bruto_ser
    '03_vlr_adic_bruto_administracao','03_vlr_adic_bruto_total','03_impostos_sobre_produto
    '03_pib','03_pib_per_capita','03_atividade_principal_primeira',
    '03_atividade_principal_segunda','03_atividade_principal_terceira']

IBGE_PIB = IBGE_PIB[cols]

print(IBGE_PIB.shape)
print(IBGE_PIB.columns)
IBGE_PIB.head()
```

```
(5570, 22)
Index(['municipio-uf', '03_ano', '03_grande_regiao', '03_nome_uf', '03_uf',
       '03_cod_municipio', '03_municipio', '03_mesorregiao', '03_microregiao',
       '03_regiao_imediata', '03_regiao_intermediaria',
       '03_vlr_adic_bruto_agropecuaria', '03_vlr_adic_bruto_industria',
       '03_vlr_adic_bruto_servicos', '03_vlr_adic_bruto_administracao',
       '03_vlr_adic_bruto_total', '03_impostos_sobre_produtos', '03_pib',
       '03_pib_per_capita', '03_atividade_principal_primeira',
       '03_atividade_principal_segunda', '03_atividade_principal_terceira'],
      dtype='object')
```

Out[37]:

		municipio-uf	03_ano	03_grande_regiao	03_nome_uf	03_uf	03_cod_municipio	03_mui	
44545		ALTA FLORESTA D'OESTE-RO	2018		Norte	Rondônia	RO	1100015	Alta FD
44546		ARIQUEMES-RO	2018		Norte	Rondônia	RO	1100023	Ariq
44547		CABIXI-RO	2018		Norte	Rondônia	RO	1100031	
44548		CACOAL-RO	2018		Norte	Rondônia	RO	1100049	
44549		CEREJEIRAS-RO	2018		Norte	Rondônia	RO	1100056	Cer

In [38]:

```

## Vamos criar variáveis com valores percentuais de outras já existentes
## (para que tragam uma informação em mesma base independentemente do tamanho do valor nominal)
## Em seguida apagar a coluna que deu origem para não influenciar indevidamente o modelo.

IBGE_PIB['03_vlr_adic_bruto_agropecuaria_%']=pd.to_numeric(IBGE_PIB['03_vlr_adic_bruto_agropecuaria'], errors='raise')
IBGE_PIB = IBGE_PIB.drop(['03_vlr_adic_bruto_agropecuaria'],1)

IBGE_PIB['03_vlr_adic_bruto_industria_%']=pd.to_numeric(IBGE_PIB['03_vlr_adic_bruto_industria'], errors='raise')
IBGE_PIB = IBGE_PIB.drop(['03_vlr_adic_bruto_industria'],1)

IBGE_PIB['03_vlr_adic_bruto_servicos_%']=pd.to_numeric(IBGE_PIB['03_vlr_adic_bruto_servicos'], errors='raise')
IBGE_PIB = IBGE_PIB.drop(['03_vlr_adic_bruto_servicos'],1)

IBGE_PIB['03_vlr_adic_bruto_administracao_%']=pd.to_numeric(IBGE_PIB['03_vlr_adic_bruto_administracao'], errors='raise')
IBGE_PIB = IBGE_PIB.drop(['03_vlr_adic_bruto_administracao'],1)

# Transforma algumas colunas em numérico
IBGE_PIB['03_impostos_sobre_produtos'] = pd.to_numeric(IBGE_PIB['03_impostos_sobre_produtos'], errors='raise')
IBGE_PIB['03_pib'] = pd.to_numeric(IBGE_PIB['03_pib'], errors='raise')
IBGE_PIB['03_pib_per_capita'] = pd.to_numeric(IBGE_PIB['03_pib_per_capita'], errors='raise')

# Transforma algumas colunas em string
IBGE_PIB['03_cod_municipio'] = IBGE_PIB['03_cod_municipio'].astype(str)

##Apagar a coluna '02_municipio', redundante com municipio-uf
IBGE_PIB = IBGE_PIB.drop(['03_municipio'],1)

# Apagar a coluna '03_ano' pois tem valor fixo (2018) por todo o dataset.
IBGE_PIB = IBGE_PIB.drop(['03_ano'],1)

# Reordenar as colunas
cols=['municipio-uf', '03_nome_uf', '03_uf', '03_cod_municipio', '03_grande_regiao',
       '03_mesorregiao', '03_microregiao', '03_regiao_imediata',
       '03_regiao_intermediaria', '03_vlr_adic_bruto_total', '03_vlr_adic_bruto_agropecuaria_%',
       '03_vlr_adic_bruto_industria_%', '03_vlr_adic_bruto_servicos_%',
       '03_vlr_adic_bruto_administracao_%',
       '03_impostos_sobre_produtos', '03_pib', '03_pib_per_capita',
       '03_atividade_principal_primeira', '03_atividade_principal_segunda',
       '03_atividade_principal_terceira']
IBGE_PIB=IBGE_PIB[cols]

print (IBGE_PIB.shape)
print (IBGE_PIB.columns)

```

(5570, 20)

```

Index(['municipio-uf', '03_nome_uf', '03_uf', '03_cod_municipio',
       '03_grande_regiao', '03_mesorregiao', '03_microregiao',
       '03_regiao_imediata', '03_regiao_intermediaria',
       '03_vlr_adic_bruto_total', '03_vlr_adic_bruto_agropecuaria_%',
       '03_vlr_adic_bruto_industria_%', '03_vlr_adic_bruto_servicos_%',
       '03_vlr_adic_bruto_administracao_%', '03_impostos_sobre_produtos',
       '03_pib', '03_pib_per_capita', '03_atividade_principal_primeira',
       '03_atividade_principal_segunda', '03_atividade_principal_terceira'],
      dtype='object')

```

In [39]:

```
# Chama a função para identificar se há valores faltantes ou NaN.
identificar_faltantes(IBGE_PIB)
#vendo as linhas que contêm variáveis com valores nulos.
print("Linhas que contém valores nulos:")
IBGE_PIB[pd.isnull(IBGE_PIB).any(axis=1)]
```

Shape do dataframe: (5570, 20)  
 Colunas que contêm valores faltantes: []  
 Número de linhas com valores faltantes: 0

Linhos que contém valores nulos:

Out[39]:

municipio-uf	03_nome_uf	03 uf	03_cod_municipio	03_grande_regiao	03_mesorregiao	03_micrорegiao

In [40]:

```
# investigando os dados originais, vemos que os valores da coluna 03_vlr_adic_bruto_agropecuaria_%
# para os municípios acima eram '-'.
# Assim, ao fazer a divisão que deu origem à coluna 03_vlr_adic_bruto_agropecuaria_%, o resultado era '-'.
# O que temos a fazer agora é substituir NaN por 0 (zero).
```

```
IBGE_PIB['03_vlr_adic_bruto_agropecuaria_%'] = IBGE_PIB['03_vlr_adic_bruto_agropecuaria_%'].replace('-', 0)
```

In [41]:

```
# Chama a função para identificar se há valores faltantes ou NaN.
identificar_faltantes(IBGE_PIB)
```

Shape do dataframe: (5570, 20)  
 Colunas que contêm valores faltantes: []  
 Número de linhas com valores faltantes: 0

Out[41]:

[]

In [42]:

IBGE\_PIB.head()

Out[42]:

		municipio-uf	03_nome_uf	03_uf	03_cod_municipio	03_grande_regiao	03_mesorregiao
44545		ALTA FLORESTA D'OESTE-RO	Rondônia	RO	1100015	Norte	Leste Rondoniense
44546		ARIQUEMES-RO	Rondônia	RO	1100023	Norte	Leste Rondoniense
44547		CABIXI-RO	Rondônia	RO	1100031	Norte	Leste Rondoniense
44548		CACOAL-RO	Rondônia	RO	1100049	Norte	Leste Rondoniense
44549		CEREJEIRAS-RO	Rondônia	RO	1100056	Norte	Leste Rondoniense

- Cria '.csv' com o resultado das transformações na base original.

In [43]:

IBGE\_PIB.to\_csv('ntbk\_01\_03 - ibge\_pib\_2018.csv', index=False)

## 2.4. Tabela 6579 - População residente estimada (tabela6579.csv)

Descrição:

As estimativas populacionais têm fundamental importância para o cálculo de indicadores sociodemográficos nos períodos intercensitários, bem como alimentam as bases de informações de Ministérios e Secretarias Estaduais e Municipais da área social para a implementação de políticas públicas e posterior avaliação de seus respectivos programas. Além disso, em cumprimento ao dispositivo constitucional, as estimativas da população constituem o principal parâmetro para a distribuição, conduzida pelo Tribunal de Contas da União, das quotas partes relativas ao Fundo de Participação de Estados e Municípios.

- Fonte: Site do Instituto Brasileiro de Geografia e Estatística na Internet (Sistema IBGE de Recuperação Automática - SIDRA);
- Endereço eletrônico: <https://sidra.ibge.gov.br/geratabela?format=us.csv&name=tabela6579.csv&terr=N&rank=-&query=t/6579/n6/all/v/all/p/2018/l,v,t%2Bp> (<https://sidra.ibge.gov.br/geratabela?format=us.csv&name=tabela6579.csv&terr=N&rank=-&query=t/6579/n6/all/v/all/p/2018/l,v,t%2Bp>)
- Data da coleta: 19/3/2021;
- Possui metadados: (<https://metadados.ibge.gov.br/consulta/estatisticos/operacoes-estatisticas/XF>) (<https://metadados.ibge.gov.br/consulta/estatisticos/operacoes-estatisticas/XF>)).

## - Leitura do arquivo '.csv'.

Iniciamos com a visualização do arquivo '.csv' para verificarmos sua estrutura original e analisarmos as linhas que efetivamente deverão ser lidas.

In [44]:

```
nome_arquivo = 'tabela6579.csv'
enconding_arquivo = ler_csv(nome_arquivo) #Chama a função para a leitura do arquivo '.csv'

32 ['Novo Horizonte do Oeste (RO)', '2018', '8751']
33 ['Cacaualândia (RO)', '2018', '6190']
34 ['Campo Novo de Rondônia (RO)', '2018', '14009']
35 ['Candeias do Jamari (RO)', '2018', '25983']
36 ['Castanheiras (RO)', '2018', '3119']
37 ['Chupinguaia (RO)', '2018', '10886']
38 ['Cujubim (RO)', '2018', '24226']
39 ['Governador Jorge Teixeira (RO)', '2018', '8095']
40 ['Itapuã do Oeste (RO)', '2018', '10272']
41 ['Ministro Andreazza (RO)', '2018', '9762']
42 ['Mirante da Serra (RO)', '2018', '11080']
43 ['Monte Negro (RO)', '2018', '15695']
44 ['Nova União (RO)', '2018', '7047']
45 ['Parecis (RO)', '2018', '5947']
46 ['Pimenteiras do Oeste (RO)', '2018', '2191']
47 ['Primavera de Rondônia (RO)', '2018', '2939']
48 ["São Felipe D'Oeste (RO)", '2018', '5280']
49 ['São Francisco do Guaporé (RO)', '2018', '19842']
50 ['Seringueiras (RO)', '2018', '11860']
```

Como se observa, acima, as primeiras 3 linhas do arquivo são apenas informativas e não deverão ser carregadas (assim temos que configurar o parâmetro skiprows = 5 quando importarmos o csv).

Vemos também que, no fim do arquivo, há linhas que não nos interessam (totalizações, comentários, notas, etc). Assim, será necessário limitar a importação dos dados, configurando o parâmetro skipfooter = 97 (no caso).

Como utilizaremos o skipfooter, é necessário configurar o parâmetro engine='python'

In [45]:

```
IBGE_Residentes=pd.read_csv(nome_arquivo, delimiter = ',', engine='python', skiprows=2, skipfooter=97)
IBGE_Residentes.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5570 entries, 0 to 5569
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Município        5570 non-null   object  
 1   Ano              5570 non-null   int64  
 2   População residente estimada (Pessoas) 5570 non-null   int64  
dtypes: int64(2), object(1)
memory usage: 130.7+ KB
```

In [46]:

IBGE\_Residentes

Out[46]:

	Município	Ano	População residente estimada (Pessoas)
0	Alta Floresta D'Oeste (RO)	2018	23167
1	Ariquemes (RO)	2018	106168
2	Cabixi (RO)	2018	5438
3	Cacoal (RO)	2018	84813
4	Cerejeiras (RO)	2018	16444
...	...	...	...
5565	Vianópolis (GO)	2018	13746
5566	Vicentinópolis (GO)	2018	8611
5567	Vila Boa (GO)	2018	6026
5568	Vila Propício (GO)	2018	5758
5569	Brasília (DF)	2018	2974703

5570 rows × 3 columns

## - Promovendo transformações nos dados

- Selecionar apenas os dados de 2018
- Criar uma nova coluna, padronizada, chamada 'municipio-uf' que será a chave para joins/merges com outros dataframes;
- Atribuir nomes mais sucintos às variáveis;
- Reordenar as colunas.

In [47]:

```
# Criando a nova coluna 'municipio-uf'.
IBGE_Residentes['municipio-uf']=IBGE_Residentes['Município'].str[:5].str.upper()+'-'+IBGE_
```

# Removendo caracteres especiais da coluna recém-criada.

```
IBGE_Residentes['municipio-uf'] = IBGE_Residentes['municipio-uf'].apply(remover_caracteres_
```

In [48]:

```
# Vamos excluir a coluna Município (que deu origem à municipio-uf) e reordenar.

IBGE_Residentes=IBGE_Residentes.drop(['Município','Ano'],1)

# Renomear a coluna '2018'
IBGE_Residentes = IBGE_Residentes.rename({'População residente estimada (Pessoas)': '04_pop'})

#... e agora reordene-as.
cols = ['municipio-uf','04_populacao_residente_2018']
IBGE_Residentes = IBGE_Residentes[cols]

print(IBGE_Residentes.shape)
print(IBGE_Residentes.columns)
IBGE_Residentes.head()
```

(5570, 2)  
Index(['municipio-uf', '04\_populacao\_residente\_2018'], dtype='object')

Out[48]:

	municipio-uf	04_populacao_residente_2018
0	ALTA FLORESTA D'OESTE-RO	23167
1	ARIQUEMES-RO	106168
2	CABIXI-RO	5438
3	CACOAL-RO	84813
4	CEREJEIRAS-RO	16444

In [49]:

```
print('Shape',IBGE_Residentes.shape)
print('Valores únicos',pd.Series(IBGE_Residentes['municipio-uf']).unique().shape)
# Veremos que não há duplicidades.
```

Shape (5570, 2)  
Valores únicos (5570,)

- Cria '.csv' com o resultado das transformações na base original.

In [50]:

```
IBGE_Residentes.to_csv('ntbk_01_04 - ibge população residente 2018.csv',index=False)
```

## 2.5. Tabela 200 - População residente, por sexo, situação e grupos de idade (tabela200.csv)

Descrição:

Censo Demográfico - 2010 O Censo Demográfico tem por objetivo contar os habitantes do território nacional, identificar suas características e revelar como vivem os brasileiros, produzindo informações imprescindíveis para a definição de políticas públicas e a tomada de decisões de investimentos da iniciativa privada ou de

qualquer nível de governo. Também constitui a única fonte de referência sobre a situação de vida da população nos municípios e em seus recortes internos, como distritos, bairros e localidades, rurais ou urbanas, cujas realidades dependem de seus resultados para serem conhecidas e terem seus dados atualizados.

Tabela 200 - População residente, por sexo, situação e grupos de idade - Amostra - Características Gerais da População

- Fonte: Site do Instituto Brasileiro de Geografia e Estatística na Internet (Sistema IBGE de Recuperação Automática - SIDRA);
- Endereço eletrônico: [[https://sidra.ibge.gov.br/geratabela?format=us.csv&name=tabela200.csv&terr=N&rank=-&query=t/200/n6/all/u/y/v/allxp/p/last%2021/c2/0/c1/0/c58/0,1140,1141,1142,1143,1144,1145,1146,1147,1148.](https://sidra.ibge.gov.br/geratabela?format=us.csv&name=tabela200.csv&terr=N&rank=-&query=t/200/n6/all/u/y/v/allxp/p/last%2021/c2/0/c1/0/c58/0,1140,1141,1142,1143,1144,1145,1146,1147,1148;) (<https://sidra.ibge.gov.br/geratabela?format=us.csv&name=tabela200.csv&terr=N&rank=-&query=t/200/n6/all/u/y/v/allxp/p/last%2021/c2/0/c1/0/c58/0,1140,1141,1142,1143,1144,1145,1146,1147,1148.>);
- Data da coleta: 19/3/2021;
- Possui metadados: sim, mas sem descrição das variáveis (<https://metadados.ibge.gov.br/consulta/estatisticos/operacoes-estatisticas/CD> (<https://metadados.ibge.gov.br/consulta/estatisticos/operacoes-estatisticas/CD>)).

## - Leitura do arquivo '.csv'.

Iniciamos com a visualização do arquivo '.csv' para verificarmos sua estrutura original e analisarmos as linhas que efetivamente deverão ser lidas.

In [51]:

```
nome_arquivo = 'tabela200.csv'
enconding_arquivo = ler_csv(nome_arquivo) #Chama a função para a leitura do arquivo '.csv'
[1]
6 ['Ariquemes (RO)', '2010', 'Total', 'Total', '90353', '7342', '8286', '9
256', '9041', '8981', '8458', '7668', '6944', '6416', '5042', '3982', '291
5', '2009', '1665', '1089', '567', '513', '109', '39', '19', '13']
7 ['Cabixi (RO)', '2010', 'Total', 'Total', '6313', '515', '507', '604',
'595', '448', '522', '462', '457', '467', '415', '406', '278', '228', '14
8', '134', '73', '44', '7', '-', '-', '3']
8 ['Cacoal (RO)', '2010', 'Total', 'Total', '78574', '5923', '6275', '747
3', '7859', '7758', '7018', '6617', '6067', '5647', '4832', '4072', '267
1', '2048', '1582', '1154', '854', '434', '237', '46', '7', '-']
9 ['Cerejeiras (RO)', '2010', 'Total', 'Total', '17029', '1218', '1357',
'1635', '1719', '1560', '1410', '1321', '1249', '1268', '1129', '836', '62
9', '508', '478', '302', '225', '111', '43', '24', '6', '-']
10 ['Colorado do Oeste (RO)', '2010', 'Total', 'Total', '18591', '1401',
'1421', '1668', '1787', '1639', '1628', '1481', '1367', '1257', '1170', '9
61', '816', '714', '435', '408', '218', '104', '79', '27', '11', '-']
11 ['Corumbiara (RO)', '2010', 'Total', 'Total', '8783', '613', '751', '90
9', '949', '736', '736', '721', '684', '635', '534', '436', '342', '229',
'199', '140', '85', '59', '21', '-', '-', '5']
12 ['Costa Marques (RO)', '2010', 'Total', 'Total', '12678', '12771', '1152
```

Como se observa, acima, as primeiras 3 linhas do arquivo são apenas informativas e não deverão ser carregadas (assim temos que configurar o parâmetro skiprows = 3 quando importarmos o csv).

Vemos também que, no fim do arquivo, há linhas que não nos interessam (totalizações, comentários, notas, etc). Assim, será necessário limitar a importação dos dados, configurando o parâmetro `skipfooter = 14` (no caso).

Como utilizaremos o `skipfooter`, é necessário configurar o parâmetro `engine='python'`

In [52]:

```
IBGE_Censo_Idade=pd.read_csv(nome_arquivo, delimiter=',', engine='python', skiprows=3, skipf
IBGE_Censo_Idade
```

Out[52]:

	Município	Ano	Situação do domicílio	Sexo	Total	0 a 4 anos	5 a 9 anos	10 a 14 anos	15 a 19 anos	20 a 24 anos
0	Alta Floresta D'Oeste (RO)	2010	Total	Total	24392	1851	2107	2401	2588	2200
1	Ariquemes (RO)	2010	Total	Total	90353	7342	8286	9256	9041	8981
2	Cabixi (RO)	2010	Total	Total	6313	515	507	604	595	448
3	Cacoal (RO)	2010	Total	Total	78574	5923	6275	7473	7859	7758
4	Cerejeiras (RO)	2010	Total	Total	17029	1218	1357	1635	1719	1560
...	...	...	...	...	...	...	...	...	...	...
5561	Vianópolis (GO)	2010	Total	Total	12548	786	1008	1129	1085	1039
5562	Vicentinópolis (GO)	2010	Total	Total	7371	592	587	622	640	697
5563	Vila Boa (GO)	2010	Total	Total	4735	447	481	475	429	456
5564	Vila Propício (GO)	2010	Total	Total	5145	357	490	508	468	360
5565	Brasília (DF)	2010	Total	Total	2570160	189171	200087	219091	220178	245521

5566 rows × 26 columns

In [53]:

IBGE\_Censo\_Idade.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5566 entries, 0 to 5565
Data columns (total 26 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Município        5566 non-null   object  
 1   Ano              5566 non-null   int64  
 2   Situação do domicílio 5566 non-null   object  
 3   Sexo              5566 non-null   object  
 4   Total             5566 non-null   object  
 5   0 a 4 anos       5566 non-null   object  
 6   5 a 9 anos       5566 non-null   object  
 7   10 a 14 anos     5566 non-null   object  
 8   15 a 19 anos     5566 non-null   object  
 9   20 a 24 anos     5566 non-null   object  
 10  25 a 29 anos     5566 non-null   object  
 11  30 a 34 anos     5566 non-null   object  
 12  35 a 39 anos     5566 non-null   object  
 13  40 a 44 anos     5566 non-null   object  
 14  45 a 49 anos     5566 non-null   object  
 15  50 a 54 anos     5566 non-null   object  
 16  55 a 59 anos     5566 non-null   object  
 17  60 a 64 anos     5566 non-null   object  
 18  65 a 69 anos     5566 non-null   object  
 19  70 a 74 anos     5566 non-null   object  
 20  75 a 79 anos     5566 non-null   object  
 21  80 a 84 anos     5566 non-null   object  
 22  85 a 89 anos     5566 non-null   object  
 23  90 a 94 anos     5566 non-null   object  
 24  95 a 99 anos     5566 non-null   object  
 25  100 anos ou mais 5566 non-null   object  
dtypes: int64(1), object(25)
memory usage: 1.1+ MB
```

## - Promovendo transformações nos dados

- Obs: os dados disponíveis são unicamente os de 2010;
- Criar uma nova coluna, padronizada, chamada 'municipio-uf' que será a chave para joins/merges com outros dataframes;
- Atribuir nomes mais sucintos às variáveis;
- Criação de colunas adicionais;
- Reordenar as colunas.

In [54]:

```
# Criando a nova coluna 'municipio-uf'.
IBGE_Censo_Idade['municipio-uf']=IBGE_Censo_Idade['Município'].str[:-5].str.upper()+'-'+'IBG'

# Removendo caracteres especiais da coluna recém-criada.
IBGE_Censo_Idade['municipio-uf'] = IBGE_Censo_Idade['municipio-uf'].apply(remover_caractere)

IBGE_Censo_Idade.head()
```

Out[54]:

	Município	Ano	Situação do domicílio	Sexo	Total	0 a 4 anos	5 a 9 anos	10 a 14 anos	15 a 19 anos	20 a 24 anos	25 a 29 anos	30 a 34 anos	35 a 39 anos
0	Alta Floresta D'Oeste (RO)	2010	Total	Total	24392	1851	2107	2401	2588	2200	2092	1867	1791
1	Ariquemes (RO)	2010	Total	Total	90353	7342	8286	9256	9041	8981	8458	7668	6944
2	Cabixi (RO)	2010	Total	Total	6313	515	507	604	595	448	522	462	451
3	Cacoal (RO)	2010	Total	Total	78574	5923	6275	7473	7859	7758	7018	6617	6061
4	Cerejeiras (RO)	2010	Total	Total	17029	1218	1357	1635	1719	1560	1410	1321	1249

In [55]:

```
print('Shape',IBGE_Censo_Idade.shape)
print('Valores únicos',pd.Series(IBGE_Censo_Idade['municipio-uf']).unique().shape)
# Veremos que não há duplicidades.
```

Shape (5566, 27)  
 Valores únicos (5566,)

In [56]:

```
#Vamos excluir as colunas em que os valores são fixos (e sem interesse): ano, situação do d
#Também excluiremos Município (que deu origem à coluna municipio-uf)
cols=['Ano','Situação do domicílio','Sexo','Município']
IBGE_Censo_Idade = IBGE_Censo_Idade.drop(cols,1)
IBGE_Censo_Idade.columns
```

Out[56]:

```
Index(['Total', '0 a 4 anos', '5 a 9 anos', '10 a 14 anos', '15 a 19 anos',
       '20 a 24 anos', '25 a 29 anos', '30 a 34 anos', '35 a 39 anos',
       '40 a 44 anos', '45 a 49 anos', '50 a 54 anos', '55 a 59 anos',
       '60 a 64 anos', '65 a 69 anos', '70 a 74 anos', '75 a 79 anos',
       '80 a 84 anos', '85 a 89 anos', '90 a 94 anos', '95 a 99 anos',
       '100 anos ou mais', 'municipio-uf'],
      dtype='object')
```

In [57]:

#Em seguida, vamos transformar as quantidades de pessoas em cada faixa em numérico.

```
cols = ['Total', '0 a 4 anos', '5 a 9 anos', '10 a 14 anos', '15 a 19 anos',
        '20 a 24 anos', '25 a 29 anos', '30 a 34 anos', '35 a 39 anos',
        '40 a 44 anos', '45 a 49 anos', '50 a 54 anos', '55 a 59 anos',
        '60 a 64 anos', '65 a 69 anos', '70 a 74 anos', '75 a 79 anos',
        '80 a 84 anos', '85 a 89 anos', '90 a 94 anos', '95 a 99 anos',
        '100 anos ou mais']

for i in cols:
    IBGE_Censo_Idade[i] = pd.to_numeric(IBGE_Censo_Idade[i], errors='coerce')

print(IBGE_Censo_Idade.info())
IBGE_Censo_Idade.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5566 entries, 0 to 5565
Data columns (total 23 columns):
 #   Column           Non-Null Count  Dtype  
 --- 
 0   Total            5565 non-null   float64
 1   0 a 4 anos       5565 non-null   float64
 2   5 a 9 anos       5565 non-null   float64
 3   10 a 14 anos     5565 non-null   float64
 4   15 a 19 anos     5565 non-null   float64
 5   20 a 24 anos     5565 non-null   float64
 6   25 a 29 anos     5565 non-null   float64
 7   30 a 34 anos     5565 non-null   float64
 8   35 a 39 anos     5565 non-null   float64
 9   40 a 44 anos     5565 non-null   float64
 10  45 a 49 anos     5565 non-null   float64
 11  50 a 54 anos     5565 non-null   float64
 12  55 a 59 anos     5565 non-null   float64
 13  60 a 64 anos     5565 non-null   float64
 14  65 a 69 anos     5565 non-null   float64
 15  70 a 74 anos     5565 non-null   float64
 16  75 a 79 anos     5565 non-null   float64
 17  80 a 84 anos     5564 non-null   float64
 18  85 a 89 anos     5531 non-null   float64
 19  90 a 94 anos     5235 non-null   float64
 20  95 a 99 anos     3842 non-null   float64
 21  100 anos ou mais 1669 non-null   float64
 22  municipio-uf      5566 non-null   object 
dtypes: float64(22), object(1)
memory usage: 1000.3+ KB
None
```

Out[57]:

Total	0 a 4 anos	5 a 9 anos	10 a 14 anos	15 a 19 anos	20 a 24 anos	25 a 29 anos	30 a 34 anos	35 a 39 anos	40 a 44 anos	45 a 49 anos	50 a 54 anos	55 a 59 anos	60 a 64 anos	65 a 69 anos	70 a 74 anos	75 a 79 anos	80 a 84 anos	85 a 89 anos	90 a 94 anos	95 a 99 anos	100 anos ou mais
0	24392.0	1851.0	2107.0	2401.0	2588.0	2200.0	2092.0	1867.0	1795.0	1715.0	1566.0	1178.0	1000.0	9256.0	8981.0	8458.0	7668.0	6944.0	6416.0	5042.0	3982.0
1	90353.0	7342.0	8286.0	9256.0	9041.0	8981.0	8458.0	7668.0	6944.0	6416.0	5042.0	3982.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0

Total	0 a 4 anos	5 a 9 anos	10 a 14 anos	15 a 19 anos	20 a 24 anos	25 a 29 anos	30 a 34 anos	35 a 39 anos	40 a 44 anos	45 a 49 anos	50 + anos
2 6313.0	515.0	507.0	604.0	595.0	448.0	522.0	462.0	457.0	467.0	415.0	406
3 78574.0	5923.0	6275.0	7473.0	7859.0	7758.0	7018.0	6617.0	6067.0	5647.0	4832.0	4072
4 17029.0	1218.0	1357.0	1635.0	1719.0	1560.0	1410.0	1321.0	1249.0	1268.0	1129.0	836

In [58]:

```
## Vemos acima que, após convertermos os valores em numéricos, em vez de 5566 informações n
## passamos a ter apenas 5565.
```

```
## Vamos tentar encontrar o valor diferente.
```

```
divergencia = IBGE_Censo_Idade.loc[IBGE_Censo_Idade['Total'].isna()]
print (divergencia.shape)
divergencia.head()
```

(1, 23)

Out[58]:

Total	0 a 4 anos	5 a 9 anos	10 a 14 anos	15 a 19 anos	20 a 24 anos	25 a 29 anos	30 a 34 anos	35 a 39 anos	40 a 44 anos	45 a 49 anos	50 a 54 anos	55 a 59 anos	60 + anos
1074	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [59]:

```
## Temos, assim, que a cidade Cococi-CE não apresenta qualquer valor para os campos. Vamos,  
IBGE_Censo_Idade = IBGE_Censo_Idade.drop(divergencia.index)  
  
# Repetindo o comando, vemos que não há mais divergências.  
divergencia = IBGE_Censo_Idade.loc[IBGE_Censo_Idade['Total'].isna()]  
print (divergencia.shape)  
  
print (IBGE_Censo_Idade.info())
```

```
(0, 23)  
<class 'pandas.core.frame.DataFrame'>  
Int64Index: 5565 entries, 0 to 5565  
Data columns (total 23 columns):  
 #   Column           Non-Null Count  Dtype     
---  --  
 0   Total            5565 non-null   float64  
 1   0 a 4 anos       5565 non-null   float64  
 2   5 a 9 anos       5565 non-null   float64  
 3   10 a 14 anos      5565 non-null   float64  
 4   15 a 19 anos      5565 non-null   float64  
 5   20 a 24 anos      5565 non-null   float64  
 6   25 a 29 anos      5565 non-null   float64  
 7   30 a 34 anos      5565 non-null   float64  
 8   35 a 39 anos      5565 non-null   float64  
 9   40 a 44 anos      5565 non-null   float64  
 10  45 a 49 anos      5565 non-null   float64  
 11  50 a 54 anos      5565 non-null   float64  
 12  55 a 59 anos      5565 non-null   float64  
 13  60 a 64 anos      5565 non-null   float64  
 14  65 a 69 anos      5565 non-null   float64  
 15  70 a 74 anos      5565 non-null   float64  
 16  75 a 79 anos      5565 non-null   float64  
 17  80 a 84 anos      5564 non-null   float64  
 18  85 a 89 anos      5531 non-null   float64  
 19  90 a 94 anos      5235 non-null   float64  
 20  95 a 99 anos      3842 non-null   float64  
 21  100 anos ou mais  1669 non-null   float64  
 22  municipio-uf       5565 non-null   object  
dtypes: float64(22), object(1)  
memory usage: 1.0+ MB  
None
```

In [60]:

```
# Chama a função para identificar se há valores faltantes ou NaN.
identificar_faltantes(IBGE_Censo_Idade)
```

Shape do dataframe: (5565, 23)

Colunas que contêm valores faltantes: ['80 a 84 anos', '85 a 89 anos', '90 a 94 anos', '95 a 99 anos', '100 anos ou mais']

Número de linhas com valores faltantes: 4258

Out[60]:

```
['80 a 84 anos',
 '85 a 89 anos',
 '90 a 94 anos',
 '95 a 99 anos',
 '100 anos ou mais']
```

In [61]:

```
## Vemos que para as colunas das idades mais elevadas (e apenas para elas) há valores NaN.

## Vamos substituir os NaN por zeros para não influenciar indevidamente a criação das novas
## Somar número com NaN geraria erro.

IBGE_Censo_Idade = IBGE_Censo_Idade.fillna(0)
print (IBGE_Censo_Idade.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5565 entries, 0 to 5565
Data columns (total 23 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Total            5565 non-null    float64
 1   0 a 4 anos       5565 non-null    float64
 2   5 a 9 anos       5565 non-null    float64
 3   10 a 14 anos     5565 non-null    float64
 4   15 a 19 anos     5565 non-null    float64
 5   20 a 24 anos     5565 non-null    float64
 6   25 a 29 anos     5565 non-null    float64
 7   30 a 34 anos     5565 non-null    float64
 8   35 a 39 anos     5565 non-null    float64
 9   40 a 44 anos     5565 non-null    float64
 10  45 a 49 anos     5565 non-null    float64
 11  50 a 54 anos     5565 non-null    float64
 12  55 a 59 anos     5565 non-null    float64
 13  60 a 64 anos     5565 non-null    float64
 14  65 a 69 anos     5565 non-null    float64
 15  70 a 74 anos     5565 non-null    float64
 16  75 a 79 anos     5565 non-null    float64
 17  80 a 84 anos     5565 non-null    float64
 18  85 a 89 anos     5565 non-null    float64
 19  90 a 94 anos     5565 non-null    float64
 20  95 a 99 anos     5565 non-null    float64
 21  100 anos ou mais 5565 non-null    float64
 22  municipio-uf      5565 non-null    object 
dtypes: float64(22), object(1)
memory usage: 1.0+ MB
None
```

In [62]:

```
# Temos, então, 5565 municípios com informações completas.

#Vamos, agora, reduzir o número de faixas de valores considerando ranges de 20 anos em vez
#transformar a informação como um percentual do total.
IBGE_Censo_Idade['05_0-19_anos_%'] = (IBGE_Censo_Idade['0 a 4 anos']+IBGE_Censo_Idade['5 a
+IBGE_Censo_Idade['10 a 14 anos']+IBGE_Censo_Idade['15 a 19 anos'])/IBGE_Censo_Idade['Total']

IBGE_Censo_Idade['05_20-39_anos_%'] = (IBGE_Censo_Idade['20 a 24 anos']+IBGE_Censo_Idade['2
+IBGE_Censo_Idade['30 a 34 anos']+IBGE_Censo_Idade['35 a 39 anos'])/IBGE_Censo_Idade['Total']

IBGE_Censo_Idade['05_40-59_anos_%'] = (IBGE_Censo_Idade['40 a 44 anos']+IBGE_Censo_Idade['4
+IBGE_Censo_Idade['50 a 54 anos']+IBGE_Censo_Idade['55 a 59 anos'])/IBGE_Censo_Idade['Total']

IBGE_Censo_Idade['05_60+_anos_%'] = (IBGE_Censo_Idade['60 a 64 anos']+IBGE_Censo_Idade['65
+IBGE_Censo_Idade['70 a 74 anos']+IBGE_Censo_Idade['75 a 79 anos']+IBGE_Censo_Idade['80 a 8
+IBGE_Censo_Idade['85 a 89 anos']+IBGE_Censo_Idade['90 a 94 anos']+IBGE_Censo_Idade['95 a 9
+IBGE_Censo_Idade['100 anos ou mais'])/IBGE_Censo_Idade['Total']*100

# Agora apagar as colunas originais.
cols=['0 a 4 anos', '5 a 9 anos', '10 a 14 anos', '15 a 19 anos',
      '20 a 24 anos', '25 a 29 anos', '30 a 34 anos', '35 a 39 anos',
      '40 a 44 anos', '45 a 49 anos', '50 a 54 anos', '55 a 59 anos',
      '60 a 64 anos', '65 a 69 anos', '70 a 74 anos', '75 a 79 anos',
      '80 a 84 anos', '85 a 89 anos', '90 a 94 anos', '95 a 99 anos',
      '100 anos ou mais']
IBGE_Censo_Idade = IBGE_Censo_Idade.drop(cols,1)

IBGE_Censo_Idade.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 5565 entries, 0 to 5565
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Total            5565 non-null   float64
 1   municipio-uf    5565 non-null   object  
 2   05_0-19_anos_%  5565 non-null   float64
 3   05_20-39_anos_% 5565 non-null   float64
 4   05_40-59_anos_% 5565 non-null   float64
 5   05_60+_anos_%   5565 non-null   float64
dtypes: float64(5), object(1)
memory usage: 304.3+ KB
```

In [63]:

```
# Chama a função para identificar se há valores faltantes ou NaN.
identificar_faltantes(IBGE_Censo_Idade)
```

Shape do dataframe: (5565, 6)  
 Colunas que contêm valores faltantes: []  
 Número de linhas com valores faltantes: 0

Out[63]:

[]

In [64]:

```
# Vamos renomar a coluna Total

IBGE_Censo_Idade = IBGE_Censo_Idade.rename({'Total': '05_total_residentes'}, axis = 1)

# ...e agora reorndeá-las.

cols = ['municipio-uf','05_total_residentes', '05_0-19_anos_%', '05_20-39_anos_%',
         '05_40-59_anos_%', '05_60+_anos_%']

IBGE_Censo_Idade = IBGE_Censo_Idade[cols]

print(IBGE_Censo_Idade.shape)
IBGE_Censo_Idade.info()
```

```
(5565, 6)
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5565 entries, 0 to 5565
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   municipio-uf      5565 non-null   object  
 1   05_total_residentes 5565 non-null   float64 
 2   05_0-19_anos_%     5565 non-null   float64 
 3   05_20-39_anos_%    5565 non-null   float64 
 4   05_40-59_anos_%    5565 non-null   float64 
 5   05_60+_anos_%      5565 non-null   float64 
dtypes: float64(5), object(1)
memory usage: 304.3+ KB
```

In [65]:

IBGE\_Censo\_Idade.head()

Out[65]:

	municipio-uf	05_total_residentes	05_0-19_anos_%	05_20-39_anos_%	05_40-59_anos_%	05_60+anos_%
0	ALTA FLORESTA D'OESTE-RO	24392.0	36.680059	32.609052	21.921122	8.785667
1	ARIQUEMES-RO	90353.0	37.547176	35.473089	20.314765	6.666076
2	CABIXI-RO	6313.0	35.181372	29.922382	24.805956	10.090290
3	CACOAL-RO	78574.0	35.037035	34.947947	21.918192	8.096826
4	CEREJEIRAS-RO	17029.0	34.817077	32.532738	22.678959	9.965353

- Cria '.csv' com o resultado das transformações na base original.

In [66]:

IBGE\_Censo\_Idade.to\_csv('ntbk\_01\_05 - ibge censo 2010 idades.csv', index=False)

## 2.6. Tabela 1554 - Pessoas de 10 anos ou mais de idade, por nível de instrução (tabela1554.csv)

Descrição:

Censo Demográfico - 2010 O Censo Demográfico tem por objetivo contar os habitantes do território nacional, identificar suas características e revelar como vivem os brasileiros, produzindo informações imprescindíveis para a definição de políticas públicas e a tomada de decisões de investimentos da iniciativa privada ou de qualquer nível de governo. Também constitui a única fonte de referência sobre a situação de vida da população nos municípios e em seus recortes internos, como distritos, bairros e localidades, rurais ou urbanas, cujas realidades dependem de seus resultados para serem conhecidas e terem seus dados atualizados.

Tabela 1554 - Pessoas de 10 anos ou mais de idade, por nível de instrução - Resultados Gerais da Amostra

- Fonte: Site do Instituto Brasileiro de Geografia e Estatística na Internet (Sistema IBGE de Recuperação Automática - SIDRA);
- Endereço eletrônico: <https://sidra.ibge.gov.br/geratabela?format=us.csv&name=tabela1554.csv&terr=N&rank=-&query=t/1554/n6/all/v/allxp/p/all/c1568/all/d/v140%200/l/v,c1568,t%2Bp> (<https://sidra.ibge.gov.br/geratabela?format=us.csv&name=tabela1554.csv&terr=N&rank=-&query=t/1554/n6/all/v/allxp/p/all/c1568/all/d/v140%200/l/v,c1568,t%2Bp>)
- Data da coleta: 19/3/2021;
- Possui metadados: sim, mas sem descrição das variáveis (<https://metadados.ibge.gov.br/consulta/estatisticos/operacoes-estatisticas/CD> (<https://metadados.ibge.gov.br/consulta/estatisticos/operacoes-estatisticas/CD>)).

## - Leitura do arquivo '.csv'.

Iniciamos com a visualização do arquivo '.csv' para verificarmos sua estrutura original e analisarmos as linhas que efetivamente deverão ser lidas.

In [67]:

```
nome_arquivo = 'tabela1554.csv'
enconding_arquivo = ler_csv(nome_arquivo) #Chama a função para a leitura do arquivo '.csv'

0
1 ['Tabela 1554 - Pessoas de 10 anos ou mais de idade, por nível de instru
ção - Resultados Gerais da Amostra']
2 ['Variável - Pessoas de 10 anos ou mais de idade (Pessoas)']
3 ['Município', 'Ano', 'Nível de instrução']
4 ['Município', 'Ano', 'Total', 'Sem instrução e fundamental incompleto',
'Fundamental completo e médio incompleto', 'Médio completo e superior inco
mpleto', 'Superior completo', 'Não determinado']
5 ["Alta Floresta D'Oeste (RO)", '2010', '20434', '13470', '2925', '3121',
'801', '117']
6 ['Ariquemes (RO)', '2010', '74725', '40351', '12630', '16799', '4275',
'671']
7 ['Cabixi (RO)', '2010', '5291', '3487', '730', '840', '193', '41']
8 ['Cacoal (RO)', '2010', '66376', '36674', '12244', '12943', '4090', '42
5']
9 ['Cerejeiras (RO)', '2010', '14454', '8642', '2256', '2562', '765', '22
9']
10 ['Colorado do Oeste (RO)', '2010', '15769', '9422', '2425', '2799', '95
1', '171']
11 ...
```

Como se observa, acima, as primeiras linhas do arquivo são apenas informativas e não deverão ser carregadas (assim temos que configurar o parâmetro skiprows quando importarmos o csv).

Vemos também que, no fim do arquivo, há linhas que não nos interessam (totalizações, comentários, notas, etc). Assim, será necessário limitar a importação dos dados, configurando o parâmetro skipfooter.

Como utilizaremos o skipfooter, é necessário configurar o parâmetro engine='python'

In [68]:

```
IBGE_Censo_Instrucao=pd.read_csv(nome_arquivo, delimiter=',',engine='python', skiprows=3, s
IBGE_Censo_Instrucao.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5565 entries, 0 to 5564
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Município        5565 non-null   object 
 1   Ano              5565 non-null   int64  
 2   Total            5565 non-null   int64  
 3   Sem instrução e fundamental incompleto 5565 non-null   int64  
 4   Fundamental completo e médio incompleto 5565 non-null   int64  
 5   Médio completo e superior incompleto    5565 non-null   int64  
 6   Superior completo      5565 non-null   int64  
 7   Não determinado       5565 non-null   object 
```

dtypes: int64(6), object(2)  
memory usage: 347.9+ KB

In [69]:

IBGE\_Censo\_Instrucao

Out[69]:

	Município	Ano	Total	Sem instrução e fundamental incompleto	Fundamental completo e médio incompleto	Médio completo e superior incompleto	Superior completo	N determina
0	Alta Floresta D'Oeste (RO)	2010	20434	13470	2925	3121	801	1
1	Ariquemes (RO)	2010	74725	40351	12630	16799	4275	1
2	Cabixi (RO)	2010	5291	3487	730	840	193	1
3	Cacoal (RO)	2010	66376	36674	12244	12943	4090	1
4	Cerejeiras (RO)	2010	14454	8642	2256	2562	765	1
...	...	...	...	...	...	...	...	...
5560	Vianópolis (GO)	2010	10754	6253	1930	2020	480	1
5561	Vicentinópolis (GO)	2010	6192	3934	936	1021	283	1
5562	Vila Boa (GO)	2010	3807	2335	600	724	80	1
5563	Vila Propício (GO)	2010	4298	2938	873	377	87	1
5564	Brasília (DF)	2010	2180903	761013	366883	656089	382917	141

5565 rows × 8 columns

## - Promovendo transformações nos dados

- Obs: os dados disponíveis são unicamente os de 2010;
- Criar uma nova coluna, padronizada, chamada 'municipio-uf' que será a chave para joins/merges com outros dataframes;
- Atribuir nomes mais sucintos às variáveis;
- Criação de colunas adicionais;
- Excluindo colunas desnecessárias;
- Reordenar as colunas.

In [70]:

```
# Criando a nova coluna 'municipio-uf'.
IBGE_Censo_Instrucao['municipio-uf']=IBGE_Censo_Instrucao['Município'].str[:-5].str.upper()

# Removendo caracteres especiais da coluna recém-criada.
IBGE_Censo_Instrucao['municipio-uf'] = IBGE_Censo_Instrucao['municipio-uf'].apply(remover_c)

IBGE_Censo_Instrucao.head()
```

Out[70]:

	Município	Ano	Total	Sem instrução e fundamental incompleto	Fundamental completo e médio incompleto	Médio completo e superior incompleto	Superior completo	Não determinado	m
0	Alta Floresta D'Oeste (RO)	2010	20434	13470	2925	3121	801	117	D'I
1	Ariquemes (RO)	2010	74725	40351	12630	16799	4275	671	AR
2	Cabixi (RO)	2010	5291	3487	730	840	193	41	
3	Cacoal (RO)	2010	66376	36674	12244	12943	4090	425	C.
4	Cerejeiras (RO)	2010	14454	8642	2256	2562	765	229	CEI

In [71]:

```
print('Shape',IBGE_Censo_Instrucao.shape)
print('Valores únicos',pd.Series(IBGE_Censo_Instrucao['municipio-uf']).unique().shape)
# Veremos que não há duplicidades.
```

Shape (5565, 9)  
 Valores únicos (5565,)

In [72]:

```
#Vamos excluir as colunas em que os valores são fixos (e sem interesse): ano.  

#Também excluiremos Município (que deu origem à coluna municipio-uf)
cols=['Ano','Município']
IBGE_Censo_Instrucao = IBGE_Censo_Instrucao.drop(cols,1)
IBGE_Censo_Instrucao.columns
```

Out[72]:

```
Index(['Total', 'Sem instrução e fundamental incompleto',
       'Fundamental completo e médio incompleto',
       'Médio completo e superior incompleto', 'Superior completo',
       'Não determinado', 'municipio-uf'],
      dtype='object')
```

In [73]:

```
# Chama a função para identificar se há valores faltantes ou NaN.
identificar_faltantes(IBGE_Censo_Instrucao)
```

Shape do dataframe: (5565, 7)  
 Colunas que contêm valores faltantes: []  
 Número de linhas com valores faltantes: 0

Out[73]:

[]

In [74]:

```
#Em seguida, vamos transformar as quantidades de pessoas em cada faixa em numérico.
```

```
cols = ['Total', 'Sem instrução e fundamental incompleto',
        'Fundamental completo e médio incompleto',
        'Médio completo e superior incompleto', 'Superior completo',
        'Não determinado']
for i in cols:
    IBGE_Censo_Instrucao[i]=pd.to_numeric(IBGE_Censo_Instrucao[i],errors='coerce')

print(IBGE_Censo_Instrucao.info())
IBGE_Censo_Instrucao.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5565 entries, 0 to 5564
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
 ---  -- 
 0   Total            5565 non-null   int64  
 1   Sem instrução e fundamental incompleto 5565 non-null   int64  
 2   Fundamental completo e médio incompleto 5565 non-null   int64  
 3   Médio completo e superior incompleto   5565 non-null   int64  
 4   Superior completo      5565 non-null   int64  
 5   Não determinado       4754 non-null   float64 
 6   municipio-uf         5565 non-null   object  
dtypes: float64(1), int64(5), object(1)
memory usage: 304.5+ KB
None
```

Out[74]:

	Total	Sem instrução e fundamental incompleto	Fundamental completo e médio incompleto	Médio completo e superior incompleto	Superior completo	Não determinado	municipio-uf
0	20434	13470	2925	3121	801	117.0	ALTA FLORESTA D'OESTE-RO
1	74725	40351	12630	16799	4275	671.0	ARIQUEMES-RO
2	5291	3487	730	840	193	41.0	CABIXI-RO
3	66376	36674	12244	12943	4090	425.0	CACOAL-RO
4	14454	8642	2256	2562	765	229.0	CEREJEIRAS-RO

In [75]:

```
# Chama a função para identificar se há valores faltantes ou NaN.
identificar_faltantes(IBGE_Censo_Instrucao)
```

Shape do dataframe: (5565, 7)

Colunas que contêm valores faltantes: ['Não determinado']

Número de linhas com valores faltantes: 811

Out[75]:

```
['Não determinado']
```

In [76]:

```
## Vemos que na coluna "Não determinado" (e apenas nela) há valores NaN.
```

```
## Vamos substituir os NaN por zeros para não influenciar indevidamente a criação das novas
## Somar número com NaN geraria erro.
```

```
IBGE_Censo_Instrucao['Não determinado'] = IBGE_Censo_Instrucao['Não determinado'].fillna(0)
print (IBGE_Censo_Instrucao.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5565 entries, 0 to 5564
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   Total            5565 non-null    int64  
 1   Sem instrução e fundamental incompleto 5565 non-null    int64  
 2   Fundamental completo e médio incompleto 5565 non-null    int64  
 3   Médio completo e superior incompleto    5565 non-null    int64  
 4   Superior completo                  5565 non-null    int64  
 5   Não determinado                 5565 non-null    float64 
 6   municipio-uf                   5565 non-null    object  
dtypes: float64(1), int64(5), object(1)
memory usage: 304.5+ KB
None
```

In [77]:

```
# Temos, então, 5565 municípios com informações completas.

#Vamos, agora, transformar a informação como um percentual do total.
IBGE_Censo_Instrucao['06_Sem_Instrução_e_Fundamental_Incompleto%'] = \
IBGE_Censo_Instrucao['Sem instrução e fundamental incompleto']/IBGE_Censo_Instrucao['Total']

IBGE_Censo_Instrucao['06_Medio_Incompleto%'] = \
IBGE_Censo_Instrucao['Fundamental completo e médio incompleto']/IBGE_Censo_Instrucao['Total']

IBGE_Censo_Instrucao['06_Superior_Incompleto%'] = \
IBGE_Censo_Instrucao['Médio completo e superior incompleto']/IBGE_Censo_Instrucao['Total']*

IBGE_Censo_Instrucao['06_Superior_Completo%'] = \
IBGE_Censo_Instrucao['Superior completo']/IBGE_Censo_Instrucao['Total']*100

IBGE_Censo_Instrucao['06_Indeterminado%'] = \
IBGE_Censo_Instrucao['Não determinado']/IBGE_Censo_Instrucao['Total']*100

#Podemos agora apagar as colunas originais.
cols=['Sem instrução e fundamental incompleto',
      'Fundamental completo e médio incompleto',
      'Médio completo e superior incompleto', 'Superior completo',
      'Não determinado']
IBGE_Censo_Instrucao = IBGE_Censo_Instrucao.drop(cols,1)

IBGE_Censo_Instrucao.head()
```

Out[77]:

	Total	municipio-uf	06_Sem_Instrução_e_Fundamental_Incompleto%	06_Medio_Incompleto%
0	20434	ALTA FLORESTA D'OESTE-RO	0.659195	14.31437%
1	74725	ARIQUEMES-RO	0.539993	16.90197%
2	5291	CABIXI-RO	0.659044	13.79701%
3	66376	CACOAL-RO	0.552519	18.44642%
4	14454	CEREJEIRAS-RO	0.597897	15.60813%

In [78]:

```
# Chama a função para identificar se há valores faltantes ou NaN.
identificar_faltantes(IBGE_Censo_Instrucao)
```

Shape do dataframe: (5565, 7)  
 Colunas que contêm valores faltantes: []  
 Número de linhas com valores faltantes: 0

Out[78]:

[]

In [79]:

```
# Vamos renomar a coluna Total

IBGE_Censo_Instrucao = IBGE_Censo_Instrucao.rename({'Total': '06_total_instrução'}, axis =
# ...e agora reordeneá-las.

cols = ['municipio-uf', '06_total_instrução', '06_Sem_Instrução_e_Fundamental_Incompleto%', 
        '06_Medio_Incompleto_%', '06_Superior_Incompleto_%',
        '06_Superior_Completo_%', '06_Indeterminado_%']

IBGE_Censo_Instrucao = IBGE_Censo_Instrucao[cols]

print(IBGE_Censo_Instrucao.shape)
IBGE_Censo_Instrucao.info()
```

(5565, 7)  
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 5565 entries, 0 to 5564  
Data columns (total 7 columns):  
 # Column Non-Null Count Dtype   
--- --   
 0 municipio-uf 5565 non-null object   
 1 06\_total\_instrução 5565 non-null int64   
 2 06\_Sem\_Instrução\_e\_Fundamental\_Incompleto% 5565 non-null float64   
 3 06\_Medio\_Incompleto\_% 5565 non-null float64   
 4 06\_Superior\_Incompleto\_% 5565 non-null float64   
 5 06\_Superior\_Completo\_% 5565 non-null float64   
 6 06\_Indeterminado\_% 5565 non-null float64   
dtypes: float64(5), int64(1), object(1)  
memory usage: 304.5+ KB

In [80]:

IBGE\_Censo\_Instrucao.head()

Out[80]:

	municipio-uf	06_total_instrução	06_Sem_Instrução_e_Fundamental_Incompleto%	06_Medio_Incompleto_%
0	ALTA FLORESTA D'OESTE-RO	20434		0.659195
1	ARIQUEMES-RO	74725		0.539993
2	CABIXI-RO	5291		0.659044
3	CACOAL-RO	66376		0.552519
4	CEREJEIRAS-RO	14454		0.597897

- Cria '.csv' com o resultado das transformações na base original.

In [81]:

```
IBGE_Censo_Instrucao.to_csv('ntbk_01_06 - ibge censo instrucao.csv', index=False)
```

## 2.7. Tabela 2030 - Pessoas de 10 anos ou mais de idade, por classes de rendimento nominal mensal (tabela2030.csv)

Descrição:

Censo Demográfico - 2010 O Censo Demográfico tem por objetivo contar os habitantes do território nacional, identificar suas características e revelar como vivem os brasileiros, produzindo informações imprescindíveis para a definição de políticas públicas e a tomada de decisões de investimentos da iniciativa privada ou de qualquer nível de governo. Também constitui a única fonte de referência sobre a situação de vida da população nos municípios e em seus recortes internos, como distritos, bairros e localidades, rurais ou urbanas, cujas realidades dependem de seus resultados para serem conhecidas e terem seus dados atualizados.

Tabela 2030 - Pessoas de 10 anos ou mais de idade, por classes de rendimento nominal mensal - Resultados Gerais da Amostra.

- Fonte: Site do Instituto Brasileiro de Geografia e Estatística na Internet (Sistema IBGE de Recuperação Automática - SIDRA);
- Endereço eletrônico: <https://sidra.ibge.gov.br/geratabela?format=us.csv&name=tabela2030.csv&terr=N&rank=-&query=t/2030/n6/all/v/allxp/p/last%201/c11570/all/d/v140%200/l/v,c11570,t%2Bp> (<https://sidra.ibge.gov.br/geratabela?format=us.csv&name=tabela2030.csv&terr=N&rank=-&query=t/2030/n6/all/v/allxp/p/last%201/c11570/all/d/v140%200/l/v,c11570,t%2Bp>)
- Data da coleta: 19/3/2021;
- Possui metadados: sim, mas sem descrição das variáveis (<https://metadados.ibge.gov.br/consulta/estatisticos/operacoes-estatisticas/CD> (<https://metadados.ibge.gov.br/consulta/estatisticos/operacoes-estatisticas/CD>)).

### - Leitura do arquivo '.csv'.

Iniciamos com a visualização do arquivo '.csv' para verificarmos sua estrutura original e analisarmos as linhas que efetivamente deverão ser lidas.

In [82]:

```
nome_arquivo = 'tabela2030.csv'
encoding_arquivo = ler_csv(nome_arquivo) #Chama a função para a Leitura do arquivo '.csv'

0
1 ['Tabela 2030 - Pessoas de 10 anos ou mais de idade, por classes de rendimento nominal mensal - Resultados Gerais da Amostra']
2 ['Variável - Pessoas de 10 anos ou mais de idade (Pessoas)']
3 ['Município', 'Ano', 'Classes de rendimento nominal mensal']
4 ['Município', 'Ano', 'Total', 'Até 1 salário mínimo', 'Mais de 1 a 2 salários mínimos', 'Mais de 2 a 3 salários mínimos', 'Mais de 3 a 5 salários mínimos', 'Mais de 5 a 10 salários mínimos', 'Mais de 10 a 20 salários mínimos', 'Mais de 20 salários mínimos', 'Sem rendimento']
5 ["Alta Floresta D'Oeste (RO)", '2010', '20434', '7202', '3525', '855', '732', '345', '76', '39', '7662']
6 ['Ariquemes (RO)', '2010', '74725', '19934', '17050', '5436', '4066', '2656', '604', '377', '24602']
7 ['Cabixi (RO)', '2010', '5291', '2033', '856', '241', '195', '77', '37', '10', '1841']
8 ['Cacoal (RO)', '2010', '66376', '18968', '13879', '4662', '3676', '2295', '677', '326', '21893']
9 ['Cerejeiras (RO)', '2010', '14454', '5565', '2518', '888', '776', '284', '81', '51', '4290']
10 ['Colniza (RO)', '2010', '157601', '151221', '122721', '102411', '162
```

Como se observa, acima, as primeiras linhas do arquivo são apenas informativas e não deverão ser carregadas (assim temos que configurar o parâmetro skiprows quando importarmos o csv).

Vemos também que, no fim do arquivo, há linhas que não nos interessam (totalizações, comentários, notas, etc). Assim, será necessário limitar a importação dos dados, configurando o parâmetro `skipfooter` (no caso).

Como utilizaremos o skipfooter, é necessário configurar o parâmetro engine='python'

In [83]:

```
IBGE_Censo_Rendimento=pd.read_csv(nome_arquivo, delimiter=',', engine='python', skiprows=3,
IBGE_Censo_Rendimento.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5565 entries, 0 to 5564
Data columns (total 11 columns):
 #   Column           Non-Null Count Dtype  
 --- 
 0   Município        5565 non-null   object  
 1   Ano              5565 non-null   int64   
 2   Total            5565 non-null   int64  
 3   Até 1 salário mínimo  5565 non-null   int64  
 4   Mais de 1 a 2 salários mínimos 5565 non-null   int64  
 5   Mais de 2 a 3 salários mínimos 5565 non-null   int64  
 6   Mais de 3 a 5 salários mínimos 5565 non-null   int64  
 7   Mais de 5 a 10 salários mínimos 5565 non-null   object  
 8   Mais de 10 a 20 salários mínimos 5565 non-null   object  
 9   Mais de 20 salários mínimos    5565 non-null   object  
 10  Sem rendimento      5565 non-null   int64  
dtypes: int64(7), object(4)
memory usage: 478.4+ KB
```

In [84]:

IBGE\_Censo\_Rendimento

Out[84]:

	Município	Ano	Total	Até 1 salário mínimo	Mais de 1 a 2 salários mínimos	Mais de 2 a 3 salários mínimos	Mais de 3 a 5 salários mínimos	Mais de 5 a 10 salários mínimos	Mais de 10 a 20 salários mínimos	+
0	Alta Floresta D'Oeste (RO)	2010	20434	7202	3525	855	732	345	76	
1	Ariquemes (RO)	2010	74725	19934	17050	5436	4066	2656	604	
2	Cabixi (RO)	2010	5291	2033	856	241	195	77	37	
3	Cacoal (RO)	2010	66376	18968	13879	4662	3676	2295	677	
4	Cerejeiras (RO)	2010	14454	5565	2518	888	776	284	81	
...	...	...	...	...	...	...	...	...	...	
5560	Vianópolis (GO)	2010	10754	3956	2356	626	496	387	91	
5561	Vicentinópolis (GO)	2010	6192	1848	1526	521	239	186	52	
5562	Vila Boa (GO)	2010	3807	1315	798	122	109	50	3	
5563	Vila Propício (GO)	2010	4298	1307	758	197	101	35	13	
5564	Brasília (DF)	2010	2180903	360150	432771	162062	158761	205364	126058	

5565 rows × 11 columns



## - Promovendo transformações nos dados

- Obs: os dados disponíveis são unicamente os de 2010;
- Criar uma nova coluna, padronizada, chamada 'municipio-uf' que será a chave para joins/merges com outros dataframes;
- Atribuir nomes mais sucintos às variáveis;
- Criação de colunas adicionais;
- Excluindo colunas desnecessárias;
- Reordenar as colunas.

In [85]:

```
# Criando a nova coluna 'municipio-uf'.
IBGE_Censo_Rendimento['municipio-uf']=IBGE_Censo_Rendimento['Município'].str[:-5].str.upper

# Removendo caracteres especiais da coluna recém-criada.
IBGE_Censo_Rendimento['municipio-uf'] = IBGE_Censo_Rendimento['municipio-uf'].apply(remover)

IBGE_Censo_Rendimento.head()
```

Out[85]:

	Município	Ano	Total	Até 1 salário mínimo	Mais de 1 a 2 salários mínimos	Mais de 2 a 3 salários mínimos	Mais de 3 a 5 salários mínimos	Mais de 5 a 10 salários mínimos	Mais de 10 a 20 salários mínimos	Mais de 20 salários mínimos
0	Alta Floresta D'Oeste (RO)	2010	20434	7202	3525	855	732	345	76	39
1	Ariquemes (RO)	2010	74725	19934	17050	5436	4066	2656	604	377
2	Cabixi (RO)	2010	5291	2033	856	241	195	77	37	10
3	Cacoal (RO)	2010	66376	18968	13879	4662	3676	2295	677	326
4	Cerejeiras (RO)	2010	14454	5565	2518	888	776	284	81	51

In [86]:

```
print('Shape',IBGE_Censo_Rendimento.shape)
print('Valores únicos',pd.Series(IBGE_Censo_Rendimento['municipio-uf']).unique().shape)
# Veremos que não há duplicidades.
```

Shape (5565, 12)  
 Valores únicos (5565,)

In [87]:

```
#Vamos excluir as colunas em que os valores são fixos (e sem interesse): ano.
#Também excluiremos Município (que deu origem à coluna municipio-uf)
cols=['Ano','Município']
IBGE_Censo_Rendimento = IBGE_Censo_Rendimento.drop(cols,1)
IBGE_Censo_Rendimento.columns
```

Out[87]:

```
Index(['Total', 'Até 1 salário mínimo', 'Mais de 1 a 2 salários mínimos',
       'Mais de 2 a 3 salários mínimos', 'Mais de 3 a 5 salários mínimos',
       'Mais de 5 a 10 salários mínimos', 'Mais de 10 a 20 salários mínimos',
       'Mais de 20 salários mínimos', 'Sem rendimento', 'municipio-uf'],
      dtype='object')
```

In [88]:

```
# Chama a função para identificar se há valores faltantes ou NaN.
identificar_faltantes(IBGE_Censo_Rendimento)
```

Shape do dataframe: (5565, 10)  
 Colunas que contêm valores faltantes: []  
 Número de linhas com valores faltantes: 0

Out[88]:

[]

In [89]:

```
#Em seguida, vamos transformar as quantidades de pessoas em cada faixa em numérico.
cols = ['Total', 'Até 1 salário mínimo', 'Mais de 1 a 2 salários mínimos',
         'Mais de 2 a 3 salários mínimos', 'Mais de 3 a 5 salários mínimos',
         'Mais de 5 a 10 salários mínimos', 'Mais de 10 a 20 salários mínimos',
         'Mais de 20 salários mínimos', 'Sem rendimento']

for i in cols:
    IBGE_Censo_Rendimento[i]=pd.to_numeric(IBGE_Censo_Rendimento[i],errors='coerce')

IBGE_Censo_Rendimento.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5565 entries, 0 to 5564
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
 ---  -- 
 0   Total            5565 non-null   int64  
 1   Até 1 salário mínimo  5565 non-null   int64  
 2   Mais de 1 a 2 salários mínimos  5565 non-null   int64  
 3   Mais de 2 a 3 salários mínimos  5565 non-null   int64  
 4   Mais de 3 a 5 salários mínimos  5565 non-null   int64  
 5   Mais de 5 a 10 salários mínimos 5553 non-null   float64 
 6   Mais de 10 a 20 salários mínimos 5175 non-null   float64 
 7   Mais de 20 salários mínimos     4377 non-null   float64 
 8   Sem rendimento             5565 non-null   int64  
 9   municipio-uf              5565 non-null   object 
```

dtypes: float64(3), int64(6), object(1)  
 memory usage: 434.9+ KB

In [90]:

```
# Chama a função para identificar se há valores faltantes ou NaN.
identificar_faltantes(IBGE_Censo_Rendimento)
```

Shape do dataframe: (5565, 10)

Colunas que contêm valores faltantes: ['Mais de 5 a 10 salários mínimos', 'Mais de 10 a 20 salários mínimos', 'Mais de 20 salários mínimos']

Número de linhas com valores faltantes: 1334

Out[90]:

```
['Mais de 5 a 10 salários mínimos',
 'Mais de 10 a 20 salários mínimos',
 'Mais de 20 salários mínimos']
```

In [91]:

```
## Vemos que para as colunas de rendimentos mais elevados (e apenas para elas) há valores
## Vamos substituir os NaN por zeros para não influenciar indevidamente a criação das novas
## Somar número com NaN geraria erro.
```

```
IBGE_Censo_Rendimento = IBGE_Censo_Rendimento.fillna(0)
IBGE_Censo_Rendimento.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5565 entries, 0 to 5564
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   Total            5565 non-null    int64  
 1   Até 1 salário mínimo  5565 non-null    int64  
 2   Mais de 1 a 2 salários mínimos 5565 non-null    int64  
 3   Mais de 2 a 3 salários mínimos 5565 non-null    int64  
 4   Mais de 3 a 5 salários mínimos 5565 non-null    int64  
 5   Mais de 5 a 10 salários mínimos 5565 non-null    float64 
 6   Mais de 10 a 20 salários mínimos 5565 non-null    float64 
 7   Mais de 20 salários mínimos    5565 non-null    float64 
 8   Sem rendimento          5565 non-null    int64  
 9   municipio-uf           5565 non-null    object 
```

dtypes: float64(3), int64(6), object(1)  
memory usage: 434.9+ KB

In [92]:

```
# Temos, então, 5565 municípios com informações completas.
```

```
#Vamos, agora, transformar a informação como um percentual do total.
```

```
IBGE_Censo_Rendimento['07_Ate_1SM_%'] = IBGE_Censo_Rendimento['Até 1 salário mínimo']/IBGE_Censo_Rendimento['Total']
IBGE_Censo_Rendimento['07_Ate_2SM_%'] = IBGE_Censo_Rendimento['Mais de 1 a 2 salários mínimos']/IBGE_Censo_Rendimento['Total']
IBGE_Censo_Rendimento['07_Ate_3SM_%'] = IBGE_Censo_Rendimento['Mais de 2 a 3 salários mínimos']/IBGE_Censo_Rendimento['Total']
IBGE_Censo_Rendimento['07_Ate_5SM_%'] = IBGE_Censo_Rendimento['Mais de 3 a 5 salários mínimos']/IBGE_Censo_Rendimento['Total']
IBGE_Censo_Rendimento['07_Ate_10SM_%'] = IBGE_Censo_Rendimento['Mais de 5 a 10 salários mínimos']/IBGE_Censo_Rendimento['Total']
IBGE_Censo_Rendimento['07_Ate_20SM_%'] = IBGE_Censo_Rendimento['Mais de 10 a 20 salários mínimos']/IBGE_Censo_Rendimento['Total']
IBGE_Censo_Rendimento['07_20SM+_%'] = IBGE_Censo_Rendimento['Mais de 20 salários mínimos']/IBGE_Censo_Rendimento['Total']
IBGE_Censo_Rendimento['07_Sem_Rendimento_%'] = IBGE_Censo_Rendimento['Sem rendimento']/IBGE_Censo_Rendimento['Total']
```

```
#Podemos agora apagar as colunas originais.
```

```
cols=[ 'Até 1 salário mínimo', 'Mais de 1 a 2 salários mínimos',
       'Mais de 2 a 3 salários mínimos', 'Mais de 3 a 5 salários mínimos',
       'Mais de 5 a 10 salários mínimos', 'Mais de 10 a 20 salários mínimos',
       'Mais de 20 salários mínimos', 'Sem rendimento']
```

```
IBGE_Censo_Rendimento = IBGE_Censo_Rendimento.drop(cols,1)
```

```
IBGE_Censo_Rendimento.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5565 entries, 0 to 5564
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
 --- 
 0   Total            5565 non-null    int64  
 1   municipio-uf     5565 non-null    object  
 2   07_Ate_1SM_%     5565 non-null    float64 
 3   07_Ate_2SM_%     5565 non-null    float64 
 4   07_Ate_3SM_%     5565 non-null    float64 
 5   07_Ate_5SM_%     5565 non-null    float64 
 6   07_Ate_10SM_%    5565 non-null    float64 
 7   07_Ate_20SM_%    5565 non-null    float64 
 8   07_20SM+_%      5565 non-null    float64 
 9   07_Sem_Rendimento_% 5565 non-null    float64 
dtypes: float64(8), int64(1), object(1)
memory usage: 434.9+ KB
```

In [93]:

```
# Chama a função para identificar se há valores faltantes ou NaN.
identificar_faltantes(IBGE_Censo_Rendimento)
```

Shape do dataframe: (5565, 10)

Colunas que contêm valores faltantes: []

Número de linhas com valores faltantes: 0

Out[93]:

```
[]
```

In [94]:

```
# Vamos renomar a coluna Total

IBGE_Censo_Rendimento = IBGE_Censo_Rendimento.rename({'Total': '07_total_rendimentos'}, axis=1)
# ...e agora reordene-as.

cols = ['municipio-uf', '07_total_rendimentos', '07_Ate_1SM_%', '07_Ate_2SM_%', '07_Ate_3SM_%',
         '07_Ate_5SM_%', '07_Ate_10SM_%', '07_Ate_20SM_%', '07_20SM+_%', '07_Sem_Rendimento_%']

IBGE_Censo_Rendimento = IBGE_Censo_Rendimento[cols]

print(IBGE_Censo_Rendimento.shape)
IBGE_Censo_Rendimento.info()

(5565, 10)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5565 entries, 0 to 5564
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   municipio-uf      5565 non-null   object  
 1   07_total_rendimentos  5565 non-null   int64   
 2   07_Ate_1SM_%       5565 non-null   float64 
 3   07_Ate_2SM_%       5565 non-null   float64 
 4   07_Ate_3SM_%       5565 non-null   float64 
 5   07_Ate_5SM_%       5565 non-null   float64 
 6   07_Ate_10SM_%      5565 non-null   float64 
 7   07_Ate_20SM_%      5565 non-null   float64 
 8   07_20SM+_%        5565 non-null   float64 
 9   07_Sem_Rendimento_% 5565 non-null   float64 
dtypes: float64(8), int64(1), object(1)
memory usage: 434.9+ KB
```

In [95]:

IBGE\_Censo\_Rendimento.head()

Out[95]:

	municipio-uf	07_total_rendimentos	07_Ate_1SM_%	07_Ate_2SM_%	07_Ate_3SM_%	07_Ate_5SM_%	07_Ate_10SM_%	07_Ate_20SM_%	07_20SM+_%	07_Sem_Rendimento_%
0	ALTA FLORESTA D'OESTE-RO	20434	35.245180	17.250661	4.184203	;	;	;	;	;
1	ARIQUEMES-RO	74725	26.676480	22.816996	7.274674	;	;	;	;	;
2	CABIXI-RO	5291	38.423738	16.178416	4.554905	;	;	;	;	;
3	CACOAL-RO	66376	28.576594	20.909666	7.023623	;	;	;	;	;
4	CEREJEIRAS-RO	14454	38.501453	17.420783	6.143628	;	;	;	;	;

- Cria '.csv' com o resultado das transformações na base original.

In [96]:

```
IBGE_Censo_Rendimento.to_csv('ntbk_01_07 - ibge censo rendimentos.csv', index=False)
```

### 3. Resumo das bases de dados criadas

In [97]:

```
#Ntbk 01_01 - RFB Arrecadacao 2018.csv  
arrecadacao_2018.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 5576 entries, 0 to 5575  
Data columns (total 2 columns):  
 #   Column           Non-Null Count  Dtype     
---  --     
 0   municipio-uf    5576 non-null   object    
 1   01_arrecadacao_2018 5576 non-null   float64  
dtypes: float64(1), object(1)  
memory usage: 87.2+ KB
```

In [98]:

```
#Ntbk 01_02 - IBGE Empresas 2018.csv  
IBGE_Empresas.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
Int64Index: 5570 entries, 0 to 5569  
Data columns (total 8 columns):  
 #   Column           Non-Null Count  Dtype     
---  --     
 0   municipio-uf    5570 non-null   object    
 1   02_unidades     5570 non-null   int64     
 2   02_unidades_atuantes_% 5570 non-null   float64  
 3   02_ocupados_total 5570 non-null   int64     
 4   02_ocupados_assalariados_% 5570 non-null   float64  
 5   02_ocupados_assalariados_medio 5570 non-null   float64  
 6   02_salario_medio_mensal_em_SM 5570 non-null   float64  
 7   02_salario_medio_mensal_em_Reais 5570 non-null   float64  
dtypes: float64(5), int64(2), object(1)  
memory usage: 391.6+ KB
```

In [99]:

```
#Ntbk 01_03 - IBGE PIB 2018.csv
IBGE_PIB.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5570 entries, 44545 to 50114
Data columns (total 20 columns):
 #   Column           Non-Null Count Dtype  
 --- 
 0   municipio-uf      5570 non-null   object  
 1   03_nome_uf        5570 non-null   object  
 2   03_uf              5570 non-null   object  
 3   03_cod_municipio   5570 non-null   object  
 4   03_grande_regiao   5570 non-null   object  
 5   03_mesorregiao    5570 non-null   object  
 6   03_microregiao    5570 non-null   object  
 7   03_regiao_imediata 5570 non-null   object  
 8   03_regiao_intermediaria 5570 non-null   object  
 9   03_vlr_adic_bruto_total 5570 non-null   float64 
 10  03_vlr_adic_bruto_agropecuaria_% 5570 non-null   float64 
 11  03_vlr_adic_bruto_industria_% 5570 non-null   float64 
 12  03_vlr_adic_bruto_servicos_% 5570 non-null   float64 
 13  03_vlr_adic_bruto_administracao_% 5570 non-null   float64 
 14  03_impostos_sobre_produtos 5570 non-null   float64 
 15  03_pib              5570 non-null   float64 
 16  03_pib_per_capita   5570 non-null   float64 
 17  03_atividade_principal_primeira 5570 non-null   object  
 18  03_atividade_principal_segunda 5570 non-null   object  
 19  03_atividade_principal_terceira 5570 non-null   object  
dtypes: float64(8), object(12)
memory usage: 913.8+ KB
```

In [100]:

```
#Ntbk 01_04 - IBGE População Residente 2018.csv
IBGE_Residentes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5570 entries, 0 to 5569
Data columns (total 2 columns):
 #   Column           Non-Null Count Dtype  
 --- 
 0   municipio-uf      5570 non-null   object  
 1   04_populacao_residente_2018 5570 non-null   int64  
dtypes: int64(1), object(1)
memory usage: 87.2+ KB
```

In [101]:

```
#Ntbk 01_05 - IBGE Censo 2010 Idades.csv
```

```
IBGE_Censo_Idade.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5565 entries, 0 to 5565
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   municipio-uf      5565 non-null   object  
 1   05_total_residentes 5565 non-null   float64 
 2   05_0-19_anos_%     5565 non-null   float64 
 3   05_20-39_anos_%    5565 non-null   float64 
 4   05_40-59_anos_%    5565 non-null   float64 
 5   05_60+_anos_%     5565 non-null   float64 
dtypes: float64(5), object(1)
memory usage: 304.3+ KB
```

In [102]:

```
#Ntbk 01_06 - IBGE Censo Instrucao.csv
```

```
IBGE_Censo_Instrucao.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5565 entries, 0 to 5564
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   municipio-uf      5565 non-null   object  
 1   06_total_instrução 5565 non-null   int64  
 2   06_Sem_Instrução_e_Fundamental_Incompleto% 5565 non-null   float64 
 3   06_Medio_Incompleto_%    5565 non-null   float64 
 4   06_Superior_Incompleto_% 5565 non-null   float64 
 5   06_Superior_Completo_%   5565 non-null   float64 
 6   06_Indeterminado_%     5565 non-null   float64 
dtypes: float64(5), int64(1), object(1)
memory usage: 304.5+ KB
```

In [103]:

```
#Ntbk 01_07 - IBGE Censo Rendimentos.csv
```

```
IBGE_Censo_Rendimento.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5565 entries, 0 to 5564
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   municipio-uf      5565 non-null   object  
 1   07_total_rendimentos  5565 non-null   int64   
 2   07_Ate_1SM_%       5565 non-null   float64 
 3   07_Ate_2SM_%       5565 non-null   float64 
 4   07_Ate_3SM_%       5565 non-null   float64 
 5   07_Ate_5SM_%       5565 non-null   float64 
 6   07_Ate_10SM_%      5565 non-null   float64 
 7   07_Ate_20SM_%      5565 non-null   float64 
 8   07_20SM+_%        5565 non-null   float64 
 9   07_Sem_Rendimento_% 5565 non-null   float64 
dtypes: float64(8), int64(1), object(1)
memory usage: 434.9+ KB
```

# Apêndice VII – Ntbk 02. Criação do dataset consolidado - merges e saneamentos.ipynb

Este notebook tem por objetivo criar uma única tabela consolidada, gerada a partir dos *merges/joins* dos *datasets* gerados pelo *Ntbk 01. Coleta dos dados e transformações preliminares* (que são resultado dos ajustes e transformações dos dataset originais coletados).

Veremos que, para isso, precisaremos promover novas transformações, agora em relação ao conteúdo dos dataset, para garantir a uniformidade das informações.

Está estruturado da seguinte maneira:

1. Ações Preliminares;
2. Comparação entre os *datasets* a serem consolidados;
3. Obtenção do *dataset* de referência para uniformização dos nomes dos municípios;
4. *Merges* dos *datasets* de interesse;
5. Criação do *dataset* consolidado ('ntbk 02 - dataset consolidado.csv').

## 1. Ações Preliminares

### - Importação das bibliotecas de interesse.

In [1]:

```
import pandas as pd
import numpy as np
from zipfile import ZipFile

pd.set_option('max_columns',200)
pd.set_option('max_rows',100)
```

### - Declaração de funções que serão utilizadas no notebook.

In [2]:

```
def descompactar(nome_arquivo_compactado):
    # Lê o arquivo zipado e extrai o conteúdo
    with ZipFile (nome_arquivo_compactado, 'r') as zip:
        zip.extractall()
        zip.printdir()
    return "Concluído"
```

In [3]:

```
## Função utilizada para remover caracteres especiais.
```

```
def remover_caracteres_especiais(nome):
    nome = nome.replace('Á', 'A').replace('Â', 'A').replace('Ã', 'A').replace('À', 'A').replace('à', 'a')
    return nome
```

In [4]:

```
## Função utilizada para uniformizar os nomes dos municípios.
```

```
## A grafia à direita ser refere à atualização mais recente publicada pelo IBGE acerca dos
```

```
def corrigir_nomes_municipios(nome):
```

```
    nome = nome.replace('AMPARO DA SERRA-MG', 'AMPARO DO SERRA-MG')\
        .replace('AMPARO DE SAO FRANCISCO-SE', 'AMPARO DO SAO FRANCISCO-SE')\
        .replace('AUGUSTO SEVERO-RN', 'CAMPO GRANDE-RN') \
        .replace('ASSU-RN', 'ACU-RN') \
        .replace('BALNEARIO DE PICARRAS-SC', 'BALNEARIO PICARRAS-SC').replace('BARAO DO MONTE AL')
        .replace('BELEM DE SAO FRANCISCO-PE', 'BELEM DO SAO FRANCISCO-PE').replace('BIRITIBA-MIR')
        .replace('BOA SAUDE-RN', 'JANUARIO CICCO-RN')\
        .replace('BOM JESUS-GO', 'BOM JESUS DE GOIAS-GO').replace('BRASOPOLIS-MG', 'BRAZOPOLIS-MG')
        .replace('DONA EUSEBIA-MG', 'DONA EUZEBIA-MG').replace('ELDORADO DOS CARAJAS-PA', 'ELDORA')
        .replace('EMBU-SP', 'EMBU DAS ARTES-SP').replace('FLORINIA-SP', 'FLORINEA-SP').replace('F')
        .replace('GRAO PARA-SC', 'GRAO-PARA-SC')\
        .replace('IGUARACI-PE', 'IGUARACY-PE').replace('LAGOA DO ITAENGA-PE', 'LAGOA DE ITAENGA-P')
        .replace('MOGI-MIRIM-SP', 'MOGI MIRIM-SP').replace('MUNHOZ DE MELLO-PR', 'MUNHOZ DE MELO-')
        .replace('MUQUEM DE SAO FRANCISCO-BA', 'MUQUEM DO SAO FRANCISCO-BA') \
        .replace("OLHO D'AGUA DOS BORGES-RN", "OLHO D'AGUA DO BORGES-RN")\
        .replace('PACAEMBU DAS ARTES-SP', 'PACAEMBU-SP').replace('PARATI-RJ', 'PARATY-RJ') \
        .replace('PASSA-VINTE-MG', 'PASSA VINTE-MG').replace("PINGO D'AGUA-MG", "PINGO-D'AGUA-MG")
        .replace('POXOREO-MT', 'POXOREU-MT').replace('PRESIDENTE CASTELO BRANCO-SC', 'PRESIDENTE')
        .replace('SANTA CRUZ DO MONTE CASTELO-PR', 'SANTA CRUZ DE MONTE CASTELO-PR') \
        .replace('SANTA ISABEL DO PARA-PA', 'SANTA IZABEL DO PARA-PA').replace('SANTA TERESINHA-')
        .replace("SANTANA DO LIVRAMENTO-RS", "SANT'ANA DO LIVRAMENTO-RS").replace('SAO DOMINGOS')
        .replace('SAO THOME DAS LETRAS-MG', 'SAO TOME DAS LETRAS-MG')\
        .replace('SAO VALERIO DA NATIVIDADE-TO', 'SAO VALERIO-TO').replace('TRAJANO DE MORAIS-RJ')

    return nome
```

In [5]:

```
## Função para identificar se há valores faltantes, ou NaN em um dataframe

def identificar_faltantes(df):

    #Mostra o tamanho do meu dataset
    print("Shape do dataframe:",df.shape)

    #mostra em que coluna estão esses elementos faltantes
    colunas_faltantes = df.columns[df.isna().any()].tolist()
    print("Colunas que contêm valores faltantes:",colunas_faltantes)

    #traz o número de linhas que restariam se eu tirasse os que seriam excluídas pelo dropna
    print("Número de linhas com valores faltantes:",df.shape[0]-df.dropna().shape[0],"\n")

    return colunas_faltantes
```

## 2. Comparação entre os datasets a serem consolidados.

- Leitura dos arquivo '.csv' gerados pelo \*Ntbk 01. Coleta dos dados.

In [6]:

```
arrecadacao_2018=pd.read_csv('ntbk_01_01 - rfb arrecadacao 2018.csv')
IBGE_Empresas=pd.read_csv('ntbk_01_02 - ibge empresas 2018.csv')
IBGE_PIB=pd.read_csv('ntbk_01_03 - ibge pib 2018.csv')
IBGE_Residentes=pd.read_csv('ntbk_01_04 - ibge população residente 2018.csv')
IBGE_Censo_Idade=pd.read_csv('ntbk_01_05 - ibge censo 2010 idades.csv')
IBGE_Censo_Instrucao=pd.read_csv('ntbk_01_06 - ibge censo instrucao.csv')
IBGE_Censo_Rendimento=pd.read_csv('ntbk_01_07 - ibge censo rendimentos.csv')

print ("Shapes:")
print ('-arrecadacao_2018',arrecadacao_2018.shape)
print ('-IBGE_Empresas',IBGE_Empresas.shape)
print ('-IBGE_PIB',IBGE_PIB.shape)
print ('-IBGE_Residentes',IBGE_Residentes.shape)
print ('-IBGE_Censo_Idade',IBGE_Censo_Idade.shape)
print ('-IBGE_Censo_Instrucao',IBGE_Censo_Instrucao.shape)
print ('-IBGE_Censo_Rendimento',IBGE_Censo_Rendimento.shape)
```

Shapes:

- arrecadacao\_2018 (5576, 2)
- IBGE\_Empresas (5570, 8)
- IBGE\_PIB (5570, 20)
- IBGE\_Residentes (5570, 2)
- IBGE\_Censo\_Idade (5565, 6)
- IBGE\_Censo\_Instrucao (5565, 7)
- IBGE\_Censo\_Rendimento (5565, 10)

Com a leitura dos shapes das bases de interesse, observa-se, de pronto, que há divergências entre os valores e, possivelmente, inconsistências.

Sabe-se que atualmente, há, no Brasil, 5570 municípios:

- Mesmo número contido nas bases indicadas por IBGE\_Empresas, IBGE\_PIB e IBGE\_Residentes;
- No entanto a base da RFB (indicada por arrecadacao\_2018) apresenta 5576 municípios;
- Já as bases indicadas por IBGE\_Censo\_Idade, IBGE\_Censo\_Instrucao, IBGE\_Censo\_Rendimento contêm 5565 municípios.

### 3. Obtenção do dataset de referência para uniformização dos nomes dos municípios

De modo a uniformizarmos nossas bases a partir de uma referência comum e, assim, sanearmos as possíveis divergências vamos importar a tabela Códigos de Municípios mais recente do site do IBGE (que, no caso, é a de 2020).

#### Dataset: Códigos dos Municípios - 2020 (DTB\_2020\_v2.zip)

Descrição:

A Tabela de Códigos de Municípios do IBGE apresenta a lista dos municípios brasileiros associados a um código composto de 7 dígitos, sendo os dois primeiros referentes ao código da Unidade da Federação.

É atualizada sistematicamente de forma a incluir as alterações decorrentes do desdobramento de municípios e, consequentemente, da criação de novos municípios, mudanças de nome dos municípios, como também de processos de fusão que resultam na extinção ou modificação de nome de algum município.

A informação mais recente é a do ano de 2020.

Trata-se de uma arquivo compactado que contém nossa base de interesse:

“RELATORIO\_DTB\_BRASIL\_MUNICIPIO.xls”

- Fonte: Site do Instituto Brasileiro de Geografia e Estatística na Internet  
([https://geoftp.ibge.gov.br/organizacao\\_do\\_territorio/estrutura\\_territorial/divisao\\_territorial/](https://geoftp.ibge.gov.br/organizacao_do_territorio/estrutura_territorial/divisao_territorial/))  
([https://geoftp.ibge.gov.br/organizacao\\_do\\_territorio/estrutura\\_territorial/divisao\\_territorial/](https://geoftp.ibge.gov.br/organizacao_do_territorio/estrutura_territorial/divisao_territorial/))
- Endereço eletrônico:  
[https://geoftp.ibge.gov.br/organizacao\\_do\\_territorio/estrutura\\_territorial/divisao\\_territorial/2020/DTB\\_2020\\_v2](https://geoftp.ibge.gov.br/organizacao_do_territorio/estrutura_territorial/divisao_territorial/2020/DTB_2020_v2)  
([https://geoftp.ibge.gov.br/organizacao\\_do\\_territorio/estrutura\\_territorial/divisao\\_territorial/2020/DTB\\_2020\\_v2](https://geoftp.ibge.gov.br/organizacao_do_territorio/estrutura_territorial/divisao_territorial/2020/DTB_2020_v2))  
Data da coleta: 19/3/2021;
- Possui metadados: não.

In [7]:

```
descompactar('DTB_2020_v2.zip')
```

File Name	Modified	S
RELATORIO_DTB_BRASIL_DISTRITO.ods	2021-02-18 08:41:42	538
RELATORIO_DTB_BRASIL_DISTRITO.xls	2021-02-18 08:42:38	2993
RELATORIO_DTB_BRASIL_MUNICIPIO.ods	2021-02-18 08:43:28	272
RELATORIO_DTB_BRASIL_MUNICIPIO.xls	2021-02-18 08:44:14	1323
RELATORIO_DTB_BRASIL_SUBDISTRITO.ods	2021-02-18 08:44:58	43
RELATORIO_DTB_BRASIL_SUBDISTRITO.xls	2021-02-18 08:47:36	230

Out[7]:

'Concluído'

In [8]:

```
nome_arquivo = 'RELATORIO_DTB_BRASIL_MUNICIPIO.xls'
IBGE_municios=pd.read_excel(nome_arquivo)
IBGE_municios.head(2)
```

Out[8]:

UF	Nome_UF	Região Geográfica Intermediária	Nome Região Geográfica Intermediária	Região Geográfica Imediata	Nome Região Geográfica Imediata	Mesorregião Geográfica	Nome_Mes
0	11	Rondônia	1102	Ji-Paraná	110005	Cacoal	2 Leste Rondônia
1	11	Rondônia	1102	Ji-Paraná	110005	Cacoal	2 Leste Rondônia

Como se vê, IBGE\_municios contém as 5570 linhas, como esperado.

**Para comparar com os demais dataset, precisaremos criar a coluna padrão municipio-uf.** Mas IBGE\_municípios não tem a sigla dos estados. Precisaremos, então, incluir em IBGE\_municios a sigla da uf (por meio de um merge com um dataframe auxiliar criado a partir de IBGE-PIB que conterá, exclusivamente, as variáveis '03\_nome\_uf' e '03\_uf').

Iniciamos coletando os valores únicos para nome do estado e a sigla do estado a partir do dataframe IBGE\_PIB ('03\_nome\_uf' e '03\_uf')

In [9]:

```
estado_sigla = IBGE_PIB[['03_nome_uf', '03_uf']].copy()
estado_sigla = estado_sigla.drop_duplicates()
estado_sigla.head()
```

Out[9]:

	03_nome_uf	03_uf
0	Rondônia	RO
52	Acre	AC
74	Amazonas	AM
136	Roraima	RR
151	Pará	PA

Vamos, agora, criar uma nova coluna com a sigla dos estados em IBGE\_municípios, resultado do merge com o df estado\_sigla acima criado.

In [10]:

```
IBGE_municípios = pd.merge(IBGE_municípios, estado_sigla, how="outer", left_on='Nome_UF', right_on='03_uf')
print(IBGE_municípios.shape)
IBGE_municípios.head(2)
```

(5570, 15)

Out[10]:

UF	Nome_UF	Região Geográfica Intermediária	Nome Região Geográfica Intermediária	Região Geográfica Imediata	Nome Região Geográfica Imediata	Mesorregião Geográfica	Nome_Mesoregião
0	11 Rondônia	1102	Ji-Paraná	110005	Cacoal	2 Leste Ronco	
1	11 Rondônia	1102	Ji-Paraná	110005	Cacoal	2 Leste Ronco	

Podemos, agora, criar a coluna padronizada 'municipio-uf' e, na sequência, excluir as variáveis que já não mais serão necessárias.

In [11]:

```
#Agora podemos criar a coluna padronizada 'municipio-uf'
IBGE_municpios['municipio-uf']=IBGE_municipios['Nome_Município'].str.upper()+'-'+'IBGE_muni

# Removendo caracteres especiais da coluna recém-criada.
IBGE_municipios['municipio-uf'] = IBGE_municipios['municipio-uf'].apply(remover_caracteres_)

#Podemos apagar a coluna "03_nome_uf" e "03_uf" criada com o merge no dataframe IBGE_munici
IBGE_municipios = IBGE_municipios.drop(['03_nome_uf','03_uf'],axis=1)

IBGE_municipios.head(2)
```

Out[11]:

	UF	Nome_UF	Região Geográfica Intermediária	Nome Região Geográfica Intermediária	Região Geográfica Imediata	Nome Região Geográfica Imediata	Mesorregião Geográfica	Nome_Mes
0	11	Rondônia	1102	Ji-Paraná	110005	Cacoal	2	Leste Ron
1	11	Rondônia	1102	Ji-Paraná	110005	Cacoal	2	Leste Ron

## 4. Merges dos datasets de interesse.

Para a criação da tabela consolidada, faremos uma sequência de merges (com os saneamentos necessários).  
e modo a uniformizar os resultados, privilegiaremos sempre as informações contidas em IBGE\_municipios (por se tratar da base mais atualizada). Assim:

- municípios que não mais constem dessa relação serão excluídos;
- municípios com grafias diferentes terão seus nomes ajustados a partir dessa referência;

O primeiro merge será entre os dataframes IBGE\_municipios e arrecadacao\_2018. Os demais merges terão por ponto de partida os merges anteriores.

### Merge 1 - 'IBGE\_municipios' e 'arrecadacao\_2018'.

In [12]:

```
# Iniciamos com um outer join para identificar as ocorrências que não são mútuas entre as t
print ('Shape de IBGE_municipios',IBGE_municipios.shape)
print ('Shape de arrecadacao_2018',arrecadacao_2018.shape)
Merge_01 = pd.merge(IBGE_municipios['municipio-uf'], arrecadacao_2018['municipio-uf'], how=
Merge_01.shape
print ('Shape de merge_01', Merge_01.shape)

## Identificando as divergências
diferencias=Merge_01[Merge_01['_merge']!='both']
print("\n", "Diferenças encontradas", diferencias.shape)
print (diferencias['_merge'].value_counts())
diferencias
```

	municipio-uf	_merge
197	ELDORADO DO CARAJAS-PA	left_only
259	SANTA IZABEL DO PARA-PA	left_only
436	SAO VALERIO-TO	left_only
440	TABOCAO-TO	left_only
1076	ACU-RN	left_only
1132	JANUARIO CICCO-RN	left_only
1165	OLHO D'AGUA DO BORGES-RN	left_only
1419	SAO DOMINGOS-PB	left_only
1482	BELEM DO SAO FRANCISCO-PE	left_only
1540	IGUARACY-PE	left_only
1561	LAGOA DE ITAENGA-PE	left_only
1752	AMARALDO DO SAO FRANCISCO-SE	left_only

Nota-se que há, a princípio, 33 municípios que aparecem somente em PIB\_municipios (indicados com o 'left\_only') e outros 39 municípios que aparecem apenas em arrecadacao\_2018 (indicados com 'right\_only').

Uma análise mais detida, porém, mostra que parte das inconsistências se dá por divergência de grafia (por exemplo: ACU-RN vs. ASSU-RN, ou GRAO-PARA-SC vs. GRAO PARA-SC) ou por uma mudança mais significativa de nomenclatura (como TABOCAO-TO vs. FORTALEZA DO TABOCAO-TO, ou EMBU DAS ARTES-SP vs. EMBU-SP). Há ainda situação curiosa de município com alteração total de seu nome (como o município de 'AUGUSTO SEVERO-RN' que passou a se chamar 'CAMPO GRANDE-RN')

Faremos, então, esses ajustes de grafia. Como privilegiaremos o contido em IBGE\_municipios, alteraremos sempre a grafia do dataframe à direita no merge.

Há uma situação interessante: oficialmente, o município de 'BOA SAUDE-RN', que passou a se chamar 'JANUARIO CICCO' em 1953 (quando elevado a município). Emenda da Câmara Municipal n.º 01, de 02-02-1991, porém, retornou seu nome para BOA SAUDE, mas como não houve até o momento homologação por Lei Estadual, o IBGE desconsidera essa alteração. Porém todas as referências são para Boa Saúde (RN) - vide o site da prefeitura local <http://www.boasaude.rn.gov.br/historia-do-municipio.html> (<http://www.boasaude.rn.gov.br/historia-do-municipio.html>). Como estamos privilegiando o definido pelo IBGE, manteremos JANUARIO CICCO.

In [13]:

```
# Como no merge1 o dataframe à direita é arrecadacao_2018, será nele que faremos as adequações
arrecadacao_2018['municipio-uf'] = arrecadacao_2018['municipio-uf'].apply(corrigar_nomes_mun)
```

Outra inconsistência diz respeito ao município de 'ASSIS CHATEAUBRIAND - PB' que passou a se chamar 'RIACHAO DO BACAMARTE-PB'. Ocorre que em arrecadacao\_2018 há informação de valores arrecadados para essas duas ocorrências. Somaremos então os dois valores, atribuiremos o resultado a 'RIACHAO DO BACAMARTE-PB' e excluiremos 'ASSIS CHATEAUBRIAND - PB'.

O mesmo ocorre com CAMPO DE SANTANA-PB e TACIMA-PB; e com LIVRAMENTO DO BRUMADO - BA e LIVRAMENTO DE NOSSA SENHORA -BA; e SANTAREM-PB e JOCA CLAUDINO-PB- serão mantidos os últimos e lhes atribuído a soma das arrecadações.

In [14]:

```
municipios_atuais = ['RIACHAO DO BACAMARTE-PB', 'TACIMA-PB', 'LIVRAMENTO DE NOSSA SENHORA-BA'
municipios_extintos = ['ASSIS CHATEAUBRIAND-PB', 'CAMPO DE SANTANA-PB', 'LIVRAMENTO DO BRUMA
duplas=zip(municipios_atuais, municipios_extintos)

for i in duplas:
    print("\n",i)

    ind_1 = arrecadacao_2018.index[arrecadacao_2018['municipio-uf']==i[0]].tolist()
    arrec_munic_1 = arrecadacao_2018.at[ind_1[0], '01_arrecadacao_2018']
    print (i[0]," - ",arrec_munic_1)

    ind_2 = arrecadacao_2018.index[arrecadacao_2018['municipio-uf']==i[1]].tolist()
    arrec_munic_2 = arrecadacao_2018.at[ind_2[0], '01_arrecadacao_2018']
    print (i[1]," - ",arrec_munic_2)

    soma_arrec = arrec_munic_1.astype(float) + arrec_munic_2.astype(float)
    print ("Soma"," - ", soma_arrec)

    arrecadacao_2018.at[ind_1[0], '01_arrecadacao_2018'] = soma_arrec
    print ("Novo valor para",i[0]," - ",arrecadacao_2018.at[ind_1[0],'01_arrecadacao_2018'])

('RIACHAO DO BACAMARTE-PB', 'ASSIS CHATEAUBRIAND-PB')
RIACHAO DO BACAMARTE-PB - 422067.12
ASSIS CHATEAUBRIAND-PB - 2696361.84000009
Soma - 3118428.9600000903
Novo valor para RIACHAO DO BACAMARTE-PB - 3118428.9600000903

('TACIMA-PB', 'CAMPO DE SANTANA-PB')
TACIMA-PB - 708332.35
CAMPO DE SANTANA-PB - 3574894.61000052
Soma - 4283226.96000052
Novo valor para TACIMA-PB - 4283226.96000052

('LIVRAMENTO DE NOSSA SENHORA-BA', 'LIVRAMENTO DO BRUMADO-BA')
LIVRAMENTO DE NOSSA SENHORA-BA - 12645132.32
LIVRAMENTO DO BRUMADO-BA - 12168718.809999
Soma - 24813851.129999
Novo valor para LIVRAMENTO DE NOSSA SENHORA-BA - 24813851.129999

('JOCA CLAUDINO-PB', 'SANTAREM-PB')
JOCA CLAUDINO-PB - 244080.8
SANTAREM-PB - 2029605.34000023
Soma - 2273686.1400002297
Novo valor para JOCA CLAUDINO-PB - 2273686.1400002297
```

Por fim, as ocorrências 'EXTERIOR-EX' e 'ZERADO--' não são de interesse e serão excluídas quando refizermos o merge.

## Refazer o merge, mas agora como o left join.

Saneadas as inconsistências acima, o Merge1 pode ser repetido, agora como LeftJoin de modo a mantermos todas as ocorrências de IBGE\_municipios e apenas as ocorrências correspondentes em "arrecadacao\_2018". Teremos, assim, mantidos os 5570 municípios.

In [15]:

```
print ('Shape de IBGE_municipios',IBGE_municipios.shape)
print ('Shape de arrecadacao_2018',arrecadacao_2018.shape)
Merge_01 = pd.merge(IBGE_municipios['municipio-uf'], arrecadacao_2018,how="left", on='muni
Merge_01.shape
print ('Shape de merge_01', Merge_01.shape)
```

Shape de IBGE\_municipios (5570, 14)  
 Shape de arrecadacao\_2018 (5576, 2)  
 Shape de merge\_01 (5570, 3)

In [16]:

```
# Chama a função para identificar se há valores faltantes ou NaN.
identificar_faltantes(Merge_01)
```

Shape do dataframe: (5570, 3)  
 Colunas que contêm valores faltantes: []  
 Número de linhas com valores faltantes: 0

Out[16]:

[]

In [17]:

```
#Podemos apagar a coluna '_merge' criada pelo comando de mesmo nome.
Merge_01 = Merge_01.drop('_merge', axis=1)
Merge_01.columns
```

Out[17]:

Index(['municipio-uf', '01\_arrecadacao\_2018'], dtype='object')

## Merge 2 - 'Merge\_01' e 'IBGE\_Empresas'.

Como já conhecemos boa parte das divergências, já podemos iniciar os procedimentos pela regularização das grafias dos nomes de municípios.

In [18]:

```
# Como no merge1 o dataframe à direita é arrecadacao_2018, será nele que faremos as adequações.
IBGE_Empresas['municipio-uf'] = IBGE_Empresas['municipio-uf'].apply(corrigir_nomes_municipi
```

Seguimos fazendo um merge ('outer') para identificar eventuais divergências.

In [19]:

```
# Iniciamos com um outer join para identificar as ocorrências que não são mútuas entre as t
print ('Shape de Merge_01',Merge_01.shape)
print ('Shape de IBGE_Empresas',IBGE_Empresas.shape)
Merge_02 = pd.merge(Merge_01['municipio-uf'], IBGE_Empresas['municipio-uf'],how="outer", o
Merge_02.shape
print ('Shape de merge_02', Merge_02.shape)

## Identificando as divergências
diferencias=Merge_02[Merge_02['_merge']!='both']
print("\n", "Diferenças encontradas", diferencias.shape)
print (diferencias['_merge'].value_counts())
diferencias
```

Shape de Merge\_01 (5570, 2)  
 Shape de IBGE\_Empresas (5570, 8)  
 Shape de merge\_02 (5570, 2)

Diferenças encontradas (0, 2)

_merge	Count
both	0
right_only	0
left_only	0

Name: \_merge, dtype: int64

Out[19]:

municipio-uf	_merge
--------------	--------

Não há divergências. Podemos fazer o merge definitivo.

## Refazer o merge, mas agora como o left join.

Não havendo divergências remanescentes, o Merge pode ser repetido, agora como LeftJoin de modo a mantermos todas as ocorrências de 'Merge\_01' e apenas as ocorrências correspondentes em "IBGE\_Empresas". Teremos, assim, mantidos os 5570 municípios.

In [20]:

```
print ('Shape de Merge_01', Merge_01.shape)
print ('Shape de IBGE_Empresas',IBGE_Empresas.shape)
Merge_02 = pd.merge(Merge_01, IBGE_Empresas,how="left", on='municipio-uf',indicator=True,s
Merge_02.shape
print ('Shape de merge_02', Merge_02.shape)

Shape de Merge_01 (5570, 2)
Shape de IBGE_Empresas (5570, 8)
Shape de merge_02 (5570, 10)
```

In [21]:

```
# Chama a função para identificar se há valores faltantes ou NaN.
identificar_faltantes(Merge_01)
```

Shape do dataframe: (5570, 2)  
 Colunas que contêm valores faltantes: []  
 Número de linhas com valores faltantes: 0

Out[21]:

[]

In [22]:

```
#Podemos apagar a coluna '_merge' criada pelo comando de mesmo nome.
Merge_02 = Merge_02.drop('_merge', axis=1)
Merge_02.columns
```

Out[22]:

```
Index(['municipio-uf', '01_arrecadacao_2018', '02_unidades',
       '02_unidades_atuantes_%', '02_ocupados_total',
       '02_ocupados_assalariados_%', '02_ocupados_assalariados_medio',
       '02_salario_medio_mensal_em_SM', '02_salario_medio_mensal_em_Reais'],
      dtype='object')
```

## Merge 3 - 'Merge\_02' e 'IBGE\_PIB'.

Como já conhecemos boa parte das divergências, já podemos iniciar os procedimentos pela regularização das grafias dos nomes de municípios.

In [23]:

```
# Como no merge1 o dataframe à direita é arrecadacao_2018, será nele que faremos as adequações
IBGE_PIB['municipio-uf'] = IBGE_PIB['municipio-uf'].apply(corrigir_nomes_municipios)
```

Seguimos fazendo um merge ('outer') para identificar eventuais divergências.

In [24]:

```
# Iniciamos com um outer join para identificar as ocorrências que não são mútuas entre as t
print ('Shape de Merge_02',Merge_02.shape)
print ('Shape de IBGE_PIB',IBGE_PIB.shape)
Merge_03 = pd.merge(Merge_02['municipio-uf'], IBGE_PIB['municipio-uf'],how="outer", on='mu
Merge_02.shape
print ('Shape de merge_03', Merge_03.shape)

## Identificando as divergências
diferencias=Merge_03[Merge_03['_merge']!='both']
print("\n", "Diferenças encontradas", diferencias.shape)
print (diferencias['_merge'].value_counts())
diferencias
```

Shape de Merge\_02 (5570, 9)  
 Shape de IBGE\_PIB (5570, 20)  
 Shape de merge\_03 (5570, 2)

Diferenças encontradas (0, 2)

_merge	count
both	0
right_only	0
left_only	0

Name: \_merge, dtype: int64

Out[24]:

municipio-uf	_merge
--------------	--------

Não há divergências. Podemos fazer o merge definitivo.

## Refazer o merge, mas agora como o left join.

Não havendo divergências remanescentes, o Merge pode ser repetido, agora como LeftJoin de modo a mantermos todas as ocorrências de 'Merge\_02' e apenas as ocorrências correspondentes em "IBGE\_PIB". Teremos, assim, mantidos os 5570 municípios.

In [25]:

```
print ('Shape de Merge_02', Merge_02.shape)
print ('Shape de IBGE_PIB',IBGE_PIB.shape)
Merge_03 = pd.merge(Merge_02, IBGE_PIB,how="left", on='municipio-uf',indicator=True,suffix_
Merge_03.shape
print ('Shape de merge_03', Merge_03.shape)

Shape de Merge_02 (5570, 9)
Shape de IBGE_PIB (5570, 20)
Shape de merge_03 (5570, 29)
```

In [26]:

```
# Chama a função para identificar se há valores faltantes ou NaN.
identificar_faltantes(Merge_02)
```

Shape do dataframe: (5570, 9)  
 Colunas que contêm valores faltantes: []  
 Número de linhas com valores faltantes: 0

Out[26]:

[]

In [27]:

```
#Podemos apagar a coluna '_merge' criada pelo comando de mesmo nome.
Merge_03 = Merge_03.drop('_merge', axis=1)
Merge_03.columns
```

Out[27]:

```
Index(['municipio-uf', '01_arrecadacao_2018', '02_unidades',
       '02_unidades_atuantes_%', '02_ocupados_total',
       '02_ocupados_assalariados_%', '02_ocupados_assalariados_medio',
       '02_salario_medio_mensal_em_SM', '02_salario_medio_mensal_em_Reais',
       '03_nome_ue', '03_ue', '03_cod_municipio', '03_grande_regiao',
       '03_mesorregiao', '03_microregiao', '03_regiao_imediata',
       '03_regiao_intermediaria', '03_vlr_adic_bruto_total',
       '03_vlr_adic_bruto_agropecuaria_%', '03_vlr_adic_bruto_industria_%',
       '03_vlr_adic_bruto_servicos_%', '03_vlr_adic_bruto_administracao_%',
       '03_impostos_sobre_produtos', '03_pib', '03_pib_per_capita',
       '03_atividade_principal_primeira', '03_atividade_principal_segunda',
       '03_atividade_principal_terceira'],
      dtype='object')
```

## Merge 4 - 'Merge\_03' e 'IBGE\_Residentes'.

Como já conhecemos boa parte das divergências, já podemos iniciar os procedimentos pela regularização das grafias dos nomes de municípios.

In [28]:

```
# Como no merge1 o dataframe à direita é arrecadacao_2018, será nele que faremos as adequações
IBGE_Residentes['municipio-uf'] = IBGE_Residentes['municipio-uf'].apply(corrigir_nomes_muni
```

Seguimos fazendo um merge ('outer') para identificar eventuais divergências.

In [29]:

```
# Iniciamos com um outer join para identificar as ocorrências que não são mútuas entre as t
print ('Shape de Merge_03',Merge_03.shape)
print ('Shape de IBGE_Residentes',IBGE_Residentes.shape)
Merge_04 = pd.merge(Merge_03['municipio-uf'], IBGE_Residentes['municipio-uf'],how="outer",
Merge_04.shape
print ('Shape de merge_04', Merge_04.shape)

## Identificando as divergências
diferencias=Merge_04[Merge_04['_merge']!='both']
print("\n", "Diferenças encontradas", diferencias.shape)
print (diferencias['_merge'].value_counts())
diferencias

```

Shape de Merge\_03 (5570, 28)  
 Shape de IBGE\_Residentes (5570, 2)  
 Shape de merge\_04 (5570, 2)

Diferenças encontradas (0, 2)

_merge	count
both	0
right_only	0
left_only	0

Name: \_merge, dtype: int64

Out[29]:

municipio-uf	_merge
--------------	--------

Não há divergências. Podemos fazer o merge definitivo.

## Refazer o merge, mas agora como o left join.

Não havendo divergências remanescentes, o Merge pode ser repetido, agora como LeftJoin de modo a mantermos todas as ocorrências de 'Merge\_03' e apenas as ocorrências correspondentes em "IBGE\_Residentes". Teremos, assim, mantidos os 5570 municípios.

In [30]:

```
print ('Shape de Merge_03', Merge_03.shape)
print ('Shape de IBGE_Residentes',IBGE_Residentes.shape)
Merge_04 = pd.merge(Merge_03, IBGE_Residentes,how="left", on='municipio-uf',indicator=True
Merge_04.shape
print ('Shape de merge_04', Merge_04.shape)


```

Shape de Merge\_03 (5570, 28)  
 Shape de IBGE\_Residentes (5570, 2)  
 Shape de merge\_04 (5570, 30)

In [31]:

```
# Chama a função para identificar se há valores faltantes ou NaN.
identificar_faltantes(Merge_04)
```

Shape do dataframe: (5570, 30)  
 Colunas que contêm valores faltantes: []  
 Número de linhas com valores faltantes: 0

Out[31]:

[]

In [32]:

```
#Podemos apagar a coluna '_merge' criada pelo comando de mesmo nome.
Merge_04 = Merge_04.drop('_merge', axis=1)
Merge_04.columns
```

Out[32]:

```
Index(['municipio-uf', '01_arrecadacao_2018', '02_unidades',
       '02_unidades_atuantes_%', '02_ocupados_total',
       '02_ocupados_assalariados_%', '02_ocupados_assalariados_medio',
       '02_salario_medio_mensal_em_SM', '02_salario_medio_mensal_em_Reais',
       '03_nome_ue', '03_ue', '03_cod_municipio', '03_grande_regiao',
       '03_mesorregiao', '03_microregiao', '03_regiao_imediata',
       '03_regiao_intermediaria', '03_vlr_adic_bruto_total',
       '03_vlr_adic_bruto_agropecuaria_%', '03_vlr_adic_bruto_industria_%',
       '03_vlr_adic_bruto_servicos_%', '03_vlr_adic_bruto_administracao_%',
       '03_impostos_sobre_produtos', '03_pib', '03_pib_per_capita',
       '03_atividade_principal_primeira', '03_atividade_principal_segunda',
       '03_atividade_principal_terceira', '04_populacao_residente_2018'],
      dtype='object')
```

## Merge 5 - 'Merge\_04' e 'IBGE\_Censo\_Idade'.

Como já conhecemos boa parte das divergências, já podemos iniciar os procedimentos pela regularização das grafias dos nomes de municípios.

In [33]:

```
# Como no merge1 o dataframe à direita é arrecadacao_2018, será nele que faremos as adequações
IBGE_Censo_Idade['municipio-uf'] = IBGE_Censo_Idade['municipio-uf'].apply(corrigir_nomes_mu
```

Seguimos fazendo um merge ('outer') para identificar eventuais divergências.

In [34]:

```
# Iniciamos com um outer join para identificar as ocorrências que não são mútuas entre as t
print ('Shape de Merge_04',Merge_04.shape)
print ('Shape de IBGE_Censo_Idade',IBGE_Censo_Idade.shape)
Merge_05 = pd.merge(Merge_04['municipio-uf'], IBGE_Censo_Idade['municipio-uf'],how="outer")
Merge_05.shape
print ('Shape de merge_05', Merge_05.shape)
```

*## Identificando as divergências*

```
diferencias=Merge_05[Merge_05['_merge']!='both']
print("\n", "Diferenças encontradas", diferencias.shape)
print (diferencias['_merge'].value_counts())
diferencias
```

```
Shape de Merge_04 (5570, 29)
Shape de IBGE_Censo_Idade (5565, 6)
Shape de merge_05 (5570, 2)
```

```
Diferenças encontradas (5, 2)
left_only      5
both          0
right_only    0
Name: _merge, dtype: int64
```

Out[34]:

	municipio-uf	_merge
224	MOJUI DOS CAMPOS-PA	left_only
4341	BALNEARIO RINCAO-SC	left_only
4505	PESCARIA BRAVA-SC	left_only
4923	PINTO BANDEIRA-RS	left_only
5160	PARAISO DAS AGUAS-MS	left_only

Vemos aqui que há 5 (cinco) ocorrências em Merge\_04 que não existem em IBGE\_Censo\_Idade. Tratam-se de municípios instalados após a realização do Censo de 2010.

Assim, como a intenção é avaliar o efeito de variáveis, como as do censo, na previsão da arrecadação, esses cinco municípios deverão ser excluídos, vez que não apresentarão a totalidade das informações.

Para isso faremos um inner join.

## Refazer o merge, mas agora como o inner join.

Havendo divergências remanescentes, o Merge pode ser repetido, agora como Inner Join de modo a mantermos todas as ocorrências mútuas de 'Merge\_04' e "IBGE\_Censo\_Idade". Passaremos a ter, então, 5565 municípios.

In [35]:

```
print ('Shape de Merge_04', Merge_04.shape)
print ('Shape de IBGE_Censo_Idade', IBGE_Censo_Idade.shape)
Merge_05 = pd.merge(Merge_04, IBGE_Censo_Idade, how="inner", on='municipio-uf', indicator=True)
Merge_05.shape
print ('Shape de merge_05', Merge_05.shape)
```

Shape de Merge\_04 (5570, 29)  
 Shape de IBGE\_Censo\_Idade (5565, 6)  
 Shape de merge\_05 (5565, 35)

In [36]:

```
# Chama a função para identificar se há valores faltantes ou NaN.
identificar_faltantes(Merge_05)
```

Shape do dataframe: (5565, 35)  
 Colunas que contêm valores faltantes: []  
 Número de linhas com valores faltantes: 0

Out[36]:

[]

In [37]:

```
#Podemos apagar a coluna '_merge' criada pelo comando de mesmo nome.
Merge_05 = Merge_05.drop('_merge', axis=1)
Merge_05.columns
```

Out[37]:

```
Index(['municipio-uf', '01_arrecadacao_2018', '02_unidades',
       '02_unidades_atuantes_%', '02_ocupados_total',
       '02_ocupados_assalariados_%', '02_ocupados_assalariados_medio',
       '02_salario_medio_mensal_em_SM', '02_salario_medio_mensal_em_Reais',
       '03_nome_ue', '03_ue', '03_cod_municipio', '03_grande_regiao',
       '03_mesorregiao', '03_microregiao', '03_regiao_imediata',
       '03_regiao_intermediaria', '03_vlr_adic_bruto_total',
       '03_vlr_adic_bruto_agropecuaria_%', '03_vlr_adic_bruto_industria_%',
       '03_vlr_adic_bruto_servicos_%', '03_vlr_adic_bruto_administracao_%',
       '03_impostos_sobre_produtos', '03_pib', '03_pib_per_capita',
       '03_atividade_principal_primeira', '03_atividade_principal_segunda',
       '03_atividade_principal_terceira', '04_populacao_residente_2018',
       '05_total_residentes', '05_0-19_anos_%', '05_20-39_anos_%',
       '05_40-59_anos_%', '05_60+_anos%'],
      dtype='object')
```

## Merge 6 - 'Merge\_05' e 'IBGE\_Censo\_Instrucao'.

Como já conhecemos boa parte das divergências, já podemos iniciar os procedimentos pela regularização das grafias dos nomes de municípios.

In [38]:

```
# Como no merge1 o dataframe à direita é arrecadacao_2018, será nele que faremos as adequações
IBGE_Censo_Instrucao['municipio-uf'] = IBGE_Censo_Instrucao['municipio-uf'].apply(corrigir_
```

Seguimos fazendo um merge ('outer') para identificar eventuais divergências.

In [39]:

```
# Iniciamos com um outer join para identificar as ocorrências que não são mútuas entre as tabelas
print ('Shape de Merge_05',Merge_05.shape)
print ('Shape de IBGE_Censo_Instrucao',IBGE_Censo_Instrucao.shape)
Merge_06 = pd.merge(Merge_05['municipio-uf'], IBGE_Censo_Instrucao['municipio-uf'],how="outer")
Merge_06.shape
print ('Shape de merge_06', Merge_06.shape)

## Identificando as divergências
diferencias=Merge_06[Merge_06['_merge']!='both']
print("\n", "Diferenças encontradas", diferencias.shape)
print (diferencias['_merge'].value_counts())
diferencias
```

```
Shape de Merge_05 (5565, 34)
Shape de IBGE_Censo_Instrucao (5565, 7)
Shape de merge_06 (5565, 2)
```

```
Diferenças encontradas (0, 2)
both          0
right_only    0
left_only     0
Name: _merge, dtype: int64
```

Out[39]:

municipio-uf	_merge
0	both
1	right_only
2	left_only

## Refazer o merge, mas agora como o inner join.

Como não há divergências remanescentes, o Merge pode ser repetido, agora como Inner Join de modo a mantermos todas as ocorrências mútuas de 'Merge\_05' e "IBGE\_Censo\_Instrucao". Continuaremos a ter, então, 5565 municípios.

In [40]:

```
print ('Shape de Merge_05', Merge_05.shape)
print ('Shape de IBGE_Censo_Instrucao',IBGE_Censo_Instrucao.shape)
Merge_06 = pd.merge(Merge_05, IBGE_Censo_Instrucao,how="inner", on='municipio-uf', indicator=True)
Merge_06.shape
print ('Shape de merge_06', Merge_06.shape)
```

```
Shape de Merge_05 (5565, 34)
Shape de IBGE_Censo_Instrucao (5565, 7)
Shape de merge_06 (5565, 41)
```

In [41]:

```
# Chama a função para identificar se há valores faltantes ou NaN.
identificar_faltantes(Merge_06)
```

Shape do dataframe: (5565, 41)  
 Colunas que contêm valores faltantes: []  
 Número de linhas com valores faltantes: 0

Out[41]:

[]

In [42]:

```
#Podemos apagar a coluna '_merge' criada pelo comando de mesmo nome.
Merge_06 = Merge_06.drop('_merge', axis=1)
Merge_06.columns
```

Out[42]:

```
Index(['municipio-uf', '01_arrecadacao_2018', '02_unidades',
       '02_unidades_atuantes_%', '02_ocupados_total',
       '02_ocupados_assalariados_%', '02_ocupados_assalariados_medio',
       '02_salario_medio_mensal_em_SM', '02_salario_medio_mensal_em_Reais',
       '03_nome_ue', '03_ue', '03_cod_municipio', '03_grande_regiao',
       '03_mesorregiao', '03_microregiao', '03_regiao_imediata',
       '03_regiao_intermediaria', '03_vlr_adic_bruto_total',
       '03_vlr_adic_bruto_agropecuaria_%', '03_vlr_adic_bruto_industria_%',
       '03_vlr_adic_bruto_servicos_%', '03_vlr_adic_bruto_administracao_%',
       '03_impostos_sobre_produtos', '03_pib', '03_pib_per_capita',
       '03_atividade_principal_primeira', '03_atividade_principal_segunda',
       '03_atividade_principal_terceira', '04_populacao_residente_2018',
       '05_total_residentes', '05_0-19_anos_%', '05_20-39_anos_%',
       '05_40-59_anos_%', '05_60+anos_%', '06_total_instrucao',
       '06_Sem_Instrucao_e_Fundamental_Incompleto%', '06_Medio_Incompleto
       _%',
       '06_Superior_Incompleto_%', '06_Superior_Completo_%',
       '06_Indeterminado_%'],
      dtype='object')
```

## Merge 7 - 'Merge\_06' e 'IBGE\_Censo\_Rendimento'.

Como já conhecemos boa parte das divergências, já podemos iniciar os procedimentos pela regularização das grafias dos nomes de municípios.

In [43]:

```
# Como no merge1 o dataframe à direita é arrecadacao_2018, será nele que faremos as adequações
IBGE_Censo_Rendimento['municipio-uf'] = IBGE_Censo_Rendimento['municipio-uf'].apply(corrigir)
```

Seguimos fazendo um merge ('outer') para identificar eventuais divergências.

In [44]:

```
# Iniciamos com um outer join para identificar as ocorrências que não são mútuas entre as t
print ('Shape de Merge_06',Merge_06.shape)
print ('Shape de IBGE_Censo_Rendimento',IBGE_Censo_Rendimento.shape)
Merge_07 = pd.merge(Merge_06['municipio-uf'], IBGE_Censo_Rendimento['municipio-uf'],how="o
Merge_07.shape
print ('Shape de merge_07', Merge_07.shape)

## Identificando as divergências
diferencias=Merge_07[Merge_07['_merge']!='both']
print("\n", "Diferenças encontradas", diferencias.shape)
print (diferencias['_merge'].value_counts())
diferencias
```

Shape de Merge\_06 (5565, 40)  
 Shape de IBGE\_Censo\_Rendimento (5565, 10)  
 Shape de merge\_07 (5565, 2)

Diferenças encontradas (0, 2)

_merge	Count
both	0
right_only	0
left_only	0

Name: \_merge, dtype: int64

Out[44]:

---

municipio-uf	_merge
--------------	--------

---

## Refazer o merge, mas agora como o inner join.

Como não há divergências remanescentes, o Merge pode ser repetido, agora como Inner Join de modo a mantermos todas as ocorrências mútuas de 'Merge\_06' e "IBGE\_Censo\_Rendimento". Continuaremos a ter, então, 5565 municípios.

In [45]:

```
print ('Shape de Merge_06', Merge_06.shape)
print ('Shape de IBGE_Censo_Rendimento',IBGE_Censo_Rendimento.shape)
Merge_07 = pd.merge(Merge_06, IBGE_Censo_Rendimento,how="inner", on='municipio-uf',indicat
Merge_07.shape
print ('Shape de merge_07', Merge_07.shape)

Shape de Merge_06 (5565, 40)
Shape de IBGE_Censo_Rendimento (5565, 10)
Shape de merge_07 (5565, 50)
```

In [46]:

```
# Chama a função para identificar se há valores faltantes ou NaN.
identificar_faltantes(Merge_07)
```

Shape do dataframe: (5565, 50)  
 Colunas que contêm valores faltantes: []  
 Número de linhas com valores faltantes: 0

Out[46]:

[]

In [47]:

```
#Podemos apagar a coluna '_merge' criada pelo comando de mesmo nome.
Merge_07 = Merge_07.drop('_merge', axis=1)
Merge_07.columns
```

Out[47]:

```
Index(['municipio-uf', '01_arrecadacao_2018', '02_unidades',
       '02_unidades_atuantes_%', '02_ocupados_total',
       '02_ocupados_assalariados_%', '02_ocupados_assalariados_medio',
       '02_salario_medio_mensal_em_SM', '02_salario_medio_mensal_em_Reais',
       '03_nome_ue', '03_ue', '03_cod_municipio', '03_grande_regiao',
       '03_mesorregiao', '03_microregiao', '03_regiao_imediata',
       '03_regiao_intermediaria', '03_vlr_adic_bruto_total',
       '03_vlr_adic_bruto_agropecuaria_%', '03_vlr_adic_bruto_industria_%',
       '03_vlr_adic_bruto_servicos_%', '03_vlr_adic_bruto_administracao_%',
       '03_impostos_sobre_produtos', '03_pib', '03_pib_per_capita',
       '03_atividade_principal_primeira', '03_atividade_principal_segunda',
       '03_atividade_principal_terceira', '04_populacao_residente_2018',
       '05_total_residentes', '05_0-19_anos_%', '05_20-39_anos_%',
       '05_40-59_anos_%', '05_60+_anos_%', '06_total_instrucao',
       '06_Sem_Instrucao_e_Fundamental_Incompleto%', '06_Medio_Incompleto
       _%',
       '06_Superior_Incompleto%', '06_Superior_Completo%',
       '06_Indeterminado%', '07_total_rendimentos', '07_Ate_1SM_%',
       '07_Ate_2SM_%', '07_Ate_3SM_%', '07_Ate_5SM_%', '07_Ate_10SM_%',
       '07_Ate_20SM_%', '07_20SM+_%', '07_Sem_Rendimento%'],
      dtype='object')
```

In [48]:

```
print(Merge_07.shape)
Merge_07.info()
```

```
(5565, 49)
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5565 entries, 0 to 5564
Data columns (total 49 columns):
 #   Column           Non-Null Count Dtype  
 --- 
 0   municipio-uf    5565 non-null   object  
 1   01_arrecadacao_2018 5565 non-null   float64 
 2   02_unidades      5565 non-null   int64   
 3   02_unidades_atuantes_% 5565 non-null   float64 
 4   02_ocupados_total 5565 non-null   int64   
 5   02_ocupados_assalariados_% 5565 non-null   float64 
 6   02_ocupados_assalariados_medio 5565 non-null   float64 
 7   02_salario_medio_mensal_em_SM 5565 non-null   float64 
 8   02_salario_medio_mensal_em_Reais 5565 non-null   float64 
 9   03_nome_uf        5565 non-null   object  
 10  03_uf             5565 non-null   object  
 11  03_cod_municipio 5565 non-null   int64   
 12  03_grande_regiao 5565 non-null   object  
 13  03_mesorregiao   5565 non-null   object  
 14  03_microregiao   5565 non-null   object  
 15  03_regiao_imediata 5565 non-null   object  
 16  03_regiao_intermediaria 5565 non-null   object  
 17  03_vlr_adic_bruto_total 5565 non-null   float64 
 18  03_vlr_adic_bruto_agropecuaria_% 5565 non-null   float64 
 19  03_vlr_adic_bruto_industria_% 5565 non-null   float64 
 20  03_vlr_adic_bruto_servicos_% 5565 non-null   float64 
 21  03_vlr_adic_bruto_administracao_% 5565 non-null   float64 
 22  03_impostos_sobre_produtos 5565 non-null   float64 
 23  03_pib            5565 non-null   float64 
 24  03_pib_per_capita 5565 non-null   float64 
 25  03_atividade_principal_primeira 5565 non-null   object  
 26  03_atividade_principal_segunda 5565 non-null   object  
 27  03_atividade_principal_terceira 5565 non-null   object  
 28  04_populacao_residente_2018 5565 non-null   int64   
 29  05_total_residentes 5565 non-null   float64 
 30  05_0-19_anos_% 5565 non-null   float64 
 31  05_20-39_anos_% 5565 non-null   float64 
 32  05_40-59_anos_% 5565 non-null   float64 
 33  05_60+_anos_% 5565 non-null   float64 
 34  06_total_instrucao 5565 non-null   int64   
 35  06_Sem_Instrucao_e_Fundamental_Incompleto% 5565 non-null   float64 
 36  06_Medio_Incompleto_% 5565 non-null   float64 
 37  06_Superior_Incompleto_% 5565 non-null   float64 
 38  06_Superior_Completo_% 5565 non-null   float64 
 39  06_Indeterminado_% 5565 non-null   float64 
 40  07_total_rendimentos 5565 non-null   int64   
 41  07_Ate_1SM_% 5565 non-null   float64 
 42  07_Ate_2SM_% 5565 non-null   float64 
 43  07_Ate_3SM_% 5565 non-null   float64 
 44  07_Ate_5SM_% 5565 non-null   float64 
 45  07_Ate_10SM_% 5565 non-null   float64 
 46  07_Ate_20SM_% 5565 non-null   float64 
 47  07_20SM+_% 5565 non-null   float64 
 48  07_Sem_Rendimento_% 5565 non-null   float64
```

```
dtypes: float64(32), int64(6), object(11)
memory usage: 2.1+ MB
```



## 5. Criação do *dataset consolidado*

Cria a tabela que será a base para a exploração dos dados e aplicação dos modelos de machine learning.

In [49]:

```
Merge_07.to_csv('ntbk 02 - dataset consolidado.csv',index=False)
```

# Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb

O objetivo deste notebook é analisar e explorar dos dados contidos no *dataset ntbk 02 - dataset consolidado.csv*, criado no *Ntbk 02. Criação do dataset consolidado - merges e saneamentos* - vide Apêndice VII.

Está estruturado da seguinte maneira:

- 1. Ações Preliminares;
- 2. Exploração e transformação dos dados (conversão de bases):
  - 2.1. Sumários estatísticos;
  - 2.2. Analisando a variável de interesse (label): '01\_arrecadacao\_2018';
  - 2.3. Análise visual das variáveis numéricas;
  - 2.4. Identificação e tratamento de multicolinearidades;
- 3. Exclusão de variáveis que não são de interesse para criação dos modelos;
- 4. Criação do dataset consolidado 'ntbk 03 - dataset consolidado - maiores correlacoes.csv'

## 1. Ações Preliminares

### - Importação das bibliotecas de interesse.

In [1]:

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

pd.set_option('max_columns',200)
pd.set_option('max_rows',500)

%matplotlib inline
```

### - Declaração de funções que serão utilizadas no notebook.

In [2]:

```
# Imprime gráfico de dispersão para comparação de duas variáveis.

def imprime_pairplot(df, colunas_a_exibir, nome_arquivo = 'Ntbk 03 - Matriz Scatter.png'):
    g = sns.pairplot(df[colunas_a_exibir], height=2)

    #O código abaixo tem por objetivo editar a posição dos labels dos eixos 'y' (deixando-o
    #e 'x' (inclinando-o para evitar a sobreposição)
    for axes in g.axes.flat:
        axes.set_ylabel(axes.get_ylabel(), rotation=0, horizontalalignment='right', fontsize=12)
        axes.set_xlabel(axes.get_xlabel(), rotation=20, fontsize=12)
    #plt.savefig(nome_arquivo)
    plt.show()

    return 'Concluído'
```

In [3]:

```
#Função para a impressão de um boxplot e de um histograma.

def plotstats(df, col):

    #plt.rc('font', size=8)
    #plt.rc('axes', titlesize=8)
    plt.rc('axes', labelsize=12)
    plt.rc('xtick', labelsize=8)
    plt.rc('ytick', labelsize=8)
    #plt.rc('Legend', fontsize=8)
    #plt.rc('figure', titlesize=12)

    ## Preparação para impressão dos dois gráficos sobrepostos
    fig, ax = plt.subplots(2, 1, figsize = (12,8))

    ## Primeiro o boxplot
    df.dropna().boxplot(col, ax = ax[0], vert=False, return_type='dict', rot=0, fontsize=12)

    ## Agora o histograma
    temp = df[col].values
    ax[1].hist(temp, bins = 30, alpha = 0.7)
    plt.ylabel('Número de Cidades')
    plt.xlabel(col)

    return [col]
```

In [4]:

```
# Converte variáveis numéricas para a base logarítmica
# Pode ser passado por parâmetro uma única variável, ou mesmo o dataframe inteiro.

def converte_para_ln(df):
    cols = df.columns.tolist()
    contador = 0
    for col in cols:

        if df[col].dtype in [np.int64, np.int32, np.float64]:
            nova_coluna = col + "_ln"
            # A linha abaixo calcula o Log de cada valor das colunas numéricas
            # O abs para x é porque o np.log retorna erro para valores negativos, usaremos,
            # A função Log retornará um erro (-inf) quando o x = 0
            # Assim, quando o valor for zero, a função retornará -13,82 (que é o Ln de 0,00
            df_transformado[nova_coluna]=df[col].apply(lambda x: np.log(abs(x)) if x!= 0 else 0)
            contador = contador + 1
    print ('Colunas criadas:',contador)
    return('Concluído')
```

In [5]:

```
# Calcula os limites do boxplot

def limites_min_max_boxplot(df, col):
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    menor_limite = Q1 - (1.5 * IQR)
    maior_limite = Q3 + (1.5 * IQR)
    u = max(df[col][df[col]<maior_limite])
    u_Reais = np.exp(u)
    l = min(df[col][df[col]>menor_limite])
    l_Reais = np.exp(l)
    return [u,l,u_Reais,l_Reais]
```

In [6]:

```
#Cria gráficos do tipo boxplot.

def cria_boxplots(df,col_ref):
    import numpy as np
    import matplotlib.pyplot as plt

    plt.rc('axes', labelsize=20)
    plt.rc('xtick', labelsize=20)
    plt.rc('ytick', labelsize=20)

    contador = 1
    cols = df.columns.tolist()[:-1]
    for col in cols:
        if(df[col].dtype in [np.int64, np.int32, np.float64]) and col != col_ref:
            fig = plt.figure(figsize = (20,10))
            fig.clf()
            ax = fig.gca()
            df.boxplot(column = [col], ax = ax, by = [col_ref])
            #nome_arquivo = 'Ntbk 03 - Boxplots - ' + str(contador) + '.png'
            #plt.savefig(nome_arquivo)
            contador = contador + 1
    return('Concluído')
```

In [7]:

```
# Classifica o dataset a partir do valor de 01_arrecadacao_2018

#K = 5
#Amplitude = 26.36 - 12.94 = 13.42
#Intervalo = Amplitude / K
#Faixa_1 0 |- 15.623999999999999
#Faixa_2 15.623999999999999 |- 18.308
#Faixa_3 18.308 |-20.991999999999997
#Faixa_4 20.991999999999997 |- 23.676000000000002
#Faixa_5 23.676000000000002 |- 26.36

def faixa_arrecadacao(row):
    if row['01_arrecadacao_2018_ln']<Faixa_1:
        return '1'
    elif row['01_arrecadacao_2018_ln']<Faixa_2:
        return '2'
    elif row['01_arrecadacao_2018_ln']<Faixa_3:
        return '3'
    elif row['01_arrecadacao_2018_ln']<Faixa_4:
        return '4'
    elif row['01_arrecadacao_2018_ln']<=Faixa_5:
        return '5'
```

## - Importação do dataframe com os dados de interesse.

Aqui importamos a base 'ntbk 02 - dataset consolidado.csv', que é resultado da integração das bases originais coletadas para o estudo aqui em andamento.

In [8]:

```
import pandas as pd
import numpy as np

df_original=pd.read_csv('ntbk 02 - dataset consolidado.csv')

print(df_original.shape)
print (df_original.dtypes)
```

```
(5565, 49)
municipio-uf                         object
01_arrecadacao_2018                   float64
02_unidades                          int64
02_unidades_atuantes_%                float64
02_ocupados_total                     int64
02_ocupados_assalariados_%           float64
02_ocupados_assalariados_medio       float64
02_salario_medio_mensal_em_SM        float64
02_salario_medio_mensal_em_Reais     float64
03_nome_uf                           object
03_uf                                 object
03_cod_municipio                      int64
03_grande_regiao                      object
03_mesorregiao                        object
03_microregiao                        object
03_regiao_imediata                   object
03_regiao_intermediaria               object
03_vlr_adic_bruto_total               float64
03_vlr_adic_bruto_agropecuaria_%    float64
03_vlr_adic_bruto_industria_%       float64
03_vlr_adic_bruto_servicos_%        float64
03_vlr_adic_bruto_administracao_%   float64
03_impostos_sobre_produtos           float64
03_pib                               float64
03_pib_per_capita                     float64
03_atividade_principal_primeira     object
03_atividade_principal_segunda       object
03_atividade_principal_terceira      object
04_populacao_residente_2018          int64
05_total_residentes                  float64
05_0-19_anos_%                      float64
05_20-39_anos_%                    float64
05_40-59_anos_%                    float64
05_60+_anos_%                      float64
06_total_instrução                  int64
06_Sem_Instrução_e_Fundamental_Incompleto% float64
06_Medio_Incompleto_%               float64
06_Superior_Incompleto_%            float64
06_Superior_Completo_%              float64
06_Indeterminado_%                 float64
07_total_rendimentos                int64
07_Ate_1SM_%                        float64
07_Ate_2SM_%                        float64
07_Ate_3SM_%                        float64
07_Ate_5SM_%                        float64
07_Ate_10SM_%                       float64
07_Ate_20SM_%                       float64
07_20SM+_%                          float64
```

```
07_Sem_Rendimento_%
dtype: object
```

float64

Vê-se que a base original contém 5565 linhas e 49 colunas. Dessas, 11 são do tipo *object* e 38 numéricas.

No entanto, a variável '03\_cod\_municipio' embora numérica, deve ser convertida para o tipo str. Passaremos a ter, assim, 12 variáveis do tipo *object* e 37 numéricas.

In [9]:

```
# Primeiramente converteremos a variável 03_cod_municipio de int64 para str.
df_original['03_cod_municipio']=df_original['03_cod_municipio'].astype(str)
```

## 2. Exploração e transformação dos dados (conversão de bases)

### 2.1. Sumários estatísticos

Iniciaremos com um sumário estatístico das colunas com dados numéricos.

Interessante notar a diferença da magnitude entre as variáveis (colunas do *dataframe* ou *features*). Por exemplo: os valores da coluna **01\_arrecadacao\_2018** chegam à ordem das centenas de bilhões (vide o **max 280440706625.21**); os da coluna **02\_ocupados\_total**, à ordem dos milhões (vide o **max 5571893.0**); os da coluna **02\_salario\_medio\_mensal\_em\_SM** são inferiores a uma dezena (vide o **máx 6,9**); e os valores da coluna **02\_ocupados\_assalariados\_%** estão, apenas, **entre 0 e 100**.

Mesmo a amplitude dos valores da própria coluna **01\_arrecadacao\_2018** é significativa. Enquanto a menor arrecadação é a de Santo Antonio dos Milagres - PI, com o valor de R\$ 416.535,96 (ordem das centenas de milhares de Reais), a maior é a do município de São Paulo - SP, com o valor de R\$ 280.440.706.625,21 (centenas de bilhões de Reais).

In [10]:

```
df_original.describe().round(2).astype(str)
```

Out[10]:

	01_arrecadacao_2018	02_unidades	02_unidades_atuantes_%	02_ocupados_total	02_ocupados_assalariados_%
count	5565.0	5565.0	5565.0	5565.0	5565.0
mean	249435087.56	978.82	97.44	9382.44	6.9
std	4865354052.58	8958.4	2.95	93409.53	1.0
min	416535.96	5.0	56.1	59.0	0.0
25%	3775315.98	73.0	96.55	564.0	0.0
50%	9175716.74	173.0	97.98	1233.0	0.0
75%	31501112.41	471.0	99.21	3536.0	100.0
max	280440706625.21	566037.0	100.0	5571893.0	100.0

In [11]:

```
print("Município com maior valor para 01_arrecadacao_2018:", "\n", \
df_original[['municipio-uf','01_arrecadacao_2018']][df_original['01_arrecadacao_2018']\ 
==max(df_original['01_arrecadacao_2018'])])

print("\n")

print("Município com menor valor para 01_arrecadacao_2018:", "\n", \
df_original[['municipio-uf','01_arrecadacao_2018']][df_original['01_arrecadacao_2018']\ 
==min(df_original['01_arrecadacao_2018'])])
```

Município com maior valor para 01\_arrecadacao\_2018:  
 municipio-uf 01\_arrecadacao\_2018  
 3830 SAO PAULO-SP 2.804407e+11

Município com menor valor para 01\_arrecadacao\_2018:  
 municipio-uf 01\_arrecadacao\_2018  
 847 SANTO ANTONIO DOS MILAGRES-PI 416535.96

Agora, um sumário estatístico das colunas com dados não numéricos. Vemos, por exemplo, que o Nordeste é a região brasileira com maior número de municípios com mais de 32% do total (variável **03\_grande\_regiao**), embora seja Minas Gerais o estado com maior concentração (variável **UF**). Interessante também notar que a maior parte das unidades (estabelecimentos de pessoas jurídicas localizados nos municípios) têm por atividade principal "Administração, defesa, educação e saúde pública" (variável **03\_atividade\_principal\_primeira**).

In [12]:

```
categ = df_original.dtypes[df_original.dtypes == "object"].index
df_original[categ].describe()
```

Out[12]:

	municipio-uf	03_nome_uf	03_uf	03_cod_municipio	03_grande_regiao	03_mesorregiao
count	5565	5565	5565	5565	5565	5565
unique	5565		27	27	5	137
top	SAO MIGUEL DO GUapore-RO	Minas Gerais	MG	4315750	Nordeste	Noroeste Rio-grandense
freq	1	853	853	1	1794	216

## 2.2. Analisando a variável de interesse (label): '01\_arrecadacao\_2018'.

O interesse do estudo será a predição da arrecadação dos municípios brasileiros em 2018.

Para conhecer melhor nossa variável de interesse (label), passa-se a examiná-la.

In [13]:

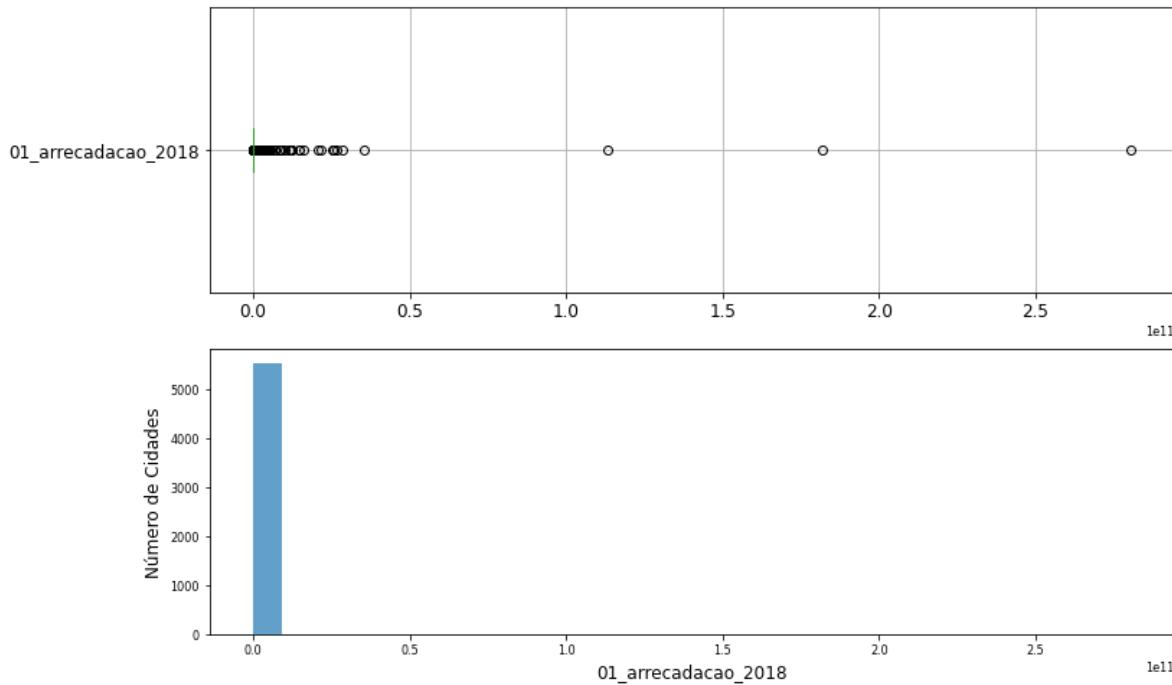
```
#Para preservar o df_original, vamos criar um novo dataframe, de nome df_transformado.
df_transformado=df_original.copy()
```

In [14]:

```
plotstats(df_original, '01_arrecadacao_2018')
df_transformado['01_arrecadacao_2018'].describe().round(2).astype(str)
```

Out[14]:

count	5565.0
mean	249435087.56
std	4865354052.58
min	416535.96
25%	3775315.98
50%	9175716.74
75%	31501112.41
max	280440706625.21
Name:	01_arrecadacao_2018, dtype: object



A princípio, vemos que a variável original (01\_arrecadacao\_2018) possui uma amplitude significativa (valor mínimo de 416,5 mil e valor máximo de 280,4 bi), com média de 249,5 mi e mediana de 9,1 mi.

Essa grande variação prejudica a visualização gráfica tanto no boxplot como no histograma e, como visto acima, dificulta a análise comparativa com as demais variáveis, com menores grandes.

Em razão disso, converteremos a variável para a escala logarítmica.

In [15]:

```
df_transformado['01_arrecadacao_2018_ln']=np.log(df_transformado['01_arrecadacao_2018']).as
```

Passaremos, abaixo, a analisar a variável que resultou da transformação.

Por meio da função abaixo, imprimirei um Gráfico de Dispersão com as arrecadações

(01\_arrecadacao\_2018\_In). No primeiro gráfico, está a distribuição por estado da federação. O segundo gráfico exibe a distribuição em ordem ascendente das arrecadações (para isso, precisei criar a variável **indice\_ordenado**, que acompanha o valor ordenado ascendente da coluna 01\_arrecadacao\_2018\_In).

In [16]:

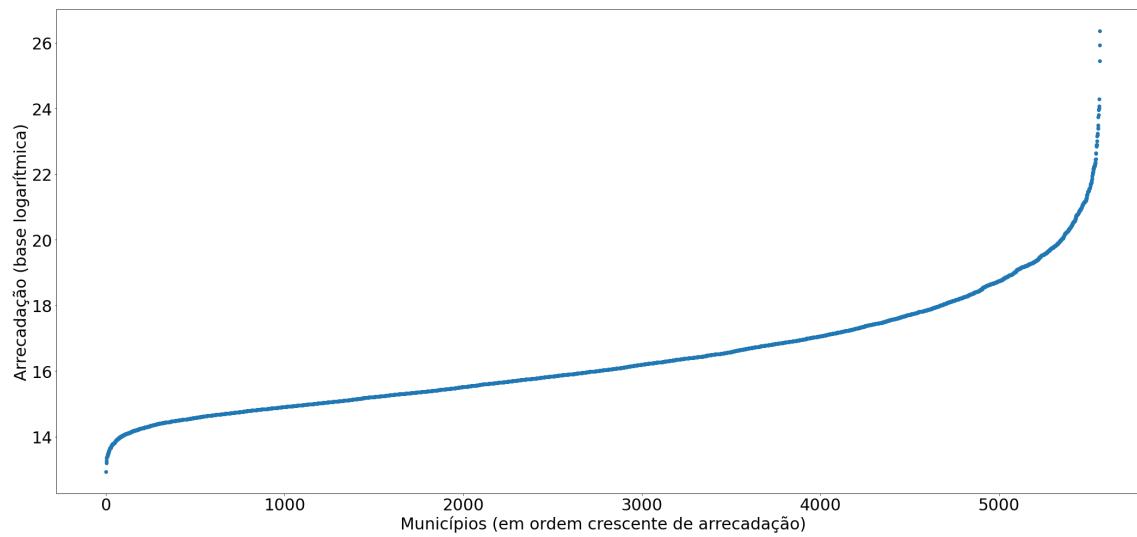
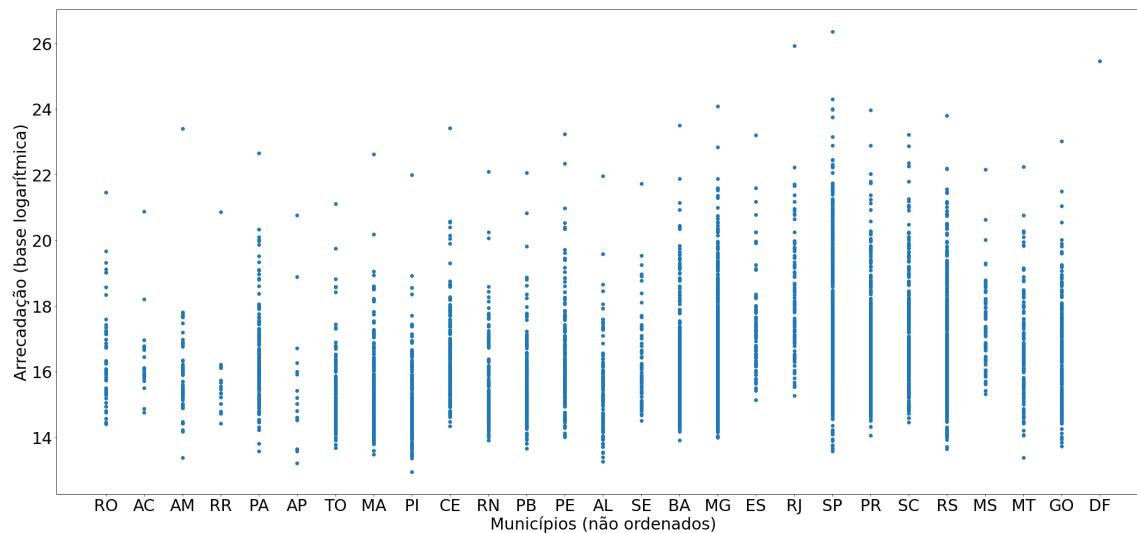
```
#Cria a coluna 'indice_ordenado'
df_transformado.sort_values(by='01_arrecadacao_2018_ln', ascending=True, inplace = True)
df_transformado['indice_ordenado']=np.arange(0,df_transformado.shape[0])
df_transformado.sort_index(ascending=True, inplace = True)

# Imprime o gráfico

#plt.rc('font', size=8)
#plt.rc('axes', titlesize=8)
plt.rc('axes', labelsize=30)
plt.rc('xtick', labelsize=30)
plt.rc('ytick', labelsize=30)
#plt.rc('Legend', fontsize=8)
#plt.rc('figure', titlesize=12)

fig, ax = plt.subplots(2, 1, figsize = (36,36))
ax[0].scatter(df_transformado['03_uf'],df_transformado['01_arrecadacao_2018_ln'])
ax[0].set_xlabel('Municípios (não ordenados)')
ax[0].set_ylabel('Arrecadação (base logarítmica)')
ax[1].scatter(df_transformado['indice_ordenado'],df_transformado['01_arrecadacao_2018_ln'])
ax[1].set_xlabel('Municípios (em ordem crescente de arrecadação)')
ax[1].set_ylabel('Arrecadação (base logarítmica)')

plt.show()
```



Abaixo o boxplot e o histograma da distribuição.

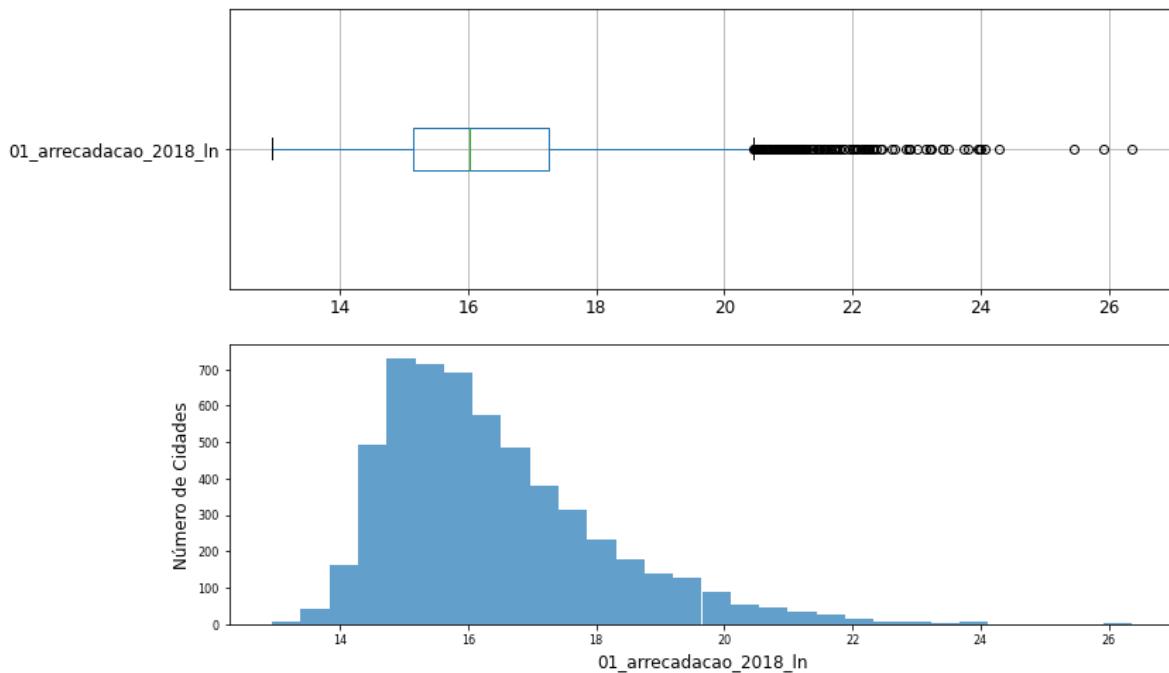
In [17]:

```
plotstats(df_transformado, '01_arrecadacao_2018_ln')
df_transformado['01_arrecadacao_2018_ln'].describe().round(8).astype(str)
```

Out[17]:

count	5565.0
mean	16.40321133
std	1.69714142
min	12.93972808
25%	15.14399464
50%	16.03207107
75%	17.26553342
max	26.35962815

Name: 01\_arrecadacao\_2018\_ln, dtype: object



Após essa transformação para a escala logarítmica, tanto o boxplot como o histograma podem ser melhor analisados.

Vemos, pelo histograma, que a distribuição da variável se aproxima da Normal, com média de 16,4 e mediana de 16,03.

Pelo boxplot, nota-se a presença de outliers.

Com a linha de comando abaixo, foram identificados 158 outliers. Cidades que atendam a algum dos seguintes critérios:

- 01\_arrecadacao\_2018\_ln < 12.939728075397491 (R\$ 416.535,96)
- 01\_arrecadacao\_2018\_ln > 20.445576801868206 (R\$ 757.532.350,88)

In [18]:

```
#A Linha abaixo utiliza a função limites_min_max_boxplot declarada anteriormente.
limites_outliers=limites_min_max_boxplot(df_transformado, '01_arrecadacao_2018_ln')
print ('Serão outliers os valores:')
print ('-acima de :',limites_outliers[0], "(base logarítmica) ou",limites_outliers[2],"(em R")
print ('-abaixo de :',limites_outliers[1],"(base logarítmica) ou",limites_outliers[3],"(em R")
```

Serão outliers os valores:

-acima de : 20.445576801868192 (base logarítmica) ou 757532350.8799902 (em Reais)  
-abaixo de : 12.93972807539753 (base logarítmica) ou 416535.96000001614 (em Reais)

Exibindo os outliers.

In [19]:

```
#Exibindo os outliers

index = np.where((df_transformado["01_arrecadacao_2018_ln"] > limites_outliers[0]) | \
                  (df_transformado["01_arrecadacao_2018_ln"] < limites_outliers[1]))

outliers = df_transformado[['municipio-uf','03_uf','01_arrecadacao_2018_ln','01_arrecadacao_2018']]
print (outliers.shape)
#outliers.sort_values(by=3).round(2).astype(str)
outliers.sort_values(by='01_arrecadacao_2018').round(2).astype(str)
```

(158, 5)

Abaixo, os Gráficos de Dispersão para os 158 outliers. No gráfico à esquerda, está a distribuição por estado da federação. O gráfico à direita exibe a distribuição em ordem ascendente das arrecadações.

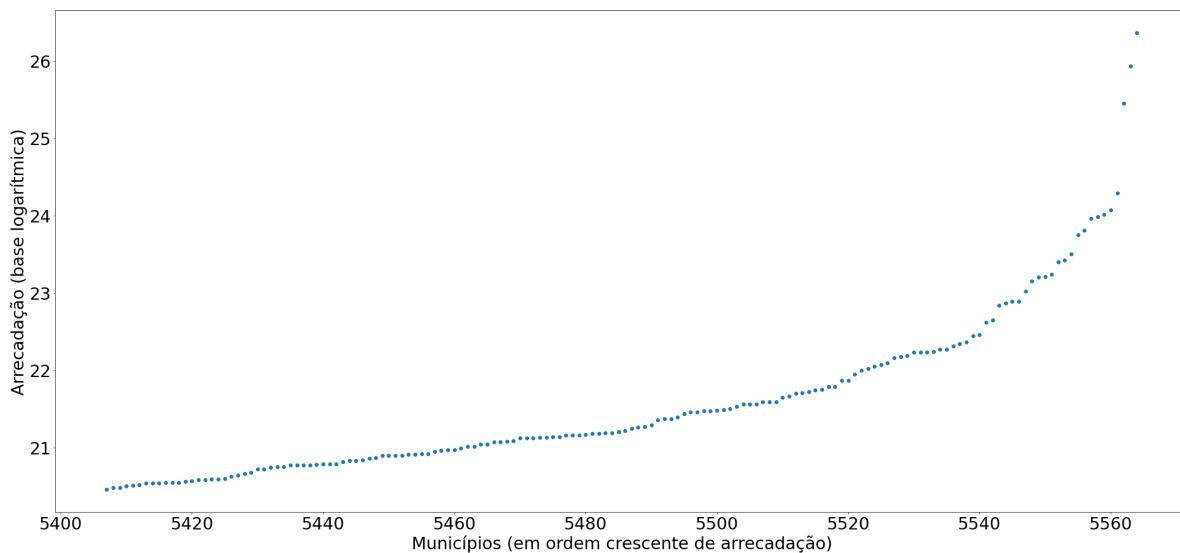
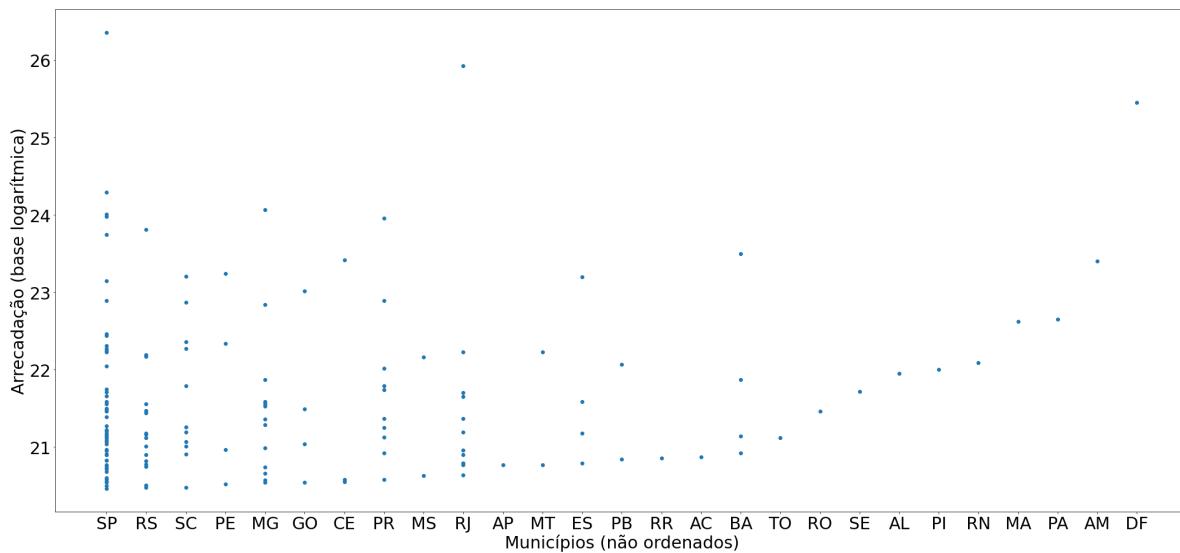
In [20]:

```
# Imprime o gráfico

# plt.rc('font', size=8)
# plt.rc('axes', titlesize=8)
plt.rc('axes', labelsize=30)
plt.rc('xtick', labelsize=30)
plt.rc('ytick', labelsize=30)
# plt.rc('legend', fontsize=8)
# plt.rc('figure', titlesize=12)

fig, ax = plt.subplots(2, 1, figsize = (36,36))
ax[0].scatter(df_transformado.index,df_transformado['01_arrecadacao_2018_ln'])
ax[0].scatter(outliers['03_ue'],outliers['01_arrecadacao_2018_ln'])
ax[0].set_xlabel('Municípios (não ordenados)')
ax[0].set_ylabel('Arrecadação (base logarítmica)')
ax[1].scatter(outliers['indice_ordenado'],outliers['01_arrecadacao_2018_ln'])
ax[1].set_xlabel('Municípios (em ordem crescente de arrecadação)')
ax[1].set_ylabel('Arrecadação (base logarítmica)')

plt.show()
```



A importância de se analisar os outliers diz respeito ao seu possível efeito negativo aos modelos de machine learning, influenciando-os indevidamente. Assim, esses outliers precisam ser avaliados e, se for o caso, tratados (transformados, ou excluídos).

No entanto, a caracterização de um valor como um outlier não é tão simples. Embora as arrecadações dos 158 municípios acima realmente destoem dos demais (lembrando que a média total das arrecadações é de 16,4, em bases logarítmicas, ou R\$ 13.299.158,62, e aqui estamos falando de arrecadações superiores a 772.262.465,32), não se tratam de outliers causados, por exemplo, por erros de valores (imagine se para uma variável que se refira à altura da pessoa, em vez de uma escala em metros - em que a altura máxima possível fosse pouco superior a 2.0 metros, na ordem das unidades - alguns valores estivessem informados em cm - o que levaria os valores à ordem das centenas).

Diante disso, ainda que influenciem o resultado final do modelo, esses valores destoantes refletem as características desse dataset e deverão ser mantidos.

## 2.3. Análise visual das variáveis numéricas.

Como visto anteriormente, há expressiva diferença de grandezas entre as variáveis, o que impõe dificuldades para a comparação entre elas, especialmente quando analisamos os relacionamentos com a variável a ser predita ("01\_arrecadacao\_2018").

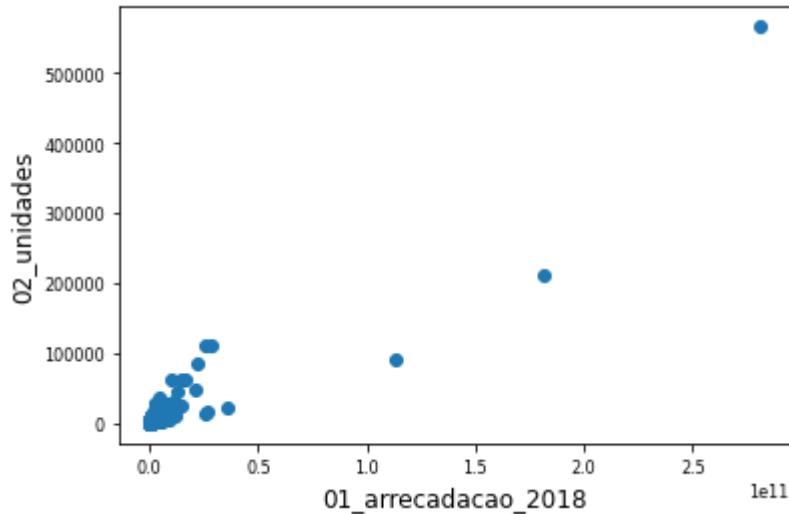
Veja o exemplo abaixo em que comparamos as variáveis **01\_arrecadacao\_2018** e **02\_unidades**. Enquanto aquela possui uma amplitude entre **416.536,96** e **280.440.706.625,21**; para esta a amplitude é entre **5** e **566.037** unidades.

Essa diferença de magnitudes faz com que o gráfico se pareça com o abaixo, a partir do qual uma análise válida fica prejudicada.

In [21]:

```
#plt.rcParams['font.size']=8
#plt.rcParams['axes.titlesize']=8
plt.rcParams['axes.labelsizes']=12
plt.rcParams['xtick.labelsizes']=8
plt.rcParams['ytick.labelsizes']=8
#plt.rcParams['legend.fontsize']=8
#plt.rcParams['figure.titlesize']=12

plt.scatter(df_transformado['01_arrecadacao_2018'], df_transformado['02_unidades'])
plt.xlabel('01_arrecadacao_2018')
plt.ylabel('02_unidades')
plt.show()
```



Faremos, então, a transformação dos dados e, em seguida, retomaremos a análise gráfica.

### - Transformação das variáveis numéricas (conversão para base logarítmica natural)

Para assim, reduzir os efeitos da diferença de magnitude entre as variáveis, vamos transformar os valores numéricos para uma escala logarítmica natural.

In [22]:

```
converte_para_ln(df_transformado)
df_transformado = df_transformado.drop(['01_arrecadacao_2018_ln_ln', 'indice_ordenado_ln'],
```

Colunas criadas: 39

Vamos agora manter apenas as variáveis numéricas recém-criadas, excluindo as demais. Preservaremos o dataframe integral criando a instância df\_transformado\_completo.

In [23]:

```

colunas_a_manter = ['municipio-uf', '01_arrecadacao_2018', '01_arrecadacao_2018_ln', '02_unidades_atuantes_%_ln', '02_ocupados_total_ln', '02_ocupados_assalariados_%_ln', '02_ocupados_assalariados_medio_ln', '02_salario_medio_mensal_em_SM_ln', '02_salario_medio_mensal_em_Reais_ln', '03_nome_uf', '03 uf', '03_cod_municipio', '03_mesorregiao', '03_microregiao', '03_regiao_imediata', '03_regiao_intermediaria', '03_atividade_principal_primeira', '03_atividade_principal', '03_atividade_principal_terceira', '03_vlr_adic_bruto_total_ln', '03_vlr_adic_bruto_agropecuaria_%_ln', '03_vlr_adic_bruto_industria_%_ln', '03_vlr_adic_bruto_servicos_%_ln', '03_vlr_adic_bruto_administracao_%_ln', '03_impostos_sobre_produtos_ln', '03_pib_ln', '03_pib_per_capita_ln', '04_populacao_residente_2018_ln', '05_total_residentes_ln', '05_0-19_anos_%_ln', '05_20-39_anos_%_ln', '05_40-59_anos_%_ln', '05_60+_anos_%_ln', '06_total_instrucao_ln', '06_Sem_Instrucao_e_Fundamental_Incompleto%_ln', '06_Medio_Incompleto_%_ln', '06_Superior_Incompleto_%_ln', '06_Superior_Completo_%_ln', '06_Indeterminado_%_ln', '07_total_rendimentos_ln', '07_Ate_1SM_%_ln', '07_Ate_2SM_%_ln', '07_Ate_3SM_%_ln', '07_Ate_5SM_%_ln', '07_Ate_10SM_%_ln', '07_Ate_20SM_%_ln', '07_20SM+_%_ln', '07_Sem_Rendimento_%_ln', 'indice_ordenado']

df_transformado_completo = df_transformado.copy()
df_transformado = df_transformado[colunas_a_manter]
print('df_transformado_completo.shape', df_transformado_completo.shape)
print('df_transformado', df_transformado.shape)

```

df\_transformado\_completo.shape (5565, 87)  
df\_transformado (5565, 51)

Nota-se, abaixo, que após a transformação dos dados para a escala logarítmica, o menor valor entre todas as variáveis passou a ser -13,82 e o maior valor apenas 26,35, trazendo-os, assim, para uma mesma magnitude facilitando a análise comparada e reduzindo viéses inadequadas decorrentes unicamente da diferença de grandezas.

In [24]:

```
df_transformado.describe()
```

Out[24]:

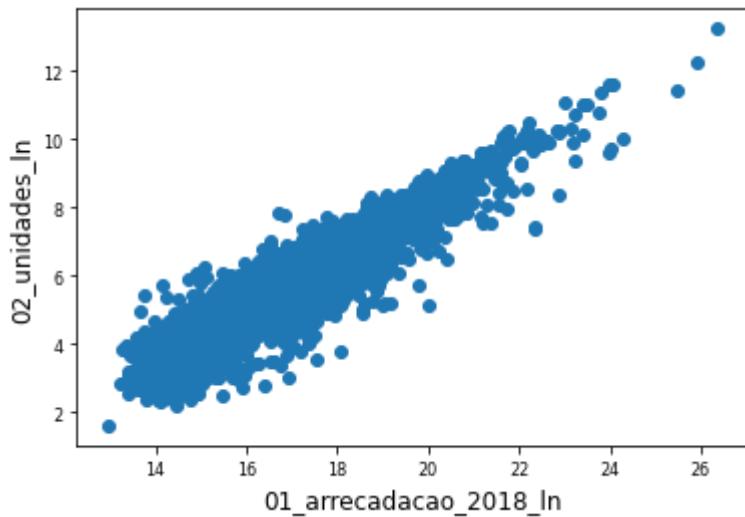
	01_arrecadacao_2018	01_arrecadacao_2018_ln	02_unidades_ln	02_unidades_atuantes_%_ln
--	---------------------	------------------------	----------------	---------------------------

<b>count</b>	5.565000e+03	5565.000000	5565.000000	5565.000000
<b>mean</b>	2.494351e+08	16.403211	5.330180	4.57874
<b>std</b>	4.865354e+09	1.697141	1.421578	0.03288
<b>min</b>	4.165360e+05	12.939728	1.609438	4.02719
<b>25%</b>	3.775316e+06	15.143995	4.290459	4.57001
<b>50%</b>	9.175717e+06	16.032071	5.153292	4.58479
<b>75%</b>	3.150111e+07	17.265533	6.154858	4.59720
<b>max</b>	2.804407e+11	26.359628	13.246415	4.60511

Após a transformação para a base logarítmica natural, o novo gráfico impresso passa a exibir mais claramente a relação entre as duas variáveis sob atenção.

In [25]:

```
plt.rc('axes', labelsize=12)
plt.rc('xtick', labelsize=8)
plt.rc('ytick', labelsize=8)
plt.scatter(df_transformado['01_arrecadacao_2018_ln'], df_transformado['02_unidades_ln'])
plt.xlabel('01_arrecadacao_2018_ln')
plt.ylabel('02_unidades_ln')
plt.show()
```



## - Boxplots por UF

Como complemento, seguem abaixo boxplots que exibem, por Estado da Federação, o comportamento das variáveis numéricas.

Como era de se esperar, há variações relevantes entre os diversos estados.

De maneira bastante resumida, pode-se destacar algumas constatações interessantes a partir da leitura dos boxplots:

Algumas constatações interessantes com a leitura do primeiro gráfico que retrata o comportamento da variável '01\_arrecadacao\_2018\_ln' agrupada por '03\_uf':

- embora os municípios com maior arrecadação nacional estejam no estado de São Paulo (observado a partir do valor máximo da distribuição, desconsiderados os outliers) - estado em que também se observa a maior dispersão (diferença entre o terceiro e o primeiro quartis) e maior amplitude (diferença entre o valor máximo e o valor mínimo) -, é o estado do Rio de Janeiro que apresenta a proporção mais significativa de municípios com alta arrecadação (mediana e segundo e terceiro quartis com valores mais elevados dentre os boxplots);
- por outro lado, é do estado do Piauí a distribuição de municípios com menor arrecadação (embora seja a Paraíba o estado com menor valor mínimo do boxplot);
- observando o conjunto de boxplots, nota-se que as distribuições são, em sua maioria, simétricas ou levemente assimétricas positivas (em razão de a mediana aproximar-se da indicação do primeiro quartil);
- há outliers (superiores, apenas) para todos os Estados da Federação.

Comportamento semelhante se observa no quinto gráfico (variável '02\_ocupados\_total\_ln' agrupada por '03\_uf') e no segundo gráfico (variável '02\_unidades\_ln' agrupada por '03\_uf'), com a diferença de que, neste último, o menor número de unidades está no estado do Tocantins (e não do Piauí).

A leitura do terceiro e quarto gráficos considera o quanto as distribuições se aproximam do percentual máximo (100%). E aí é interessante notar que é no estado de São Paulo onde estão os piores números relativos à variável "02\_unidades\_atuantes\_%ln" e, no Rio Grande do Sul, em relação à variável "02\_ocupados\_assalariados%\_ln".

In [26]:

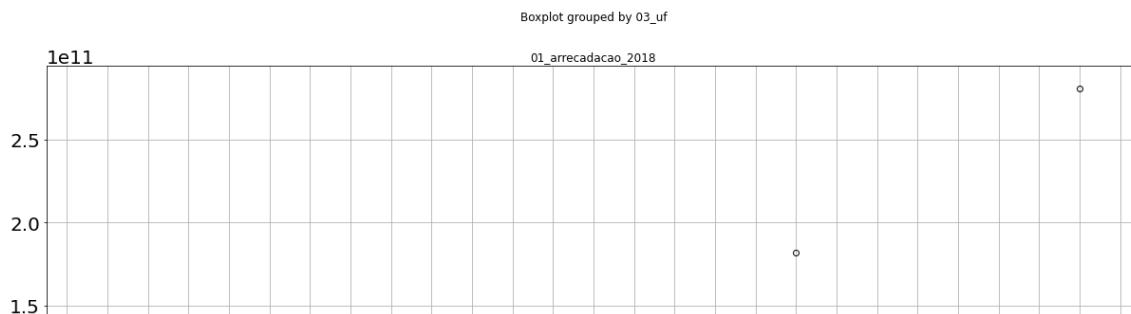
```
cria_boxplots(df_transformado, '03_uf') #Chama a função declarada anteriormente.
```

```
<ipython-input-6-e95bb038c515>:15: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
```

```
fig = plt.figure(figsize = (20,10))
```

Out[26]:

'Concluído'



## - Categorização do dataset por "faixas de arrecadação".

Por fim, vamos criar no dataset df\_transformado\_completo uma coluna chamada 'faixa\_arrecadacao', útil para as análises dos resultados finais.

Serão criadas 5 classes .

- K = 5
- Amplitude = 26.36 - 12.94 = 13.42
- Intervalo = Amplitude / K
- Faixa\_1 0 |- 15.623999999999999
- Faixa\_2 15.62399999999999 |- 18.308
- Faixa\_3 18.308 |-20.991999999999997
- Faixa\_4 20.99199999999997 |- 23.676000000000000002
- Faixa\_5 23.676000000000002 |-| 26.36

In [27]:

```
K=5
Amplitude = 26.36 - 12.94
Intervalo = Amplitude/K
Faixa_1 = 12.94 + (1*Intervalo)
Faixa_2 = 12.94 + (2*Intervalo)
Faixa_3 = 12.94 + (3*Intervalo)
Faixa_4 = 12.94 + (4*Intervalo)
Faixa_5 = 12.94 + (5*Intervalo)

print('K',K)
print('Amplitude',Amplitude)
print('Intervalo',Intervalo)
print('Faixa_1',Faixa_1)
print('Faixa_2',Faixa_2)
print('Faixa_3',Faixa_3)
print('Faixa_4',Faixa_4)
print('Faixa_5',Faixa_5)

#Chama a função 'faixa_arrecadacao' que categoriza o dataset ns faixas definidas.
df_transformado_completo['faixa_arrecadacao'] = df_transformado_completo.apply(lambda row:f
```

```
K 5
Amplitude 13.42
Intervalo 2.684
Faixa_1 15.623999999999999
Faixa_2 18.308
Faixa_3 20.991999999999997
Faixa_4 23.676000000000002
Faixa_5 26.36
```

In [28]:

```
%matplotlib inline
import matplotlib.pyplot as plt

contador = df_transformado_completo['faixa_arrecadacao'].value_counts().sort_index() # Conta o número de municípios por faixa
fig = plt.figure(figsize=(8,6)) # define área plotagem

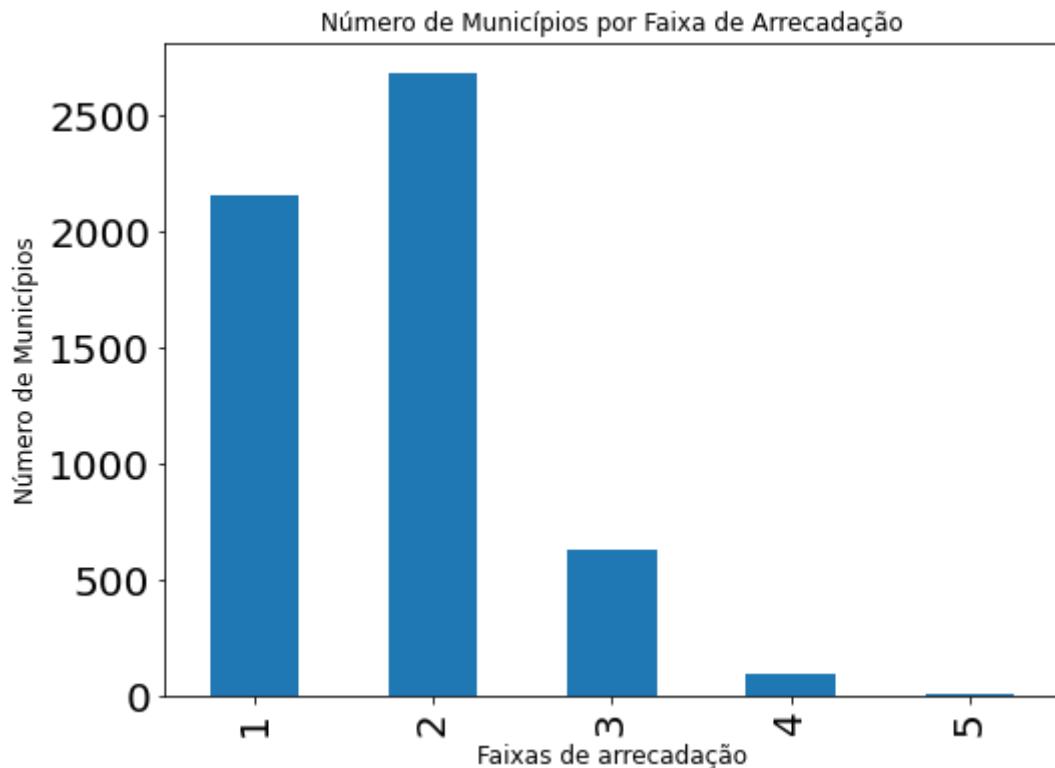
ax = fig.gca() # define eixos
contador.plot.bar(ax = ax) # Usa o método plot.bar para o contador

ax.set_title('Número de Municípios por Faixa de Arrecadação') # Título principal
ax.set_xlabel('Faixas de arrecadação', fontsize=12) # Título eixo x
ax.set_ylabel('Número de Municípios', fontsize=12) # Título eixo y
print ('Número de municípios por Faixa')
print (contador.sort_index())
```

Número de municípios por Faixa

1	2152
2	2676
3	633
4	94
5	10

Name: faixa\_arrecadacao, dtype: int64



Vemos acima que as duas primeiras faixas de arrecadação compreendem a maior parte dos municípios (aproximadamente 86,75% dos 5.565 aqui sob análise).

Abaixo são exibidos boxplots com o comportamento das variáveis numéricas de interesse para cada faixa de arrecadação.

- com o primeiro gráfico temos que ao analisarmos a variável **01\_arrecadacao\_2018\_In** a partir das 5 faixas de arrecadação, temos que a primeira faixa apresenta outliers inferiores (lembrando que quando

considerando quando havíamos analisado a totalidade das distribuições, o boxplot exibia 158 outliers superiores, que, aqui, englobam parte da faixa 3 e a totalidade das faixas 4 e 5); e que, com exceção da faixa 1, as distribuições apresentam-se positivamente assimétricas;

- no segundo gráfico vemos que, diferente do gráfico anterior em que não havia sobreposições nos limites dos boxplots, aqui vemos que há (por exemplo, os limites mínimos das faixas (com exceção, é óbvio da faixa 1) são inferiores ao limite máximo da faixa imediatamente anterior - isso denota que, não necessariamente, uma maior arrecadação (movimento entre as faixas) decorre de um número maior de unidades (estabelecimentos de pessoas jurídicas); vemos também a presença de outliers;
- nota-se no terceiro e quarto gráficos uma maior variação nos valores das faixas o que conduz à presença de uma maior quantidade de outliers por faixas de arrecadação.

In [29]:

```
cria_boxplots(df_transformado_completo, 'faixa_arrecadacao')
```

```
<ipython-input-6-e95bb038c515>:15: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
```

```
fig = plt.figure(figsize = (20,10))
```

Out[29]:

'Concluído'



## - Transformação das variáveis categóricas (One Hot Encoding)

Do total de variáveis do dataset, as colunas abaixo são categóricas nominais (não ordinais).

In [30]:

```
#Para a análise, retornaremos o conteúdo completo do dataframe df_transformado
df_transformado = df_transformado_completo.copy()

variaveis_categoricas = df_transformado.dtypes[df_transformado.dtypes == "object"].index
variaveis_categoricas
```

Out[30]:

```
Index(['municipio-uf', '03_nome_uf', '03_uf', '03_cod_municipio',
       '03_grande_regiao', '03_mesorregiao', '03_microregiao',
       '03_regiao_imediata', '03_regiao_intermediaria',
       '03_atividade_principal_primeira', '03_atividade_principal_segunda',
       '03_atividade_principal_terceira', 'faixa_arrecadacao'],
      dtype='object')
```

Vamos analisar a quantidade de valores únicos em cada uma delas (a quantidade de novas variáveis que

seriam criadas)

In [31]:

```
for i in variaveis_categoricas:
    print ('Valores únicos para',i,df_transformado[i].nunique())
```

Valores únicos para municipio-uf 5565  
 Valores únicos para 03\_nome\_uf 27  
 Valores únicos para 03\_uf 27  
 Valores únicos para 03\_cod\_municipio 5565  
 Valores únicos para 03\_grande\_regiao 5  
 Valores únicos para 03\_mesorregiao 137  
 Valores únicos para 03\_microregiao 554  
 Valores únicos para 03\_regiao\_imediata 508  
 Valores únicos para 03\_regiao\_intermediaria 133  
 Valores únicos para 03\_atividade\_principal\_primeira 10  
 Valores únicos para 03\_atividade\_principal\_segunda 10  
 Valores únicos para 03\_atividade\_principal\_terceira 10  
 Valores únicos para faixa\_arrecadacao 5

In [32]:

```
# Para se ter uma visão dos atributos categóricos, os atributos não numéricos
# são descartados.
```

```
categ = df_transformado.dtypes[df_transformado.dtypes == "object"].index
df_transformado[categ].describe()
```

Out[32]:

	municipio-uf	03_nome_uf	03_uf	03_cod_municipio	03_grande_regiao	03_mesorregiao
count	5565	5565	5565	5565	5565	5565
unique	5565	27	27	5565	5	137
top	SAO MIGUEL DO GUAPORÉ-RO	Minas Gerais	MG	4315750	Nordeste	Noroeste Rio-grandense
freq	1	853	853	1	1794	216

Diante da quantidade de valores únicos, dentre as variáveis relativas a localização regional, vamos manter apenas UF. Assim, transformaremos apenas as variáveis abaixo. Utilizaremos a técnica One Hot Encoding, que, então, criará 57 novas colunas.

In [33]:

```

variaveis_a_transformar = ['03_uf','03_atividade_principal_primeira',
                           '03_atividade_principal_segunda', '03_atividade_principal_terceira']

for i in variaveis_a_transformar:
    df_transformado = pd.concat([df_transformado,pd.get_dummies(df_transformado[i],prefix=i)])
    df_transformado = df_transformado.drop(i,axis=1)

## as linhas abaixo totalizam o número de colunas criadas
cols = df_transformado.columns.tolist()

print(df_transformado.shape)
#print(df_transformado.columns)

## as linhas abaixo exibem as colunas criadas
colunas_criadas=[]

for i in variaveis_a_transformar:
    x=0
    while x < len(cols):
        if cols[x][0:len(i)] == i: #procura pelas colunas que se iniciam com o nome da coluna
            colunas_criadas.append(cols[x])
        x = x+1

print("Colunas criadas:",len(colunas_criadas))
print(colunas_criadas)

```

(5565, 141)

Colunas criadas: 57

['03\_uf\_AC', '03\_uf\_AL', '03\_uf\_AM', '03\_uf\_AP', '03\_uf\_BA', '03\_uf\_CE', '03\_uf\_DF', '03\_uf\_ES', '03\_uf\_GO', '03\_uf\_MA', '03\_uf\_MG', '03\_uf\_MS', '03\_uf\_MT', '03\_uf\_PA', '03\_uf\_PB', '03\_uf\_PE', '03\_uf\_PI', '03\_uf\_PR', '03\_uf\_RJ', '03\_uf\_RN', '03\_uf\_RO', '03\_uf\_RR', '03\_uf\_RS', '03\_uf\_SC', '03\_uf\_SE', '03\_uf\_SP', '03\_uf\_TO', '03\_atividade\_principal\_primeira\_Administração, defesa, educação e saúde públicas e segurança social', '03\_atividade\_principal\_primeira\_Agricultura, inclusive apoio à agricultura e a pós colheita', '03\_atividade\_principal\_primeira\_Comércio e reparação de veículos automotores e motocicletas', '03\_atividade\_principal\_primeira\_Construção', '03\_atividade\_principal\_primeira\_Demais serviços', '03\_atividade\_principal\_primeira\_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação', '03\_atividade\_principal\_primeira\_Indústrias de transformação', '03\_atividade\_principal\_primeira\_Indústrias extractivas', '03\_atividade\_principal\_primeira\_Pecuária, inclusive apoio à pecuária', '03\_atividade\_principal\_primeira\_Produção florestal, pesca e aquicultura', '03\_atividade\_principal\_segunda\_Administração, defesa, educação e saúde públicas e segurança social', '03\_atividade\_principal\_segunda\_Agricultura, inclusive apoio à agricultura e a pós colheita', '03\_atividade\_principal\_segunda\_Comércio e reparação de veículos automotores e motocicletas', '03\_atividade\_principal\_segunda\_Construção', '03\_atividade\_principal\_segunda\_Demais serviços', '03\_atividade\_principal\_segunda\_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação', '03\_atividade\_principal\_segunda\_Indústrias de transformação', '03\_atividade\_principal\_segunda\_Indústrias extractivas', '03\_atividade\_principal\_segunda\_Pecuária, inclusive apoio à pecuária', '03\_atividade\_principal\_segunda\_Produção florestal, pesca e aquicultura', '03\_atividade\_principal\_terceira\_Administração, defesa, educação e saúde públicas e segurança social', '03\_atividade\_principal\_terceira\_Agricultura, inclusive apoio à agricultura e a pós colheita', '03\_atividade\_principal\_terceira\_Comércio e reparação de veículos automotores e motocicletas', '03\_atividade\_principal\_terceira\_Construção', '03\_atividade\_principal\_terceira\_Demais serviços', '03\_atividade\_principal\_terceira\_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação', '03\_atividade\_principal\_terceira\_Indústrias de transformação', '03\_atividade\_principal\_terceira\_Indústrias extractivas', '03\_atividade\_principal\_terceira\_Pecuária, inclusive apoio à pecuária', '03\_atividade\_principal\_terceira\_Produção florestal, pesca e aquicultura', '03\_atividade\_principal\_terceira\_Turismo', '03\_atividade\_principal\_terceira\_Uso de solo', '03\_atividade\_principal\_terceira\_Veículos automotores e motocicletas']

```
al_terceira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação', '03_atividade_principal_terceira_Indústrias de transformação', '03_atividade_principal_terceira_Indústrias extractivas', '03_atividade_principal_terceira_Pecuária, inclusive apoio à pecuária', '03_atividade_principal_terceira_Produção florestal, pesca e aquicultura']
```

## 2.4. Identificação e tratamento de multicolinearidades

### - Impressão das Matrizes de Gráficos de Dispersão e das correlações.

Para a análise visual das variáveis numéricas (inclusive aquelas que resultaram da transformação das variáveis categóricas) e os relacionamentos entre elas, criaremos uma Matriz de Gráficos de Dispersão (*scatter plots*). Assim, em vez de examinar, isoladamente, um grande número de gráficos (um para cada combinação de uma dupla de variáveis), a matriz aqui criada permite examinar as relações entre as diversas variáveis em uma única visão.

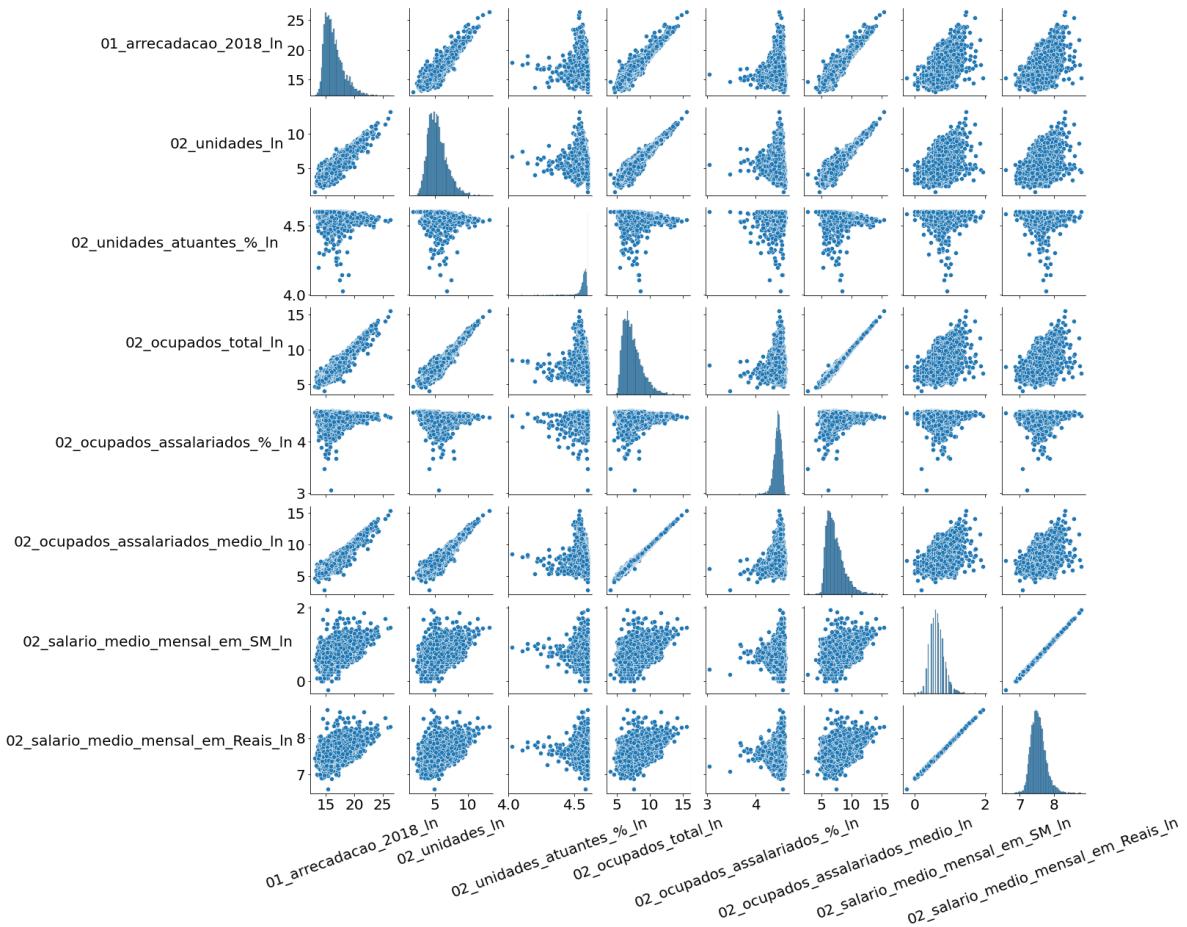
Veremos que o gráfico é formado por um conjunto de Gráficos de Dispersão. Para cada variável há tanto uma linha e uma coluna. A variável é exibida no eixo vertical para cada linha, e no eixo horizontal para cada coluna. Desse modo, toda combinação cruzada para todas as variáveis são mostradas nessas duas possíveis orientações.

Segue abaixo os gráficos com o relacionamento entre as variáveis da base *ii. Tabela 1685 - Unidades locais, empresas e outras organizações atuantes, pessoal ocupado total, pessoal ocupado assalariado, pessoal assalariado médio, salários e outras remunerações e salário médio mensal (tabela1685.csv)* (indicadas com o prefixo '02\_'), convertidas para a base logarítmica, e a variável de interesse (*label*), também convertida: *01\_arrecadacao\_2018\_ln*.

In [34]:

```
colunas_a_exibir = ['01_arrecadacao_2018_ln', '02_unidades_ln',
                     '02_unidades_atuantes_%_ln', '02_ocupados_total_ln',
                     '02_ocupados_assalariados_%_ln', '02_ocupados_assalariados_medio_ln',
                     '02_salario_medio_mensal_em_SM_ln', '02_salario_medio_mensal_em_Reais_ln']

imprime_pairplot(df_transformado, colunas_a_exibir)
```



Out[34]:

'Concluído'

**Análise dos gráficos acima:**

- A partir da análise da primeira linha dos gráficos, é possível constatar que as variáveis **02\_unidades\_ln**, **02\_ocupados\_total\_ln**, **02\_ocupados\_assalariados\_medio\_ln**, guardam evidente colinearidade com a variável de interesse (que após a transformação passou a ser a **01\_arrecadacao\_2018\_ln**), de modo que poderão, a princípio, contribuir com a predição desta;
- Interessante também notar que o relacionamento entre as variáveis **02\_ocupados\_total\_ln** e **02\_ocupados\_assalariados\_medio\_ln** (interseção entre a quarta linha e a sexta coluna) denota uma altíssima colinearidade, o que pode indicar que seriam redundantes a determinar a exclusão de uma delas sem prejuízo para o modelo de predição, tornando-o mais simples;
- Da mesma forma, há forte relação entre as variáveis **02\_salario\_medio\_mensal\_em\_SM\_ln** e **02\_salario\_medio\_mensal\_em\_Reais\_ln** (interseção entre a sétima linha e a oitava coluna), o que, igualmente, determinará a escolha por apenas uma delas para a simplificação, sem prejuízos, ao modelo de predição.

**Diante do exposto acima, é fundamental procedermos a uma análise mais detida entre as correlações das variáveis de modo a identificarmos possíveis multicolinearidades e evitarmos \*\*a maldição da**

**dimensionalidade\*\*.**

(vide [<https://blog.minitab.com/pt/basta-lidando-com-a-multicolinearidade-na-analise-de-regressao>]  
(<https://blog.minitab.com/pt/basta-lidando-com-a-multicolinearidade-na-analise-de-regressao%5D>))

A maldição da dimensionalidade diz que a quantidade de dados de que você precisa, para alcançar o conhecimento desejado, impacta exponencialmente o número de atributos necessários. Com isso, o desempenho do classificador tende a se degradar a partir de um determinado número de atributos, mesmo que sejam úteis. [<https://medium.com/@fabioleneine/como-selecionar-atributos-para-resolver-a-maldi%C3%A7%C3%A3o-da-dimensionalidade-5c810bc8449f#:~:text=A%20maldi%C3%A7%C3%A3o%20da%20dimensionalidade,atributos%2C%20mesmo%20demonstrado>]  
(<https://medium.com/@fabioleneine/como-selecionar-atributos-para-resolver-a-maldi%C3%A7%C3%A3o-da-dimensionalidade-5c810bc8449f#:~:text=A%20maldi%C3%A7%C3%A3o%20da%20dimensionalidade,atributos%2C%20mesmo%20demonstrado>)]

Uma das causas desse efeito é a multicolinearidade, que ocorre quando o modelo inclui vários fatores correlacionados não apenas à sua variável de resposta, mas também uns aos outros. Em outras palavras, resulta quando você tem fatores que são, em certa medida, redundantes.

A multicolinearidade faz aumentar os erros padrão dos coeficientes, e esse aumento significa que os coeficientes para algumas variáveis independentes podem não ser significativamente diferentes de 0. Em outras palavras, ao superinflacionar os erros padrão, a multicolinearidade torna algumas variáveis estatisticamente insignificantes quando deveriam ser significativas. Sem multicolinearidade (e, portanto, com erros padrão mais baixos), esses coeficientes poderiam ser significativos.

Assim, remover do modelo as preditoras que são altamente correlacionadas é uma estratégia para lidar com a multicolinearidade. Como eles fornecem informações redundantes, a remoção de um dos fatores altamente correlacionados geralmente não reduz drasticamente o resultado do modelo

Foi por esse motivo que identificamos e sugerimos, acima, a exclusão de variáveis com forte correlação transversal (não apenas com a variável de interesse).

## - Análise das correlações e identificação de multicolinearidades

Passaremos a examinar as correlações das diversas variáveis com nossa variável de interesse (01\_arrecadacao\_2018).

Primeiro, computaremos a correlação entre as variáveis e nosso *label*.

In [35]:

```
print(len(df_transformado.columns.tolist()))
print(df_transformado.columns.tolist())
```

141

'03\_atividade\_principal\_terceira\_Comércio e reparação de veículos automotores e motocicletas', '03\_atividade\_principal\_terceira\_Construção', '03\_atividade\_principal\_terceira\_Demais serviços', '03\_atividade\_principal\_terceira\_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação', '03\_atividade\_principal\_terceira\_Indústrias de transformação', '03\_atividade\_principal\_terceira\_Indústrias extractivas', '03\_atividade\_principal\_terceira\_Pecuária, inclusive apoio à pecuária', '03\_atividade\_principal\_terceira\_Produção florestal, pesca e aquicultura']

In [36]:

```

colunas_a_manter = [
    '01_arrecadacao_2018', '02_unidades',
    '02_unidades_atuantes__', '02_ocupados_total',
    '02_ocupados_assalariados__', '02_ocupados_assalariados_medio',
    '02_salario_medio_mensal_em_SM', '02_salario_medio_mensal_em_Reais',
    '03_vlr_adic_bruto_total', '03_vlr_adic_bruto_agropecuaria__', '03_vlr_adic_bruto_in',
    '03_vlr_adic_bruto_servicos__', '03_vlr_adic_bruto_administracao__',
    '03_impostos_sobre_produtos', '03_pib', '03_pib_per_capita',
    '04_populacao_residente_2018',
    '05_total_residentes', '05_0-19_anos__', '05_20-39_anos__',
    '05_40-59_anos__', '05_60+anos__', '06_total_instrucao',
    '06_Sem_Instrucao_e_Fundamental_Incompleto%', '06_Medio_Incompleto%',
    '06_Superior_Incompleto__', '06_Superior_Completo__',
    '06_Indeterminado__', '07_total_rendimentos', '07_Ate_1SM__',
    '07_Ate_2SM__', '07_Ate_3SM__', '07_Ate_5SM__', '07_Ate_10SM__',
    '07_Ate_20SM__', '07_20SM+__', '07_Sem_Rendimento__',
    '03_uf_AC', '03_uf_AL', '03_uf_AM', '03_uf_AP', '03_uf_BA', '03_uf_CE', '03_uf_DF',
    '03_uf_ES', '03_uf_GO', '03_uf_MA', '03_uf_MG', '03_uf_MS', '03_uf_MT', '03_uf_PA',
    '03_uf_PB', '03_uf_PE', '03_uf_PI', '03_uf_PR', '03_uf_RJ', '03_uf_RN', '03_uf_RO',
    '03_uf_RR', '03_uf_RS', '03_uf_SC', '03_uf_SE', '03_uf_SP', '03_uf_TO',
    '03_atividade_principal_primeira_Administração, defesa, educação e saúde públicas e',
    '03_atividade_principal_primeira_Agricultura, inclusive apoio à agricultura e a pós',
    '03_atividade_principal_primeira_Comércio e reparação de veículos automotores e mot',
    '03_atividade_principal_primeira_Construção', '03_atividade_principal_primeira_Dema',
    '03_atividade_principal_primeira_Eletricidade e gás, água, esgoto, atividades de ge',
    '03_atividade_principal_primeira_Indústrias de transformação', '03_atividade_princi',
    '03_atividade_principal_primeira_Pecuária, inclusive apoio à pecuária',
    '03_atividade_principal_primeira_Produção florestal, pesca e aquicultura',
    '03_atividade_principal_segunda_Administração, defesa, educação e saúde públicas e',
    '03_atividade_principal_segunda_Agricultura, inclusive apoio à agricultura e a pós',
    '03_atividade_principal_segunda_Comércio e reparação de veículos automotores e moto',
    '03_atividade_principal_segunda_Construção', '03_atividade_principal_segunda_Demais',
    '03_atividade_principal_segunda_Eletricidade e gás, água, esgoto, atividades de ges',
    '03_atividade_principal_segunda_Indústrias de transformação', '03_atividade_princip',
    '03_atividade_principal_segunda_Pecuária, inclusive apoio à pecuária',
    '03_atividade_principal_segunda_Produção florestal, pesca e aquicultura',
    '03_atividade_principal_terceira_Administração, defesa, educação e saúde públicas e',
    '03_atividade_principal_terceira_Agricultura, inclusive apoio à agricultura e a pós',
    '03_atividade_principal_terceira_Comércio e reparação de veículos automotores e mot',
    '03_atividade_principal_terceira_Construção', '03_atividade_principal_terceira_Dema',
    '03_atividade_principal_terceira_Eletricidade e gás, água, esgoto, atividades de ge',
    '03_atividade_principal_terceira_Indústrias de transformação', '03_atividade_princi',
    '03_atividade_principal_terceira_Pecuária, inclusive apoio à pecuária',
    '03_atividade_principal_terceira_Produção florestal, pesca e aquicultura'
]

```

```
#imprime_pairplot(df_transformado, colunas_a_exibir, 'Ntbk 03 - Matriz Scatter - Base Total
```

```

correlacoes=df_transformado[colunas_a_manter].corr().abs()
print (df_transformado[colunas_a_manter].shape)
correlacoes=correlacoes['01_arrecadacao_2018'].sort_values(ascending=False)
print("Correlações com '01_arrecadacao_2018'")
df_correlacoes = pd.DataFrame(correlacoes)
print(df_correlacoes)
#df_correlacoes.to_csv('ntbk 03 - correlacoes.csv')

```

(5565, 94)

Correlações com '01\_arrecadacao\_2018'

01\_arrecadacao\_2018

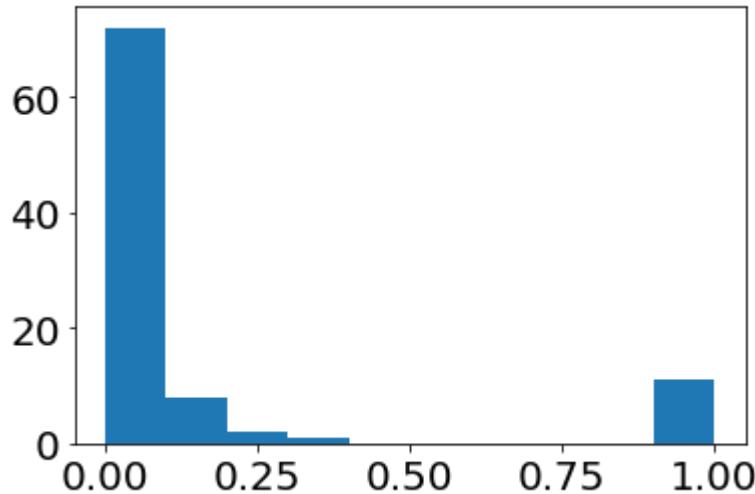
1.000000

01\_arrecadacao\_2018

03_pib	0.974050
03_impostos_sobre_produtos	0.973514
03_vlr_adic_bruto_total	0.967767
02_ocupados_assalariados_medio	0.942503
02_ocupados_total	0.942168
02_unidades	0.930497
07_total_rendimentos	0.919989
06_total_instrução	0.919989
05_total_residentes	0.918294
04_populacao_residente_2018	0.916867
03_uf_DF	0.312061
07_20SM+_%	0.258040
07_Ate_20SM_%	0.234498
06_Superior_Completo_%	0.183179
02_salario_medio_mensal_com_Bonus	0.175182

In [37]:

```
plt.hist(correlacoes)
plt.show()
```



**Vemos, acima, que:**

- há um pequeno número de variáveis (10, no total) com correlação superior a 0.9 com nosso *label*;
- outras 11, possuem correlação entre 0.10 e 0.32;
- na faixa de correlações entre 0,01 e 0,10 há 31 variáveis.
- há 41 variáveis com correlação de valor inferior a 0.01.

Se, porém, analisarmos as correlações para as variáveis em sua versão convertida, a associação entre elas fica ainda mais evidente:

- passa a haver apenas 6 com correlação superior a 0.9 com nosso *label*;
- porém, outras 9, agora, possuem correlação entre 0.60 e 0.89;
- na faixa de correlações entre 0,1 e 0,59 há 40 variáveis.
- na faixa de correlações entre 0,01 e 0,09 há 32 variáveis.
- há, apenas, 6 variáveis com correlação de valor inferior a 0.01.

In [38]:

```

colunas_a_manter_ln = [
    '01_arrecadacao_2018_ln', '02_unidades_ln',
    '02_unidades_atuantes_%_ln', '02_ocupados_total_ln',
    '02_ocupados_assalariados_%_ln', '02_ocupados_assalariados_medio_ln',
    '02_salario_medio_mensal_em_SM_ln', '02_salario_medio_mensal_em_Reais_ln',
    '03_vlr_adic_bruto_total_ln', '03_vlr_adic_bruto_agropecuaria_%', '03_vlr_adic_bruto',
    '03_vlr_adic_bruto_servicos_%_ln', '03_vlr_adic_bruto_administracao_%_ln',
    '03_impostos_sobre_produtos_ln', '03_pib_ln', '03_pib_per_capita_ln',
    '04_populacao_residente_2018_ln',
    '05_total_residentes_ln', '05_0-19_anos_%_ln', '05_20-39_anos_%_ln',
    '05_40-59_anos_%_ln', '05_60+_anos_%_ln', '06_total_instrucao_ln',
    '06_Sem_Instrucao_e_Fundamental_Incompleto%_ln', '06_Medio_Incompleto%_ln',
    '06_Superior_Incompleto%_ln', '06_Superior_Completo%_ln',
    '06_Indeterminado%_ln', '07_total_rendimentos_ln', '07_Ate_1SM%_ln',
    '07_Ate_2SM%_ln', '07_Ate_3SM%_ln', '07_Ate_5SM%_ln', '07_Ate_10SM%_ln',
    '07_Ate_20SM%_ln', '07_20SM+_%_ln', '07_Sem_Rendimento%_ln',
    '03_uf_AC', '03_uf_AL', '03_uf_AM', '03_uf_AP', '03_uf_BA', '03_uf_CE', '03_uf_DF',
    '03_uf_ES', '03_uf_GO', '03_uf_MA', '03_uf_MG', '03_uf_MS', '03_uf_MT', '03_uf_PA',
    '03_uf_PB', '03_uf_PE', '03_uf_PI', '03_uf_PR', '03_uf_RJ', '03_uf_RN', '03_uf_RO',
    '03_uf_RR', '03_uf_RS', '03_uf_SC', '03_uf_SE', '03_uf_SP', '03_uf_TO',
    '03_atividade_principal_primeira_Administração, defesa, educação e saúde públicas e',
    '03_atividade_principal_primeira_Agricultura, inclusive apoio à agricultura e a pós',
    '03_atividade_principal_primeira_Comércio e reparação de veículos automotores e mot',
    '03_atividade_principal_primeira_Construção', '03_atividade_principal_primeira_Dema',
    '03_atividade_principal_primeira_Eletricidade e gás, água, esgoto, atividades de ge',
    '03_atividade_principal_primeira_Indústrias de transformação', '03_atividade_princi',
    '03_atividade_principal_primeira_Pecuária, inclusive apoio à pecuária',
    '03_atividade_principal_primeira_Produção florestal, pesca e aquicultura',
    '03_atividade_principal_segunda_Administração, defesa, educação e saúde públicas e',
    '03_atividade_principal_segunda_Agricultura, inclusive apoio à agricultura e a pós',
    '03_atividade_principal_segunda_Comércio e reparação de veículos automotores e moto',
    '03_atividade_principal_segunda_Construção', '03_atividade_principal_segunda_Demais',
    '03_atividade_principal_segunda_Eletricidade e gás, água, esgoto, atividades de ges',
    '03_atividade_principal_segunda_Indústrias de transformação', '03_atividade_princip',
    '03_atividade_principal_segunda_Pecuária, inclusive apoio à pecuária',
    '03_atividade_principal_segunda_Produção florestal, pesca e aquicultura',
    '03_atividade_principal_terceira_Administração, defesa, educação e saúde públicas e',
    '03_atividade_principal_terceira_Agricultura, inclusive apoio à agricultura e a pós',
    '03_atividade_principal_terceira_Comércio e reparação de veículos automotores e mot',
    '03_atividade_principal_terceira_Construção', '03_atividade_principal_terceira_Dema',
    '03_atividade_principal_terceira_Eletricidade e gás, água, esgoto, atividades de ge',
    '03_atividade_principal_terceira_Indústrias de transformação', '03_atividade_princi',
    '03_atividade_principal_terceira_Pecuária, inclusive apoio à pecuária',
    '03_atividade_principal_terceira_Produção florestal, pesca e aquicultura'
]

```

```
#imprime_pairplot(df_transformado, colunas_a_exibir, 'Ntbk 03 - Matriz Scatter - Base Total
```

```

correlacoes_ln=df_transformado[colunas_a_manter_ln].corr().abs()
print (df_transformado[colunas_a_manter_ln].shape)
correlacoes_ln=correlacoes_ln['01_arrecadacao_2018_ln'].sort_values(ascending=False)
print("Correlações com '01_arrecadacao_2018_ln'")
df_correlacoes_ln = pd.DataFrame(correlacoes_ln)
print(df_correlacoes_ln)
#df_correlacoes_ln.to_csv('ntbk 03 - correlacoes_ln.csv')

```

(5565, 94)

Correlações com '01\_arrecadacao\_2018\_ln'

	01_arrecadacao_2018_ln
01_arrecadacao_2018_ln	1.000000
03_impostos_sobre_produtos_ln	0.956657

02_ocupados_total_ln	0.954628
02_ocupados_assalariados_medio_ln	0.950757
03_pib_ln	0.945561
03_vlr_adic_bruto_total_ln	0.941368
02_unidades_ln	0.928730
07_total_rendimentos_ln	0.816781
06_total_instrução_ln	0.816781
04_populacao_residente_2018_ln	0.810196
05_total_residentes_ln	0.804971
06_Sem_Instrução_e_Fundamental_Incompleto%_ln	0.725350
07_Ate_1SM_%_ln	0.646870
03_vlr_adic_bruto_administracao_%_ln	0.641945
03_vlr_adic_bruto_industria_%_ln	0.623916
03_vlr_adic_bruto_servicos_%_ln	0.618521
06_Superior_Incompleto_%_ln	0.597164
06_Superior_Completo_%_ln	0.572580
07_Ate_5SM_%_ln	0.572464
03_atividade_principal_primeira_Demais serviços	0.572182
03_atividade_principal_primeira_Administração, ...	0.561534
03_pib_per_capita_ln	0.556917
02_salario_medio_mensal_em_Reais_ln	0.544382
02_salario_medio_mensal_em_SM_ln	0.543254
07_Ate_3SM_%_ln	0.543168
07_Ate_10SM_%_ln	0.538032
05_20-39_anos_%_ln	0.496667
06_Medio_Incompleto_%_ln	0.484070
07_Ate_2SM_%_ln	0.468599
07_20SM+_%_ln	0.443790
07_Ate_20SM_%_ln	0.399069
03_atividade_principal_segunda_Indústrias de tr...	0.366802
03_vlr_adic_bruto_agropecuaria %	0.342171
03_atividade_principal_segunda_Demais serviços	0.338820
03_atividade_principal_terceira_Administração, ...	0.323146
06_Indeterminado_%_ln	0.302916
03_atividade_principal_terceira_Pecuária, inclu...	0.301295
03_atividade_principal_segunda_Comércio e repar...	0.294363
02_unidades_atuantes_%_ln	0.256424
05_0-19_anos_%_ln	0.252028
07_Sem_Rendimento_%_ln	0.249932
03_uf_SP	0.232818
03_atividade_principal_terceira_Comércio e repa...	0.212600
03_atividade_principal_terceira_Demais serviços	0.211907
05_40-59_anos_%_ln	0.209231
03_atividade_principal_primeira_Indústrias de t...	0.196653
05_60+_anos_%_ln	0.192257
03_atividade_principal_terceira_Indústrias de t...	0.175997
03_uf_PI	0.174980
03_atividade_principal_terceira_Agricultura, in...	0.149311
03_atividade_principal_segunda_Administração, d...	0.149219
03_uf_RJ	0.142822
03_uf_PB	0.124762
03_atividade_principal_segunda_Pecuária, inclus...	0.113692
03_uf_MA	0.112188
03_uf_TO	0.105870
03_uf_SC	0.097952
03_atividade_principal_primeira_Agricultura, in...	0.096144
03_atividade_principal_terceira_Produção flores...	0.095931
03_atividade_principal_primeira_Indústrias extr...	0.083784
03_uf_RN	0.082464
03_uf_DF	0.071509
03_uf_AL	0.069716

03_uf_ES	0.067822
03_atividade_principal_primeira_Comércio e repa...	0.067530
03_atividade_principal_segunda_Agricultura, inc...	0.063582
03_atividade_principal_primeira_Pecuária, inclu...	0.061731
03_uf_BA	0.061634
03_uf_MS	0.060629
03_uf_PR	0.059631
03_atividade_principal_segunda_Produção florest...	0.047306
02_ocupados_assalariados_%_ln	0.044188
03_uf_MG	0.044051
03_atividade_principal_segunda_Indústrias extra...	0.041009
03_uf_MT	0.033687
03_uf_AM	0.032714
03_uf_RS	0.031529
03_atividade_principal_primeira_Eletricidade e ...	0.029148
03_uf_AP	0.026301
03_uf_RR	0.019245
03_atividade_principal_segunda_Eletricidade e g...	0.017343
03_atividade_principal_segunda_Construção	0.015799
03_atividade_principal_terceira_Eletricidade e ...	0.012815
03_uf_GO	0.012225
03_uf_SE	0.011968
03_atividade_principal_terceira_Construção	0.011057
03_uf_PA	0.010477
03_atividade_principal_primeira_Produção flores...	0.010472
03_atividade_principal_primeira_Construção	0.007363
03_uf_RO	0.005454
03_uf_PE	0.003323
03_uf_CE	0.003065
03_atividade_principal_terceira_Indústrias extr...	0.002743
03_uf_AC	0.001884

Abaixo, iniciaremos a investigação das multicolinearidades.

In [39]:

```
correlacoes = df_transformado[colunas_a_manter].corr()
correlacoes = pd.DataFrame(correlacoes)
correlacoes['indice'] = correlacoes.index
correlacoes
df_correlacoes = pd.DataFrame(correlacoes)
df_correlacoes
#df_correlacoes.to_csv('ntbk 03 - matriz de correlacoes - completa.csv', index=True)
```

Out[39]:

	01_arrecadacao_2018	02_unidades	02_unidades_atuantes_%
01_arrecadacao_2018	1.000000	0.930497	-0.049776
02_unidades	0.930497	1.000000	-0.079235
02_unidades_atuantes_%	-0.049776	-0.079235	1.000000
02_ocupados_total	0.942168	0.990256	-0.080499
02_ocupados_assalariados_%	0.018330	0.007482	0.113011
02_ocupados_assalariados_medio	0.942503	0.987375	-0.081272
02_salario_medio_mensal_em_SM	0.174743	0.208032	-0.200304
02_salario_medio_mensal_em_Reais	0.175183	0.208084	-0.200430

O quadro acima, mostra as diferentes correlações cruzadas entre as diversas variáveis.

Selecionaremos aquelas que possuem elevada correlação com outras variáveis, além daquela já vista em relação a 01\_arrecadacao\_2018.

In [40]:

```
#O dataframe criado acima está no formato pivot (tabela de referência cruzada).
#O comando exibe os valores para cada coordenada linha e coluna
correlacoes = correlacoes.melt(id_vars='indice').copy()

#O comando abaixo, então, seleciona as altas correlações (valores absolutos superiores a ,9
altas_correlacoes = correlacoes[abs(correlacoes['value']) >=.95) \
    & (correlacoes['indice']!= correlacoes['variable']) \
    & (correlacoes['indice']!= '01_arrecadacao_2018') \
    & (correlacoes['indice']!= '01_arrecadacao_2018_ln') \
    & (correlacoes['variable']!= '01_arrecadacao_2018') \
    & (correlacoes['variable']!= '01_arrecadacao_2018_ln')].sort_index()
print (altas_correlacoes.shape)
altas_correlacoes
```

(78, 3)

Há, portanto, as seguintes ocorrências em que, além da correlação com a variável de interesse, apresentam grande correlação entre si (acima de 0.99), denotando multicolaridade.

- Para a dupla abaixo, serão mantidas as variáveis que têm maior correlação com 01\_arrecadacao\_2018 ('02\_ocupados\_total' e '02\_salario\_medio\_mensal\_em\_Reais') e excluídas as demais (em itálico)
  - **02\_ocupados\_total** e **02\_ocupados\_assalariados\_medio**
  - **02\_salario\_medio\_mensal\_em\_Reais** e **02\_salario\_medio\_mensal\_em\_SM**
- Já para o trio de variáveis abaixo, será mantida a variável 03\_pib, escolhida em detrimento de 03\_impostos\_sobre\_produtos (que tem a maior correlação com 01\_arrecadacao\_2018), por ser esta última uma variável própria das Administrações Tributárias e, então, vamos privilegiar aquela a que essas administrações não têm acesso por si.
  - **03\_impostos\_sobre\_produtos**, **03\_pib** e **03\_vlr\_adic\_bruto\_total**
- No tocante à correlação múltipla entre 04\_populacao\_residente\_2018, 05\_total\_residentes, 06\_total\_instrucao e 07\_total\_rendimentos (todas refirindo-se à população residente), será mantida a variável 04\_populacao\_residente\_2018 por se tratar da informação mais recente do mesmo período que o da variável de interesse, embora não tivesse a maior correlação com esta.
  - **04\_populacao\_residente\_2018**, **05\_total\_residentes**, **06\_total\_instrucao** e **07\_total\_rendimentos**
- Assim, seriam mantidas quatro variáveis ('02\_ocupados\_total', '02\_salario\_medio\_mensal\_em\_Reais', '03\_pib' e '04\_populacao\_residente\_2018') e excluídas sete (em itálico).

Se considerarmos como elevada uma correlação o valor acima de 0.95, então haveria dois grupos com multicolinearidade. Nesse caso, seriam mantidas apenas duas variáveis (*03\_pib* e *02\_salario\_medio\_mensal\_em\_Reais*) e excluídas dez (em itálico, abaixo).

**03\_pib, 02\_ocupados\_assalariados\_medio, 02\_ocupados\_total, 02\_unidades, 03\_vlr\_adic\_bruto\_total, 03\_impostos\_sobre\_produtos, 04\_populacao\_residente\_2018, 05\_total\_residentes, 06\_total\_instrução, 07\_total\_rendimentos**

### **02\_salario\_medio\_mensal\_em\_Reais e 02\_salario\_medio\_mensal\_em\_SM**

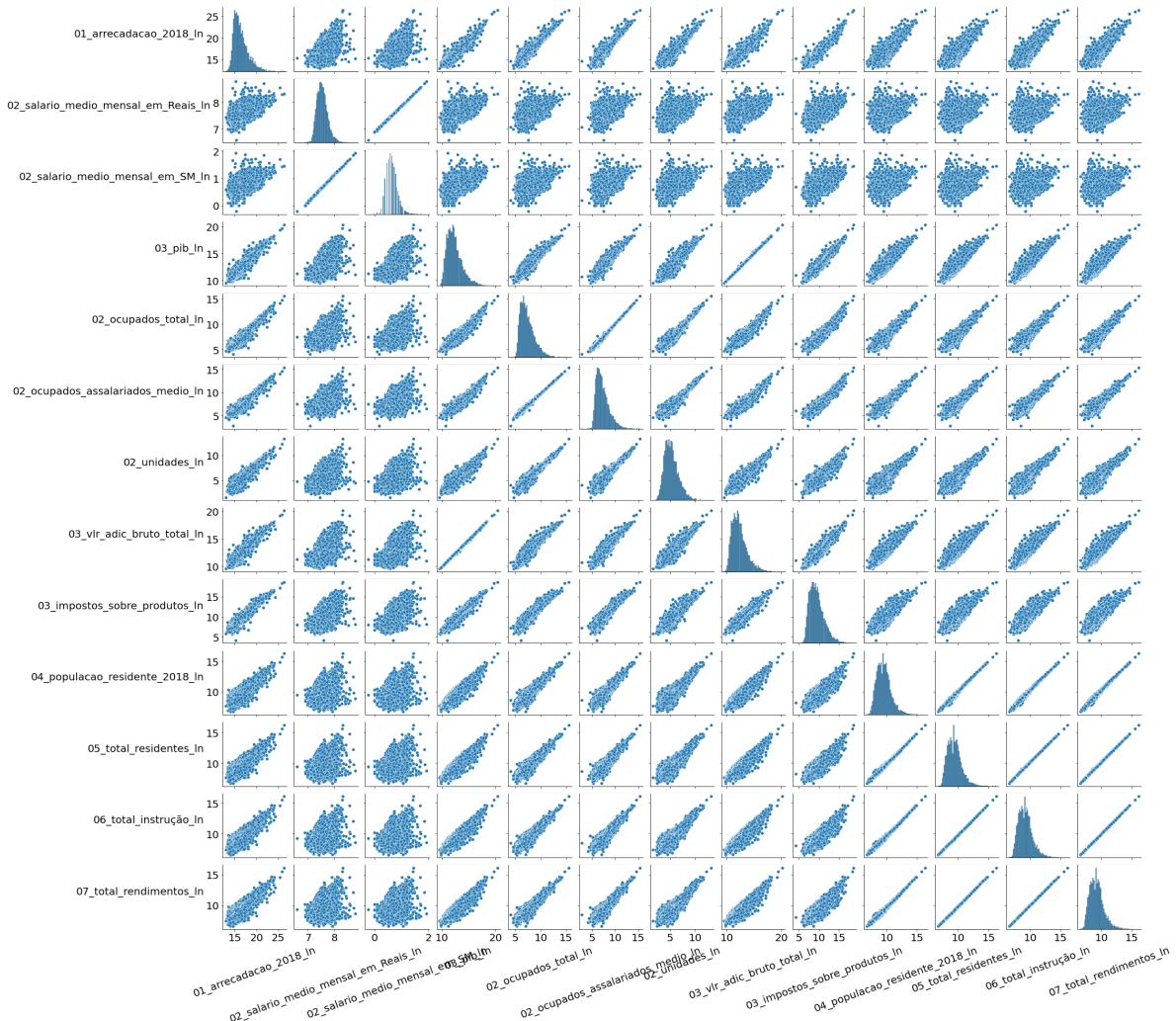
Observação: A variável que não aparecia quando selecionamos as correlações acima de 0.99, e foi incluída aqui, é a '02\_unidades'.

No gráfico abaixo, essas relações mútuas entre as variáveis ficam bastante evidentes (utilizamos para a impressão os valores convertidos em base logarítmica para permitir uma melhor comparação dos valores). Os gráficos da primeira linha mostram a significativa correlação entre as variáveis acima e a variável de interesse *01\_arrecadacao\_2018*. Já nas linhas seguintes, nota-se que nas interseções mencionadas acima há colinearidade explícita, denotada por uma distribuição que praticamente coincide com uma linha reta.

In [41]:

```
colunas_a_exibir = ['01_arrecadacao_2018_ln', '02_salario_medio_mensal_em_Reais_ln', '02_sala  
'03_pib_ln', '02_ocupados_total_ln', '02_ocupados_assalariados_medio_ln',  
'02_unidades_ln', '03_vlr_adic_bruto_total_ln', '03_impostos_sobre_produto  
'04_populacao_residente_2018_ln', '05_total_residentes_ln', '06_total_insu  
'07_total_rendimentos_ln']
```

```
imprime_pairplot(df_transformado, colunas_a_exibir)
```



Out[41]:

'Concluído'

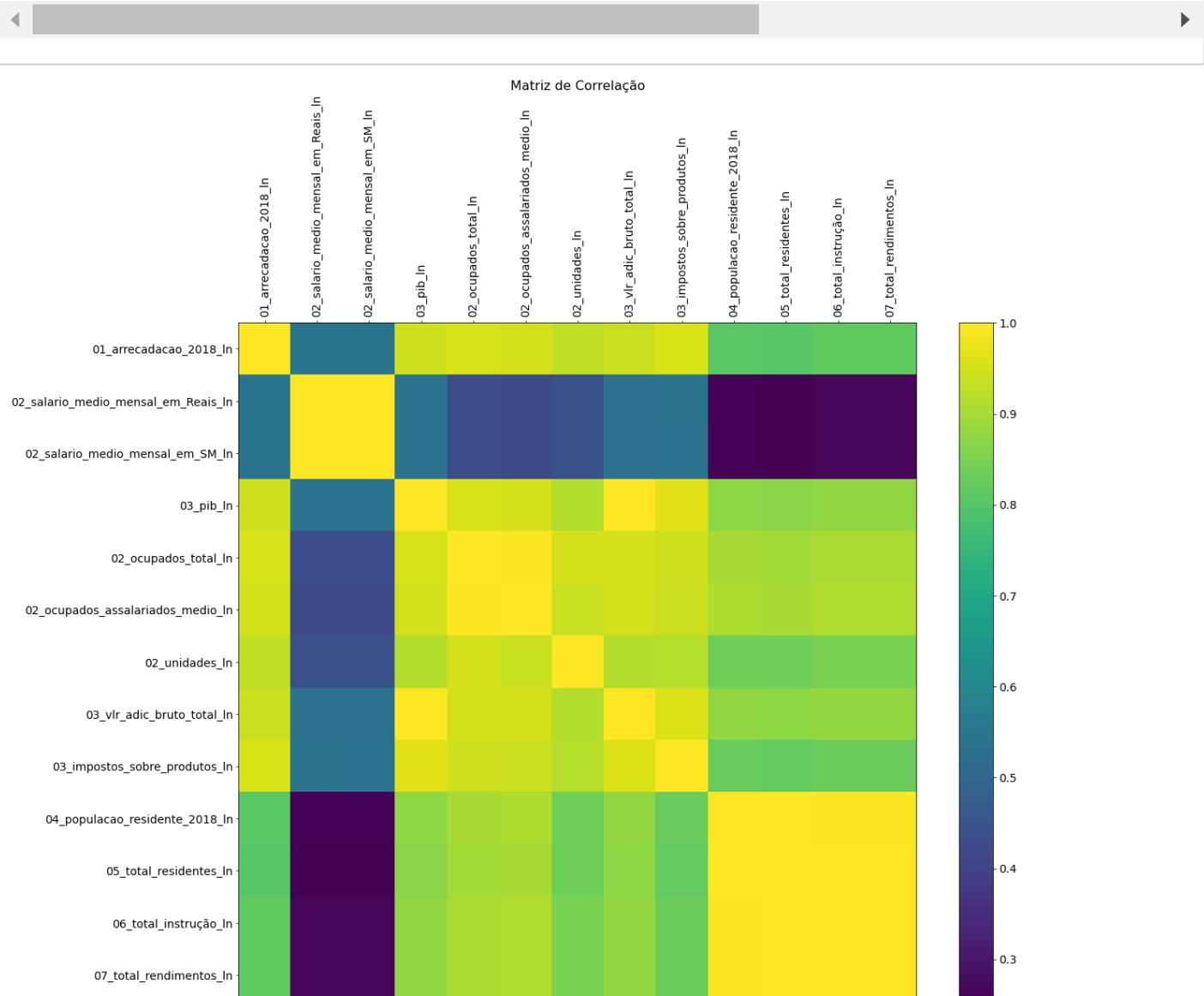
Abaixo imprimiremos a Matriz de Correlações na forma de mapa de calor, por meio da qual também é possível confirmar essas multicolinearidades.

In [42]:

```
f = plt.figure(figsize=(19, 15))

# plt.rc('font', size=8)
# plt.rc('axes', titlesize=8)
plt.rc('axes', labelsize=20)
plt.rc('xtick', labelsize=8)
plt.rc('ytick', labelsize=8)
# plt.rc('legend', fontsize=8)
# plt.rc('figure', titlesize=12)

plt.matshow(df_transformado[colunas_a_exibir].corr(), fignum=f.number)
plt.xticks(ticks=range(df_transformado[colunas_a_exibir].shape[1]), labels=df_transformado[colunas_a_exibir].columns, rotation=90)
plt.yticks(ticks=range(df_transformado[colunas_a_exibir].shape[1]), labels=df_transformado[colunas_a_exibir].columns, rotation=0)
cb = plt.colorbar()
cb.ax.tick_params(labelsize=14)
plt.title('Matriz de Correlação', fontsize=16);
```



## 3. Exclusão de variáveis que não são de interesse para criação dos modelos

### 3.1. Exclusão das variáveis independentes com baixa correlação com o *label*

Na sequência, identificaremos e excluiremos as variáveis independentes com baixa correlação com o label (assim consideradas as correlações inferiores a 0.01).

In [43]:

```
correlacoes=df_transformado[colunas_a_manter].corr().abs()
print (df_transformado[colunas_a_manter].shape)
correlacoes=pd.DataFrame(correlacoes['01_arrecadacao_2018'].sort_values(ascending=False))
baixas_correlacoes = correlacoes.index[correlacoes['01_arrecadacao_2018']<=.01]
print(baixas_correlacoes.shape)
baixas_correlacoes
```

(5565, 94)

(41,)

Out[43]:

```
Index(['03_uf_BA',
       '03_atividade_principal_segunda_Pecuária, inclusive apoio à pecuária',
       '03_uf_PI', '03_uf_PB',
       '03_atividade_principal_terceira_Produção florestal, pesca e aquicultura',
       '03_uf_MA', '03_uf_RN', '03_uf_TO', '03_uf_GO', '03_uf_RS',
       '03_atividade_principal_primeira_Pecuária, inclusive apoio à pecuária',
       '03_atividade_principal_primeira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação',
       '03_atividade_principal_terceira_Indústrias de transformação',
       '03_atividade_principal_segunda_Indústrias extractivas', '03_uf_AL',
       '03_uf_MT', '03_uf_PA', '03_uf_CE',
       '03_atividade_principal_segunda_Produção florestal, pesca e aquicultura',
       '03_uf_SE', '03_uf_PR', '03_atividade_principal_terceira_Construção',
       '03_uf_PE',
       '03_atividade_principal_primeira_Produção florestal, pesca e aquicultura',
       '03_atividade_principal_terceira_Indústrias extractivas', '03_uf_RO',
       '03_uf_MS',
       '03_atividade_principal_terceira_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação',
       '03_atividade_principal_primeira_Comércio e reparação de veículos automotores e motocicletas',
       '03_atividade_principal_segunda_Eletricidade e gás, água, esgoto, atividades de gestão de resíduos e descontaminação',
       '03_uf_AC', '03_atividade_principal_primeira_Indústrias extractivas',
       '03_uf_SC', '03_atividade_principal_segunda_Construção', '03_uf_AP',
       '03_uf_RR', '03_atividade_principal_primeira_Construção', '03_uf_ES',
       '03_atividade_principal_primeira_Indústrias de transformação',
       '03_atividade_principal_segunda_Administração, defesa, educação e saúde públicas e segurança social',
       '03_uf_AM'],
      dtype='object')
```

In [44]:

```
# Exclusão das variáveis com correlação inferior a 0.01.

print ('Shape:')
print ('- antes das exclusões',df_transformado.shape)

#apaga as variáveis (transformadas) com multicolinearidade do dataset df_transformado, utilizando
colunas_a_apagar = ['02_ocupados_assalariados_medio', '02_ocupados_total', '02_unidades', '03_impostos_sobre_produtos', '04_populacao_residente_2018', '05_total_07_total_rendimentos','03_uf_BA','03_atividade_principal_segunda_Pecuária','03_uf_PI','03_uf_PB','03_atividade_principal_terceira_Produção florestal','03_uf_MA','03_uf_RN','03_uf_TO','03_uf_GO','03_uf_RS','03_atividade_principal_primeira_Pecuária, inclusive apoio à pecuária','03_atividade_principal_primeira_Eletricidade e gás, água, esgoto, atividade','03_atividade_principal_terceira_Indústrias de transformação','03_atividade_principal_segunda_Indústrias extractivas','03_uf_AL','03_uf_MT','03_uf_PA','03_uf_CE','03_atividade_principal_03_uf_SE','03_uf_PR','03_atividade_principal_terceira_Construção','03_atividade_principal_primeira_Produção florestal, pesca e aquicultura','03_atividade_principal_terceira_Indústrias extractivas','03_uf_RO','03_atividade_principal_terceira_Eletricidade e gás, água, esgoto, atividade','03_atividade_principal_primeira_Comércio e reparação de veículos automóveis','03_atividade_principal_segunda_Eletricidade e gás, água, esgoto, atividade','03_uf_AC','03_atividade_principal_primeira_Indústrias extractivas','03_atividade_principal_segunda_Construção','03_uf_AP','03_uf_RR','03_uf_ES','03_atividade_principal_primeira_Indústrias de transformação','03_atividade_principal_segunda_Administração, defesa, educação e saúde','03_uf_AM']

df_transformado = df_transformado.drop(colunas_a_apagar, axis=1)
print ('- após exclusões.shape',df_transformado.shape)
```

Shape:

- antes das exclusões (5565, 141)
- após exclusões.shape (5565, 90)

### 3.2. Exclusão das variáveis criadas na conversão para a base logarítmica natural

In [45]:

```
#Apaga as variáveis criadas na conversão para a base Logarítmica
print ('Shape:')
print ('- antes das exclusões',df_transformado.shape)

colunas_a_apagar = ['01_arrecadacao_2018_ln','02_unidades_ln', '02_unidades_atuantes_%_ln',
'02_ocupados_assalariados_%_ln', '02_ocupados_assalariados_medio_ln',
'02_salario_medio_mensal_em_SM_ln',
'02_salario_medio_mensal_em_Reais_ln', '03_vlr_adic_bruto_total_ln',
'03_vlr_adic_bruto_agropecuaria_%_ln',
'03_vlr_adic_bruto_industria_%_ln', '03_vlr_adic_bruto_servicos_%_ln',
'03_vlr_adic_bruto_administracao_%_ln', '03_impostos_sobre_produtos_ln',
'03_pib_ln', '03_pib_per_capita_ln', '04_populacao_residente_2018_ln',
'05_total_residentes_ln', '05_0-19_anos_%_ln', '05_20-39_anos_%_ln',
'05_40-59_anos_%_ln', '05_60+_anos_%_ln', '06_total_instrução_ln',
'06_Sem_Instrução_e_Fundamental_Incompleto%_ln',
'06_Medio_Incompleto_%_ln', '06_Superior_Incompleto_%_ln',
'06_Superior_Completo_%_ln', '06_Indeterminado_%_ln',
'07_total_rendimentos_ln', '07_Ate_1SM_%_ln', '07_Ate_2SM_%_ln',
'07_Ate_3SM_%_ln', '07_Ate_5SM_%_ln', '07_Ate_10SM_%_ln',
'07_Ate_20SM_%_ln', '07_20SM+_%_ln', '07_Sem_Rendimento_%_ln']
df_transformado = df_transformado.drop(colunas_a_apagar, axis=1)
df_transformado.shape

print ('- após exclusões.shape',df_transformado.shape)
```

Shape:

- antes das exclusões (5565, 90)
- após exclusões.shape (5565, 53)

### 3.3. Exclusão de variáveis categóricas sem interesse

In [46]:

```
#Exclusão de variáveis que não serão úteis à aplicação dos modelos de machine learning.

print ('Shape:')
print ('- antes das exclusões',df_transformado.shape)

colunas_a_apagar = ['03_nome_uf', '03_cod_municipio', '03_grande_regiao', '03_mesorregiao',
'03_microregiao', '03_regiao_imediata', '03_regiao_intermediaria']

df_transformado = df_transformado.drop(colunas_a_apagar, axis=1)

print ('- após as exclusões',df_transformado.shape)
```

Shape:

- antes das exclusões (5565, 53)
- após as exclusões (5565, 46)

### 3.4. Exclusão da variável auxiliar criada ('indice\_ordenado')

In [47]:

```
#Exclusão da variável auxiliar criada (indice_ordenado)

print ('Shape:')
print ('- antes das exclusões',df_transformado.shape)

colunas_a_apagar = ['indice_ordenado']

df_transformado = df_transformado.drop(colunas_a_apagar, axis=1)

print ('- após as exclusões',df_transformado.shape)
```

Shape:

- antes das exclusões (5565, 46)
- após as exclusões (5565, 45)

## 4. Criação do dataset consolidado 'ntbk 03 - dataset consolidado - maiores correlacoes.csv'

In [48]:

```
# reordena colunas

colunas = ['municipio-uf', 'faixa_arrecadacao', '01_arrecadacao_2018', '02_unidades_atuante',
           '02_ocupados_assalariados_%', '02_salario_medio_mensal_em_Reais',
           '03_uf_DF', '03_uf_MG',
           '03_uf_RJ', '03_uf_SP',
           '03_vlr_adic_bruto_agropecuaria_%', '03_vlr_adic_bruto_industria_%',
           '03_vlr_adic_bruto_servicos_%', '03_vlr_adic_bruto_administracao_%',
           '03_pib', '03_pib_per_capita',
           '03_atividade_principal_primeira_Administração, defesa, educação e saúde públicas e',
           '03_atividade_principal_primeira_Agricultura, inclusive apoio à agricultura e a pós',
           '03_atividade_principal_primeira_Demais serviços',
           '03_atividade_principal_segunda_Agricultura, inclusive apoio à agricultura e a pós',
           '03_atividade_principal_segunda_Comércio e reparação de veículos automotores e motocicletas',
           '03_atividade_principal_segunda_Demais serviços',
           '03_atividade_principal_segunda_Indústrias de transformação',
           '03_atividade_principal_terceira_Administração, defesa, educação e saúde públicas e',
           '03_atividade_principal_terceira_Agricultura, inclusive apoio à agricultura e a pós',
           '03_atividade_principal_terceira_Comércio e reparação de veículos automotores e motocicletas',
           '03_atividade_principal_terceira_Demais serviços',
           '03_atividade_principal_terceira_Pecuária, inclusive apoio à pecuária',
           '05_0-19_anos_%', '05_20-39_anos_%',
           '05_40-59_anos_%', '05_60+_anos_%',
           '06_Sem_Instrução_e_Fundamental_Incompleto%', '06_Medio_Incompleto_%',
           '06_Superior_Incompleto_%', '06_Superior_Completo_%',
           '06_Indeterminado_%', '07_Ate_1SM_%', '07_Ate_2SM_%', '07_Ate_3SM_%',
           '07_Ate_5SM_%', '07_Ate_10SM_%', '07_Ate_20SM_%', '07_20SM+_%',
           '07_Sem_Rendimento_%']

df_transformado = df_transformado[colunas]
```

In [49]:

```
print(df_transformado.shape)
df_transformado.columns
```

(5565, 45)

Out[49]:

```
Index(['municipio-uf', 'faixa_arrecadacao', '01_arrecadacao_2018',
       '02_unidades_atuantes_%', '02_ocupados_assalariados_%',
       '02_salario_medio_mensal_em_Reais', '03_uf_DF', '03_uf_MG', '03_uf_R
J',
       '03_uf_SP', '03_vlr_adic_bruto_agropecuaria_%',
       '03_vlr_adic_bruto_industria_%', '03_vlr_adic_bruto_servicos_%',
       '03_vlr_adic_bruto_administracao_%', '03_pib', '03_pib_per_capita',
       '03_atividade_principal_primeira_Administração, defesa, educação e sa
úde públicas e seguridade social',
       '03_atividade_principal_primeira_Agricultura, inclusive apoio à agric
ultura e a pós colheita',
       '03_atividade_principal_primeira_Demais serviços',
       '03_atividade_principal_segunda_Agricultura, inclusive apoio à agricu
ltura e a pós colheita',
       '03_atividade_principal_segunda_Comércio e reparação de veículos auto
motores e motocicletas',
       '03_atividade_principal_segunda_Demais serviços',
       '03_atividade_principal_segunda_Indústrias de transformação',
       '03_atividade_principal_terceira_Administração, defesa, educação e sa
úde públicas e seguridade social',
       '03_atividade_principal_terceira_Agricultura, inclusive apoio à agric
ultura e a pós colheita',
       '03_atividade_principal_terceira_Comércio e reparação de veículos aut
omotores e motocicletas',
       '03_atividade_principal_terceira_Demais serviços',
       '03_atividade_principal_terceira_Pecuária, inclusive apoio à pecuári
a',
       '05_0-19_anos_%', '05_20-39_anos_%', '05_40-59_anos_%', '05_60+_anos
_%',
       '06_Sem_Instrução_e_Fundamental_Incompleto%', '06_Medio_Incompleto
_%',
       '06_Superior_Incompleto_%', '06_Superior_Completo_%',
       '06_Indeterminado_%', '07_Ate_1SM_%', '07_Ate_2SM_%', '07_Ate_3SM_%',
       '07_Ate_5SM_%', '07_Ate_10SM_%', '07_Ate_20SM_%', '07_20SM+_%',
       '07_Sem_Rendimento_%'],
      dtype='object')
```

In [50]:

```
df_transformado.to_csv('ntbk 03 - dataset consolidado - maiores correlacoes.csv', index=False)
```

In [ ]:

# Apêndice IX – Ntbk 04. Criação dos modelos de machine learning-Regressão.ipynb

Neste notebook serão avaliados modelos de aprendizado de máquina (machine learning), operacionalizados com a aplicação de algoritmos para Regressão. Na sequência, será feita uma análise de resíduos e, por fim, será feita uma análise comparada entre o melhor e o segundo melhor modelos.

O objetivo deste projeto é explorar atributos econômicos, sociais, demográficos e geopolíticos disponibilizados de maneira pública pelo IBGE (variáveis independentes) e avaliar sua capacidade de predizer a arrecadação dos municípios brasileiros para o ano de 2018 (variável dependente ou label: 01\_arrecadacao\_2018), informação essa também pública e disponibilizada pela Receita Federal do Brasil

Para a identificação do melhor modelo foram implementadas uma série de ações e experimentos, a seguir enumerados:

1. Ações Preliminares;
2. Importação do dataset que contém os dados de interesse;
3. Criação dos *dataframes* de estudo;
4. Separação das bases de treinamento e de teste;
5. Definição de uma baseline (valor de referência) par avaliação dos modelos;
6. Aplicação dos algoritmos e comparação da capacidade de previsão em relação a cada um dos dataframes de estudo;
7. Análise dos resíduos, inclusive comparando a performance entre os dois modelos de melhor resultado e analisando os impactos da exclusão dos municípios de maior arrecadação para o algoritmo ElasticNet;
8. Aprimoramento do modelo mais promissor por meio da otimização de hiperparâmetros e uso de cross validation.

## 1. Ações Preliminares

### - Importação das bibliotecas de interesse.

In [1]:

```
import pandas as pd
from pandas import DataFrame

import numpy as np
from numpy import arange

import matplotlib
import matplotlib.pyplot as plt

from sklearn.preprocessing import FunctionTransformer
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.ensemble import RandomForestRegressor
from sklearn import svm
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RepeatedKFold

from math import sqrt

pd.set_option('max_columns', 200)
pd.set_option('max_rows', 200)

%matplotlib inline
```

- Declaração de funções que serão utilizadas no notebook.

In [2]:

```
# Converte variáveis numéricas para a base logarítmica
# Pode ser passado por parâmetro uma única variável, ou mesmo o dataframe inteiro.

def converte_para_ln(df,cols=''):
    if cols == '':
        cols = df.columns.tolist()
    contador = 0
    for col in cols:

        if df[col].dtype in [np.int64, np.int32, np.float64]:
            nova_coluna = col + "_ln"
            # A linha abaixo calcula o Log de cada valor das colunas numéricas
            # O abs para x é porque o np.log retorna erro para valores negativos, usaremos,
            # A função Log retorna um erro (-inf) quando o x = 0
            # Assim, quando o valor for zero, a função retornará -13,82 (que é o Ln de 0,00
            df[nova_coluna]=df[col].apply(lambda x: np.log(abs(x)) if x!= 0 else -13,82).as
            contador = contador + 1
    print ('Colunas criadas:',contador)
    return('Concluído')
```

In [3]:

```
# Converte variáveis numéricas usando a técnica MinMax

def converte_para_MinMax(df,cols):

    contador = 0
    for col in cols:

        #define um novo nome para as colunas transformadas
        nova_coluna = col + "_minmax"

        # transforma os dados para o formato exigido pelo MinMaxScaler() e procede à transf
        dados_a_transformar = df[col].values.reshape(-1, 1)
        scaler = MinMaxScaler()
        dados_transformados = scaler.fit_transform(dados_a_transformar)

        # restabelece o formato DataFrame e atribui o valor à nova coluna criada
        df[nova_coluna] = DataFrame(dados_transformados)
        #print (col, nova_coluna, " - Min", scaler.data_min_, " - Máx", scaler.data_max_, " "

        contador = contador + 1
    print ('Colunas criadas:',contador)
    return('Concluído')
```

In [4]:

```
# Calcula a métrica Root Mean Squared Error
def rmse(y_true, y_pred):
    return sqrt(mean_squared_error(y_true,y_pred))
```

In [5]:

```
#Cria um scatter plot com o resultado do modelo de machine Learning.
def imprime_scatter_plot(y_test,y_pred):
    plt.scatter(y_test, y_pred)
    range = [y_test.min(), y_pred.max()]
    plt.plot(range, range, 'red')
    plt.xlabel('Arrecadação real (em log neperiano)')
    plt.ylabel('Arrecadação predita (em log neperiano)')
    plt.show()
    return ''
```

In [6]:

```
#aplica modelos de machine Learning

def aplicar_modelo_ML(df, modelo_definido, x_train, y_train, x_test, y_test):

    print('Dataframe:',df)

    modelo = modelo_definido
    print("Modelo:",modelo)

    modelo.fit(x_train,y_train)
    y_pred = modelo.predict(x_train)
    rmse_train=rmse(y_train,y_pred)
    mae_train = mean_absolute_error(y_train,y_pred)
    print("MAE na base de treinamento: ",mae_train)
    print("RMSE na base de treinamento: ",rmse_train)
    print("R² na base de treinamento: {:.2f}".format(modelo.score(x_train, y_train)))

    modelo.fit(x_train,y_train)
    y_pred = modelo.predict(x_test)
    rmse_test=rmse(y_test,y_pred)
    mae_test = mean_absolute_error(y_test,y_pred)
    print("MAE na base de teste: ",mae_test)
    print("RMSE na base de teste: ",rmse_test)
    print("R² na base de teste: {:.2f}".format(modelo.score(x_test, y_test)))
    print("\n")

    y_test_ln = np.log(abs(np.where(y_test==0,0.00001,y_test)))
    y_pred_ln = np.log(abs(np.where(y_pred==0,0.00001,y_pred)))

    #imprime_scatter_plot(np.log(y_test),np.log(y_pred))
    imprime_scatter_plot(y_test_ln,y_pred_ln)

    resultados = {'Dataframe': df,
                  'Modelo': [str(modelo)],
                  'MAE - Base Treinamento':mae_train,
                  'MAE - Base Teste':mae_test,
                  'RMSE - Base Treinamento': rmse_train,
                  'RMSE - Base Teste': rmse_test,
                  'R2 (Acurácia) - Base Treinamento': modelo.score(x_train, y_train),
                  'R2 (Acurácia) - Base Teste': modelo.score(x_test, y_test)}

    return resultados
```

In [7]:

```
def apura_coeficientes(modelo_usado,X,x_train,y_train):

    modelo = modelo_usado.fit(x_train,y_train)
    nome_coluna = 'Coeficientes_' + str(modelo)[-2]
    nome_coluna_modulo = 'Coeficientes_' + str(modelo)[-2] + '_módulo'

    coeff = pd.Series(X.columns,name='Variáveis independentes').to_frame()
    coeff[nome_coluna] = pd.Series(modelo.coef_,name='Coeficientes').astype(float).to_frame()
    coeff[nome_coluna_modulo] = (coeff[nome_coluna]**2)**0.5
    coeff=coeff.sort_values(by=[nome_coluna_modulo],ascending=False)

    features = coeff['Variáveis independentes']
    coeficientes = coeff[nome_coluna_modulo]

    plt.figure(figsize=(10,5))
    plt.ylabel('Coeficientes (em módulo)')
    plt.xlabel('Variáveis independentes')
    plt.title('Variáveis independentes e Coeficientes (em módulo)')
    plt.bar(features,coeficientes)
    plt.xticks(rotation = 90)
    plt.show()

    return coeff
```

In [8]:

```
def plotstats(df, col):
    import matplotlib.pyplot as plt

    plt.rc('axes', labelsize=20)
    plt.rc('xtick', labelsize=20)
    plt.rc('ytick', labelsize=20)

    ## Setup for plotting two charts one over the other
    fig, ax = plt.subplots(2, 1, figsize = (12,8))
    ## First a box plot
    #df.dropna().boxplot(col, ax = ax[0], vert=False,return_type='dict')
    df.boxplot(col, ax = ax[0], vert=False,return_type='dict')
    ## Plot the histogram
    ## temp = df[col].as_matrix()
    temp = df[col].values
    ax[1].hist(temp, bins = 30, alpha = 0.7)
    plt.ylabel('Número de Cidades')
    plt.xlabel(col)

    return [col]
```

In [9]:

```
def plot_residuo(df,x_indicado,y_indicado,coluna_residuo,predicao,predictor):  
  
    nrow = df.shape[0]  
    df.sort_values(by=y_indicado, ascending=True, inplace = True)  
  
    fig, ax = plt.subplots(2, 1, figsize = (36,36))  
  
    fonte = 40  
  
    plt.rc('axes', labelsize=fonte)  
    plt.rc('xtick', labelsize=fonte)  
    plt.rc('ytick', labelsize=fonte)  
  
    df.plot(kind = 'scatter', x = x_indicado, y = y_indicado , ax = ax[0], alpha = 0.5)  
  
    ax[0].set_xlabel('Municípios (ordem ascendente de arrecadação)',size=fonte)  
    ax[0].set_ylabel('Resíduos (base logarítmica)', size=fonte)  
    #ax[0].set_title('X vs. Y',size=fonte)  
  
    #ax[0].scatter(df_Log['indice_ordenado'],df_Log['01_arrecadacao_2018_Ln'],color='black'  
    #ax[0].set_xlabel('Municípios (em ordem crescente de arrecadação)')  
    #ax[0].set_ylabel('Arrecadação (base Logarítmica)')  
  
    ax[1].hist(df[coluna_residuo], bins = 30, alpha = 0.7)  
    #ax[1].hist(df['01_arrecadacao_2018_Ln'], bins = 30, alpha = 0.7, histtype='step', colo  
    ax[1].set_xlabel('Resíduos (base logaritmica)',size=fonte)  
    ax[1].set_ylabel('Número de municípios',size=fonte)  
    ax[1].set_title('Histograma dos Resíduos',size=fonte)  
  
    return ''
```

In [10]:

```
def plot_arrecadacao(df, faixa_arrec):  
  
    matplotlib.use('agg')  
  
    df.sort_values(by='indice_ordenado', ascending=True, inplace = True)  
    for tm in faixa_arrec:  
        fig = plt.figure(figsize=(50, 10))  
        fig.clf()  
        ax = fig.gca()  
        df[df['faixa_arrecadacao'] == tm].plot(kind = 'line',  
                                                x = 'indice_ordenado', y = '01_arrecadacao_2018_ln', color =  
                                                df[df['faixa_arrecadacao'] == tm].plot(kind = 'line',  
                                                x = 'indice_ordenado', y = 'predicao_melhor_modelo_ln', color =  
  
        fonte = 40  
  
        plt.xlabel("Faixas Arrecadação", size = fonte)  
        plt.ylabel("Arrecadação", size = fonte)  
        plt.title("Arrecadação por Faixa Arrecadação = " + str(tm), size = fonte)  
        plt.legend(fontsize = fonte/1.2)  
  
        #fig.savefig('faixa_arrec_' + str(tm) + '.png')  
  
    return 'Concluído'
```

In [11]:

```
def box_residuos(df):

    matplotlib.use('agg') # Define backend

    #df['residuos_%'] = df['residuos_Ln']/df['01_arrecadacao_2018_Ln']*100
    df['residuos_%'] = df['residuos']/df['01_arrecadacao_2018']*100

    fig = plt.figure(figsize=(12, 6))
    fig.clf()
    ax = fig.gca()
    df[df['residuos_%'] <= 200].boxplot(column = ['residuos_%'], by = ['faixa_arrecadacao'])
    #df.boxplot(column = ['residuos_Ln'], by = ['faixa_arrecadacao'], ax = ax)

    fonte = 15

    plt.xlabel('',size = fonte)
    plt.ylabel('Resíduos %',size=fonte)
    plt.title('',size=fonte)
    #plt.set_xticklabels(x_ticks, rotation=0, fontsize=fonte)
    #plt.set_yticklabels(y_ticks, rotation=0, fontsize=fonte)

    #plt.rc('axes', labelsize=fonte)
    #plt.rc('xtick', labelsize=fonte)
    #plt.rc('ytick', labelsize=fonte)

    #fig.savefig('boxes' + '.png')
    return 'Concluído'
```

In [12]:

```
def resids_hist(df, faixa_arrec):

    matplotlib.use('agg')

    plt.rc('axes', labelsize=20)
    plt.rc('xtick', labelsize=20)
    plt.rc('ytick', labelsize=20)

    for tm in faixa_arrec:
        fig = plt.figure(figsize=(8, 6))
        fig.clf()
        ax = fig.gca()
        ax.hist(df.loc[df['faixa_arrecadacao'] == tm, 'residuos_Ln'].values, bins = 30)
        plt.xlabel("Residuals")
        plt.ylabel("Density")
        plt.title("Histograma dos resíduos por faixa de valor = " + str(tm))
        #fig.savefig('hist_' + str(tm) + '.png')

    return 'Concluído'
```

## 2. Importação do dataset que contém os dados de interesse

A base utilizada para a realização dos experimentos foi “ntbk 03 – dataset consolidado - maiores correlacoes.csv”. Detalhes sobre a criação dessa base pode ser obtida no **Apêndice VIII – Ntbk 03. Análise e exploração dos dados.ipynb**.

O dataset possui 5565 linhas e 45 variáveis. Com exceção de ‘municipio-uf’, todas as demais variáveis já estão em formatos numéricos. Embora numérica, a variável ‘faixa\_arrecadacao’, criada neste projeto para classificação das arrecadações, não foi utilizada para o modelo, vez que sua criação exige conhecer previamente o valor das arrecadações o que, obviamente, não estará disponível em uma eventual predição de futuras arrecadações – ela, porém, será útil adiante para a análise dos resíduos.

In [13]:

```
df_integral=pd.read_csv('ntbk 03 - dataset consolidado - maiores correlacoes.csv')

print(df_integral.shape)
print (df_integral.dtypes)
df_integral.head()

(5565, 45)
municipio-uf
object
faixa_arrecadacao
int64
01_arrecadacao_2018
float64
02_unidades_atuantes_%
float64
02_ocupados_assalariados_%
float64
02_salario_medio_mensal_em_Reais
float64
03_uf_DF
int64
03_uf_MG
int64
03_uf_RJ
int64
03_uf_SP
int64
03_vlr_adic_bruto_agropecuaria_%
float64
03_vlr_adic_bruto_industria_%
float64
03_vlr_adic_bruto_servicos_%
float64
03_vlr_adic_bruto_administracao_%
float64
03_pib
float64
03_pib_per_capita
float64
03_atividade_principal_primeira_Administração, defesa, educação e saúde públ
icas e segurança social      int64
03_atividade_principal_primeira_Agricultura, inclusive apoio à agricultura e
a pós colheita              int64
03_atividade_principal_primeira_Demais serviços
int64
03_atividade_principal_segunda_Agricultura, inclusive apoio à agricultura e
a pós colheita              int64
03_atividade_principal_segunda_Comércio e reparação de veículos automotores
e motocicletas              int64
03_atividade_principal_segunda_Demais serviços
int64
03_atividade_principal_segunda_Indústrias de transformação
int64
03_atividade_principal_terceira_Administração, defesa, educação e saúde públ
icas e segurança social      int64
03_atividade_principal_terceira_Agricultura, inclusive apoio à agricultura e
a pós colheita              int64
03_atividade_principal_terceira_Comércio e reparação de veículos automotores
e motocicletas              int64
```

```

03_atividade_principal_terceira_Demais serviços
int64
03_atividade_principal_terceira_Pecuária, inclusive apoio à pecuária
int64
05_0-19_anos_%
float64
05_20-39_anos_%
float64
05_40-59_anos_%
float64
05_60+anos_%
float64
06_Sem_Instrução_e_Fundamental_Incompleto%
float64
06_Medio_Incompleto_%
float64
06_Superior_Incompleto_%
float64
06_Superior_Completo_%
float64
06_Indeterminado_%
float64
07_Ate_1SM_%
float64
07_Ate_2SM_%
float64
07_Ate_3SM_%
float64
07_Ate_5SM_%
float64
07_Ate_10SM_%
float64
07_Ate_20SM_%
float64
07_20SM+_%
float64
07_Sem_Rendimento_%
float64
dtype: object

```

Out[13]:

	municipio-uf	faixa_arrecadacao	01_arrecadacao_2018	02_unidades_atuantes_%	02_ocupado
0	ALTA FLORESTA D'OESTE-RO	2	2.787022e+07	98.464491	
1	ALTO ALEGRE DOS PARECIS-RO	2	9.658836e+06	98.773006	
2	ALTO PARAISO-RO	2	1.178047e+07	99.528302	
3	ALVORADA D'OESTE-RO	2	6.824321e+06	98.591549	
4	ARIQUEMES- RO	3	1.826154e+08	96.011169	

### 3. Criação dos *dataframes* de estudo

Serão criados três dataframes de estudo, derivados do df\_integral (instanciado com o dataset “ntbk 03 - dataset consolidado - maiores Correlacoes.csv” – vide sequência de comandos da figura acima). O primeiro (df\_original) manterá os dados em sua forma original (excluídas as variáveis ‘municipio-uf’ e ‘faixa\_arrecadacao’). Os dois seguintes, serão resultado da conversão das variáveis dependentes: um para a base logarítmica natural (df\_log) e, outro, utilizando a técnica MinMax (df\_MinMax). A variável dependente (label: 01\_arrecadacao\_2018) será mantida em sua forma original, por ser este o objetivo declarado para este projeto.

#### - df\_original

Contém os dados originais da base.

In [14]:

```
df_original = df_integral.copy()
df_original = df_original.drop('faixa_arrecadacao', axis=1)
print(df_original.shape)
df_original.describe()
```

(5565, 44)

Out[14]:

	01_arrecadacao_2018	02_unidades_atuantes_%	02_ocupados_assalariados_%	02_salario_
count	5.565000e+03	5565.000000	5565.000000	
mean	2.494351e+08	97.442228	85.305250	
std	4.865354e+09	2.952571	6.761261	
min	4.165360e+05	56.103286	21.124207	
25%	3.775316e+06	96.551724	81.526104	
50%	9.175717e+06	97.983015	85.862069	
75%	3.150111e+07	99.206349	89.880263	
max	2.804407e+11	100.000000	99.489796	

#### - df\_log

Abaixo a rotina para a conversão das variáveis independentes para a base logarítmica natural

In [15]:

```
df_log = df_original.copy()

# Seleciona as colunas numéricas...
colunas_a_converter = df_log.dtypes[df_log.dtypes != "object"].index.tolist()
#... exceto a primeira: '01_arrecadacao_2018'.
colunas_a_converter.pop(0)

# Chama a função que fará a conversão
converte_para_ln(df_log, colunas_a_converter)

# Apaga as colunas originais
df_log = df_log.drop(colunas_a_converter, axis=1)
print (df_log.shape)
df_log.describe()
```

Colunas criadas: 42  
(5565, 44)

Out[15]:

	01_arrecadacao_2018	02_unidades_atuantes_%_ln	02_ocupados_assalariados_%_ln	02_sa
<b>count</b>	5.565000e+03	5565.000000	5565.000000	
<b>mean</b>	2.494351e+08	4.578749	4.442744	
<b>std</b>	4.865354e+09	0.032886	0.086424	
<b>min</b>	4.165360e+05	4.027194	3.050420	
<b>25%</b>	3.775316e+06	4.570079	4.400923	
<b>50%</b>	9.175717e+06	4.584794	4.452742	
<b>75%</b>	3.150111e+07	4.597202	4.498478	
<b>max</b>	2.804407e+11	4.605170	4.600055	

	01_arrecadacao_2018	02_unidades_atuantes_%_ln	02_ocupados_assalariados_%_ln	02_sa
<b>count</b>	5.565000e+03	5565.000000	5565.000000	
<b>mean</b>	2.494351e+08	4.578749	4.442744	
<b>std</b>	4.865354e+09	0.032886	0.086424	
<b>min</b>	4.165360e+05	4.027194	3.050420	
<b>25%</b>	3.775316e+06	4.570079	4.400923	
<b>50%</b>	9.175717e+06	4.584794	4.452742	
<b>75%</b>	3.150111e+07	4.597202	4.498478	
<b>max</b>	2.804407e+11	4.605170	4.600055	

- df\_MinMax

In [16]:

```
df_MinMax = df_original.copy()

# Seleciona as colunas numéricas...
colunas_a_converter = df_MinMax.dtypes[df_MinMax.dtypes != "object"].index.tolist()
#... exceto a primeira: '01_arrecadacao_2018'.
colunas_a_converter.pop(0)

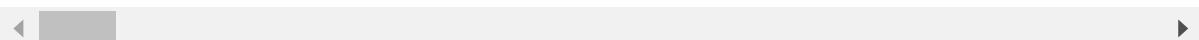
# Chama a função que fará a conversão
converte_para_MinMax(df_MinMax, colunas_a_converter)

# Apaga as colunas originais
df_MinMax = df_MinMax.drop(colunas_a_converter, axis=1)
print(df_MinMax.shape)
df_MinMax.describe()
```

Colunas criadas: 42  
(5565, 44)

Out[16]:

	01_arrecadacao_2018	02_unidades_atuantes_%_minmax	02_ocupados_assalariados_%_mir
count	5.565000e+03	5565.000000	5565.00
mean	2.494351e+08	0.941732	0.81
std	4.865354e+09	0.067262	0.08
min	4.165360e+05	0.000000	0.00
25%	3.775316e+06	0.921446	0.77
50%	9.175717e+06	0.954052	0.82
75%	3.150111e+07	0.981920	0.87
max	2.804407e+11	1.000000	1.00



## 4. Separação dos dataframes em base de treinamento e de teste

Etapa seguinte será separar os dataframes em base treinamento e base teste. Faremos uma distribuição de 70% e 30% entre as bases de treinamento e teste.

Nossa variável *label* (a variável a ser predita) será '01\_arrecadacao\_2018'.

- Dataset df\_original

In [17]:

```
#Identificar a coluna que contém o Label (Y)
Y_orig=df_original['01_arrecadacao_2018']

#X será as demais colunas. Dado X tenho Y.
X_orig=df_original.drop(['municipio-uf','01_arrecadacao_2018'],axis=1)

x_orig_train, x_orig_test, y_orig_train, y_orig_test = train_test_split(X_orig,Y_orig,test_
```

## - Dataset df\_MinMax

In [18]:

```
#Identificar a coluna que contém o Label (Y)
Y_minmax=df_MinMax['01_arrecadacao_2018']

#X será as demais colunas. Dado X tenho Y.
X_minmax=df_MinMax.drop(['municipio-uf','01_arrecadacao_2018'],axis=1)

x_minmax_train, x_minmax_test, y_minmax_train, y_minmax_test = train_test_split(X_minmax,Y_
```

## - Dataset df\_log

In [19]:

```
#Identificar a coluna que contém o Label (Y)
Y_log=df_log['01_arrecadacao_2018']

#X será as demais colunas. Dado X tenho Y.
X_log=df_log.drop(['municipio-uf','01_arrecadacao_2018'],axis=1)

x_log_train, x_log_test, y_log_train, y_log_test = train_test_split(X_log,Y_log,test_size=0
```

# 5. Definição da métrica e de uma *baseline* para avaliação dos resultados.

A qualidade do modelo será aferida a partir da métrica MAE (Mean Absolute Error), embora também seja feita referência as métricas RMSE (Root Mean Square Error) e R<sup>2</sup>:

- Para o cálculo utilizaremos a métrica mean\_absolute\_error, disponível no módulo sklearn.metrics;
- Para o cálculo do RMSE foi definida a função “rmse(y\_true, y\_pred)”, abaixo, que calcula a raiz quadrada do valor da métrica mean\_squared\_error, disponível no módulo sklearn.metrics;
- Já o R<sup>2</sup> é calculado a partir do método “.score” para a classe LinearRegression.

Como baseline, vamos atribuir indistintamente ao y\_pred (valor predito pelo modelo) o valor R\$ 249.435.087,56 (que é a média para a variável "01\_arrecadacao\_2018").

Assim, np.full(y\_test.shape,2.494351e+08,dtype=float) cria uma matriz com o valor mencionado no formato passado entre os parêntesis (no caso, como definimos, y\_xxxx\_teste.shape é 30% do total de linhas, ou 1709 linhas).

Desse modo, para ter alguma utilidade real **nossos modelos deverão ter um RMSE inferiores aos valores exibidos pela linha de comando abaixo para superarem um modelo que considera a média da distribuição total como previsão indistinta.**

In [20]:

```
#rmse(y_test, np.zeros(y_test.shape))
baseline_orig = mean_absolute_error(y_orig_test, np.full(y_orig_test.shape, 249435087.56), dtype='float64')
print ('Baseline para o modelo utilizando os dados originais', baseline_orig)

#rmse(y_test, np.zeros(y_test.shape))
baseline_minmax = mean_absolute_error(y_minmax_test, np.full(y_minmax_test.shape, 249435087.56), dtype='float64')
print ('Baseline para o modelo utilizando os dados transformados - MinMax', baseline_minmax)

#rmse(y_test, np.zeros(y_test.shape))
baseline_log = mean_absolute_error(y_log_test, np.full(y_log_test.shape, 249435087.56), dtype='float64')
print ('Baseline para o modelo utilizando os dados transformados - Log Natural', baseline_log)
```

Baseline para o modelo utilizando os dados originais 504698969.2189041  
Baseline para o modelo utilizando os dados transformados - MinMax 504698969.  
2189041  
Baseline para o modelo utilizando os dados transformados - Log Natural 50469  
8969.2189041

## 6.Aplicação dos algoritmos e comparação da capacidade de previsão em relação a cada um dos dataframes de estudo;

### 6.1. Linear Regression.

Será avaliada, aqui, a capacidade de predição do algoritmo  
sklearn.linear\_model.LinearRegression().

In [21]:

```

modelo = LinearRegression()

#Aplicação LinearRegression() no df_original
resultado = aplicar_modelo_ML('df_original', modelo, x_orig_train, y_orig_train, x_orig_test)
df_resultados_modelos = pd.DataFrame(resultado)

#Aplicação LinearRegression() no df_MinMax
resultado = aplicar_modelo_ML('df_MinMax', modelo, x_minmax_train, y_minmax_train, x_minmax_test)
df_resultados_modelos = df_resultados_modelos.append(resultado, ignore_index=True)

#Aplicação LinearRegression() no df_Log
resultado = aplicar_modelo_ML('df_Log', modelo, x_log_train, y_log_train, x_log_test, y_log_test)
df_resultados_modelos = df_resultados_modelos.append(resultado, ignore_index=True)

#Aponta o melhor modelo.
print ('O melhor resultado foi com o modelo', df_resultados_modelos['Modelo'][df_resultados_modelos['MAE - Base Treinamento'] == min(df_resultados_modelos['MAE - Base Treinamento'])])
print ('Para o dataframe', df_resultados_modelos['Dataframe'][df_resultados_modelos['MAE - Base Teste'] == min(df_resultados_modelos['MAE - Base Teste'])])
print('MAE', min(df_resultados_modelos['MAE - Base Teste']))
print('R²', max(df_resultados_modelos['R² - Base Teste']))

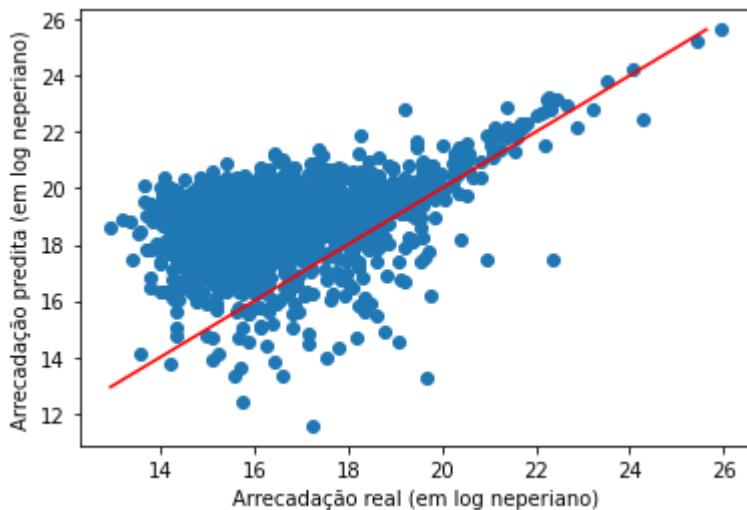
df_resultados_modelos.round(2)

```

```

Dataframe: df_original
Modelo: LinearRegression()
MAE na base de treinamento: 281738172.4810764
RMSE na base de treinamento: 717694796.4431043
R² na base de treinamento: 0.98
MAE na base de teste: 345469132.17838025
RMSE na base de teste: 1593123933.6593137
R² na base de teste: 0.91

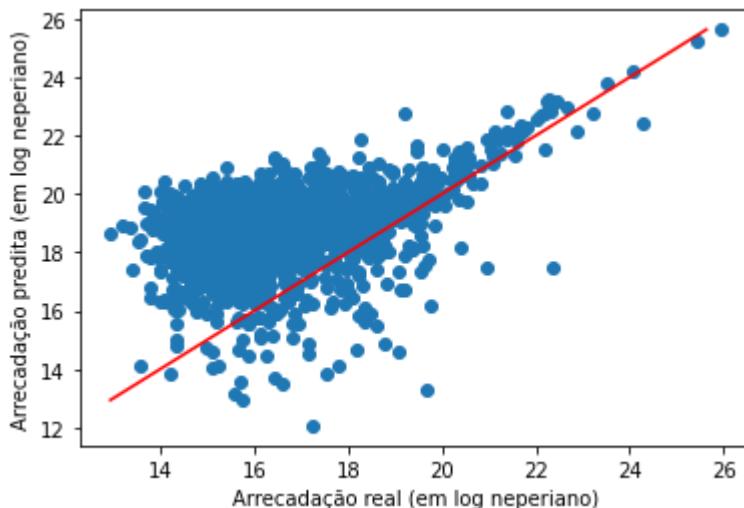
```



```

Dataframe: df_MinMax
Modelo: LinearRegression()
MAE na base de treinamento: 281744618.94495916
RMSE na base de treinamento: 717694787.2375323
R² na base de treinamento: 0.98
MAE na base de teste: 343549603.80602807
RMSE na base de teste: 1566404383.0268245
R² na base de teste: 0.92

```



Dataframe: df\_log

Modelo: LinearRegression()

MAE na base de treinamento: 810437865.5824752

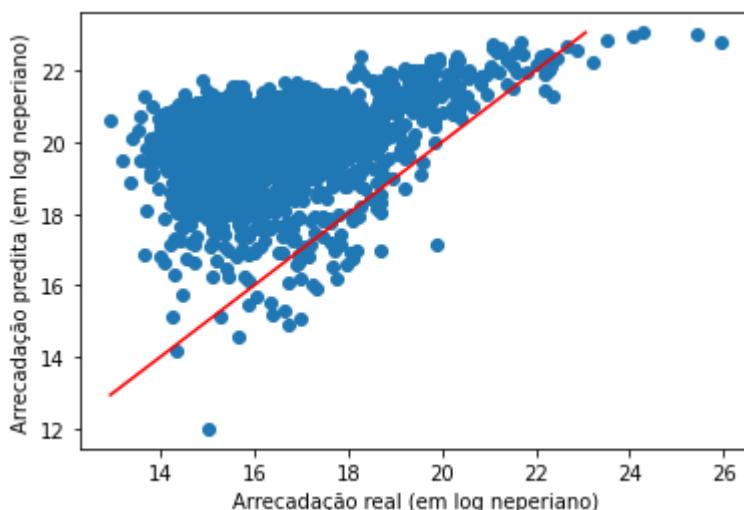
RMSE na base de treinamento: 4432093869.473925

R<sup>2</sup> na base de treinamento: 0.08

MAE na base de teste: 909510154.3941041

RMSE na base de teste: 5123285529.584597

R<sup>2</sup> na base de teste: 0.10



O melhor resultado foi com o modelo [LinearRegression()]

Para o dataframe 1 df\_MinMax

Name: Dataframe, dtype: object

MAE 343549603.80602807

Out[21]:

	Dataframe	Modelo	MAE - Base Treinamento	MAE - Base Teste	RMSE - Base Treinamento	RMSE - Base Teste	(A)
							Treinamento
0	df_original	LinearRegression()	2.817382e+08	3.454691e+08	7.176948e+08	1.593124e+09	
1	df_MinMax	[LinearRegression()]	2.817446e+08	3.435496e+08	7.176948e+08	1.566404e+09	
2	df_log	[LinearRegression()]	8.104379e+08	9.095102e+08	4.432094e+09	5.123286e+09	

Nota-se que o melhor modelo foi o obtido com a aplicação do modelo ao dataset df\_MinMax, com um RMSE de R\$ 1.566.404.383,03 e um  $R^2$  de 91,57%, ambos calculados em relação à base de testes.

Esse RMSE, embora bastante elevado, é consideravelmente inferior à baseline calculada.

Tem-se, porém, um claro indício de overfitting do modelo dada a diferença da performance da base de treinamento, comparada à base de testes.

## 6.2. Ridge (Regressão linear com regularização L2)

Força uma redução do valor dos coeficientes, penalizando coeficientes grandes que não contribuem significativamente para a explicação da variância do sinal.

A força da regularização é dada pelo atributo `alpha`, com valor *default* igual a 1.

In [22]:

```

modelo = Ridge()

#Aplicação Ridge() no df_original
resultado = aplicar_modelo_ML('df_original', modelo, x_orig_train, y_orig_train, x_orig_test)
df_resultados_modelos = df_resultados_modelos.append(resultado, ignore_index=True)

#Aplicação Ridge() no df_MinMax
resultado = aplicar_modelo_ML('df_MinMax', modelo, x_minmax_train, y_minmax_train, x_minmax_test)
df_resultados_modelos = df_resultados_modelos.append(resultado, ignore_index=True)

#Aplicação Ridge() no df_Log
resultado = aplicar_modelo_ML('df_log', modelo, x_log_train, y_log_train, x_log_test, y_log_test)
df_resultados_modelos = df_resultados_modelos.append(resultado, ignore_index=True)

#Aponta o melhor modelo.
print ('O melhor resultado foi com o modelo', df_resultados_modelos['Modelo'][df_resultados_modelos['MAE - Base Treinamento'] == min(df_resultados_modelos['MAE - Base Treinamento'])])
print ('Para o dataframe', df_resultados_modelos['Dataframe'][df_resultados_modelos['MAE - Base Teste'] == min(df_resultados_modelos['MAE - Base Teste'])])

```

df\_resultados\_modelos

```

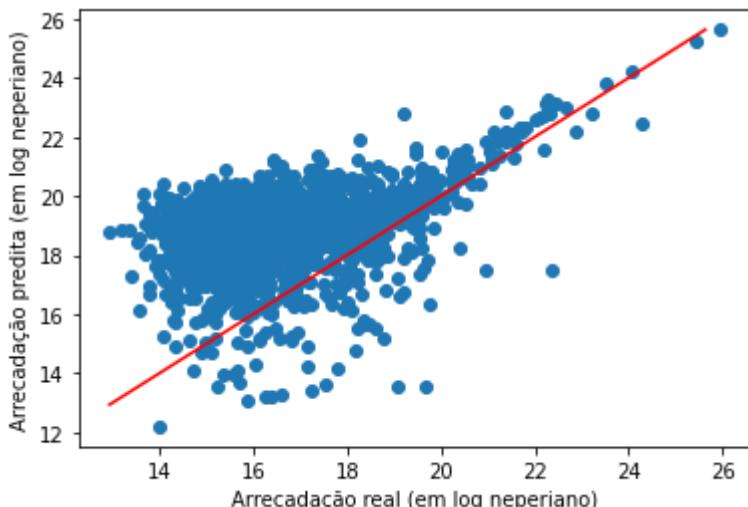
Dataframe: df_original
Modelo: Ridge()
MAE na base de treinamento: 281666714.41100645
RMSE na base de treinamento: 717737977.7999246
R2 na base de treinamento: 0.98
MAE na base de teste: 343151529.8016909
RMSE na base de teste: 1566275894.3813317
R2 na base de teste: 0.92

```

```

C:\ANACONDA\lib\site-packages\sklearn\linear_model\_ridge.py:147: LinAlgWarning: Ill-conditioned matrix (rcond=1.64184e-18): result may not be accurate.
    return linalg.solve(A, Xy, sym_pos=True,
C:\ANACONDA\lib\site-packages\sklearn\linear_model\_ridge.py:147: LinAlgWarning: Ill-conditioned matrix (rcond=1.64184e-18): result may not be accurate.
    return linalg.solve(A, Xy, sym_pos=True,

```



```

Dataframe: df_MinMax
Modelo: Ridge()
MAE na base de treinamento: 343627563.12838566

```

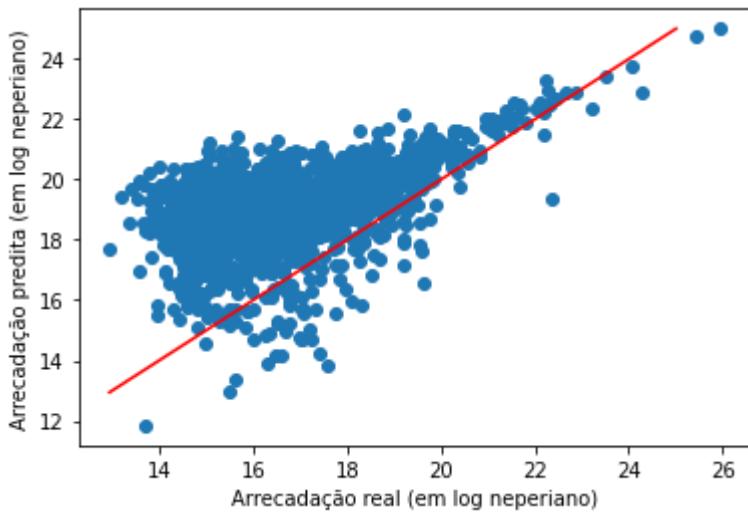
RMSE na base de treinamento: 2299321851.833422

R<sup>2</sup> na base de treinamento: 0.75

MAE na base de teste: 428231869.77576995

RMSE na base de teste: 3152693064.013641

R<sup>2</sup> na base de teste: 0.66



Dataframe: df\_log

Modelo: Ridge()

MAE na base de treinamento: 808023516.6327575

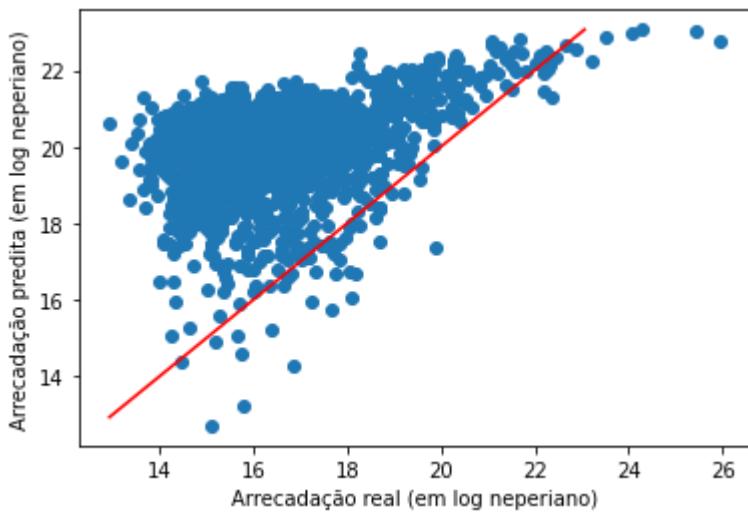
RMSE na base de treinamento: 4432133378.09394

R<sup>2</sup> na base de treinamento: 0.08

MAE na base de teste: 907983022.679373

RMSE na base de teste: 5123834246.806392

R<sup>2</sup> na base de teste: 0.10



O melhor resultado foi com o modelo [Ridge()]

Para o dataframe 3 df\_original

Name: Dataframe, dtype: object

MAE 343151529.8016909

Out[22]:

	Dataframe	Modelo	MAE - Base Treinamento	MAE - Base Teste	RMSE - Base Treinamento	RMSE - Base Teste	
0	df_original	LinearRegression()	2.817382e+08	3.454691e+08	7.176948e+08	1.593124e+09	
1	df_MinMax	[LinearRegression()]	2.817446e+08	3.435496e+08	7.176948e+08	1.566404e+09	
2	df_log	[LinearRegression()]	8.104379e+08	9.095102e+08	4.432094e+09	5.123286e+09	
3	df_original	[Ridge()]	2.816667e+08	3.431515e+08	7.177380e+08	1.566276e+09	▼

### 6.3. LASSO (Regressão linear com regularização L1)

Força uma redução do valor dos coeficientes, podendo zerar diversos coeficientes cujos atributos não contribuem significativamente para a previsão. Muito utilizado no apoio à tarefa de seleção de atributos (*feature selection*).

A força da regularização é dada pelo atributo `alpha`, com valor `default` igual a 1. `Alpha=0` resulta na regressão linear tradicional.

In [23]:

```
# O parâmetro max_iter foi definido abaixo por conta de um aviso exibido pelo python no momento da execução do comando
modelo = Lasso()

#Aplicação Lasso() no df_original
resultado = aplicar_modelo_ML('df_original', modelo, x_orig_train, y_orig_train, x_orig_test)
df_resultados_modelos = df_resultados_modelos.append(resultado, ignore_index=True)

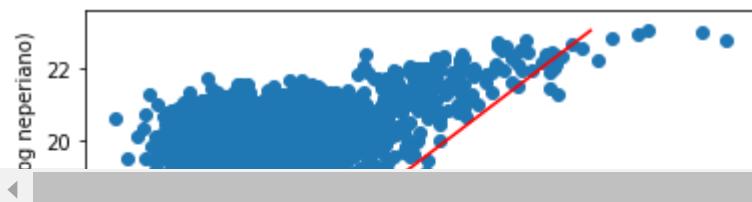
#Aplicação Lasso() no df_MinMax
resultado = aplicar_modelo_ML('df_MinMax', modelo, x_minmax_train, y_minmax_train, x_minmax_test)
df_resultados_modelos = df_resultados_modelos.append(resultado, ignore_index=True)

#Aplicação Lasso() no df_Log
resultado = aplicar_modelo_ML('df_log', modelo, x_log_train, y_log_train, x_log_test, y_log_test)
df_resultados_modelos = df_resultados_modelos.append(resultado, ignore_index=True)

#Aponta o melhor modelo.
print ('O melhor resultado foi com o modelo', df_resultados_modelos['Modelo'][df_resultados_modelos['MAE'] == min(df_resultados_modelos['MAE'])])
print ('Para o Dataframe', df_resultados_modelos['Dataframe'][df_resultados_modelos['MAE'] == min(df_resultados_modelos['MAE'])])
print ('MAE', min(df_resultados_modelos['MAE - Base Teste'])))

df_resultados_modelos
```

Dataframe: df\_log  
 Modelo: Lasso()  
 MAE na base de treinamento: 810437851.6983606  
 RMSE na base de treinamento: 4432093869.473924  
 R<sup>2</sup> na base de treinamento: 0.08  
 MAE na base de teste: 909510144.1128656  
 RMSE na base de teste: 5123285531.611351  
 R<sup>2</sup> na base de teste: 0.10



## 6.4. ElasticNet ()

Será avaliada, aqui, a capacidade de predição do algoritmo `sklearn.linear_model.ElasticNet()`.

In [24]:

```

modelo = ElasticNet()

#Aplicação ElasticNet() no df_original
resultado = aplicar_modelo_ML('df_original', modelo, x_orig_train, y_orig_train, x_orig_test)
df_resultados_modelos = df_resultados_modelos.append(resultado, ignore_index=True)

#Aplicação ElasticNet() no df_MinMax
resultado = aplicar_modelo_ML('df_MinMax', modelo, x_minmax_train, y_minmax_train, x_minmax_test)
df_resultados_modelos = df_resultados_modelos.append(resultado, ignore_index=True)

#Aplicação ElasticNet() no df_Log
resultado = aplicar_modelo_ML('df_log', modelo, x_log_train, y_log_train, x_log_test, y_log_test)
df_resultados_modelos = df_resultados_modelos.append(resultado, ignore_index=True)

#Aponta o melhor modelo.
print ('O melhor resultado foi com o modelo', df_resultados_modelos['Modelo'][df_resultados_modelos['MAE - Base Treinamento'] == min(df_resultados_modelos['MAE - Base Treinamento'])])
print ('Para o dataframe', df_resultados_modelos['Dataframe'][df_resultados_modelos['MAE - Base Teste'] == min(df_resultados_modelos['MAE - Base Teste'])])

df_resultados_modelos

```

Dataframe: df\_original  
Modelo: ElasticNet()

C:\ANACONDA\lib\site-packages\sklearn\linear\_model\\_coordinate\_descent.py:  
529: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1.045335048506352e+21, tolerance: 8.306369131538909e+18  
model = cd\_fast.enet\_coordinate\_descent()  
C:\ANACONDA\lib\site-packages\sklearn\linear\_model\\_coordinate\_descent.py:  
529: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1.045335048506352e+21, tolerance: 8.306369131538909e+18  
model = cd\_fast.enet\_coordinate\_descent()

MAE na base de treinamento: 278135661.7503823  
RMSE na base de treinamento: 728470892.1102945  
R<sup>2</sup> na base de treinamento: 0.98  
MAE na base de teste: 336341664.68788195  
RMSE na base de teste: 1559529789.6382786

## 6.5. Random Forest Regressor

Será avaliada, aqui, a capacidade de predição do algoritmo `sklearn.ensemble.RandomForestRegressor()`.

In [25]:

```

modelo = RandomForestRegressor()

#Aplicação RandomForestRegressor() no df_original
resultado = aplicar_modelo_ML('df_original', modelo, x_orig_train, y_orig_train, x_orig_test)
df_resultados_modelos = df_resultados_modelos.append(resultado, ignore_index=True)

#Aplicação RandomForestRegressor() no df_MinMax
resultado = aplicar_modelo_ML('df_MinMax', modelo, x_minmax_train, y_minmax_train, x_minmax_test)
df_resultados_modelos = df_resultados_modelos.append(resultado, ignore_index=True)

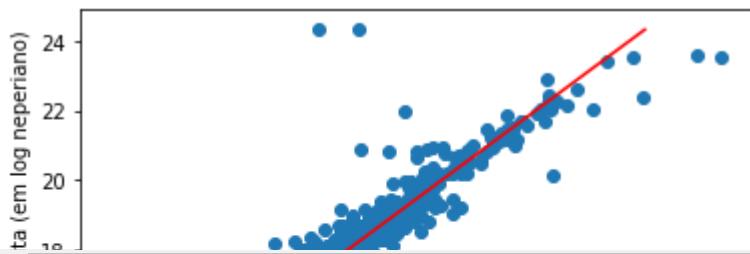
#Aplicação RandomForestRegressor() no df_Log
resultado = aplicar_modelo_ML('df_log', modelo, x_log_train, y_log_train, x_log_test, y_log_test)
df_resultados_modelos = df_resultados_modelos.append(resultado, ignore_index=True)

#Aponta o melhor modelo.
print ('O melhor resultado foi com o modelo', df_resultados_modelos['Modelo'][df_resultados_modelos['MAE - Base Treinamento'] == min(df_resultados_modelos['MAE - Base Treinamento'])])
print ('Para o dataframe', df_resultados_modelos['Dataframe'][df_resultados_modelos['MAE - Base Teste'] == min(df_resultados_modelos['MAE - Base Teste'])])
print('MAE', min(df_resultados_modelos['MAE - Base Teste']))

```

df\_resultados\_modelos

Dataframe: df\_original  
 Modelo: RandomForestRegressor()  
 MAE na base de treinamento: 51512928.039721884  
 RMSE na base de treinamento: 1538062536.6340742  
 R<sup>2</sup> na base de treinamento: 0.89  
 MAE na base de teste: 266248471.73027498  
 RMSE na base de teste: 4910176551.936895  
 R<sup>2</sup> na base de teste: 0.17



## 6.6. - Melhor modelo identificado para este primeiro experimento

In [26]:

```
print ('O melhor resultado foi com o modelo:',df_resultados_modelos['Modelo'][df_resultados_
print ('Para o dataframe:',df_resultados_modelos['Dataframe'][df_resultados_modelos['MAE -
print('MAE:',min(df_resultados_modelos['MAE - Base Teste'])))
print('RMSE:',min(df_resultados_modelos['RMSE - Base Teste'])))
print('R²:',max(df_resultados_modelos['R2 (Acurácia) - Base Teste'])))
df_resultados_modelos
```

O melhor resultado foi com o modelo: [RandomForestRegressor()]  
 Para o dataframe: df\_log  
 MAE: 149559791.88940054  
 RMSE: 1559539289.6383786  
 R<sup>2</sup>: 0.9164734130787983

Out[26]:

	Dataframe	Modelo	MAE - Base Treinamento	MAE - Base Teste	RMSE - Base Treinamento	RMSE - Base Teste
0	df_original	LinearRegression()	2.817382e+08	3.454691e+08	7.176948e+08	1.593124e+
1	df_MinMax	[LinearRegression()]	2.817446e+08	3.435496e+08	7.176948e+08	1.566404e+
2	df_log	[LinearRegression()]	8.104379e+08	9.095102e+08	4.432094e+09	5.123286e+
3	df_original	[Ridge()]	2.816667e+08	3.431515e+08	7.177380e+08	1.566276e+
4	df_MinMax	[Ridge()]	3.436276e+08	4.282319e+08	2.299322e+09	3.152693e+
5	df_log	[Ridge()]	8.080235e+08	9.079830e+08	4.432133e+09	5.123834e+
6	df_original	[Lasso()]	2.817792e+08	3.431821e+08	7.177482e+08	1.566357e+
7	df_MinMax	[Lasso()]	2.817792e+08	3.431820e+08	7.177482e+08	1.566357e+
8	df_log	[Lasso()]	8.104379e+08	9.095101e+08	4.432094e+09	5.123286e+
9	df_original	[ElasticNet()]	2.781357e+08	3.363417e+08	7.284709e+08	1.559539e+
10	df_MinMax	[ElasticNet()]	3.167394e+08	4.377077e+08	4.590456e+09	5.370983e+
11	df_log	[ElasticNet()]	5.628429e+08	6.833164e+08	4.467609e+09	5.245745e+
12	df_original	[RandomForestRegressor()]	5.151293e+07	2.662485e+08	1.538063e+09	4.910177e+
13	df_MinMax	[RandomForestRegressor()]	5.345473e+07	2.571010e+08	1.471183e+09	4.864734e+
14	df_log	[RandomForestRegressor()]	5.330153e+07	1.495598e+08	1.643934e+09	1.950479e+

## 7. Análise dos Resíduos

Passa-se à análise dos resíduos, que são a diferença entre o valor da arrecadação real para 2018 e os valores preditos pelos modelos.

### 7.1. Resíduos gerados pelo melhor modelo (RandomForestRegressor e df\_log)

Para se avaliar o erro, primeiro será reaplicado o melhor algoritmo (RandomForestRegressor) ao dataframe df\_log (integralmente, não mais dividido entre base de treinamento e de teste), de modo a que tenhamos o valor original e o previsto para todas as linhas.

Será criada uma variável ('predicao\_melhor\_modelo') com os valores previstos apontados pelo modelo e uma variável ('residuos') com a diferença entre o valor original e o valor previsto.

In [27]:

```
# Isolar as features e o label
Y=df_log['01_arrecadacao_2018']
X=df_log.drop(['municipio-uf','01_arrecadacao_2018'],axis=1)

#reaplicar o modelo com os melhores hiperparâmetros
modelo = RandomForestRegressor(random_state=12).fit(X,Y)
y_pred = modelo.predict(X)

#cria coluna com as previsões
df_log['predicao_melhor_modelo'] = y_pred

#cria coluna auxiliar
df_log['UF']=df_log['municipio-uf'][-2:]

#df_experimento['faixa_arrecadacao']=df_transformado['faixa_arrecadacao']
df_log['faixa_arrecadacao']=df_integral['faixa_arrecadacao'].astype(str)

#df_experimento['faixa_arrecadacao'] = df_experimento.apply(Lambda row:faixa_arrecadacao(ro

# Calculando o RMSE-total (base logarítmica)
mae_total = mean_absolute_error(Y,y_pred)
print("MAE-total", mae_total)

# Calcular os resíduos
df_log['residuos'] =abs(df_log['01_arrecadacao_2018'] - df_log['predicao_melhor_modelo'])
```

MAE-total 41443789.69337023

Para melhorar a exibição dos erros nos gráficos, preciso fazer um novo índice que acompanhe o valor ordenado ascendente da coluna 01\_arrecadacao\_2018.

In [28]:

```
#Cria a coluna 'indice_ordenado'
df_log.sort_values(by='01_arrecadacao_2018', ascending=True, inplace = True)
df_log['indice_ordenado']=np.arange(0,df_log.shape[0])
df_log.sort_index(ascending=True, inplace = True)
```

Por meio da função abaixo, imprimirei um Gráfico de Dispersão com as arrecadações (01\_arrecadacao\_2018) em ordem ascendente e o histograma dos resíduos.

In [29]:

```
colunas = ['01_arrecadacao_2018', 'residuos', 'predicao_melhor_modelo']
converte_para_ln(df_log, colunas)
df_log[['01_arrecadacao_2018_ln', 'residuos_ln', 'predicao_melhor_modelo_ln']]
#df_log.to_csv('ntbk 04 - predições.csv')
```

Colunas criadas: 3

Out[29]:

	01_arrecadacao_2018_ln	residuos_ln	predicao_melhor_modelo_ln
0	17.143069	14.868635	17.034531
1	16.083384	13.222956	16.024436
2	16.281953	13.971343	16.376537
3	15.736003	13.900184	15.883977
4	19.022893	15.612820	19.055398
...	...	...	...
5560	17.089410	13.470348	17.115865
5561	17.253865	14.275486	17.201650
5562	15.019310	14.774951	15.597723
5563	15.950101	13.156530	15.886946
5564	25.454995	23.192885	25.554053

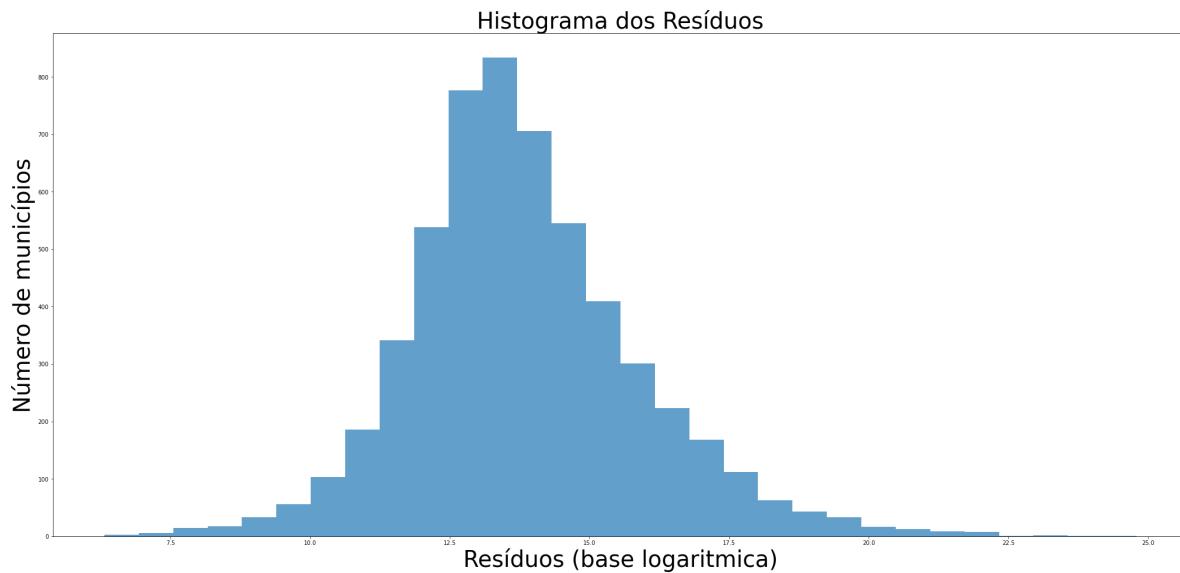
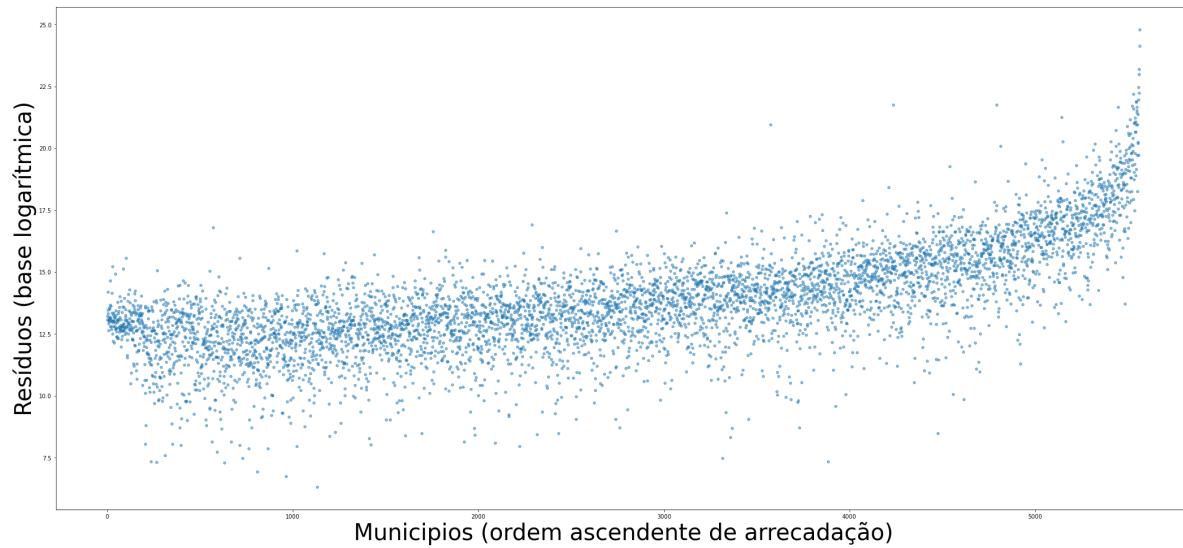
5565 rows × 3 columns

Feita essa conversão, o gráfico abaixo mostra a distribuição dos resíduos considerando uma curva ascendente da arrecadação real (variável 01\_arrecadacao\_2018) e o histograma dos resíduos.

In [30]:

```
plot_residuo(df_log, 'indice_ordenado', 'residuos_ln', 'residuos_ln', 'predicao_melhor_modelo')
```

Out[30]:



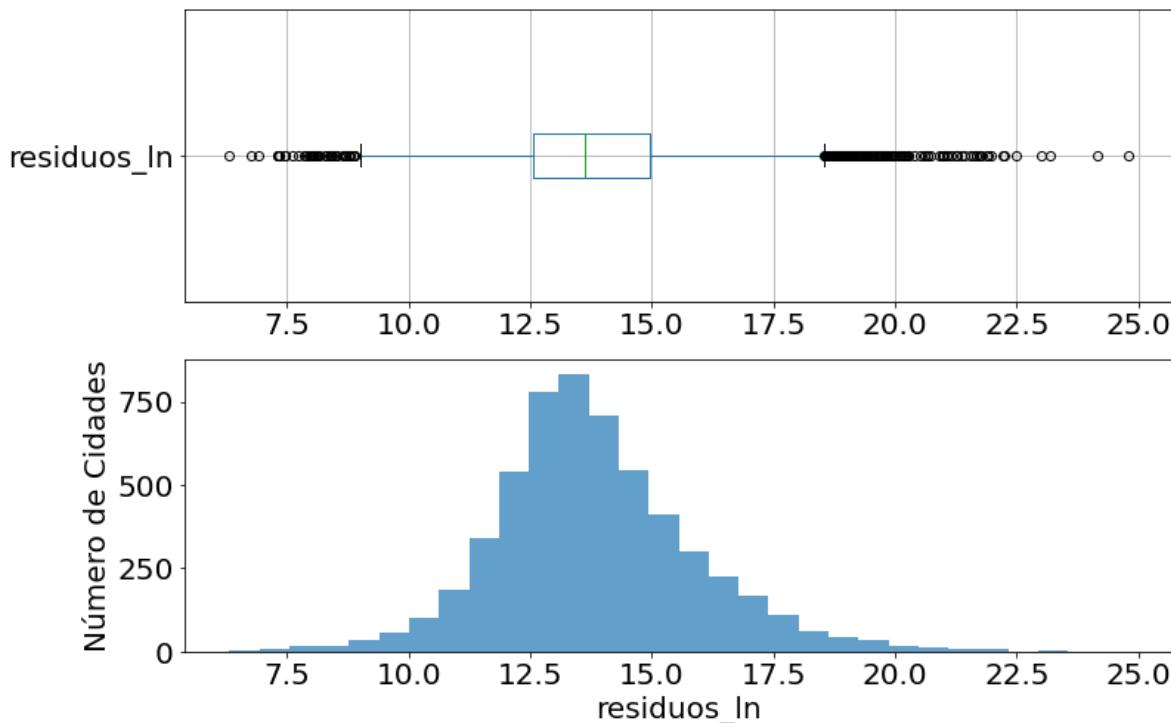
Abaixo, chamarei a função plotstats (declarada anteriormente) que exibirá o bloxpot e o histograma dos resíduos.

In [31]:

```
plotstats(df_log, 'residuos_ln')
print(df_log['residuos'].describe().apply(lambda x: '%.6f' % x))
print("\n",df_log['residuos_ln'].describe().apply(lambda x: '%.6f' % x))
```

```
count      5565.000000
mean      41443789.693370
std       928719272.901685
min       554.452200
25%      283028.850200
50%      826400.210000
75%      3116983.301500
max      58645034880.819336
Name: residuos, dtype: object
```

```
count      5565.000000
mean      13.841317
std       2.075045
min       6.317981
25%      12.553304
50%      13.624834
75%      14.952376
max      24.794769
Name: residuos_ln, dtype: object
```



A análise do boxplot e do histograma acima demonstram que a distribuição dos resíduos se aproxima da curva normal e se concentra praticamente no intervalo entre R\$22 mil e R\$ 270 mi (10 e 17.5, em valores logarítmicos).

Esses resíduos, embora não possam ser considerados desprezíveis, demonstram que o modelo teve boa eficácia, visto que o valor médio para os resíduos é de cerca de R 41 mi (ou 13,83 em base logarítmica) e o valor médio das arrecadações é de cerca de R 249 mi (vide capítulo 4.2), ou seja o erro médio representa cerca de 16,44% do valor da arrecadação média.

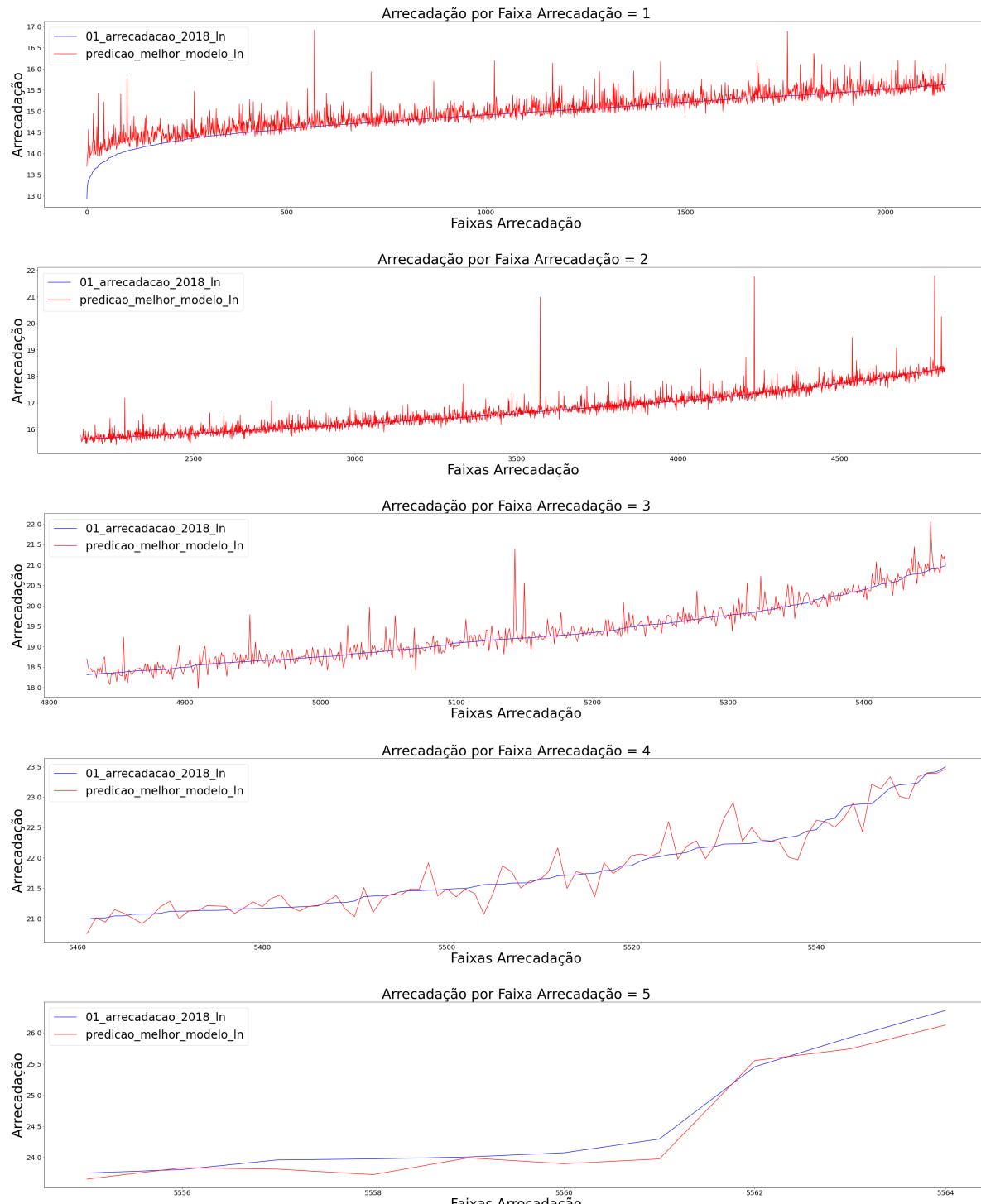
Por fim, os Gráficos de Linhas abaixo permitem comparar a evolução da arrecadação (em base logarítmica) - linha azul - e a capacidade de predição do modelo - linha vermelha – para cada faixa de arrecadação.

In [32]:

```
faixas = ['1', '2', '3', '4', '5']
plot_arrecadacao(df_log, faixas)
```

Out[32]:

'Concluído'



Interessante observar acima que, não obstante haja variações, inclusive alguns picos relevantes, especialmente nas faixas 1 a 3, a linha vermelha acompanha a linha azul em sua tendência para a grande parte dos casos. maior parte dos casos.

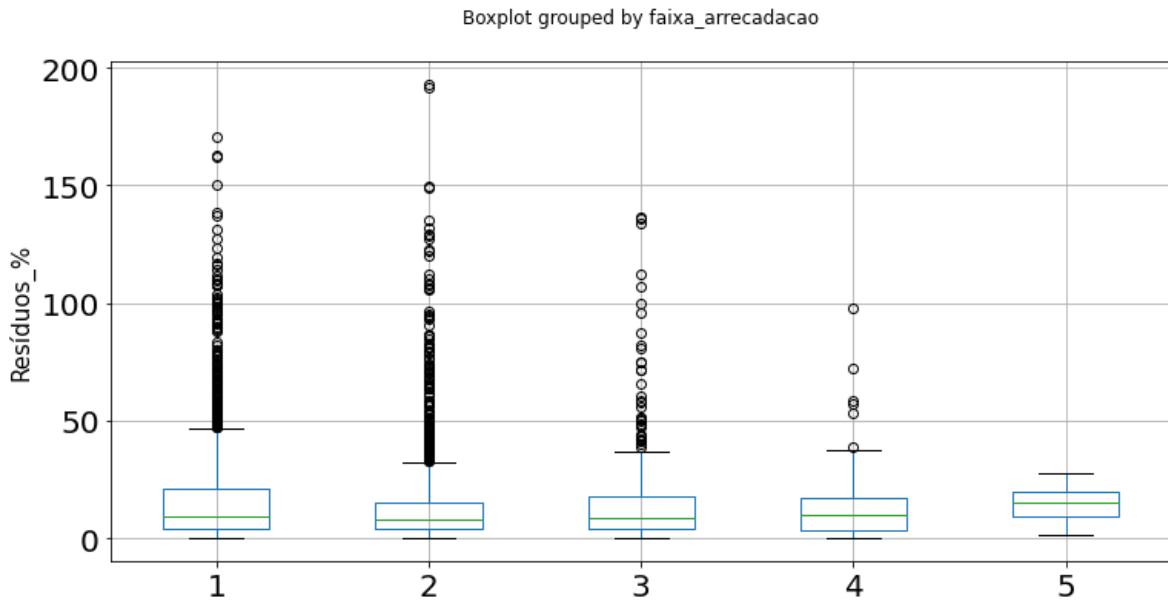
O boxplot abaixo exibe os valores percentuais dos resíduos em relação ao valor da arrecadação real.

In [33]:

```
box_residuos(df_log) #chama função
```

Out[33]:

'Concluído'



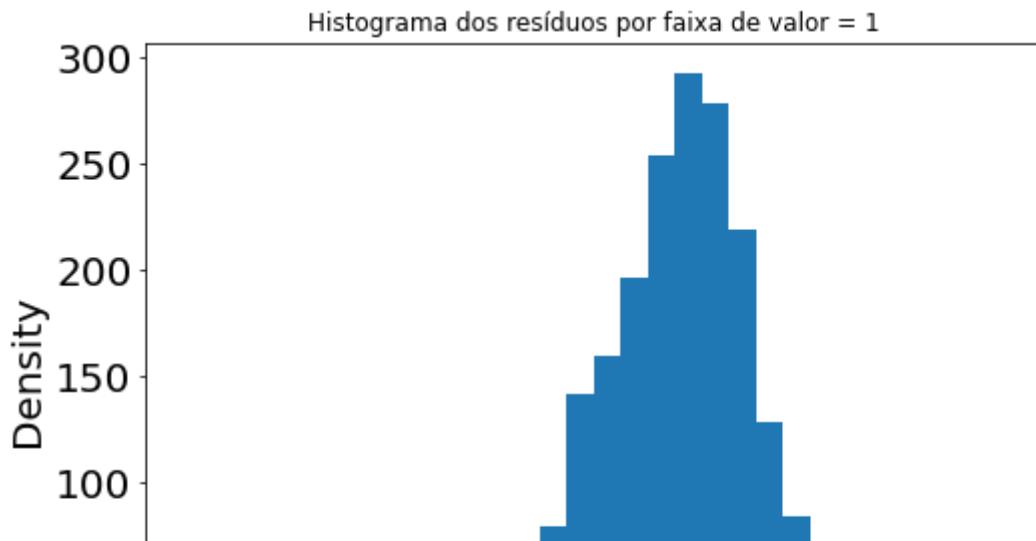
Esses resíduos, embora não possam ser considerados desprezíveis, demonstram que o modelo teve boa eficácia, visto que o valor médio para os resíduos é de cerca de R 41mi (ou 13,83em base logarítmica) e o valor médio das arrecadações é de cerca de R 249 mi (vide capítulo 4.2), ou seja o erro médio representa cerca de 16,44% do valor da arrecadação média.

In [34]:

```
faixas = ['1','2','3','4','5']
resids_hist(df_log, faixas) #chama função
```

Out[34]:

'Concluído'



Aqui podemos observar os histogramas dos resíduos por cada faixa de valor.

## 7.2. Comparando com o segundo melhor modelo

O modelo com o segundo melhor resultado foi com o algoritmo ElasticNet(). Para a comparação dos resíduos, esse algoritmo será reaplicado ao dataframe df\_original (integralmente, não mais dividido entre base de treinamento e de teste), pois essa foi a melhor combinação (vide título ElasticNet do capítulo 5.2).

Vê-se que o MAE do modelo aplicado à base completa foi de R\$ 303.356.514,04.

In [35]:

```
# Isolar as features e o label
Y=df_original['01_arrecadacao_2018']
X=df_original.drop(['municipio-uf','01_arrecadacao_2018'],axis=1)

#reaplicar o modelo com os melhores hiperparâmetros
#modelo = ElasticNet(alpha= 1000.0, fit_intercept= False, l1_ratio= 0.99, max_iter = 10000)
modelo = ElasticNet().fit(X,Y)
y_pred = modelo.predict(X)

#cria coluna com as predições
df_original['predicao_melhor_modelo'] = y_pred

#cria coluna auxiliar
df_original['UF']=df_original['municipio-uf'][-2:]

#df_experimento['faixa_arrecadacao']=df_transformado['faixa_arrecadacao']
df_original['faixa_arrecadacao']=df_integral['faixa_arrecadacao'].astype(str)

#df_experimento['faixa_arrecadacao'] = df_experimento.apply(Lambda row:faixa_arrecadacao(ro

# Calculando o RMSE-total (base Logarítmica)
mae_total = mean_absolute_error(Y,y_pred)
print("MAE-total", mae_total)

# Calcular os resíduos
#df_original['residuos'] =df_original['01_arrecadacao_2018'] - df_original['predicao_melhor'
df_original['residuos'] =abs(df_original['01_arrecadacao_2018'] - df_original['predicao_mel

#Cria a coluna 'indice_ordenado'
df_original.sort_values(by='01_arrecadacao_2018', ascending=True, inplace = True)
df_original['indice_ordenado']=np.arange(0,df_original.shape[0])
df_original.sort_index(ascending=True, inplace = True)

colunas = ['01_arrecadacao_2018','residuos','predicao_melhor_modelo']
converte_para_ln(df_original,colunas)
df_original[['01_arrecadacao_2018_ln','residuos_ln','predicao_melhor_modelo_ln']]
```

MAE-total 303356514.0401894

Colunas criadas: 3

C:\ANACONDA\lib\site-packages\sklearn\linear\_model\\_coordinate\_descent.py:52  
9: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations. Duality gap: 2.833967611579133e+21, tolerance: 1.31709172196656e+19  
model = cd\_fast.enet\_coordinate\_descent(

Out[35]:

	01_arrecadacao_2018_In	residuos_In	predicao_melhor_modelo_In
0	17.143069	14.378676	17.204178
1	16.083384	18.862673	18.922904
2	16.281953	18.677296	18.764520

	<b>01_arrecadacao_2018_In</b>	<b>residuos_In</b>	<b>predicao_melhor_modelo_In</b>
<b>3</b>	15.736003	17.802597	17.921815
<b>4</b>	19.022893	17.027066	19.150319
...	...	...	...
<b>5560</b>	17.089410	19.117044	18.975887
<b>5561</b>	17.253865	19.787609	19.704922
<b>5562</b>	15.019310	19.568827	19.558199
<b>5563</b>	15.950101	19.612186	19.637541
<b>5564</b>	25.454995	23.331431	25.327611

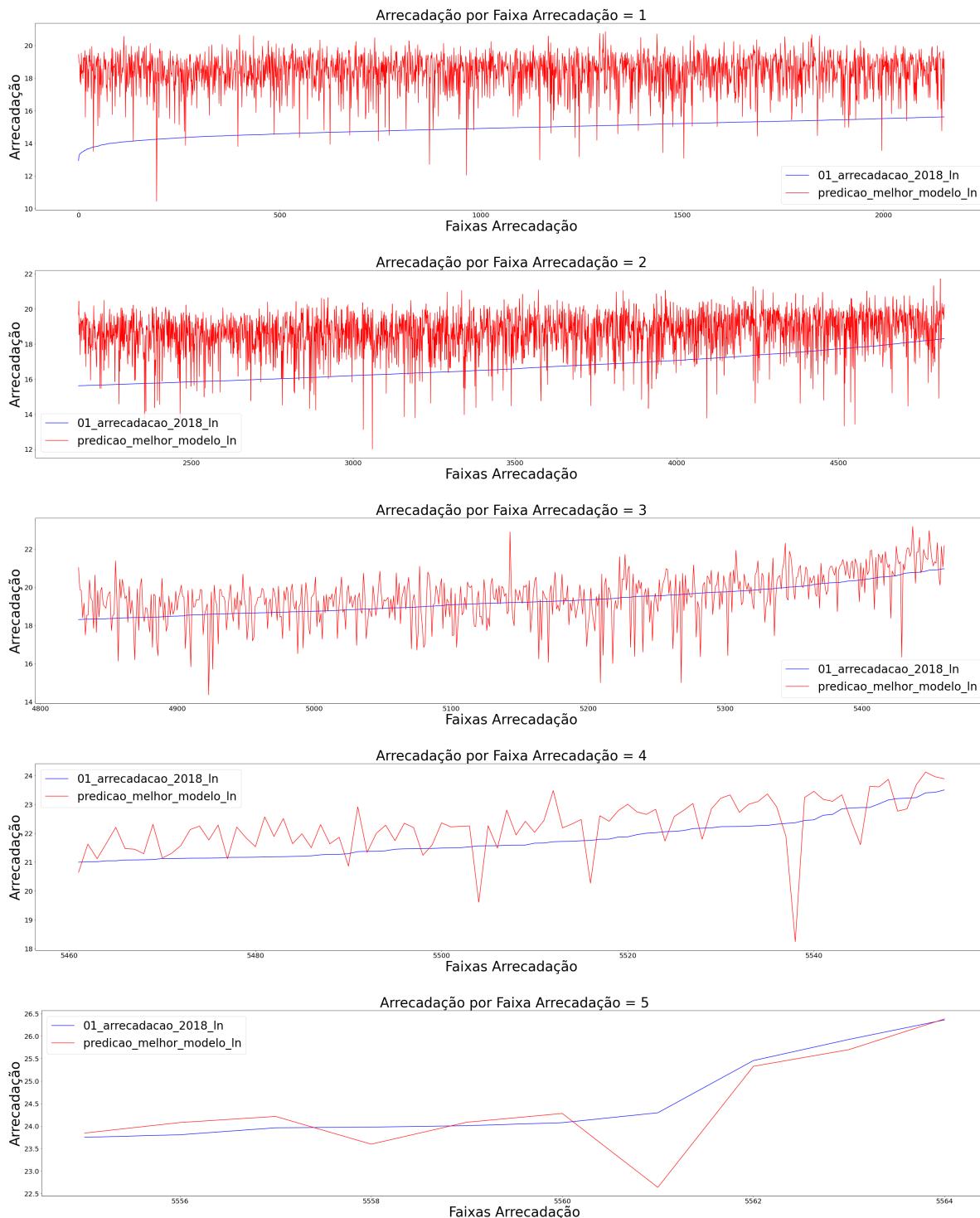
Visualizando os Gráficos de Linhas que comparam a evolução da arrecadação (em base logarítmica) – linha azul – e a capacidade de predição do modelo – linha vermelha, para cada faixa de arrecadação, vemos que diferentemente do resultado obtido para o modelo RandomForestRegressor(), as predições somente se aproximam dos valores reais de arrecadação para as faixas de valores maiores (faixas 3 a 5), apontando que o modelo penaliza mais severamente os maiores resíduos e assim busca reduzi-los ao máximo. Considerando que as faixas de valores menores, porém, possuem o maior número de municípios, isso ajuda a explicar o desempenho inferior deste modelo.

In [36]:

```
faixas = ['1','2','3','4','5']
plot_arrecadacao(df_original, faixas)
```

Out[36]:

'Concluído'



Mas, o que ocorreria se as maiores arrecadações fossem excluídas deste modelo?

### 7.3. Excluindo os outliers

Viu-se no Ntbk 03 que a distribuição das arrecadações (variável 01\_arrecadacao\_2018) indica a existência de 158 outliers: municípios com arrecadação superior a R\$ 757.532.350.87.

Foi mencionado que esses 158 municípios foram mantidos no modelo porque, embora suas arrecadações de fato destoem dos demais, não são incorretas e ajudam a caracterizar o universo sob análise, além de também ser de nosso interesse predizer a arrecadação para eles.

Também foi mencionado que esses valores, no entanto, influenciam os resultados do modelo, que precisa moldar-se ao conjunto de valores em análise. Será aqui avaliada, então, essa influência.

Ao reaplicar o algoritmo ElasticNet (que foi o segundo melhor modelo e para o qual a manutenção dos 158 municípios tiveram importante impacto) a uma nova base, sem as maiores arrecadações, tem-se que o MAE passa a ser de 20.695.422,23, uma redução extraordinária quando comparado com a base em que incluídos os outliers (que foi de 303.356.514,04 – vide título imediatamente anterior).

In [37]:

```
df_original = df_integral.copy()
df_original = df_original.drop('faixa_arrecadacao', axis=1)
print (df_original.shape)

df_original = df_original[df_original['01_arrecadacao_2018'] < 757532350.8799902]

print (df_original.shape)
df_original.describe()

(5565, 44)
(5407, 44)
```

Out[37]:

	01_arrecadacao_2018	02_unidades_atuantes_%	02_ocupados_assalariados_%	02_salario_
count	5.407000e+03	5407.000000	5407.000000	
mean	4.002709e+07	97.504228	85.249309	
std	9.129486e+07	2.959289	6.820964	
min	4.165360e+05	56.103286	21.124207	
25%	3.663282e+06	96.666667	81.440719	
50%	8.700600e+06	98.058252	85.784793	
75%	2.753385e+07	99.270073	89.876840	
max	7.575324e+08	100.000000	99.489796	

In [38]:

```
# Isolar as features e o Label
Y=df_original['01_arrecadacao_2018']
X=df_original.drop(['municipio-uf','01_arrecadacao_2018'],axis=1)

#reaplicar o modelo com os melhores hiperparâmetros
modelo = ElasticNet().fit(X,Y)
y_pred = modelo.predict(X)

#cria coluna com as previsões
df_original['predicao_melhor_modelo'] = y_pred

#cria coluna auxiliar
df_original['UF']=df_original['municipio-uf'][-2:]

#df_experimento['faixa_arrecadacao']=df_transformado['faixa_arrecadacao']
df_original['faixa_arrecadacao']=df_integral['faixa_arrecadacao'].astype(str)

#df_experimento['faixa_arrecadacao'] = df_experimento.apply(lambda row:faixa_arrecadacao(ro

# Calculando o RMSE-total (base Logarítmica)
mae_total = mean_absolute_error(Y,y_pred)
print("MAE-total", mae_total)

# Calcular os resíduos
df_original['residuos'] =abs(df_original['01_arrecadacao_2018'] - df_original['predicao_melhor_modelo'])
```

MAE-total 20695422.2336082

```
C:\ANACONDA\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:52
9: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5.760209544331866e+18, tolerance: 4
505766749686870.0
```

model = cd\_fast.enet\_coordinate\_descent(

In [39]:

```
#Cria a coluna 'indice_ordenado'
df_original.sort_values(by='01_arrecadacao_2018', ascending=True, inplace = True)
df_original['indice_ordenado']=np.arange(0,df_original.shape[0])
df_original.sort_index(ascending=True, inplace = True)

#Converte colunas para a base Logarítmica
colunas = ['01_arrecadacao_2018','residuos','predicao_melhor_modelo']
converte_para_ln(df_original,colunas)
#df_original[['01_arrecadacao_2018_ln','residuos_ln','predicao_melhor_modelo_ln']]
#df_original.to_csv('ntbk 04 - Previsões.csv')
```

Colunas criadas: 3

Out[39]:

'Concluído'

É visualizando e comparando, porém, os gráficos de linha dos resultados da base sem os outliers com o da base completa que uma diferença relevante se apresenta. Os valores preditos (linhas vermelhas) agora sobreponem-se aos da arrecadação real (linha azul) já nas primeiras faixas de arrecadação, quando no gráfico

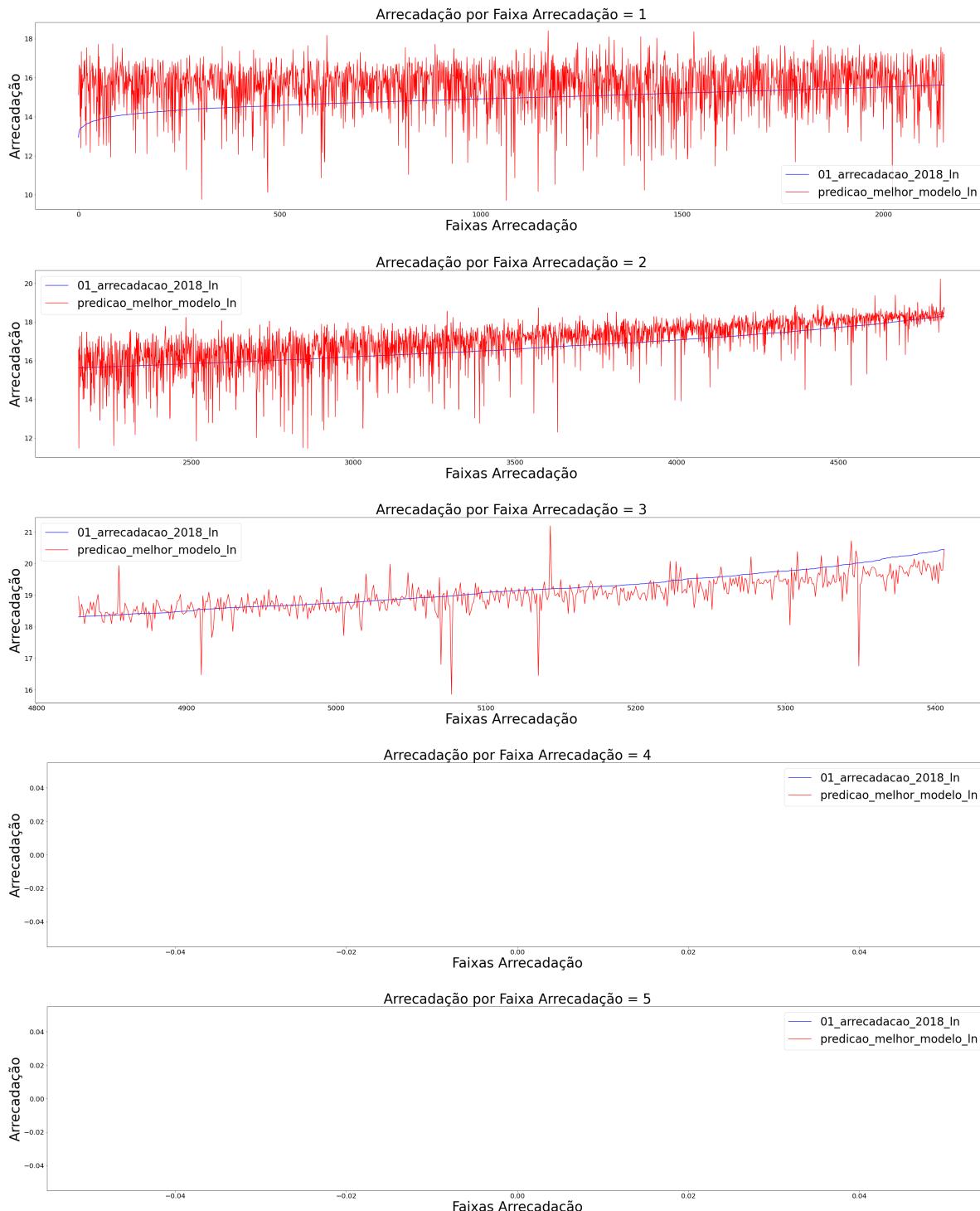
da base completa (visto acima) essa sobreposição somente ocorria para as faixas de arrecadação mais elevadas. Observação: não há elementos nos últimos dois gráficos visto que não há mais elementos nas faixas 4 e 5 após a exclusão dos outliers.

In [40]:

```
faixas = ['1', '2', '3', '4', '5']
plot_arrecadacao(df_original, faixas) #chama função
```

Out[40]:

'Concluído'



Ou seja, com a exclusão das 158 maiores arrecadações, há uma sensível melhora na capacidade de predição das arrecadações de menor valor, pelo algoritmo ElasticNet.

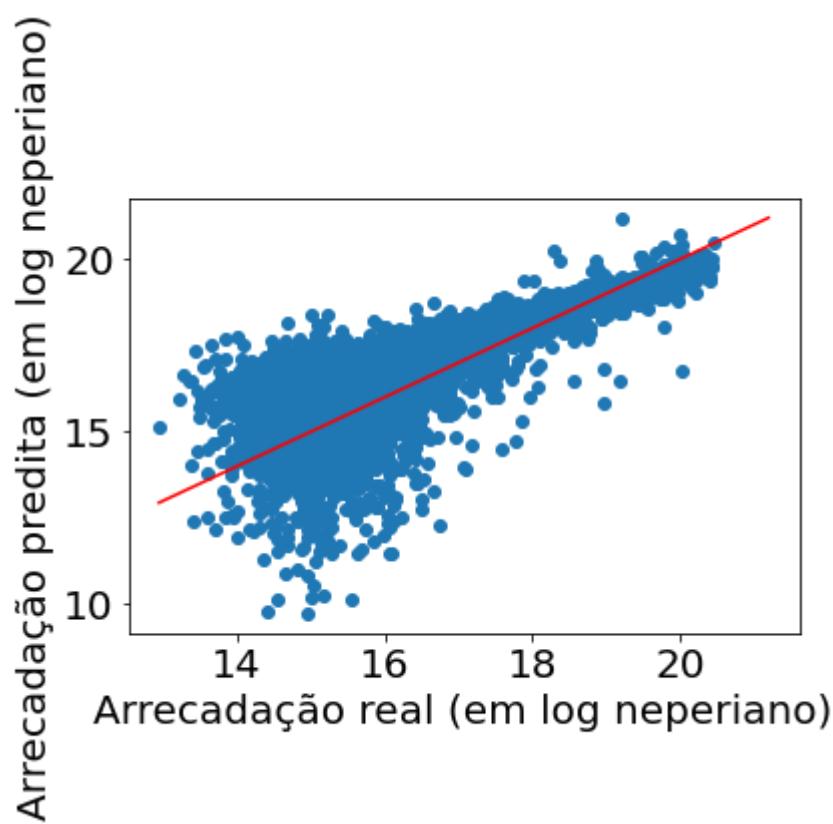
Diante desse resultado, uma estratégia que considere a divisão do dataset entre municípios com altíssima arrecadação e os demais, merece ser considerada para uma predição mais precisa para esses dois estratos.

In [41]:

```
Y_ln = np.log(np.where(Y==0, 0.00001, Y))
y_pred_ln = np.log(np.where(y_pred==0, 0.00001, y_pred))
imprime_scatter_plot(Y_ln,y_pred_ln)
```

```
<ipython-input-5-4c5f0a291225>:8: UserWarning: Matplotlib is currently using
agg, which is a non-GUI backend, so cannot show the figure.
plt.show()
```

Out[41]:



## 8. Ajustando os hiperparâmetros para o melhor modelo (tunning)

Avaliada a melhor combinação entre os dataframes de estudo e os algoritmos, passa-se, agora, a buscar o aprimoramento do resultado por meio da otimização dos hiperparâmetros e do uso da cross validation.

Este estudo seguirá, então, tendo por base a combinação de melhor resultado considerando o dataset completo (sem a exclusão dos municípios com maiores arrecadações): algoritmo RandomForestRegressor() e dataframe df\_log.

No aprendizado de máquina, a otimização ou ajuste de hiperparâmetro é a ação de escolher um conjunto de configurações ideais para um algoritmo de aprendizado. Cada algoritmo possui um conjunto de parâmetros que podem ser definidos de modo personalizado com vistas a otimizar sua performance ante à base de dados em estudo.

In [42]:

```

df_log = df_integral.copy()
df_log=df_log.drop('faixa_arrecadacao',axis=1)

#Seleciona as colunas numéricas...
colunas_a_converter = df_log.dtypes[df_log.dtypes != "object"].index.tolist()
#... exceto a primeira: '01_arrecadacao_2018'.
colunas_a_converter.pop(0)

# Chama a função que fará a conversão
converte_para_ln(df_log, colunas_a_converter)

# Apaga as colunas originais
df_log = df_log.drop(colunas_a_converter, axis=1)
print(df_log.shape)
df_log.describe()

```

Colunas criadas: 42  
(5565, 44)

Out[42]:

	01_arrecadacao_2018	02_unidades_atuantes_%_ln	02_ocupados_assalariados_%_ln	02_sa
count	5.565000e+03	5565.000000	5565.000000	
mean	2.494351e+08	4.578749	4.442744	
std	4.865354e+09	0.032886	0.086424	
min	4.165360e+05	4.027194	3.050420	
25%	3.775316e+06	4.570079	4.400923	
50%	9.175717e+06	4.584794	4.452742	
75%	3.150111e+07	4.597202	4.498478	
max	2.804407e+11	4.605170	4.600055	

No próximo experimento, será buscada a combinação ideal considerando os seis hiperparâmetros abaixo.

Para tanto, será utilizada a classe Random Search Cross Validation , da biblioteca Scikit Learn:  
sklearn.model\_selection.RandomizedSearchCV().

Esse algoritmo promove avaliações a partir de combinações aleatórias de valores para os hiperparâmetros indicados e retorna a melhor delas.

Para otimizar a identificação dos melhores hiperparâmetros, em vez de se fazer uma única separação, fixa, do dataset em apenas duas bases (treinamento e teste), o RandomizedSearchCV utiliza a técnica cross validation.

In [43]:

```
# Busca pelos melhores hiperparâmetros usando RandomizedSearchCV

# define o modelo
modelo = RandomForestRegressor(random_state=12)

#Número de árvores da Random Forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 100)]

# Número de variáveis a considerar a cada split
max_features = ['auto', 'sqrt']

# Número máximo de níveis na árvore
max_depth = [int(x) for x in np.linspace(10, 110, num = 20)]
max_depth.append(None)

# Número mínimo de exemplos para fazer o split do nó
min_samples_split = [2, 5, 10]

# Número mínimo de exemplos para cada nó-folha
min_samples_leaf = [1, 2, 4]

# Métrica de seleção de exemplos para treinar cada árvore
bootstrap = [True, False]

# Cria o random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}

print(random_grid)

# Procura randômica por parâmetros, usando cross validation com 3 folds.
# Busca em 20 diferentes combinações, e use todas as variáveis independentes
modelo_randomizado = RandomizedSearchCV(estimator = modelo, error_score=mean_absolute_error)

# Roda o modelo de busca randômica.

Y=df_log['01_arrecadacao_2018']
X=df_log.drop(['municipio-uf','01_arrecadacao_2018'],axis=1)
modelo_randomizado.fit(X, Y)

#x_train = x_Log_train
#y_train = y_Log_train
#modelo_randomizado.fit(x_train, y_train)

print('MAE: %.3f' % modelo_randomizado.best_score_)
print('Config: %s' % modelo_randomizado.best_params_)
```



```
{'n_estimators': [200, 218, 236, 254, 272, 290, 309, 327, 345, 363, 381, 400, 418, 436, 454, 472, 490, 509, 527, 545, 563, 581, 600, 618, 636, 654, 672, 690, 709, 727, 745, 763, 781, 800, 818, 836, 854, 872, 890, 909, 927, 945, 963, 981, 1000, 1018, 1036, 1054, 1072, 1090, 1109, 1127, 1145, 1163, 1181, 1200, 1218, 1236, 1254, 1272, 1290, 1309, 1327, 1345, 1363, 1381, 1400, 1418, 1436, 1454, 1472, 1490, 1509, 1527, 1545, 1563, 1581, 1600, 1618, 1636,
```

```
1654, 1672, 1690, 1709, 1727, 1745, 1763, 1781, 1800, 1818, 1836, 1854, 187  
2, 1890, 1909, 1927, 1945, 1963, 1981, 2000], 'max_features': ['auto', 'sqr  
t'], 'max_depth': [10, 15, 20, 25, 31, 36, 41, 46, 52, 57, 62, 67, 73, 78, 8  
3, 88, 94, 99, 104, 110, None], 'min_samples_split': [2, 5, 10], 'min_sample  
s_leaf': [1, 2, 4], 'bootstrap': [True, False]}\nFitting 3 folds for each of 20 candidates, totalling 60 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.  
[Parallel(n_jobs=-1)]: Done 25 tasks      | elapsed: 9.5min  
[Parallel(n_jobs=-1)]: Done 60 out of 60 | elapsed: 17.2min finished
```

MAE: 0.750

Config: {'n\_estimators': 563, 'min\_samples\_split': 2, 'min\_samples\_leaf': 1,  
'max\_features': 'auto', 'max\_depth': 88, 'bootstrap': False}

Executada a célula acima (cerca de 17 minutos em um computador pessoal), a melhor combinação de valores identificada para os hiperparâmetros indicados foi:

Config:{'n\_estimators': 563, 'min\_samples\_split': 2, 'min\_samples\_leaf': 1, 'max\_features': 'auto', 'max\_depth':  
88, 'bootstrap': False}

Será, então, reaplicado o algoritmo RandomForestRegressor (com os hiperparâmetros apontados) sobre o dataframe df\_log integral (sem a separação entre base de treinamento e de teste) de modo a comparar os resultados com o do experimento descrito no título Resíduos gerados pelo melhor modelo, deste mesmo capítulo.

In [44]:

```

modelo = RandomForestRegressor(n_estimators= 563, min_samples_split= 2, min_samples_leaf= 1
                               max_features= 'auto', max_depth= 88, bootstrap = False, random_state= 42)

# Isolar as features e o Label
Y=df_log['01_arrecadacao_2018']
X=df_log.drop(['municipio-uf','01_arrecadacao_2018'],axis=1)

#reaplicar o modelo com os melhores hiperparâmetros
modelo = RandomForestRegressor(n_estimators= 654, min_samples_split= 5, min_samples_leaf= 1
                               max_features= 'auto', max_depth= 88, bootstrap = False).fit(X,Y)

y_pred = modelo.predict(X)

#cria coluna com as previsões
df_log['predicao_melhor_modelo'] = y_pred

#cria coluna auxiliar
df_log['UF']=df_log['municipio-uf'][-2:]

#o['faixa_arrecadacao']
df_log['faixa_arrecadacao']=df_integral['faixa_arrecadacao'].astype(str)

#df_experimento['faixa_arrecadacao'] = df_experimento.apply(lambda row:faixa_arrecadacao(row['UF'],row['faixa_arrecadacao']))

# Calculando o RMSE-total (base Logarítmica)
mae_total = mean_absolute_error(Y,y_pred)
print("MAE-total", mae_total)

# Calcular os resíduos
df_log['residuos'] =abs(df_log['01_arrecadacao_2018'] - df_log['predicao_melhor_modelo'])
df_log['residuos_%'] =abs(df_log['residuos']/df_log['01_arrecadacao_2018'])*100
df_log.to_csv('ntbk 04 - predicoes e residuos.csv')

```

MAE-total 41072249.73690276

Promovendo o ajuste dos hiperparâmetros houve uma discreta melhora em relação ao modelo em que o algoritmo não teve os hiperparâmetros definidos:

- MAE do modelo sem definição de hiperparâmetros: R\$ 41.443.789,69;
- MAE do modelo com ajuste dos hiperparâmetros: R\$ 41.072.249,73

In [ ]: