PROJETO EM CIÊNCIA DE DADOS

SEMESTRE: 2025/2

PROJETO: `O custo das eleições`

COMPONENTES DO GRUPO:
Augusto Peroni Baldino
Bruno Grigoletti Laitano
Daniel Chin Tay Lee
João Pedro Zarth Dias

Breve descrição do problema: Este trabalho extensionista consiste em realizar processos de coleta, preparação e análise de dados a partir dos registros abertos da Câmara dos Deputados, com informações a respeito do trabalho de deputados federais. De forma específica, o texto analisa o impacto das últimas duas eleições gerais (2018 e 2022) sobre as atividades parlamentares, em especial sobre o valor líquido gasto por cada partido e representante, em comparação a estas mesmas atividades, quando realizadas em anos não eleitorais.

Breve descrição da solução proposta: Realizamos o *download* dos arquivos existentes no portal em formato `.json` referentes aos gastos parlamentares em um período de cinco anos — de 2018 a 2022. Esse período de tempo compreende as duas últimas eleições gerais e os anos de atuação da legislatura eleita no primeiro pleito. Dada a série temporal, analisamos a variância dos gastos realizados nos dois anos eleitorais (2018 e 2022) e nos anos não eleitorais (2019, 2020 e 2021), buscando compreender as razões da divergência de valores, em especial o impacto dos pleitos sobre as atividades de deputados federais. Como ano eleitoral, considera-se apenas as eleições gerais.

Fases da Metodologia CRISP-DM:

Fase	**Tarefas**	**%**
Compreensão do Negócio	Entendimento dos objetivos e requisitos sob a perspectiva do negócio	100%
Compreensão dos Dados	Avaliação das fontes de dados disponíveis e determinação de coletas adicionais	100%
Preparação dos Dados	Limpeza dos dados e preparo do *dataset* a ser utilizado na fase de modelagem	100%

Resumo do que foi concluído até o momento: Na primeira fase do projeto, que compreende a Compreensão do Negócio, debatemos o escopo do projeto, baseando-nos nas informações disponíveis no portal. A definição da proposta se deu com o intuito de

trabalhar com uma ampla gama de dados, além de trazer à pesquisa questões teóricas sobre o tema. Além disso, coletamos os arquivos relevantes disponíveis no portal que nos permitirão observar os gastos dos parlamentares no período selecionado. Na sequência, implementamos um algoritmo em Python para ler esses arquivos e organizar os dados mais relevantes em planilhas. Também criamos uma classe de acesso a API da Câmara dos Deputados para organizar localmente todas as informações desejadas sobre os deputados.

Na fase de Compreensão dos Dados, implementamos estratégias de coleta de dados, construindo localmente planilhas baseadas em arquivos `.json` e em chamadas à API do portal de dados abertos da Câmara dos Deputados. Na sequência, utilizamos operações nativas da biblioteca *Pandas* para descrever os dados coletados, além de construir uma análise exploratória inicial do tema central da nossa investigação. Nessa etapa do trabalho, também abordamos a qualidade dos dados, lançando mão de operações de limpeza e identificação de *outliers*.

Na fase de Preparação dos Dados, importamos as nossas planilhas para o *PowerBI* e efetivamente realizamos os processos de limpeza mapeados na fase anterior. Na sequência, produzimos *dashboards* para auxiliar na visualização dos principais dados selecionados, com o intuito de demonstrar graficamente os resultados da nossa investigação. Na contramão do que se acredita em meio ao senso comum, concluímos que os nossos representantes na Câmara dos Deputados não gastam mais durante períodos eleitorais.

Sumário

Av. Ipiranga, 6681 — Prédio 30 — Bloco C — Sala 101
CEP 90619-900 — Porto Alegre, RS — Brasil

Fone: (51) 3320-3558 — E-mail: politecnica@pucrs.br — www.pucrs.br/politecnica

1 Compreensão do Negócio

1.1 Background

De forma geral, a pesquisa explora temas da política brasileira, em especial a atividade financeira dos nossos parlamentares em anos eleitorais e não eleitorais. Com isso em mente, coletando informações a respeito dos gastos de deputados federais entre os anos de 2018 e 2022, espera-se adquirir uma visão ampla sobre o quanto variam esses gastos. Nesse sentido, a mineração servirá para organizar localmente os dados disponibilizados pelo portal da Câmara dos Deputados — tanto a respeito das despesas quanto a respeito dos parlamentares.

1.2 Objetivos de negócio e critérios de sucesso (CS)

1. Analisar o impacto das eleições sobre os gastos parlamentares.

   CS: Espera-se identificar se há variações significativas nos valores gastos por deputados federais nos últimos dois anos eleitorais em comparação a anos não eleitorais.

2. Comparar o comportamento financeiro entre partidos políticos.

   CS: Espera-se compreender se determinados partidos apresentam padrões distintos de despesa em períodos eleitorais e não eleitorais.

3. Avaliar mudanças no perfil das despesas parlamentares.

   CS: Espera-se verificar se há alteração na distribuição por categorias de despesa (ex.: transporte, consultoria, manutenção de escritório) em função da ocorrência das eleições.

1.3 Inventário de recursos

1. API Rest do portal de dados abertos da Câmara dos Deputados;
2. Arquivos `.json` disponibilizados no portal de dados abertos da Câmara dos Deputados;
3. Planilhas geradas a partir dos nossos algoritmos de captura e leitura de dados;
4. Aplicação para visualizar dados e gerar *dashboards*, como o *PowerBI*.

1.4 Requisitos, suposições e restrições

Em relação à variância nas despesas realizadas por deputados federais, supomos que exista um aumento de gastos em anos eleitoriais, especificamente durante as eleições gerais, em que a população decide, entre outros cargos, a legislatura que comporá a Câmara dos Deputados nos quatro anos seguintes ao pleito.

Inicialmente, esperava-se observar o fluxo de valores durante todo o período da Nova República, iniciada em 1985, com o término da ditadura civil-militar. No entanto, embora a API Rest do portal de dados abertos da Câmara dos Deputados disponibilize dados a

respeito de todas as legislaturas da história da Casa, os arquivos `.json` compreendem apenas os gastos realizados a partir do ano de 2008.

Além disso, há um empecilho técnico importante. Com base em alguns testes, revelou-se impraticável a coleta de dados referentes a todos os gastos disponíveis no portal. São muitas informações, e infelizmente não dispomos do poder computacional necessário para executar o nosso plano original. Nesse sentido, o período de tempo analisado compreenderá apenas as duas últimas eleições gerais, ocorridas em 2018 e em 2022, e os anos subsequentes da legislatura eleita no primeiro pleito — 2019, 2020, 2021 e 2022.

1.5 Terminologia

A seguir, apresentamos os principais termos de domínio mencionados neste projeto:

- **Deputado federal**: Representante eleito para a Câmara dos Deputados, com mandato de quatro anos. Deputados federais são responsáveis por legislar e manter-se fiéis às leis e dogmas constitucionais nacionais. Podem propor, emendar, alterar e revogar leis, leis complementares ou emendas à Constituição Federal [**?**];
- **Partido político**: Entidade formada pela livre associação de pessoas, com uma ideologia em comum, cujas finalidades são assegurar, no interesse do regime democrático, a autenticidade do sistema representativo e defender os direitos humanos fundamentais [**?**];
- **Legislatura**: Período de quatro anos correspondente ao mandato dos deputados federais;
- **Eleições gerais**: Processo eleitoral no qual são escolhidos deputados federais, estaduais, senadores, governadores e o presidente da República;
- **Gastos parlamentares**: Recursos públicos disponibilizados aos deputados para custear despesas relacionadas ao exercício do mandato. A Cota para o Exercício da Atividade Parlamentar (CEAP) custeia as despesas do mandato, como passagens aéreas e conta de celular. Algumas são reembolsadas, como as com os Correios, e outras são pagas por débito automático, como a compra de passagens [**?**].

1.6 Objetivos de mineração e critérios de sucesso (CS)

1. Os dados coletados devem nos permitir observar despesas parlamentares a cada ano.

   CS: Ao gerarmos os *dashboards* para visualizar os dados, espera-se observar o fluxo das despesas de deputados federais durante anos específicos.

2. Os dados coletados devem nos permitir observar o conjunto de despesas de cada um dos partidos em anos específicos.

   CS: Ao visualizar os dados, espera-se observar diferenças nos gastos de cada partido que compunha a Câmara dos Deputados nos anos selecionados.

3. Os dados coletados devem nos permitir observar o tipo de despesa realizada nos anos selecionados.

   CS: Ao visualizar os dados, espera-se observar o tipo de gasto realizado.

1.7 Plano de Projeto

- **Etapa 1 — Definição do escopo da pesquisa**: nesta fase inicial, debatemos possibilidades de investigação com base nos dados disponibilizados pelo portal de dados abertos da Câmara dos Deputados, visando a definição do tema central do trabalho;

- **Etapa 2 — Coleta de dados**: na sequência, iniciaremos o desenvolvimento dos algoritmos de captura dos dados a serem utilizados na pesquisa. Nesse sentido, utilizaremos a API da Câmara para coletar os principais dados a respeito dos deputados federais que tenham composto a legislatura iniciada a partir das eleições gerais de 2018. Além disso, realizamos o *download* de todos os arquivos `.json` relevantes disponíveis no portal, os quais contêm informações detalhadas sobre gastos parlamentares;

- **Etapa 3 — Análise dos dados**: os algoritmos desenvolvidos durante a segunda etapa da investigação devem produzir planilhas organizadas, que servirão de base para a análise dos dados e para a produção de *dashboards* para visualização dos dados.

1.8 Avaliação inicial de técnicas e ferramentas

Neste projeto, utilizaremos um conjunto de ferramentas e técnicas de ciência de dados para apoiar as diferentes fases do CRISP-DM:

- **Linguagem de programação**: Python, por sua ampla gama de bibliotecas voltadas para coleta, análise e visualização de dados;

- **Bibliotecas e aplicações de manipulação e análise de dados**: `pandas` (para organização e limpeza de dados), `openpyxl` (para geração de planilhas) e `PowerBI` (para visualização dos dados e geração de *dashboards*);

- **Técnicas estatísticas**: análise comparativa entre anos eleitorais e não eleitorais, além da utilização de séries temporais para observar tendências ao longo do período estudado;

- **Metodologia**: aplicação do CRISP-DM como guia para as fases do projeto, desde a compreensão do negócio até a análise e interpretação dos resultados.

1.9 Status do escopo

Acreditamos que será possível cumprir 100% do escopo pretendido. Entretanto, obstáculos serão encontrados ao longo da elaboração desta pesquisa, uma vez que trabalharemos com um conjunto bastante expressivo de dados. Além disso, devido ao tempo disponível para desenvolver o trabalho, é possível que não possamos fundamentá-lo de forma ideal em termos de referenciais bibliográficos.

2 Compreensão dos Dados

2.1 Coleta dos dados

Tendo em vista o principal objetivo da nossa pesquisa — entender se há mudanças relevantes no comportamento fiscal de deputados federais em anos de eleições gerais — definimos dois escopos de coleta distintos.

O primeiro deles utiliza os documentos `.json` disponíveis no portal de dados abertos da Câmara dos Deputados a fim de coletar informações sobre gastos parlamentares realizados entre os anos de 2018 e 2022. O período compreende todos os anos da legislatura eleita no penúltimo pleito, incluindo dois anos eleitorais.

No código abaixo, percorremos todos os arquivos `.json` transferidos do portal para um diretório local. Nomeamos os arquivos a partir do ano a que correspondem, facilitando não só a leitura, mas também a interpretação dos dados brutos. Capturamos as informações que julgamos mais relevantes para a nossa proposta, resgitrando-as em um dicionário local de nome `gastos`.

```
gastos = []

def ler_arquivo(ano):
    with open(f'arquivos/{ano}.json', encoding='utf-8') as f:
        data = json.load(f)
        processar_dados(data)

def processar_dados(data):
    for i in data['dados']:
        id = i.get('idDeputado')
        if id is None:
            continue

        nome = i['nomeParlamentar'].title()

        descricao = i.get('descricao', 'N/A').title()

        fornecedor = i.get('fornecedor', 'N/A').title()

        data_raw = i.get('dataEmissao', 'N/A')
        if data_raw != 'N/A':
            try:
                data_hora = datetime.strptime(data_raw, %Y-%m-%
                    dT%H:%M:%S ).strftime( %d/%m/%Y %H:%M )
            except ValueError:
                data_hora = data_raw
        else:
            data_hora = 'N/A'

        valor_raw = i.get('valorLiquido', 0)
        try:
```

```
            valor = float(valor_raw)
        except (TypeError, ValueError):
            continue

        gastos.append({
             id : id,
             nome : nome,
             descricao : descricao,
             fornecedor : fornecedor,
             data : data_hora,
             valor : valor
        })

def get_gastos():
    for ano in range(2018, 2023):
        ler_arquivo(ano)

    return gastos

def gerar_planilha():
    gastos = get_gastos()

    wb = Workbook()
    ws = wb.active
    ws.title = Gastos
    ws.append([ ID , Nome , Descrição , Fornecedor , Data ,
       Valor (R$) ])

    for gasto in gastos:
        ws.append([
            gasto[ id ],
            gasto[ nome ],
            gasto[ descricao ],
            gasto[ fornecedor ],
            gasto[ data ],
            gasto[ valor ]
        ])

    wb.save( planilhas/gastos.xlsx )
```

A segunda coleta foi realizada com base na API RESTful do portal. São diversos os *endpoints* disponíveis, mas utilizamos um em especial — `/deputados`. Tendo em vista que a API retorna um conjunto muito mais abrangente em termos de tempo em comparação aos arquivos utilizados na primeira coleta, definimos alguns parâmetros na URL para capturar apenas os dados desejados.

No código abaixo, percorremos todas as páginas disponíveis a partir de uma dada URL, registrando os dados encontrados e coletados em um dicionário local de nome `deputados`:

```
# URL da API para obter os deputados da 56a legislatura
```

```
url =  https://dadosabertos.camara.leg.br/api/v2/deputados?idLegislatura=56&ordem=ASC&ordenarPor=nome

def obter_deputados(url):
    deputados = {}
    while url:
        response = requests.get(url)
        if response.status_code != 200:
            print( Erro ao acessar a API: , response.status_code)
            break

        data = response.json()
        for deputado in data['dados']:
            deputados[deputado['id']] = {
                'id': deputado['id'],
                # Normalizar nome do deputado
                'nome': deputado['nome'].title(),
                'partido': deputado['siglaPartido'],
                'estado': deputado['siglaUf'],
                'legislatura': deputado['idLegislatura']
            }

        url = proxima_pagina(data)
    return deputados

def proxima_pagina(data):
    # Verifica se há um link para a próxima página
    for link in data.get('links', []):
        if link.get('rel') == 'next':
            return link.get('href')
    return None

def gerar_planilha():
    deputados = obter_deputados(url)

    wb = Workbook()
    ws = wb.active
    ws.title =  Deputados
    ws.append([ ID ,  Nome ,  Partido ,  Estado ,  Legislatura ])

    for id, info in deputados.items():
        ws.append([
            info[ id ],
            info[ nome ],
            info[ partido ],
            info[ estado ],
            info[ legislatura ]
        ])

```

```
50      wb.save( planilhas/deputados.xlsx )
```

Coletados os dados a respeito dos deputados e as informações relacionadas aos gastos de deputados federais nos anos mencionados, guardamos cada conjunto de dados em planilhas — `deputados.xlsx` e `gastos.xlsx`. Em ambos os casos, esse processo foi gerenciado por uma biblioteca do Python destinada a ler e a escrever arquivos Excel, chamada `openpyxl`. Na próxima seção deste trabalho, serão especificadas em detalhes as informações presentes em cada um dos conjuntos de dados criados a partir das nossas coletas.

2.2 Descrição dos dados

As colunas de cada uma das planilhas indicam exatamente os dados e os tipos de dados que decidimos preservar sobre cada deputado e cada gasto. Essa decisão baseou-se em uma análise prévia à coleta das informações disponíveis tanto na API quanto nos arquivos `.json` disponíveis no portal.

A fim de visualizar mais detalhadamente a descrição dos dados incluídos em cada planilha, utilizamos a biblioteca *Pandas*. Transformamos cada planilha em um *DataFrame* e recorremos às operações disponíveis nessa biblioteca para entender de forma objetiva a natureza das informações com as quais estamos lidando. Abaixo, os códigos criados para visualizar as descrições, seguidos pelos resultados impressos no terminal:

```
1   import pandas as pd
2   from tabulate import tabulate
3
4   deputados = pd.read_excel( planilhas/deputados.xlsx )
5   gastos = pd.read_excel( planilhas/gastos.xlsx )
6
7   deputados_info = pd.DataFrame({
8       Coluna : deputados.columns,
9       Tipo de dado : [str(deputados[col].dtype) for col in
          deputados.columns],
10      Exemplo : [deputados[col].dropna().iloc[0] if deputados[
          col].notna().any() else    for col in deputados.columns
          ]
11  })
12
13  gastos_info = pd.DataFrame({
14      Coluna : gastos.columns,
15      Tipo de dado : [str(gastos[col].dtype) for col in gastos
          .columns],
16      Exemplo : [gastos[col].dropna().iloc[0] if gastos[col].
          notna().any() else    for col in gastos.columns]
17  })
18
19  print( Informações sobre o DataFrame 'deputados': )
20  print(tabulate(deputados_info, headers='keys', tablefmt='grid
      '))
```

```
21
22      print( \nInformações sobre o DataFrame 'gastos': )
23      print(tabulate(gastos_info, headers='keys', tablefmt='grid'))
```

No caso da planilha `deputados.xlsx`, os seguintes dados são listados:

#	Coluna	Tipo de dado	Exemplo
0	ID	int64	74784
1	Nome	object	Luiza Erundina
2	Partido	object	SP
3	Estado	object	PSOL
4	Legislatura	int64	56

Tabela 2: Informações sobre o *DataFrame* `deputados`

No caso da planilha `gastos.xlsx`, listamos as seguintes informações:

#	Coluna	Tipo de dado	Exemplo
0	ID	int64	62881
1	Nome	object	Danilo Forte
2	Descrição	object	Manutenção de Escritório de Apoio à Atividade Parlamentar
3	Fornecedor	object	Companhia Energética do Ceará
4	Data	object	16/03/2022 00:00
5	Valor (R$)	float64	1439.49

Tabela 3: Informações sobre o *DataFrame* `gastos`

O tipo `int64` representa números inteiros de 64 bits, usado para identificadores ou valores numéricos sem casas decimais. O tipo `float64` armazena números reais (com casas decimais), sendo comum em colunas de valores monetários. Já o tipo `object` é utilizado para dados de texto ou mistos, como nomes, partidos e descrições.

2.3 Análise exploratória dos dados

Do ponto de vista da exploração inicial dos dados, executamos uma série de operações nativas da biblioteca *Pandas* sobre as nossas planilhas, visando listar os principais parâmetros estatísticos existentes em meio aos nossos *datasets*. Também utilizamos funções para nos indicar as dimensões de cada arquivo e informações gerais, como a quantidade de valores e os tipos de dados, já analisados no capítulo anterior.

```
1       def geral(deputados, gastos):
2           # Primeiras e últimas linhas
3           print( Primeiras linhas do DataFrame 'deputados': )
4           print(deputados.head())
5           print( \nÚltimas linhas do DataFrame 'deputados': )
```

```
6           print(deputados.tail())
7
8           print( \nPrimeiras linhas do DataFrame 'gastos': )
9           print(gastos.head())
10          print( \nÚltimas linhas do DataFrame 'gastos': )
11          print(gastos.tail())
12
13          # Dimensões
14          print( \nDimensões do DataFrame 'deputados': , deputados.
               shape)
15          print( Dimensões do DataFrame 'gastos': , gastos.shape)
16
17          # Nomes
18          print( Nomes das colunas do DataFrame 'deputados': ,
               deputados.columns.tolist())
19          print( Nomes das colunas do DataFrame 'gastos': , gastos.
               columns.tolist())
20
21          # Informções gerais
22          print( \nInformações gerais do DataFrame 'deputados': )
23          deputados.info()
24
25          print( \nInformações gerais do DataFrame 'gastos': )
26          gastos.info()
```

Ao executar as operações do *Pandas* aninhadas na nossa função `geral` obtivemos os seguintes resultados:

#	ID	Nome	Partido	Estado	Legislatura
0	204554	Abílio Santana	PSC	BA	56
1	204521	Abou Anni	UNIÃO	SP	56
2	204379	Acácio Favacho	MDB	AP	56
3	204560	Adolfo Viana	PSDB	BA	56
4	204528	Adriana Ventura	NOVO	SP	56

Tabela 4: Primeiras linhas do *DataFrame* `deputados`

#	ID	Nome	Partido	Estado	Legislatura
608	204559	Zé Neto	PT	BA	56
609	160632	Zé Silva	SOLIDARIEDADE	MG	56
610	204517	Zé Vitor	PL	MG	56
611	178923	Zeca Cavalcanti	PTB	PE	56
612	160592	Zeca Dirceu	PT	PR	56

Tabela 5: Últimas linhas do *DataFrame* `deputados`

#	ID	Nome	Descrição	Fornecedor	Data	Valor (R$)
0	4930	Silvio Costa	Manutenção De Escritório De Apoio À Atividade Parlamentar	Antonio Carlos Brito Maciel	06/01/2018 00:00	2160.00
1	4930	Silvio Costa	Manutenção De Escritório De Apoio À Atividade Parlamentar	Antonio Carlos Brito Maciel	07/03/2018 00:00	2160.00
2	4930	Silvio Costa	Manutenção De Escritório De Apoio À Atividade Parlamentar	Antonio Carlos Brito Maciel	01/04/2018 00:00	2160.00
3	4930	Silvio Costa	Manutenção De Escritório De Apoio À Atividade Parlamentar	Antonio Carlos Brito Maciel	12/06/2018 00:00	2160.00
4	4930	Silvio Costa	Manutenção De Escritório De Apoio À Atividade Parlamentar	Antonio Carlos Brito Maciel	10/07/2018 00:00	2160.00

Tabela 6: Primeiras linhas do *DataFrame* `gastos`

#	ID	Nome	Descrição	Fornecedor	Data	Valor (R$)
807846	220008	Eliza Virgínia	Passagem Aérea - Sigepa	Tam	04/10/2022 12:00	1820.45
807847	220008	Eliza Virgínia	Passagem Aérea - Sigepa	Tam	16/10/2022 12:00	1827.79
807848	220008	Eliza Virgínia	Passagem Aérea - Sigepa	Tam	19/10/2022 12:00	1376.56
807849	220008	Eliza Virgínia	Passagem Aérea - Sigepa	Tam	04/11/2022 12:00	1820.45
807850	220008	Eliza Virgínia	Passagem Aérea - Sigepa	Tam	09/11/2022 12:00	1818.56

Tabela 7: Últimas linhas do *DataFrame* `gastos`

```
Dimensões do DataFrame 'deputados': (613, 5)

Dimensões do DataFrame 'gastos': (807851, 6)

Nomes das colunas do DataFrame 'deputados': ['ID', 'Nome', 'Partido', 'Estado',
↪  'Legislatura']

Nomes das colunas do DataFrame 'gastos': ['ID', 'Nome', 'Descrição', 'Fornecedor',
↪  'Data', 'Valor (R$)']

Informações gerais do DataFrame 'deputados':
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 613 entries, 0 to 612
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   ID           613 non-null    int64
 1   Nome         613 non-null    object
 2   Partido      613 non-null    object
 3   Estado       613 non-null    object
 4   Legislatura  613 non-null    int64
dtypes: int64(2), object(3)
memory usage: 24.1+ KB

Informações gerais do DataFrame 'gastos':
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 807851 entries, 0 to 807850
Data columns (total 6 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   ID          807851 non-null  int64
 1   Nome        807851 non-null  object
 2   Descrição   807851 non-null  object
 3   Fornecedor  807818 non-null  object
 4   Data        801736 non-null  object
```

```
 5   Valor (R$)  807851 non-null  float64
dtypes: float64(1), int64(1), object(4)
memory usage: 37.0+ MB
```

Com base nos resultados acima, podemos ver que ambos os nossos *datasets* são bastante robustos em termos de quantidade de dados, principalmente o *DataFrame* `gastos`. Este possui mais de 800 mil registros, com alguns valores nulos em comparação ao total de informações disponíveis, principalmente no que se refere ao atributo `Data`.

Já o *DataFrame* `deputados` é composto de 613 registros. No entanto, sabe-se que, em 2018, foram eleitos 513 deputados para a 56ª legislatura (2019-2023) da Câmara dos Deputados. Nesse sentido, vale dizer que a API retorna todos os deputados que exerceram mandato em qualquer período durante a legislatura em questão. Cada item retornado representa uma pessoa eleita, uma pessoa que assumiu como suplente, ou uma pessoa que teve exercício parcial (por afastamento, licença, substituição etc.).

Do ponto de vista dos dados estatísticos, utilizamos as seguintes funções do *Pandas*, aninhadas em nosso método `estatisticas`:

```
1  def estatisticas(deputados, gastos):
2      # Estatísticas descritivas
3      print( \nEstatísticas descritivas do DataFrame 'deputados': )
4      print(deputados.describe())
5
6      print( \nEstatísticas descritivas do DataFrame 'gastos': )
7      print(gastos.describe())
8
9      # Estatísticas gerais
10     print( \nEstatísticas gerais do DataFrame 'deputados': )
11     print(deputados.describe(include='all'))
12
13     print( \nEstatísticas gerais do DataFrame 'gastos': )
14     print(gastos.describe(include='all'))
15
16     # Contagem de valores únicos por coluna
17     print( \nContagem de valores únicos por coluna no DataFrame '
           deputados': )
18     print(deputados.nunique())
19
20     print( \nContagem de valores únicos por coluna no DataFrame '
           gastos': )
21     print(gastos.nunique())
22
23     # Contagem de valores por coluna
24     print( \nContagem de valores por coluna no DataFrame '
           deputados': )
25     for col in deputados.columns:
26         print(f \nColuna: {col} )
27         print(deputados[col].value_counts().head())
28
29     print( \nContagem de valores por coluna no DataFrame 'gastos
```

```
    ': )
for col in gastos.columns:
    print(f '\nColuna: {col} )
    print(gastos[col].value_counts().head())
```

Medida	ID	Legislatura
count	613.000000	613.000000
mean	168367.101142	56.000000
std	46439.986835	0.000000
min	62881.000000	56.000000
25%	141552.000000	56.000000
50%	178963.000000	56.000000
75%	204473.000000	56.000000
max	221183.000000	56.000000

Tabela 8: Estatísticas descritivas do *DataFrame* `deputados`

Medida	ID	Valor (R$)
count	807851.000000	807851.000000
mean	162162.867758	1032.277138
std	46216.308322	2975.694118
min	4930.000000	-5500.290000
25%	141464.000000	73.000000
50%	178881.000000	199.870000
75%	204415.000000	599.000000
max	220008.000000	168000.000000

Tabela 9: Estatísticas descritivas do *DataFrame* `gastos`

Medida	ID	Nome	Partido	Estado	Legislatura
count	613.000000	613	613	613	613.000000
unique	—	613	23	27	—
top	—	Abílio Santana	PL	SP	—
freq	—	1	85	81	—
mean	168367.101142	—	—	—	56.000000
std	46439.986835	—	—	—	0.000000
min	62881.000000	—	—	—	56.000000
25%	141552.000000	—	—	—	56.000000
50%	178963.000000	—	—	—	56.000000
75%	204473.000000	—	—	—	56.000000
max	221183.000000	—	—	—	56.000000

Tabela 10: Estatísticas gerais do *DataFrame* `deputados`

Medida	ID	Nome	Descrição	Fornecedor	Data	Valor (R$)
count	807851.000000	807851	807851	807818	801736	807851.000000
unique	—	879	20	63368	7280	—
top	—	Diego Garcia	Combustíveis E Lubrificantes.	Tam	01/03/2018 00:00	—
freq	—	5493	273193	25961	1186	—
mean	162162.867758	—	—	—	—	1032.277138
std	46216.308322	—	—	—	—	2975.694118
min	4930.000000	—	—	—	—	-5500.290000
25%	141464.000000	—	—	—	—	73.000000
50%	178881.000000	—	—	—	—	199.870000
75%	204415.000000	—	—	—	—	599.000000
max	220008.000000	—	—	—	—	168000.000000

Tabela 11: Estatísticas gerais do *DataFrame* `gastos`

```
Contagem de valores únicos por coluna no DataFrame 'deputados':
ID             613
Nome           613
Partido         23
Estado          27
Legislatura      1
dtype: int64

Contagem de valores únicos por coluna no DataFrame 'gastos':
ID               880
Nome             879
Descrição         20
Fornecedor     63368
Data            7280
Valor (R$)    106581
dtype: int64

Contagem de valores por coluna no DataFrame 'deputados':

Coluna: ID
ID
204554    1
204521    1
204379    1
204560    1
204528    1
Name: count, dtype: int64

Coluna: Nome
Nome
Abílio Santana     1
Abou Anni          1
Acácio Favacho     1
Adolfo Viana       1
Adriana Ventura    1
Name: count, dtype: int64

Coluna: Partido
Partido
PL       85
PP       66
PT       63
```

```
UNIÃO    63
PSD      58
Name: count, dtype: int64

Coluna: Estado
Estado
SP    81
RJ    58
MG    57
BA    44
RS    37
Name: count, dtype: int64

Coluna: Legislatura
Legislatura
56    613
Name: count, dtype: int64

Contagem de valores por coluna no DataFrame 'gastos':

Coluna: ID
ID
178929    5493
178857    5360
160599    4833
177282    4600
74398     4560
Name: count, dtype: int64

Coluna: Nome
Nome
Diego Garcia          5493
Jorge Solla           5360
Dimas Fabiano         4833
Subtenente Gonzaga    4600
Maria Do Rosário      4560
Name: count, dtype: int64

Coluna: Descrição
Descrição
Combustíveis E Lubrificantes.                                  273193
Serviço De Táxi, Pedágio E Estacionamento                      108119
Manutenção De Escritório De Apoio À Atividade Parlamentar      104924
Telefonia                                                       56661
Divulgação Da Atividade Parlamentar.                            56480
Name: count, dtype: int64

Coluna: Fornecedor
Fornecedor
Tam                                                 25961
Uber Do Brasil Tecnologia Ltda.                     23153
Cascol Combustiveis Para Veiculos Ltda              16164
Gol                                                 13047
Servico Nacional De Aprendizagem Comercial Senac     8084
Name: count, dtype: int64
```

```
Coluna: Data
Data
01/03/2018 00:00    1186
01/10/2019 00:00    1114
02/04/2018 00:00    1103
01/06/2022 00:00    1091
02/05/2018 00:00    1078
Name: count, dtype: int64

Coluna: Valor (R$)
Valor (R$)
100.0     31117
200.0     14964
150.0     13723
50.0      12428
1000.0     5004
Name: count, dtype: int64
```

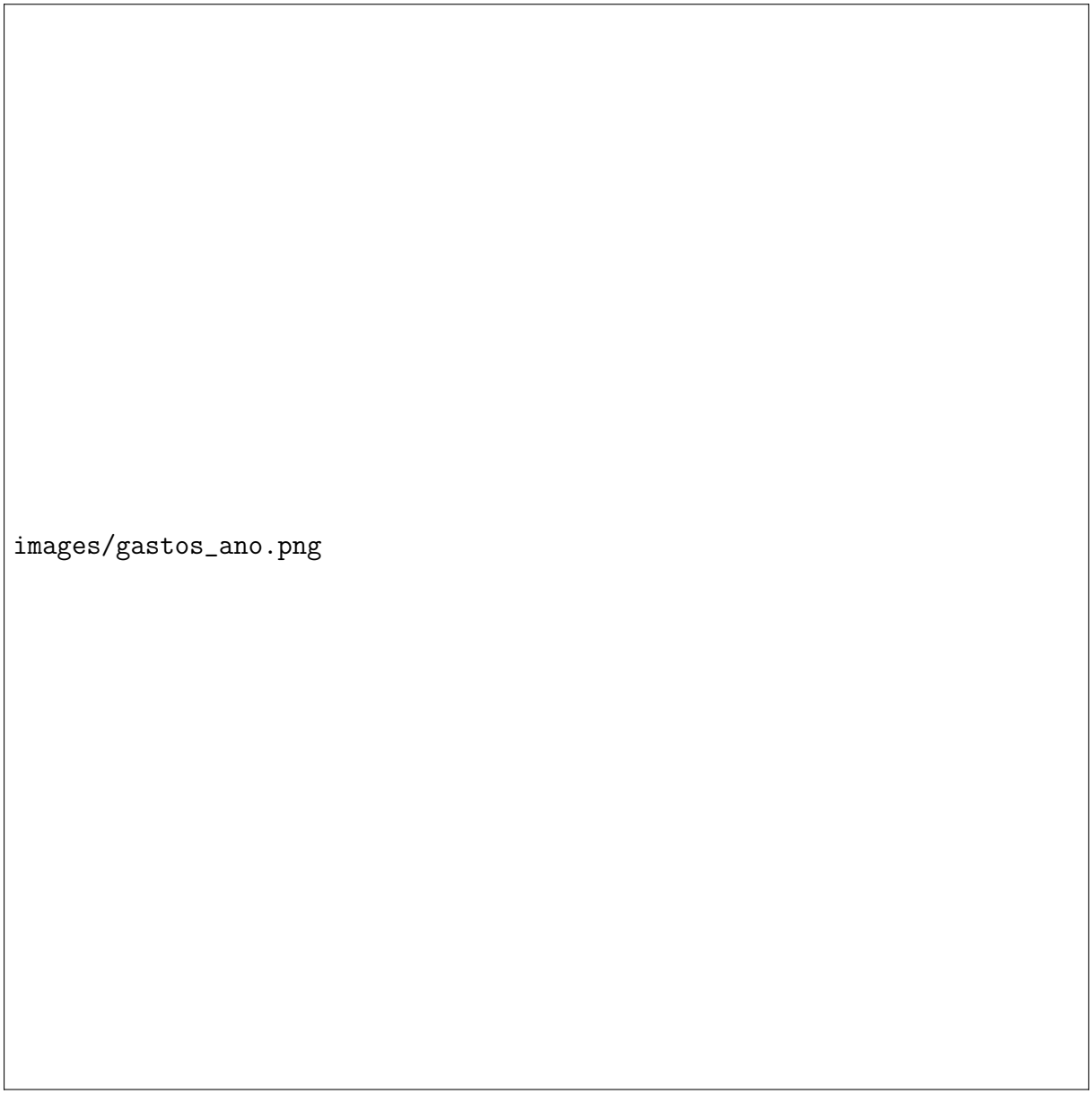
É claro que há alguns desses resultados que nos dizem pouco, como é o caso das estatísticas descritivas de um *DataFrame* como o `deputados`. Não há qualquer sentido, por exemplo, em calcular uma média de valores listados na coluna `ID`. No entanto, optamos por incluir em meio aos resultados todas as operações do *Pandas* utilizadas, para ambos os *datasets*. Assim, podemos demonstrar não só a robustez da biblioteca, mas também o quanto seu uso foi amplamente empregado ao longo deste trabalho.

Finalmente, no que tange a conclusões em relação aos objetivos de mineração, construímos algumas visualizações que devem nos auxiliar futuramente em nossa análise principal — houve uma mudança de comportamento nos gastos durante a última eleição geral?

A fim de mapear as possibilidades de resolução do nosso problema de pesquisa, utilizamos o *DataFrame* `gastos` para gerar um *scatter plot* baseado na distribuição dos gastos parlamentares por ano. O método desenvolvido para gerar o gráfico, bem como o gráfico propriamente dito, podem ser vistos abaixo:

```
1  def scatterplot_gastos(df_gastos):
2      # Converter a coluna 'Data' para datetime
3      df_gastos['Data'] = pd.to_datetime(df_gastos['Data'], format=
           %d/%m/%Y %H:%M , errors='coerce')
4
5      # Calcular a correlação entre 'Valor (R$)' e 'Data'
6      correlacao_gastos_ano = df_gastos['Valor (R$)'].corr(
           df_gastos['Data'].astype('int64'))
7      print(f Correlação entre 'Valor' e 'Data': {
           correlacao_gastos_ano} )
8
9      # Scatter plot dos gastos ao longo do tempo
10     sns.scatterplot(data=df_gastos, x= Data , y= Valor (R$) )
11     plt.gca().xaxis.set_major_locator(mdates.YearLocator())
12     plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'
           ))
13     plt.gcf().autofmt_xdate()
14     plt.title( Distribuição dos gastos por ano )
```

```
15      plt.show()
```



O resultado preliminar já sugere uma série de questões a serem abordadas tanto no próximo subcapítulo, *Verificação de qualidade dos dados*, quanto na próxima parte deste trabalho. Podemos ver que há *outliers* evidentes no nosso conjunto de dados, bem como uma quantidade de gastos bastante relevante para cada ano analisado, dificultando um tanto a visualização do gráfico.

Além disso, parece haver alguns registros em meio ao *DataFrame* `gastos` com datas inválidas — isto é, anteriores ao ano de 2018 e posteriores ao ano de 2022, pontos de início e de fim do nosso recorte temporal. No próximo subcapítulo, ao tratarmos da limpeza de dados, esses registros serão removidos do nosso *DataFrame*, e uma nova visualização será gerada e apresentada.

Apesar dessas dificuldades, podemos ver que de fato há uma concentração de gastos em anos específicos. Esse fenômeno será analisado em maior nível de detalhes no próximo capítulo deste trabalho. Por ora, passamos à verificação da qualidade dos dados, em que abordaremos a presença de valores que extrapolam os valores limítrofes dos nossos dados, bem como a limpeza de dados através da remoção de valores nulos, inválidos ou numéricos negativos.

2.4 Verificação de qualidade dos dados

Uma vez que o nosso problema central de pesquisa gira em torno de valores numéricos, é preciso garantir que não existam valores nulos ou negativos em meio ao *dataset*. Além disso, como indicado anteriormente, é preciso garantir que as datas dos gastos listados no *dataset* estejam dentro do período esperado, de 2018 a 2022. A remoção de valores nulos, inválidos e negativos foi operada com o suporte do seguinte método:

```
def limpeza_dados(df):
    print( \nDados antes da limpeza: )
    print(df.info())

    # Identifica colunas com pelo menos um valor ausente
    colunas_com_na = df.columns[df.isnull().any()].tolist()

    # Remove linhas com valores ausentes
    gastos_limpos = df.dropna(subset=colunas_com_na)

    # Remove linhas com valores negativos em todas as colunas num
       éricas
    cols_numericas = gastos_limpos.select_dtypes(include='number'
       ).columns
    for col in cols_numericas:
        gastos_limpos = gastos_limpos[gastos_limpos[col] >= 0]

    # Remove linhas com ano fora do intervalo 2018-2022
    if 'Data' in gastos_limpos.columns:
        datas = pd.to_datetime(gastos_limpos['Data'], format= %d
           /%m/%Y %H:%M , errors='coerce')
        anos_validos = (datas.dt.year >= 2018) & (datas.dt.year
           <= 2022)
        gastos_limpos = gastos_limpos[anos_validos]

    print( \nDados após limpeza: )
    print(gastos_limpos.info())

    # Quantidade de linhas removidas
    linhas_removidas = len(df) - len(gastos_limpos)
    print(f \nQuantidade de linhas removidas: {linhas_removidas}
       )

    return gastos_limpos
```

Após a remoção dos valores nulos, inválidos e negativos, resolvemos novamente aplicar sobre o *DataFrame* `gastos` a função `scatterplot_gastos`, apresentada na seção anterior. O gráfico de dispersão resultante pode ser visto abaixo:

Quanto às informações impressas localmente no terminal, você pode vê-las abaixo:

```
Dados antes da limpeza:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 807851 entries, 0 to 807850
Data columns (total 6 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   ID          807851 non-null  int64
 1   Nome        807851 non-null  object
 2   Descrição   807851 non-null  object
```

```
 3   Fornecedor  807818 non-null  object
 4   Data        801736 non-null  object
 5   Valor (R$)  807851 non-null  float64
dtypes: float64(1), int64(1), object(4)
memory usage: 37.0+ MB
None

Dados após limpeza:
<class 'pandas.core.frame.DataFrame'>
Index: 797387 entries, 0 to 807850
Data columns (total 6 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   ID          797387 non-null  int64
 1   Nome        797387 non-null  object
 2   Descrição   797387 non-null  object
 3   Fornecedor  797387 non-null  object
 4   Data        797387 non-null  object
 5   Valor (R$)  797387 non-null  float64
dtypes: float64(1), int64(1), object(4)
memory usage: 42.6+ MB
None

Quantidade de linhas removidas: 10464
```

O resultado demonstra o sucesso da operação de limpeza dos dados. Já não há valores nulos ou negativos, que não fariam qualquer sentido em nossa investigação. Além disso, contamos apenas com gastos corretamente cadastrados com datas compreendidas no intervalo esperado.

A título de curiosidade, nota-se pela variação existente no eixo das ordenadas, bem como nos pontos dispersos na parte superior do plano cartesiano, que há *outliers* em meio ao nosso conjunto de dados. Como estamos observando o total gasto em cada ano na série estudada, não há sentido em removê-los da nossa análise. Seus gastos, ainda que exorbitantes, são relevantes para a nossa investigação. Porém, vale a pena observá-los do ponto de vista estatístico.

Tendo em vista o uso de *boxplots* para visualizar a presença de *outliers*, entre outras métricas estatísticas relevantes, resolvemos codificar uma operação para produzir esse tipo de gráfico sobre o nosso *DataFrame* `gastos` "limpo" — isto é, sem valores nulos, inválidos ou negativos na coluna `Valor (R$)`. Essa operação pode ser vista abaixo:

```python
def boxplot_gastos(df_gastos):
    # O eixo X do boxplot deve ser categórico
    if 'Ano' not in df_gastos.columns:
        df_gastos['Ano'] = pd.to_datetime(df_gastos['Data'],
           format= %d/%m/%Y %H:%M , errors='coerce').dt.year
    df_gastos['Ano'] = df_gastos['Ano'].astype('Int64')

    plt.figure(figsize=(12, 6))
    sns.boxplot(data=df_gastos, x= Ano , y= Valor (R$) )
    plt.title( Boxplot dos gastos por ano )
    plt.xlabel( Ano )
    plt.ylabel( Valor (R$) )
```

```
12      plt.grid(True, linestyle='--', alpha=0.7)
13      plt.show()
```

O *boxplot* resultante da operação pode ser visto a seguir:

Podemos ver que, a cada ano sobre os quais coletamos informações no portal de dados abertos da Câmara dos Deputados, de fato há muitos *outliers*. Eles são tão impactantes que outros dados exibidos pelo *boxplot*, como a mediana, os quartis, o mínimo e o máximo para cada ano observado sequer são visíveis. Mais um motivo para mantê-los em nosso *DataFrame*. E mais um motivo para nos preocuparmos com os gastos dos nossos representantes.

No próximo capítulo desta pesquisa, utilizaremos o *PowerBI* para construir *dashboards* a partir das informações coletadas, tratadas e organizadas ao longo da investigação até o

momento. Assim, esperamos complexificar a visão preliminar obtida durante a segunda etapa do trabalho, visualizando a distribuição dos gastos parlamentares a cada ano a partir de diferentes recortes, como organização partidária, região do Brasil e parlamentares individualmente.

3 Preparação dos Dados

Nesta última parte da nossa pesquisa, importamos as nossas planilhas geradas anteriormente para um novo projeto no *Power BI*, com o objetivo de produzir *dashboards* que nos auxiliem a resolver a principal interrogação que levantamos neste trabalho. Durante esse processo, os dados foram transformados a fim de que os ruídos identificados e mencionados na seção anterior fossem removidos dos nossos *datasets*.

No caso da planilha `gastos.xlsx`, removemos linhas em branco e filtramos duas colunas a fim de remover valores inválidos — a coluna `Valor (R$)` foi filtrada para remover as linhas com números negativos, e a coluna `Data` foi filtrada a fim de remover linhas com valores nulos, bem como registros anteriores a 2018 e posteriores a 2022. No caso da planilha `deputados.xlsx`, removemos linhas em branco e linhas com valores duplicados. No entanto, essas funções foram aplicadas sobre o arquivo sem que tivessem um efeito real, já que esta planilha em específico não sofria com ruídos de qualquer natureza.

3.1 Principais visualizações

A principal motivação da nossa investigação é descobrir se de fato gasta-se mais durante o período das eleições gerais do que ao longo dos quatro anos de atuação da legislatura eleita. Trata-se de uma visão fortemente presente em meio ao senso comum. Vejamos, então, os dois primeiros gráficos que geramos a partir dos nossos *datasets*:

images/ano_quarter.pdf

Dashboard 1: Total gasto por ano e por quarter

Os gráficos acima, gerados por meio do *PowerBI*, demonstram uma inversão daquilo que supunhamos representar a realidade da política brasileira, estimulados pelo senso comum. Acredita-se, com base em uma desconfiança em relação à classe política brasileira, que nossos representantes em Brasília gastam dinheiro em demasia, e que ainda o fazem mal. Ainda, acredita-se que essa "gastança" é ainda pior durante as eleições gerais, quando muitos deputados federais arriscam a chance de se reeleger. Porém, o que os nossos dados demonstram é justamente o oposto.

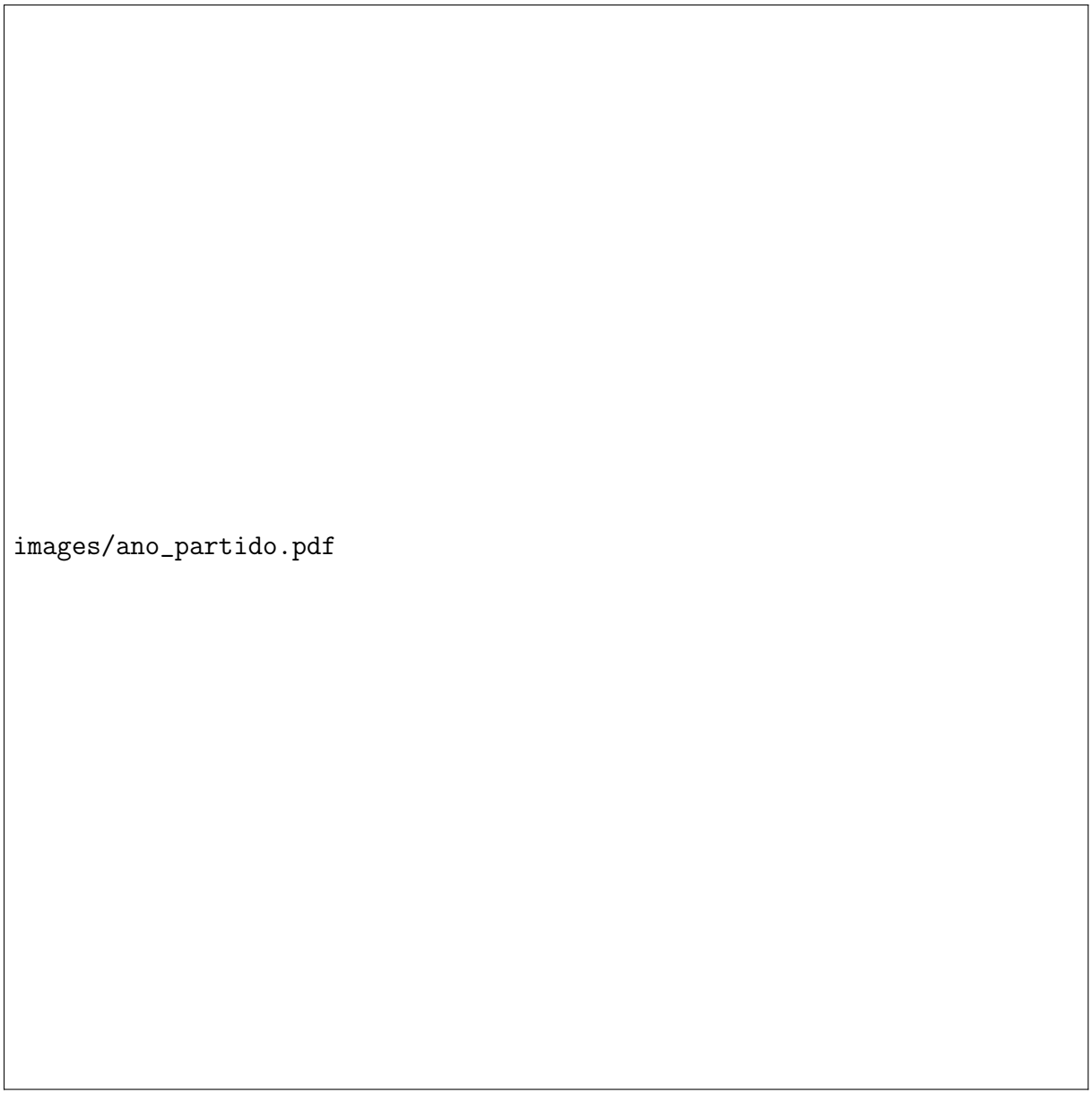
Nos histogramas representados pelo `Dashboard 1`, percebe-se que, dos cinco anos selecionados para a nossa investigação, não são necessariamente os anos eleitorais que se sobressaem em relação ao valor total gasto no período. O ano de 2022, quando foram eleitos os 513 deputados e deputadas da atual legislatura, é o mais imponente do intervalo temporal em termos de gastos, mas o ano de 2018 — quando ocorreu o penúltimo

pleito — é superado por 2021 no que se refere ao total gasto pelos parlamentares. Neste mesmo *dashboard*, ainda podemos observar um resultado segmentado por quarter, com destaques (pontos vermelhos) para os meses de outubro de cada um dos anos selecionados. O mês de outubro é justamente quando ocorrem o primeiro e o segundo turnos (quando necessário) das eleições presidenciais, sendo a Câmara dos Deputados definida na primeira volta do pleito.

Em nenhum dos casos o mês de outubro representa o ápice no montante gasto pelos parlamentares. Podemos visualizar que, no contexto dos dois anos eleitorais, há períodos em que se gasta mais do que durante o principal mês das eleições. Além disso, ao compararmos o mês de outubro dos anos eleitorais com este mesmo mês em anos não eleitorais, podemos ver que o total gasto em ao menos um ano eleitoral é menor em comparação a anos não eleitoriais — contrapondo o ano de 2018 com todos os outros anos do período selecionado, por exemplo, a exceção de 2022.

3.2 Outras visualizações

Com tantos dados coletados, resolvemos ir além do escopo original da pesquisa e, também, verificar algumas informações extras que possam contribuir para uma apreciação qualitativa do fato descoberto — que os nossos parlamentares não gastam mais em períodos eleitorais. Uma das visualizações criadas é, justamente, o valor total gasto por cada partido da Câmara dos Deputados, filtrado por ano. Vejamos, então, os gráficos construídos com este propósito, também utilizando o *PowerBI*:

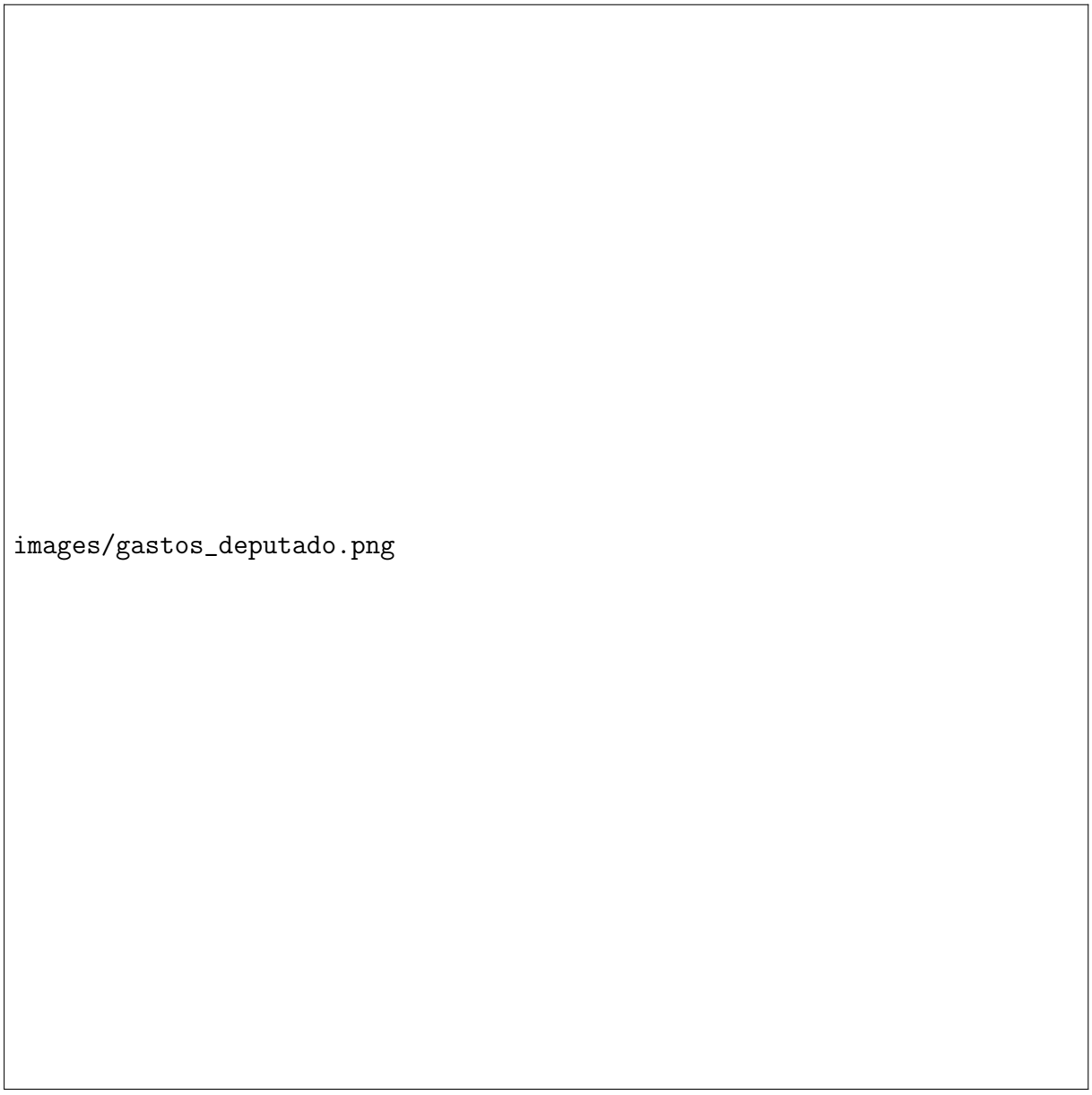


Dashboard 2: Total gasto por ano por partido

As mudanças de posição dos partidos listados neste *ranking* representam as diferentes correlações de forças que compuseram a Câmara dos Deputados nos anos selecionados. Em meio aos gastos registrados no ano de 2018, por exemplo, o Partido dos Trabalhadores (PT) surge como o campeão de despesas totais na Casa, seguido pelo Progressistas (PP) e pelo Movimento Democrático Brasileiro (MDB). No ano seguinte, a sigla que desponta na liderança dos gastos é o Partido Liberal (PL). Esse fenômeno, inclusive, nos levou a perceber que, nos arquivos originais, as despesas foram registradas com as siglas dos partidos mais atuais dos deputados, não com as siglas dos partidos a que pertenciam à época em que os gastos foram realizados. Afinal, sabemos que o ex-presidente Jair Bolsonaro foi eleito pelo Partido Social Liberal (PSL), não pelo PL, no qual ingressou apenas em 2021. Nas eleições de 2018, muitos novos parlamentares foram eleitos pelo PSL em função da campanha de Bolsonaro, mas não vemos esse partido em meio ao *ranking*.

Devido à quantidade de dados trabalhados, no entanto, seria impraticável corrigir um problema dessa magnitude.

A última visualização criada apresenta um *top 10* dos parlamentares com maior gasto total no período observado. A soma das despesas considera os gastos de todos os anos em questão. A razão pela qual produzimos este gráfico é justamente observar o custo que muitos dos nossos parlamentares representam em sua atuação na Câmara dos Deputados.



Dashboard 3: Os 10 deputados com maior gasto total no período

Esta e todas as outras visualizações podem ser melhor observadas no nosso *dashboard*, disponível para *download* neste link. Criamos filtros relativos a cada um dos anos selecionados, de modo que os *dashboards* interativos possam auxiliar interessados no assunto a tirar suas próprias conclusões a partir das informações que coletamos ao longo desta investigação.

4 Referências

[1] CÂMARA DOS DEPUTADOS. *Gastos dos deputados federais*. Agência Câmara de Notícias, [s.d.]. Disponível em: https://www.camara.leg.br/transparencia/gastos-parlamentares/. Acesso em: 9 set. 2025.

[2] CÂMARA DOS DEPUTADOS. *Saiba quais são as atribuições do deputado federal*. Agência Câmara de Notícias, 22 ago. 2022. Disponível em: https://www.camara.leg.br/noticias/903471-SAIBA-QUAIS-SAO-AS-ATRIBUICOES-DO-DEPUTADO-FEDERAL/. Acesso em: 9 set. 2025.

[3] TRIBUNAL REGIONAL ELEITORAL DO PIAUÍ (TRE-PI). *Dos Partidos Políticos*. [s.d.]. Disponível em: https://www.tre-pi.jus.br/partidos/duvidas-frequentes/dos-partidos-politicos-e-das-coligacoes. Acesso em: 9 set. 2025.

[4] XAVIER, Renan Melo. Em período eleitoral, Câmara gasta R$ 18,7 milhões em cota parlamentar. *Metrópoles*, Brasília, 23 set. 2018. Disponível em: https://www.metropoles.com/brasil/politica-brasil/em-periodo-eleitoral-camara-gasta-r-187-milhoes-em-cota-parlamentar. Acesso em: 30 out. 2025.