



# Programa Capacita Brasil/C-jovem

**Ciência de Dados - Imersão**

Coordenadora



Apoio



Instituição Executora



Parceiros





# Apresentação Projeto

Capacita Brasil/C-jovem

Processos do desenvolvimento do projeto:

- Analise dos Dados;
- Limpeza dos Dados;
- Pré-processamento dos Dados;
- Merge dos Dados;
- Dataset Final;
- Análise Exploratória (Python e BI).



UNIVERSIDADE  
ESTADUAL DO CEARÁ



Tecnologia para antecipar o futuro





# Merge de Dados

## Limpeza dos Dados

- Identificação de colunas com valores nulos ou inconsistentes, seguida da aplicação de estratégias;
- Padronização dos Números de telefones.

## Merge

- Datasets mensais em Dataset anual para análise.





# Limpeza dos dados

```
▶ #Importa as bibliotecas necessarias e as chama por uma nome encurtado ('pd', 'np')
import pandas as pd
import numpy as np
```

```
[ ] #Para acessar arquivos contidos em drives, usamos a biblioteca do google colab.
    #o codigo abaixo 'monta' o drive no notebook e pede o acesso ao drive do usuario
    #na qual contem os dados necessarios.
    from google.colab import drive
    drive.mount('/content/drive')
```

```
▶ #Apos importar as bibliotecas, iremos ler o csv mandado utilizando o Pandas
  #e o tornando em um dataframe, uma tabela que nos podemos processar usando tanto o Pandas quanto o Numpy.

df = pd.read_csv('Caminho do arquivo', dtype={'TELEFONE': str})

#Este codigo necessita do caminho/path do arquivo, que significa o local onde está
#armazenado este arquivo, para facil entendimento, um caminho exemplo foi adicionado.
#a segunda parte do codigo muda a coluna 'TELEFONE' para o tipo de dado 'string'.
```



# Limpeza dos dados

```
▶ # No Python, dados faltantes são geralmente representados por 'NaN' (Not a Number).  
# Esta linha substitui todas as ocorrências de ' - ' (um traço entre espaços) por NaN.  
# Isso ajuda o pandas a entender que esses campos estão vazios ou sem informação.  
# 'inplace=True' significa que a mudança é feita diretamente na nossa tabela 'df'.  
df.replace(' - ', np.nan, inplace=True)  
  
# Verificar valores faltantes por coluna  
print("\nValores faltantes por coluna:")  
# df.isna() cria uma tabela de True (se o valor for NaN) e False (se não for).  
# .sum() então soma os 'True' por coluna, nos dando a contagem de NaN.  
print(df.isna().sum())  
  
# Define uma função chamada 'clean_phone' para limpar os números de telefone.  
# Uma função é um bloco de código que realiza uma tarefa específica e pode ser reutilizado.  
def clean_phone(phone):  
    # Verifica se o valor do telefone é NaN (faltante).  
    # pd.isna(phone) retorna True se 'phone' for NaN.  
    if pd.isna(phone):  
        # Se for faltante, retorna None (que é outra forma de representar ausência de valor em Python).  
        return None
```



# Limpeza dos dados

```
# Converte o número de telefone para texto (string) para garantir que podemos processá-lo.
# Em seguida, 'filter(str.isdigit, str(phone))' pega apenas os caracteres que são dígitos (0-9).
# ''.join(...) junta esses dígitos de volta em um texto único.
# Exemplo: '(99) 1234-5678' se tornaria '9912345678'.
cleaned = ''.join(filter(str.isdigit, str(phone)))

# Se, após remover tudo que não é dígito, o resultado for um texto vazio
# (por exemplo, se o telefone original era algo como "não informado" ou "---"),
# então retorna None (valor ausente). Caso contrário, retorna o número limpo.
return cleaned if cleaned else None

# Aplicar a função feita acima à coluna de telefone
df['TELEFONE'] = df['TELEFONE'].apply(clean_phone).astype('string')
# Aplica a função 'clean_phone' a cada valor na coluna 'TELEFONE' da nossa tabela 'df'.
# O método '.apply()' passa cada número de telefone para a função 'clean_phone'.
# '.astype('string')' garante que a coluna, após a limpeza, seja tratada como texto.

# Converter a coluna 'DATA_CIRURGIA' para datetime64, um formato de data que o pandas entende
df['DATA_CIRURGIA'] = pd.to_datetime(df['DATA_CIRURGIA'], errors='coerce')

# Salvar em novo CSV
df.to_csv('Dadoprocessado.csv', index=False)
```




# Limpeza dos dados

```
[ ] df.dtypes #diz os tipos de cada coluna
```

```
[ ] df['DATA_CIRURGIA'] #exibindo apenas a coluna data para verificar se a transformação foi bem sucedida
```

```
[ ] df['TELEFONE'] #exibindo apenas a coluna telefone para verificar se a transformação foi bem sucedida
```



# Merge em Ano

```
[ ] # caminho dos csv para pegar todos os arquivos
caminho = "/content/2022/*.csv"
# usando glob para buscar os arquivos do padrão usado e sorted para que fiquem na ordem alfabética
arquivos = sorted(glob.glob(caminho))
```

```
▶ # lista vazia onde será armazenado os dataframes
lista_df = []

# for para percorrer os arquivos
for arquivo in arquivos:
    # leitura dos csv
    df = pd.read_csv(arquivo, encoding='latin1')
    # adicionando ANO_CIRURGIA e convertendo a data
    df['ANO_CIRURGIA'] = 2022
    df['DATA_CIRURGIA'] = pd.to_datetime(df['DATA_CIRURGIA'], format='%d-%b-%y', dayfirst=True, errors='coerce')
    # Tirando traços, espaços, parenteses dos telefones
    df['TELEFONE'] = df['TELEFONE'].str.replace(r'\D', '', regex=True)
    lista_df.append(df)

df = pd.concat(lista_df, ignore_index=True)
df.to_csv("planilha_2022_completa.csv", index=False)
```



# Merge Unificado

```
[ ] # Lista dos caminhos dos CSV, juntando os em um para ser lidos.
    # É necessário colocar em ordem para evitar erro de ordenação dos meses.
    arquivos_csv = ['Adicione aqui o caminho de cada CSV que voce deseja juntar',
                    'Adicione aqui o caminho de cada CSV que voce deseja juntar',
                    'Adicione aqui o caminho de cada CSV que voce deseja juntar',
                    'Adicione aqui o caminho de cada CSV que voce deseja juntar'
                    ]

    # Cria uma lista para armazenar os dataframes lidos
    # que sera usada para ler eles em sequencia.
    lista_dfs = []

    # Loop através da lista de arquivos, lendo cada um e adicionando à lista_dfs
    # para concatenar(Juntar/Unir) os arquivos em um só.
    for arquivo in arquivos_csv:
        try:
            df_temp = pd.read_csv(arquivo, dtype={'TELEFONE':str})
            lista_dfs.append(df_temp)
            print(f"Arquivo '{arquivo}' lido com sucesso.")
        except FileNotFoundError:
            print(f"Erro: O arquivo '{arquivo}' não foi encontrado.")
        except Exception as e:
            print(f"Ocorreu um erro ao ler o arquivo '{arquivo}': {e}")
```



# Merge Unificado

```
# Junta todos os dataframes na lista verticalmente (concatenação) e adiciona a coluna ANOS
# para a facil identificação da epoca em que foi feito o procedimento.
if lista_dfs:
    df_final = pd.concat(lista_dfs, ignore_index=True)
    df_final['ANO'] = '2024'

    # Aplica a função usada previamente na tabela unificada para evitar
    # erros de formatação dos tipos de dados.
    df_final['TELEFONE'] = df_final['TELEFONE'].apply(clean_phone).astype('string') # Mantém como string

    # Converte as colunas 'DATA_CIRURGIA' para datetime64 novamente para
    # evitar quaisquer erros de formatação dos tipos de dados.
    df_final['DATA_CIRURGIA'] = pd.to_datetime(df_final['DATA_CIRURGIA'], errors='coerce')

    # Exibe as primeiras linhas do dataframe final
    print("\nPrimeiras linhas do dataframe final:")
    print(df_final.head())

    # Exibe informações sobre o dataframe final
    print("\nInformações sobre o dataframe final:")
    df_final.info()

    # print("\nDataframe combinado salvo em 'dataframe_combinado_vertical.csv'")

else:
    print("\nNenhum dataframe foi lido com sucesso para ser combinado.")
#Salva a junção de dataframes em um só, sendo ele combinado verticalmente.
df_final.to_csv('dataframe_combinado_vertical.csv', index=False,)
```



# Análise estratégica de procedimentos hospitalares da Santa Casa

## 📌 Objetivo principal

Extrair indicadores gerenciais, identificar tendências e gerar visualizações que possam auxiliar na tomada de decisão

## 👥 Como a análise foi feita

Começamos com a coleta e limpeza de dados de aproximadamente 33 mil procedimentos que foram estruturados no formato CSV. Depois, processamos com a linguagem Python, usando bibliotecas estatísticas e de visualização (Pandas, Matplotlib e Seaborn) para identificar padrões e tendências.





# Análise estatística e geração de indicadores (KPIs)

A partir da base limpa, extraímos os principais KPIs (Indicadores-Chave de Desempenho), que são as métricas fundamentais para a gestão:



**Total de procedimentos realizados**



**Média mensal**



**Crescimento ou declínio por ano**

Monitorar KPIs auxilia  
agir preventivamente!

Também aplicamos análises comparativas para avaliar tendências ao longo do tempo.



# Análise temporal e sazonal

Utilizamos a função `seasonal_decompose`, que nos permite decompor as séries temporais em tendência, sazonalidade e ruído. Isso ajuda a entender:



Se há períodos do ano com mais ou menos procedimentos.



Se há queda sustentada ou picos sazonais.

Auxilia na gestão de recursos e tomada de decisão.

## Saídas:

- Um dashboard executivo em imagem (dashboard\_executivo.png) - centralizando todos os gráficos gerados na análise;
- Um relatório de KPIs em texto (relatorio\_kpis.txt).



# Visualização dos resultados - Relatório de KPIs

```
Carregando dados hospitalares...
Dados carregados: 33,867 procedimentos de 2020 a 2025
Gerando relatório de gestão hospitalar...
Dashboard salvo em: relatorio_gestao_hospitalar/dashboard_executivo.png
Relatório completo gerado em: relatorio_gestao_hospitalar/
Dashboard: dashboard_executivo.png
KPIs: relatorio_kpis.txt
```

## RESUMO EXECUTIVO:

```
Total de procedimentos: 33,867
Crescimento anual: -69.6%
Média mensal: 529 procedimentos
```

## ALERTAS:

```
Tendência de DECLÍNIO: -13.2% nos últimos 6 meses
ALTA concentração: 99.1% em apenas 3 convênios
```

Alguns destaques importantes:

- **Total de procedimentos analisados:** 33.867.
- **Período:** 2020 a 2025.
- **Média mensal:** 529 procedimentos.
- **Crescimento anual:** **-69,6%** — ou seja, houve uma queda expressiva no número de procedimentos ao longo dos anos.

Esses números acendem um alerta importante: **algo está impactando negativamente a quantidade de atendimentos**. Isso requer atenção da gestão para investigar causas e pensar em possíveis soluções.

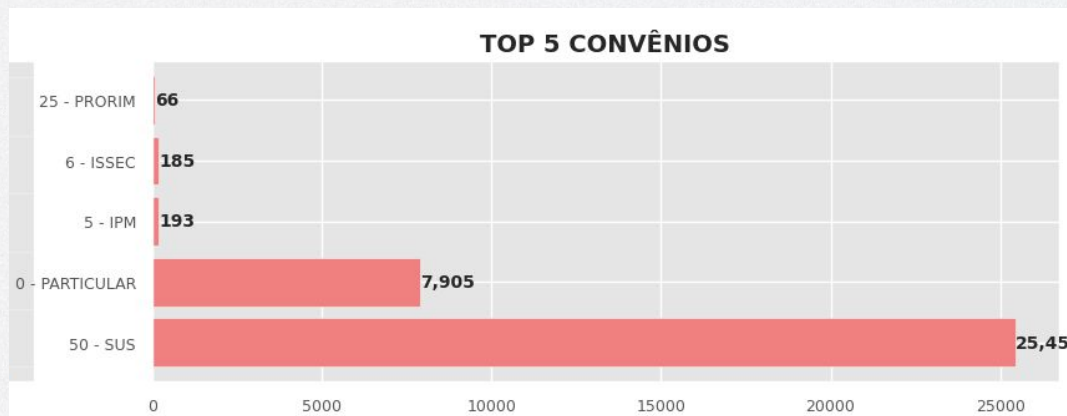


# Visualização dos resultados - gráficos



## Evolução do volume de procedimentos

- Este gráfico mostra a **quantidade de procedimentos por mês ao longo dos anos**, sendo o principal indicador da atividade da Santa Casa. A linha azul representa os dados reais, e a linha vermelha indica a tendência.
- Observamos uma **queda expressiva** a partir de 2023, que segue até 2025.

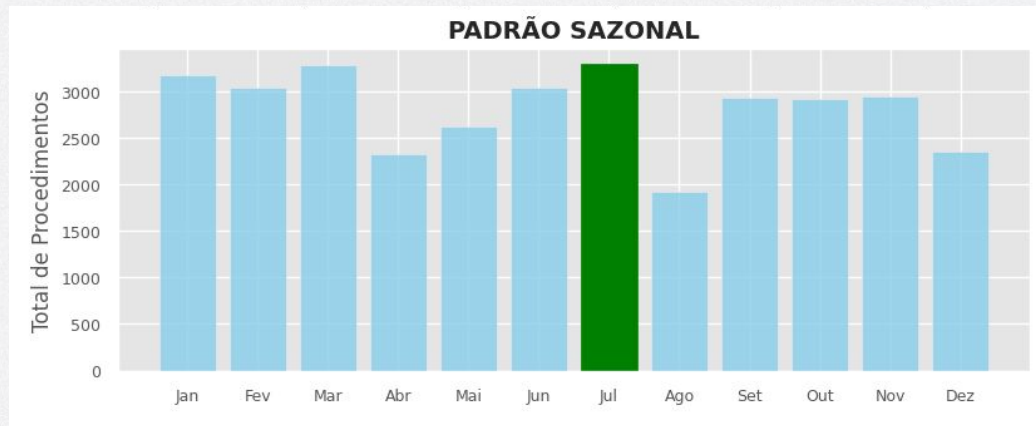


## Top 5 convênios

- Dos 33.867 procedimentos registrados, **SUS e Particular** representam juntos mais de 99% dos atendimentos.
- Isso demonstra uma alta dependência desses dois convênios.

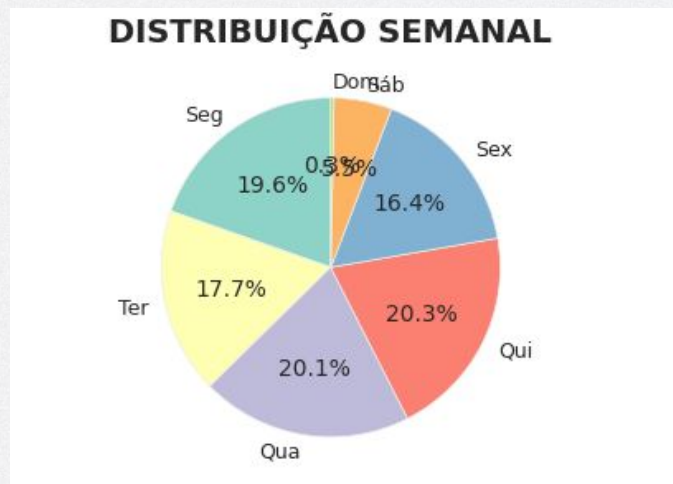


# Visualização dos resultados - gráficos



## Padrão sazonal

- Aqui observamos como os procedimentos se distribuem ao longo dos meses do ano.
- Julho é o mês com maior concentração de atendimentos, e abril e dezembro tendem a ter os menores volumes. Apesar de no gráfico parecer agosto, como não tivemos os dados desse mês em um dos anos, decidimos desconsiderar.
- Essa análise ajuda no **planejamento de recursos e equipes**.

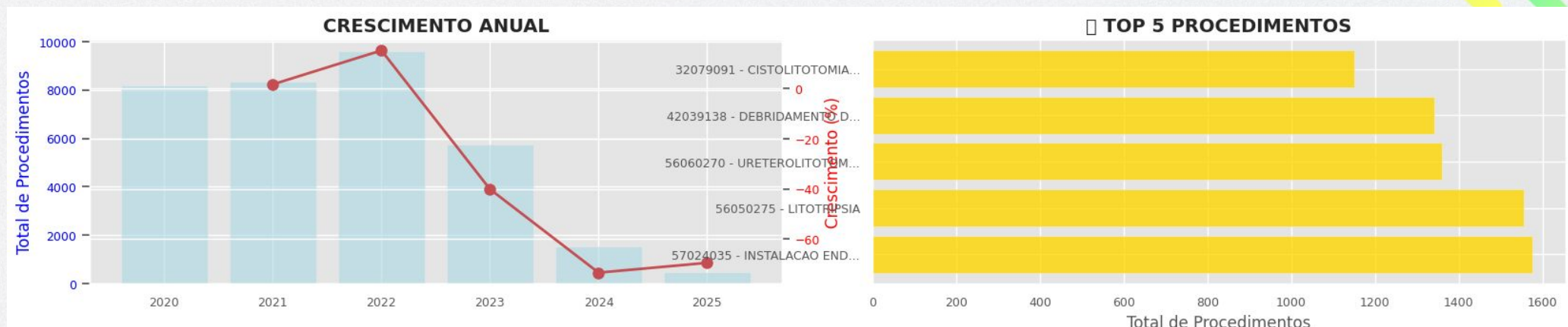


## Distribuição semanal

- Este gráfico de pizza mostra os dias da semana com mais procedimentos, mostrando a distribuição da carga de trabalho. Afins de esclarecimento: domingo concentra 0.3% e sábado concentra 5.5%.
- As quintas-feiras (20,3%) e quartas-feiras (20,1%) são os dias de maior movimento, enquanto domingos quase não registram atividade.
- Ajuda a otimizar o agendamento de salas de cirurgia, alocar equipes de enfermagem e administrar o fluxo de pacientes, **evitando gargalos em dias de pico**.



# Visualização dos resultados - gráficos



## Crescimento anual

- Neste gráfico de barras, vemos claramente a redução do número de procedimentos a partir de 2023.
- O pico foi em 2022, com quase 10 mil procedimentos.
- Em 2025, esse número caiu para menos de 1.000, já que nossos dados eram apenas dos 3 primeiros meses.
- O declínio acumulado é de -69,6%, com queda de -13,2% apenas nos últimos 6 meses.

## Top 5 procedimentos

- Os procedimentos mais realizados são:
  - **CISTOLOTOTOMIA** – 1.558 procedimentos;
  - **DEBRIDAMENTO DE PELE E TECIDO SUBCUTÂNEO** – 1.453 procedimentos;
  - **URETEROLITOTOMIA** – 1.441 procedimentos
  - **LITOTRIPSIA** – 1.334 procedimentos;
  - **INSTALAÇÃO DE ENDOPRÓTESE** – 1.230 procedimentos.
- Útil para planejar estoques e insumos específicos; capacitar profissionais de acordo com a demanda e avaliar custos e especialidades mais críticas.



## Coordenadora



## Parceiros Locais

SECRETARIA DO  
PLANEJAMENTO  
E GESTÃO

SECRETARIA DO  
DESENVOLVIMENTO  
ECONÔMICO

SECRETARIA DA  
EDUCAÇÃO

SECRETARIA DA  
CIÊNCIA, TECNOLOGIA  
E EDUCAÇÃO SUPERIOR



**CEARÁ**  
GOVERNO DO ESTADO

## Instituição Executora



**UECE**



**UNIVERSIDADE  
FEDERAL DO CEARÁ**



**INSTITUTO FEDERAL  
Ceará**

## Parceiros



**IRACEMA DIGITAL**  
ARTICULAÇÃO - TECNOLOGIA DIGITAL



**DELL** Technologies