Limpeza e Preparação de Dados

"Antes de analisar, é preciso arrumar."

Analogia do Quarto Bagunçado

Imagine tentar encontrar seu livro favorito em um quarto completamente bagunçado. Mesmo sabendo que o livro está lá, você gastará muito tempo procurando e pode até desistir.

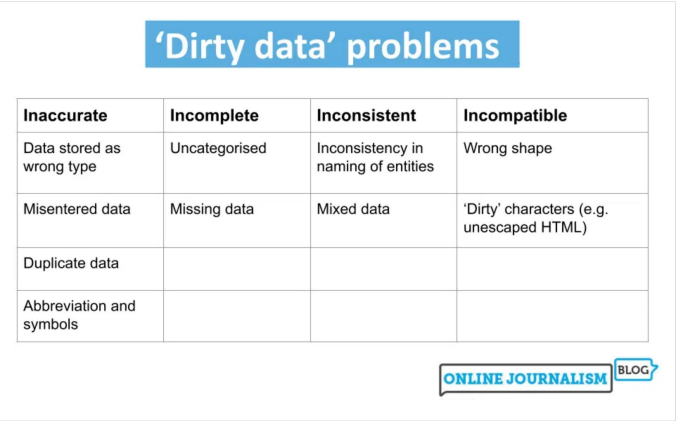
Com dados é a mesma coisa: antes de extrair valor, precisamos organiza-los.

Por que a limpeza é importante?

Dados sujos levam a conclusões erradas. Na ciência de dados, temos um princípio: **"Garbage In, Garbage Out"** (Lixo entra, lixo sai).

Impacto na análise

Estudos mostram que cientistas de dados gastam até 80% do tempo limpando e preparando dados, não analisando-os.



O que é "dado sujo"?

Dados Faltando (Nulos)

Definição:

Dados Faltando (Nulos) são valores ausentes em um conjunto de dados. Ocorrem quando uma informação simplesmente não existe ou não foi registrada.

Exemplo Prático

Em um formulário de cadastro, um cliente não preencheu o campo de e-mail.

Nome	Idade	Email	Cidade
João	25	joao@email.com	São Paulo
Maria	30	NaN	Rio de Janeiro
Carlos	22	carlos@email.com	Belo Horizonte

Impacto na Análise

Dados nulos podem causar:

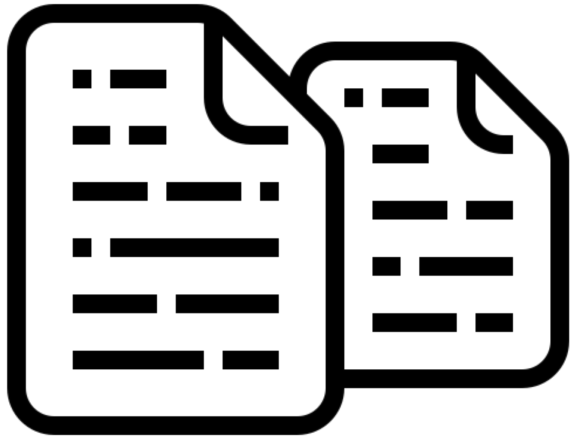
- Erros em cálculos estatísticos
- Falhas em algoritmos de aprendizado
- Análises incompletas e enviesadas

O que é "dado sujo"?

Dados Repetidos (Duplicatas)

Definição:

Dados Repetidos (Duplicatas) ocorrem quando a mesma informação aparece mais de uma vez em um conjunto de dados.



Exemplo Prático

Um cliente cadastrado duas vezes na mesma lista:

ID	Nome	Email
1	João Silva	joao@email.com
2	Maria Souza	maria@email.com
3	João Silva	joao@email.com
4	Carlos Lima	carlos@email.com

Impacto na Análise

Dados duplicados podem causar:

- Contagens inflacionadas
- Estatísticas enviesadas
- Decisões baseadas em informações distorcidas

O que é "dado sujo"?

Dados Inconsistentes

Definição:

Dados Inconsistentes são informações que representam o mesmo valor, mas estão registradas de formas diferentes, dificultando a análise e agrupamento correto dos dados.

Tipos de Inconsistências

1. Erros de Digitação

"São Paulo" vs. "Sao Paulo" vs. "SP" vs. "S. Paulo"

2. Diferentes Formatos

"janeiro" vs. "JAN" vs. "01" vs. "Jan/2023"

3. Unidades Diferentes

"1,70m" vs. "170cm" vs. "5'7""

Impacto na Análise

Dados inconsistentes dificultam agrupamentos e contagens corretas, levando a análises fragmentadas e conclusões imprecisas sobre os dados.

Por que dados sujos são um problema?

Análises Estragadas

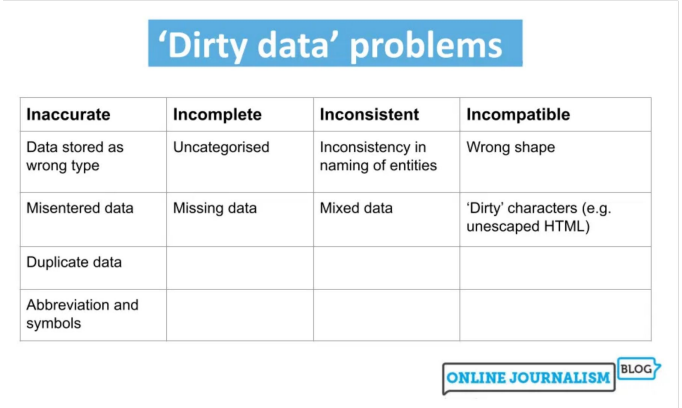
Dados sujos levam a **contagens incorretas**, **gráficos enganosos** e **estatísticas distorcidas**. Uma única entrada duplicada pode alterar significativamente médias e percentuais.

Decisões Erradas

Quando decisões são baseadas em **informações imprecisas**, as consequências podem ser graves. Empresas podem direcionar recursos para o público errado ou ignorar oportunidades importantes.

Perda de Tempo e Recursos

Cientistas de dados gastam até **80% do seu tempo** limpando dados. Quanto mais sujos os dados, mais tempo é desperdiçado em tarefas que não geram valor direto.



> *"A qualidade de suas análises nunca será melhor que a qualidade dos seus dados. Dados sujos são o inimigo número um da análise confiável."*
>
> **- Princípio fundamental da Ciência de Dados**

Atividade A: Demonstração Guiada

Olhando Dados de Verdade

Vamos explorar um dataset real para identificar problemas comuns de dados sujos.

1 Carregar um Dataset Simples

Vamos usar um dataset público de uma pesquisa escolar.

```
import pandas as pd
df = pd.read_csv('pesquisa_escolar.csv')
```

2 Mostrar as Primeiras Linhas

Visualize as primeiras linhas para entender a estrutura.

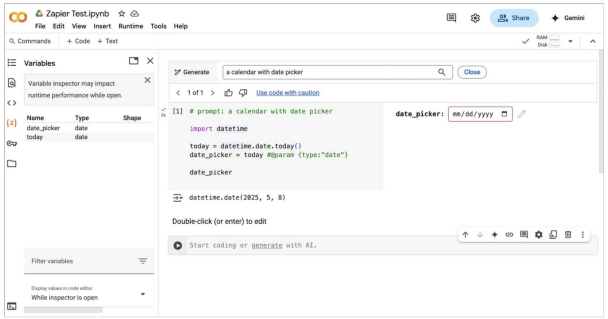
```
df.head()
```

3 Destacar um Problema

Observe os valores faltantes e inconsistências.

```
# Verificar valores nulos
df.isnull().sum()

# Verificar valores únicos
df['cidade'].unique()
```



Pergunta para Reflexão

"Se fôssemos contar os alunos por cidade, esses erros iriam atrapalhar nossa contagem?"

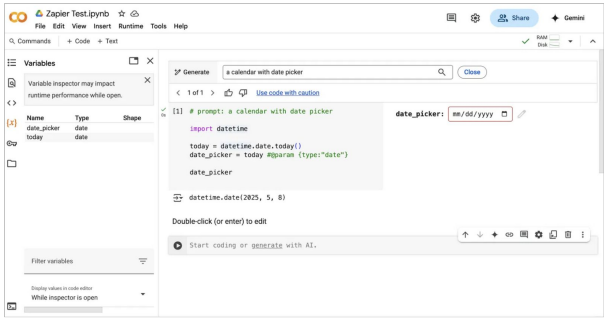
Atividade B: Criando um Dataset Sujo

Passo a Passo - Parte 1

Nesta atividade, todos trabalharão juntos para criar um dataset com problemas propositais:

1 Abram um Notebook

Acesse o Google Colab e crie um novo notebook em colab.research.google.com .



2 Crie as Colunas

Vamos usar as colunas: "Nome", "Idade", "Cidade" e "Curso Favorito".

```
import pandas as pd

# Criando um dicionário vazio
data = {
    'Nome': [],
    'Idade': [],
    'Cidade': [],
    'Curso Favorito': []
}
```

Atividade B: Criando um Dataset Sujo

Passo a Passo - Parte 2

Continuando nossa atividade, vamos coletar dados da turma e introduzir erros propositais para criar um dataset sujo.

3 Colete Dados

Fornecer suas informações: nome, idade, cidade e curso favorito.

```
data = {
    'Nome': ['Ana Silva', 'João Pereira', 'Maria Santos', 'Pedro Alves'],
    'Idade': [16, 17, 16, None],  # Valor nulo proposital
    'Cidade': ['São Paulo', 'Sao Paulo', 'Rio de Janeiro', 'Belo horizonte'],
    'Curso_Favorito': ['Matemática', 'Ciências', 'Matemática', 'História']
}
```

4 Introduza Erros Propositais

Crie erros específicos no dataset:

- **Valor nulo:** Deixe um campo em branco (idade do Pedro)
- **Duplicata:** Adicione o mesmo aluno duas vezes
- **Inconsistência:** Escreva "São Paulo" e "Sao Paulo"

5 Crie o DataFrame

Use o Pandas para criar um DataFrame com os dados coletados:

```
import pandas as pd

# Criar o DataFrame
df = pd.DataFrame(data)

# Visualizar o DataFrame
print(df)
```

Dica:

Identifique os problemas no dataset criado. Perguntas: "Quais problemas vocês conseguem identificar neste dataset? Como esses problemas poderiam afetar uma análise?"

Atividade C: Limpeza em Grupo

Encontrando e Tratando Valores Nulos

Agora vamos trabalhar juntos para limpar o dataset que acabamos de criar. Primeiro, vamos identificar e tratar os valores nulos.

1 Verificando Valores Nulos

Use o comando abaixo para contar quantos dados estão faltando:

```
df.isnull().sum()
```

```
Nome            0
Idade           2
Cidade          1
Curso Favorito  3
dtype: int64
```

2 Decidir o que fazer com os nulos

? O que fazer com valores nulos?

Remover a linha	Preencher com valor
`df.dropna()`	`df.fillna(valor)`

3 Aplicando a solução escolhida

Exemplo: Preencher "Cidade" e remover linhas sem idade:

```
# Preencher cidade com "Não informado"
df['Cidade'] = df['Cidade'].fillna('Não informado')

# Remover linhas onde idade está faltando
df = df.dropna(subset=['Idade'])
```

Atividade C: Limpeza em Grupo

Encontrando e Removendo Duplicatas

Agora vamos identificar e remover dados duplicados no dataset que criamos juntos.

1 Verificar Duplicatas

Primeiro, vamos contar quantos registros duplicados existem:

```
df.duplicated().sum()
```

```
2
```

2 Visualizar as Duplicatas

Para ver quais são as linhas duplicadas:

```
df[df.duplicated()]
```

3 Remover Duplicatas

Agora vamos remover as duplicatas:

```
df.drop_duplicates(inplace=True)
```

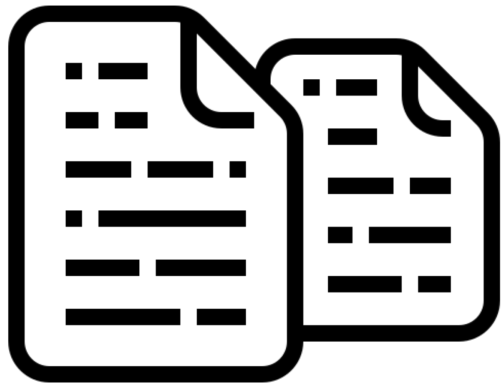
O parâmetro `inplace=True` modifica o DataFrame original.

4 Confirmar a Remoção

Verificar se as duplicatas foram removidas:

```
df.duplicated().sum()
```

```
0
```



Atividade C: Limpeza em Grupo

Padronizando o Texto

Agora vamos padronizar os textos para corrigir inconsistências como "São Paulo" vs. "Sao Paulo".

1 Verificar Valores Únicos

Primeiro, vamos verificar os valores únicos na coluna "Cidade":

```
df['Cidade'].unique()
```

```
array(['São Paulo', 'Sao Paulo', 'Rio de Janeiro', 'Belo horizonte'], dtype=object)
```

2 Padronizar o Texto

Vamos usar o método `str.title()` para padronizar o texto:

```
# Padronizar a primeira letra de cada palavra em maiúscula
df['Cidade'] = df['Cidade'].str.title()

# Verificar o resultado
df['Cidade'].unique()
```

```
array(['São Paulo', 'Sao Paulo', 'Rio De Janeiro', 'Belo Horizonte'], dtype=object)
```

3 Corrigir Inconsistências Específicas

Para corrigir inconsistências específicas, podemos usar o método `replace` :

```
# Corrigir "Sao Paulo" para "São Paulo"
df['Cidade'] = df['Cidade'].replace('Sao Paulo', 'São Paulo')
```

`.str.lower()`
Converte para minúsculas

`.str.upper()`
Converte para MAIÚSCULAS

`.str.strip()`
Remove espaços extras

`.str.replace()`
Substitui texto específico