

Departamento de Engenharia de Sistemas e Computação

Pandas na Prática

Agrupamentos, agregações, valores ausentes e gráficos

João Araujo

Algoritmos Computacionais I

Aula 29

Roteiro

- Cálculos agrupados: `groupby` e agregações
- Contagem de frequências: `value_counts`, `nunique`
- Valores ausentes: `isna`, `fillna`, `dropna`
- Datas e horas
- Gráficos direto do Pandas
- Roteiro de uma análise exploratória

Perguntas típicas de quem analisa dados

- Qual o tempo de ciclo **médio de cada máquina**?
- Qual a **taxa de peças com defeito** por turno?
- Quantas peças **de cada tipo** cada operador produziu?
- (Gapminder) Qual foi a expectativa de vida média **por ano**? E **por continente**?

Padrão: dividir–aplicar–combinar (*split–apply–combine*)

1. dividir os dados em grupos → **2.** aplicar um cálculo a cada grupo → **3.** combinar os resultados numa tabela.

groupby: médias agrupadas

```
media_por_maquina = df.groupby("maquina")["tempo_ciclo_seg"].mean()
print(media_por_maquina.round(2))
```

```
maquina
M1    32.92
M2    28.34
M3    45.28
M4    30.25
Name: tempo_ciclo_seg, dtype: float64
```

- `groupby("maquina")` agrupa; `["tempo_ciclo_seg"]` escolhe a coluna; `.mean()` calcula
- O objeto agrupado sozinho não mostra nada útil: falta aplicar a agregação

Várias estatísticas de uma vez: agg

```
resumo = (
    df
    .groupby("maquina")["tempo_ciclo_seg"]
    .agg(["count", "mean", "std", "min", "max"])
    .round(2)
)
print(resumo)
```

```
         count   mean   std   min   max
maquina
M1          17  32.92  3.55  28.0  38.9
M2          18  28.34  3.49  20.9  33.6
M3          12  45.28  3.84  39.9  52.4
M4          13  30.25  2.00  26.8  34.6
```

Os parênteses externos permitem o *encadeamento de métodos* em várias linhas (estilo preferido, como no notebook).

Agrupando por mais de uma coluna

```
tab = df.groupby(["maquina", "status"]).size().unstack(fill_value=0)
print(tab)
```

```
status    Aprovado  Descarte  Retrabalho
maquina
M1              13         1           3
M2              13         0           5
M3              10         0           2
M4              12         0           1
```

- Agrupar por duas colunas gera um *índice composto*; `unstack()` transforma o segundo nível em colunas
- `reset_index()` volta a ter um DataFrame "plano" (os nomes viram colunas comuns)
- No notebook: `df.groupby(["year", "continent"])[["lifeExp", "gdpPercap"]].mean()`

Taxa de defeitos por máquina e por turno

```python
df["com_defeito"] = df["status"] != "Aprovado"          # True/False
print(df.groupby("maquina")["com_defeito"].mean().round(2))
print(df.groupby("turno")["com_defeito"].mean().round(3))
```

```
maquina: M1 0.24 | M2 0.28 | M3 0.17 | M4 0.08
turno:   Manha 0.222 | Noite 0.240 | Tarde 0.118
```

- A média de uma coluna booleana ($True = 1$) é a proporção de `True`
- M2 tem a maior taxa de problemas, apesar de M3 ser a mais lenta: tempo e qualidade são perguntas diferentes!

Frequências: `value_counts` e `nunique`

```
print(df["maquina"].value_counts())             # M2 18, M1 17, M4 13, M3 12
print(df["operador"].nunique())                 # quantos operadores distintos
print(df.groupby("turno")["operador"].nunique())
print(df["defeito"].value_counts())             # ignora NaN por padrão
print(df["turno"].value_counts(normalize=True).round(2))   # proporções
```

- `value_counts()`: frequência de cada valor (ordenada)
- `nunique()`: número de valores *distintos*
- No *gapminder*: `df.groupby("continent")["country"].nunique()` → África 52, Américas 25, Ásia 33, Europa 30, Oceania 2

Valores ausentes

```
print(df.isna().sum())                # ausentes por coluna
# defeito: 48 ausentes  (= as 48 peças aprovadas)

df["defeito"] = df["defeito"].fillna("sem defeito")   # preenche
df.dropna(subset=["tempo_ciclo_seg"])                 # remove linhas
print(df["defeito"].value_counts().head(3))
```

```
defeito
sem defeito    48
Risco           5
Trinca          4
```

Decida com critério

Ausente significa "não se aplica" (aqui: peça sem defeito) ou "não foi medido"? Preencher, remover ou manter muda os resultados: documente a decisão.

Datas e horas

```
df["timestamp"] = pd.to_datetime(df["timestamp"])
print(df["timestamp"].min(), df["timestamp"].max())
# 2026-08-03 06:04:00   2026-08-03 12:54:00
df["hora"] = df["timestamp"].dt.hour
pecas_por_hora = df.groupby("hora").size()
print(df["timestamp"].dt.day_name().head(1))    # Monday
```

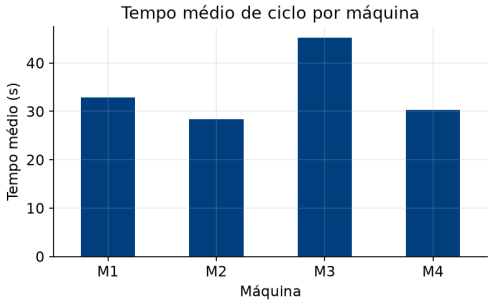
- `pd.to_datetime` converte texto em data/hora; o acessor `.dt` extrai `year`, `month`, `day`, `hour`...
- Permite agrupar por mês, semana ou hora, e calcular *intervalos* entre eventos
- Em dados reais, datas costumam vir em formatos variados: confira `dtypes`

Gráficos direto do Pandas

```python
import matplotlib.pyplot as plt

# média por máquina (barras)
(df.groupby("maquina")["tempo_ciclo_seg"]
   .mean()
   .plot(kind="bar", rot=0))
plt.ylabel("Tempo médio (s)")
plt.show()

# histograma de uma coluna
df["tempo_ciclo_seg"].plot(kind="hist",
                           bins=10)
```



- `.plot(kind="bar "hist "box "line "scatter")`: usa o Matplotlib por baixo
- Para personalizar, use `fig, ax = plt.subplots()` e passe `ax=ax` ao `.plot`

Série temporal do *gapminder*

```
expect_global = df.groupby("year")["lifeExp"].mean()
print(expect_global.head(3))
# year
# 1952    49.057620
# 1957    51.507401
# 1962    53.609249

expect_global.plot()
plt.ylabel("Expectativa de vida média (anos)")
plt.show()
```

- Exemplo final do notebook: o gráfico de linha mostra a **tendência de crescimento** (de 49 para 67 anos entre 1952 e 2007)
- Visualização e agrupamento juntos: é assim que se explora dados

Roteiro de uma análise exploratória

1. **Carregar** (`read_csv`) e **conhecer**: `shape`, `dtypes`, `head`, `describe`
2. **Limpar**: ausentes, tipos (`to_datetime`), duplicatas, valores impossíveis
3. **Perguntar**: o que eu quero saber? (taxa, média, comparação, tendência)
4. **Calcular**: filtros, `groupby`, agg
5. **Visualizar**: o gráfico certo para cada pergunta (Aula 27)
6. **Concluir e documentar**: o que os dados mostram? Quais as limitações?

Este é o esqueleto de muitos projetos finais

Combine: arquivos CSV + Pandas + funções bem organizadas + gráficos + testes (Aulas 30 a 33).

Atividade no laboratório

Com `dadosproducao.csv`:

1. Calcule o tempo médio por máquina *e* por turno (tabela com as duas chaves).
2. Qual operador produziu mais peças com problema (`Retrabalho` ou `Descarte`)?
3. Qual o defeito mais comum em cada máquina?
4. Quantas peças foram produzidas por hora? Faça um gráfico de linha.
5. Escreva uma função `relatorio(df)` que imprime um resumo (totais, taxas e a pior máquina) e uma função que salva dois gráficos em PNG.

Dica: para o item 3 use `df.groupby(["maquina","defeito"]).size()`.

Resumo da aula

- `groupby` + agregação responde perguntas por grupo; `agg` calcula várias estatísticas
- `value_counts` e `nunique` fazem contagens de frequências
- Valores ausentes: `isna`, `fillna`, `dropna`; decida e documente
- `.plot()` cria gráficos direto de Series e DataFrames
- Próxima aula: especificação do projeto final