

Departamento de Engenharia de Sistemas e Computação

Introdução ao Pandas

DataFrames, leitura de arquivos, seleção e filtros

João Araujo

Algoritmos Computacionais I

Aula 28

Roteiro

- Por que Pandas? Tabelas de dados em Python
- `DataFrame` e `Series`
- Carregando dados: `read_csv`
- Conhecendo os dados: `shape`, `columns`, `dtypes`, `info`, `head`
- Selecionando colunas e linhas: `[ ]`, `loc`, `iloc`
- Filtros, novas colunas e ordenação

Baseado no notebook `notebooks/introducao_ao_pandas.ipynb`, que por sua vez segue *Pandas for Everyone* (D. Chen).

De listas de dicionários ao DataFrame

- Na Aula 20 modelamos uma tabela como **lista de dicionários**
- Funciona, mas cada operação (filtrar, agrupar, média) exige *laços* escritos por nós
- O **Pandas** oferece a tabela pronta: o `DataFrame`

Pandas

Biblioteca para análise de dados tabulares, construída sobre o NumPy. Instalação:

```
pip install pandas
```

```
import pandas as pd
```

Duas estruturas: `DataFrame` (tabela: linhas $\times$ colunas) e `Series` (uma coluna).

Carregando dados

Arquivo `dadosproducao.csv` (60 peças produzidas em 4 máquinas):

```
import pandas as pd

df = pd.read_csv("dadosproducao.csv")
print(df.head(3))
```

```
   id_peca maquina operador  turno         timestamp  tempo_ciclo_seg    status defeito
0     1000      M1      Ana  Noite  2026-08-03 06:04             31.6  Aprovado     NaN
1     1001      M1     Elis  Tarde  2026-08-03 06:07             34.5  Aprovado     NaN
2     1002      M2    Bruno  Noite  2026-08-03 06:18             24.9  Aprovado     NaN
```

`NaN` = valor ausente (célula vazia no CSV). Arquivos separados por tabulação (`.tsv`): `pd.read_csv(arq, sep='\t')`, como no *gapminder* do notebook.

Conhecendo os dados

```
print(df.shape)        # (60, 8)   atributo: SEM parênteses
print(df.columns)      # nomes das colunas
print(df.dtypes)       # tipo de cada coluna
df.info()              # resumo: colunas, tipos, não-nulos
print(df.describe())   # estatísticas das colunas numéricas
```

```
       id_peca  tempo_ciclo_seg
count    60.00            60.00
mean   1029.50            33.44
std      17.46             7.02
min    1000.00            20.90
50%    1029.50            31.00
...
```

- `shape`, `columns`, `dtypes` são *atributos*; `info()`, `head()`, `describe()` são *métodos* (com parênteses!)
- Cada coluna tem **um tipo**; cada linha pode misturar tipos (`int64`, `float64`, texto)

Selecionando colunas

```
maquinas = df["maquina"]              # UMA coluna -> Series
print(type(maquinas))                 # pandas Series
print(maquinas.head(3).tolist())      # ['M1', 'M1', 'M2']

sub = df[["maquina", "status"]]       # VÁRIAS colunas -> DataFrame
print(sub.head(2))                    # (lista dentro dos colchetes!)
```

- Uma coluna: `df["nome"]` devolve **Series**
- Várias: precisa de uma lista dentro dos colchetes (dois pares de colchetes) e devolve **DataFrame**
- `df[0]` dá erro: colchetes simples aceitam *nomes de colunas*, não posições

Selecionando linhas: `loc` e `iloc`

`.loc[ ]`

seleciona pelo **rótulo** (nome) do índice

```
df.loc[0]
df.loc[[0, 5, 59]]
df.loc[0, "maquina"]
```

`.iloc[ ]`

seleciona pela **posição** (número da linha)

```
df.iloc[0]
df.iloc[[0, 5, -1]]
df.iloc[0, 1]
```

```
print(df.iloc[-1])                    # última linha (iloc aceita -1)
print(df.loc[df.shape[0] - 1])        # equivalente, com loc
print(df.loc[:, ["id_peca", "tempo_ciclo_seg"]])  # todas as linhas
print(df.iloc[:5, :3])                # 5 primeiras linhas, 3 primeiras colunas
```

Sintaxe `df.loc[linhas, colunas]`; o `:` significa "todos" e funciona como o fatiamento de listas. Use **nomes** sempre que possível: o código fica legível.

Filtros booleanos (como na Aula 21)

```
retrabalho = df[df["status"] == "Retrabalho"]
print(len(retrabalho))                          # 11

lentas = df[df["tempo_ciclo_seg"] > 40]
m3_lentas = df[(df["maquina"] == "M3") & (df["tempo_ciclo_seg"] > 50)]
print(m3_lentas[["id_peca", "tempo_ciclo_seg"]])
```

```
    id_peca  tempo_ciclo_seg
46     1046            52.4
51     1051            52.3
```

- Condição → máscara `True`/`False` por linha → `df[máscara]` mantém as linhas verdadeiras
- Combine com `&` (e), `|` (ou), `~` (não), cada condição entre parênteses
- Também: `df["maquina"].isin(["M1", "M2"])`

Novas colunas e ordenação

```
df["tempo_min"] = df["tempo_ciclo_seg"] / 60          # coluna calculada
df["com_defeito"] = df["status"] != "Aprovado"        # coluna booleana

mais_lentas = df.sort_values("tempo_ciclo_seg", ascending=False)
print(mais_lentas[["id_peca", "maquina", "tempo_ciclo_seg"]].head(3))
```

```
    id_peca maquina  tempo_ciclo_seg
46     1046      M3             52.4
51     1051      M3             52.3
12     1012      M3             47.8
```

- Operações são *vetorizadas* (como no NumPy): nenhum `for` necessário
- `sort_values(coluna, ascending=False)` ordena; `df.drop(columns=[...])` remove colunas

Estatísticas rápidas de uma coluna

```
t = df["tempo_ciclo_seg"]
print(t.mean(), t.median(), t.std())      # 33.44  31.0  7.02
print(t.min(), t.max())                   # 20.9  52.4
print(df["status"].value_counts())
```

```
status
Aprovado      48
Retrabalho    11
Descarte       1
Name: count, dtype: int64
```

- `mean`, `median`, `std`, `min`, `max`, `sum`: como no NumPy
- `value_counts()` = tabela de frequências (o padrão "contar ocorrências" da Aula 20, pronto!)

Exemplo do notebook: *gapminder*

```
df = pd.read_csv("gapminder.tsv", sep="\t")
print(df.shape)                    # (1704, 6)
print(df.columns.tolist())
# ['country', 'continent', 'year', 'lifeExp', 'pop', 'gdpPercap']
print(df.loc[[0, 99, 999], ["country", "lifeExp", "gdpPercap"]])
print(df.iloc[:, :3].head(2))      # fatia de colunas por posição
```

- Dados históricos (Gapminder): expectativa de vida, população e PIB per capita por país e ano
- O notebook explora as variações de `loc`, `iloc`, `range` e fatias `[:, ::2]`: experimente cada uma!
- Pergunta do notebook: o que acontece se o `range` passa do número de colunas?

Atividade no laboratório

Com `dadosproducao.csv`:

1. Mostre o número de linhas e colunas e o tipo de cada coluna.
2. Selecione apenas `id_peca`, `operador` e `tempo_ciclo_seg`.
3. Liste as peças da máquina M2 produzidas no turno da manhã.
4. Quantas peças têm tempo de ciclo maior que a média? E qual a porcentagem?
5. Mostre as 5 peças mais rápidas.
6. Crie a coluna `tempo_min` e a coluna booleana `rapida` (tempo $< 30\,\text{s}$).

Para consultar e praticar

- Notebook: `notebooks/introducao_ao_pandas.ipynb`
- Documentação: `pandas.pydata.org/docs/`
- Guia *10 minutes to pandas*
- Livro de apoio: D. Chen, *Pandas for Everyone*

Dica

Em Jupyter, a última expressão da célula é exibida sozinha: um `df.head()` dispensa o `print`.

Resumo da aula

- `DataFrame` = tabela; `Series` = coluna
- `pd.read_csv` carrega; `shape`, `dtypes`, `info()`, `head()` exploram
- `df["col"]`, `df[["a","b"]]`, `loc` (rótulos) e `iloc` (posições)
- Filtros: `df[df.col > valor]`; combine com `&` e `|`
- Próxima aula: agrupamentos, valores ausentes e gráficos com Pandas