

Estudo de Python e dados II

Manipulação de dados: Preparação de dados, dados ausentes e Tidy data

Sergio Pedro Rodrigues Oliveira

07 janeiro 2026

SUMÁRIO

| | | |
|----------|--|-----------|
| 1 | Objetivo | 1 |
| 2 | Preparação dos dados | 2 |
| 2.1 | Introdução | 2 |
| | Mapa conceitual | 2 |
| | Objetivos | 2 |
| 2.2 | Tidy data | 3 |
| 2.2.1 | Combinando conjuntos de dados | 4 |
| 2.3 | Concatenação | 5 |
| 2.3.1 | Adicionando linhas | 5 |
| | 2.3.1.1 Concatenar Series | 8 |
| | 2.3.1.2 Ignorando o índice | 9 |
| 2.3.2 | Adicionando colunas | 10 |
| 2.3.3 | Concatenação com índices diferentes | 10 |
| | 2.3.3.1 Concatenando linhas com colunas diferentes | 10 |
| | 2.3.3.2 Concatenando colunas com linhas diferentes | 10 |
| 2.4 | Combinando vários conjuntos de dados | 11 |
| 2.5 | Conclusão | 11 |
| 3 | Dados Ausentes | 12 |
| 4 | Tidy data (dados organizados) | 12 |

LISTA DE FIGURAS

| | | |
|---|--|---|
| 1 | Preparação de dados, principais tópicos. | 2 |
|---|--|---|

LISTA DE TABELAS

1 Objetivo

O objetivo deste estudo é explorar e documentar as funcionalidades essenciais das principais bibliotecas científicas do Python, como **NumPy**, **Pandas** e outras, através de exemplos práticos e casos de uso selecionados. Pretende-se consolidar o conhecimento sobre a manipulação, análise e visualização de dados, servindo como um guia de referência pessoal para futuros projetos de programação científica.

2 Preparação dos dados

2.1 Introdução

A essa altura, você deverá ser capaz de carregar dados no **Pandas** e fazer algumas visualização básica. Essa parte do livro tem como foco várias tarefas de limpeza dos dados. Começaremos com a preparação de um conjunto de dados para análise por meio da combinação de diversos conjuntos.

Mapa conceitual

1. Conhecimento prévio
 - a) Carga de dados;
 - b) Obtenção de subconjuntos de dados;
 - c) Funções e métodos de classe.

Objetivos

Este capítulo abordará:

1. *Tidy data* (dados organizados);
2. Concatenação de dados;
3. Combinação (merge) de conjunto de dados.

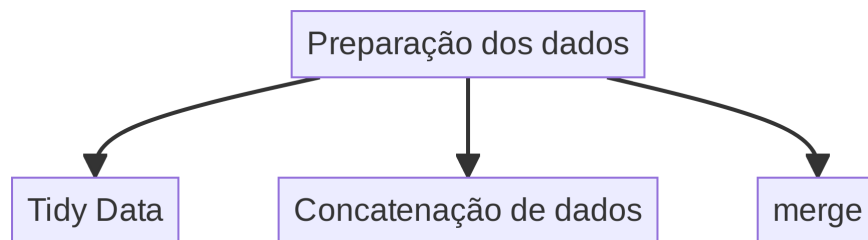


Figura 1: Preparação de dados, principais tópicos.

2.2 Tidy data

Hadley Wickham, um dos mais proeminentes membros da comunidade **R**, fala sobre a ideia de *tidy data* (dados organizados).

Com efeito, ele escreveu um artigo sobre esse conceito no *Journal of Statistical Software*. *Tidy data* é um framework para estruturar conjuntos de dados a fim de que sejam facilmente analisados. É usado principalmente como um objetivo a que devemos visar quando limpamos os dados. Depois que você compreender o que é o conceito de *tidy data*, esse conhecimento fará com que a coleta de dados seja muito mais fácil.

Então o que é *tidy data*? O artigo de Hadley Wickham o define como um conceito que atende aos seguintes critérios:

1. “Cada observação deve formar uma linha” (Observation)

Uma observação é o conjunto de todas as medidas feitas em uma única unidade (ex: uma pessoa em um exame, um país em um ano específico).

- O erro comum: Repetir a mesma observação em várias colunas ou espalhar os dados de uma mesma pessoa em tabelas diferentes sem necessidade.
- O modo Tidy: Se você está analisando a saúde de pacientes, cada linha deve representar um paciente em um momento específico.

2. “Cada variável deve formar uma coluna” (Variable)

Uma variável é um atributo que você mede (ex: Peso, Data, Temperatura).

- O erro comum: Ter colunas chamadas “Janeiro”, “Fevereiro” e “Março”. Aqui, o nome da variável é “Mês”, e Janeiro/Fevereiro são apenas valores.
- O modo Tidy: Criar uma coluna única chamada Mes onde os valores são listados.

3. “Cada tipo de unidade observacional forma uma tabela”

Esta regra foca na organização macro.

Sobre o terceiro critério do tidy data, as observações devem ser coerentes com a tabela, tornando-a objetiva quanto ao tipo de informação que deve armazenar. A ideia é que a tabela tenha um propósito único; ao misturar assuntos diferentes em uma mesma estrutura, fere-se a normalização dos dados.

- O erro comum: Misturar dados de “Clientes” com dados de “Vendas” na mesma tabela, causando redundância (ex: repetir o endereço do cliente toda vez que ele compra algo).
- O modo Tidy: Ter uma tabela para Clientes e outra para Vendas, relacionando-as por um ID. Isso facilita a manutenção e evita erros de digitação.

2.2.1 Combinando conjuntos de dados

Começaremos com o último critério de Hadley Wickham para *tidy data*: “cada tipo de unidade de observação forma uma tabela”.

Quando os dados estão organizados, é necessário combinar várias tabelas para responder a uma pergunta. Por exemplo, pode haver uma tabela separada que armazene informações de empresas e outra tabela contendo preços de ações. Se quisermos observar os preços de todas as ações no mercado de tecnologia, talvez antes tenhamos de encontrar todas as empresas de tecnologia na tabela de informações sobre empresas e então combinar esses dados com os preços das ações a fim de obter as informações de que precisamos para responder à nossa pergunta.

Os dados podem ter sido separados em tabelas distintas para reduzir a quantidade de informações redundantes (não precisamos armazenar informações sobre as empresas em cada entrada de preço das ações), mas essa organização implica que, como analistas de dados, teremos de combinar os dados relevantes por conta própria para responder à nossa pergunta.

Em outras ocasiões, um único conjunto de dados pode estar dividido em várias partes. Por exemplo, em dados de séries temporais, cada data pode estar em um arquivo separado. Em outro caso, um arquivo pode ter sido separado em partes para que os arquivos individuais fossem menores. Talvez você precise combinar dados de diversas origens para responder a uma pergunta (por exemplo, como combinar latitudes em longitudes com CEPs). Nos dois casos, você terá que combinar dados em um único dataframe de análise.

2.3 Concatenação

Uma das maneiras mais (conceitualmente) fáceis de combinar dados é por meio da concatenação.

Podemos pensar na concatenação como uma junção de linhas ou colunas em seus dados. Essa abordagem é possível se seus dados estiverem separados em partes ou se você fez um cálculo que queira concatenar ao seu conjunto de dados existente.

A concatenação é feita usando a função concat do **Pandas**.

2.3.1 Adicionando linhas

Vamos começar com alguns conjuntos de dados de exemplo para que você veja o que realmente acontece.

```
import pandas as pd

df1 = pd.read_csv("Cap_04-Preparacao_dados/01-Concatenacao/concat_1.csv")
df2 = pd.read_csv("Cap_04-Preparacao_dados/01-Concatenacao/concat_2.csv")
df3 = pd.read_csv("Cap_04-Preparacao_dados/01-Concatenacao/concat_3.csv")

print("dataframe_csv_1:")
print(df1)
print("dataframe_csv_2:")
print(df2)
print("dataframe_csv_3:")
print(df3)
```

```
dataframe_csv_1:
   A  B  C  D
0 a0 b0 c0 d0
1 a1 b1 c1 d1
2 a2 b2 c2 d2
3 a3 b3 c3 d3
dataframe_csv_2:
   A  B  C  D
0 a4 b4 c4 d4
1 a5 b5 c5 d5
2 a6 b6 c6 d6
3 a7 b7 c7 d7
dataframe_csv_3:
```

| | A | B | C | D |
|---|-----|-----|-----|-----|
| 0 | a8 | b8 | c8 | d8 |
| 1 | a9 | b9 | c9 | d9 |
| 2 | a10 | b10 | c10 | d10 |
| 3 | a11 | b11 | c11 | d11 |

A operação de empilhar dataframes uns sobre os outros é feita com a função `concat` do **Pandas**. Todos os dataframes a serem concatenados são passados em uma `list`.

```
row_concat = pd.concat([df1,df2,df3])
print(row_concat)
```

| | A | B | C | D |
|---|-----|-----|-----|-----|
| 0 | a0 | b0 | c0 | d0 |
| 1 | a1 | b1 | c1 | d1 |
| 2 | a2 | b2 | c2 | d2 |
| 3 | a3 | b3 | c3 | d3 |
| 0 | a4 | b4 | c4 | d4 |
| 1 | a5 | b5 | c5 | d5 |
| 2 | a6 | b6 | c6 | d6 |
| 3 | a7 | b7 | c7 | d7 |
| 0 | a8 | b8 | c8 | d8 |
| 1 | a9 | b9 | c9 | d9 |
| 2 | a10 | b10 | c10 | d10 |
| 3 | a11 | b11 | c11 | d11 |

Como podemos ver, `concat` empilha cegamente os dataframes. Se observar os nomes das linhas (isto é, seus índices), verá que eles são apenas uma versão empilhada dos índices originais das linhas.

Se aplicarmos os diversos métodos para obtenção de subconjuntos, os subconjuntos serão obtidos conforme esperado.

```
#Obtém o subconjunto da quarta linha do dataframe concatenado
print(row_concat.iloc[3,])
```

```
A    a3
B    b3
C    c3
D    d3
Name: 3, dtype: object
```

O que acontece se você usar `loc` para obter o subconjunto do novo dataframe?

A função `loc` pega os subconjuntos pelo rótulo (nome da linha), logo pega todos os rótulos com mesmo nome, pegando assim 3 linhas diferentes com mesmo nome.

```
print(row_concat.loc[3])
```

| | A | B | C | D |
|---|-----|-----|-----|-----|
| 3 | a3 | b3 | c3 | d3 |
| 3 | a7 | b7 | c7 | d7 |
| 3 | a11 | b11 | c11 | d11 |

2.3.1.1 Concatenar Series

Anteriormente apresentamos o processo de criar uma **Series**. No entanto, se criássemos uma nova série para concatenar em um dataframe, ela não seria concatenada corretamente.

2.3.1.2 Ignorando o índice

2.3.2 Adicionando colunas

2.3.3 Concatenação com índices diferentes

2.3.3.1 Concatenando linhas com colunas diferentes

2.3.3.2 Concatenando colunas com linhas diferentes

2.4 Combinando vários conjuntos de dados

2.5 Conclusão

3 Dados Ausentes

4 Tidy data (dados organizados)