Introdução às tabelas de dados em Python

As tabelas de dados que usaremos aqui correspondem à classe `DataFrame` do pacote `pandas`.

`pandas` é uma ferramenta para manipulação de dados em alto nível.

- A estrutura de dados principal desse pacote é o `DataFrame`, que se assemelha a uma tabela de um banco de dados:
- As colunas possuem dados de mesma natureza, e é implementada através da classe `Series` do pandas, que se assemelha ao array do `numpy`
- Cada linha corresponde a uma entrada de dados

Carregando um `DataFrame` de uma tabela de dados

Vamos carregar um arquivo de dados CSV da base online da UCI:

In [1]:

```
import pandas as pd

url = "http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
df = pd.read_csv(url)

# Mostra as primeiras linhas do data frame
df.head()
```

Out[1]:

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa

In [2]:

```
# Mostra as últimas linhas do data frame
df.tail()
```

Out[2]:

	5.1	3.5	1.4	0.2	Iris-setosa
144	6.7	3.0	5.2	2.3	Iris-virginica
145	6.3	2.5	5.0	1.9	Iris-virginica
146	6.5	3.0	5.2	2.0	Iris-virginica
147	6.2	3.4	5.4	2.3	Iris-virginica
148	5.9	3.0	5.1	1.8	Iris-virginica

O dataframe do pandas é composto por:

- Índice das linhas
- Índice das colunas, ou cabeçalho
- Colunas de dados homogêneos chamadas `Series`

O primeiro problema que podemos notar é que o cabeçalho do arquivo está errado. Geralmente um arquivo CSV contém o cabeçalho na primeira linha, porém neste caso a primeira linha já é uma linha de dados.

Para evitar esse problema, indicamos que não há cabeçalho na chamada do método `read_csv`:

In [3]:

```
df = pd.read_csv(url, header=None)

# Mostra as primeiras linhas do data frame
df.head()
```

Out[3]:

	0	1	2	3	4
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Acessando as informações sobre o `DataFrame`

In [4]:

```
# Dimensões do DataFrame
df.shape
```

Out[4]:

```
(150, 5)
```

In [5]:

```
# Índices das linhas
df.index
```

Out[5]:

```
RangeIndex(start=0, stop=150, step=1)
```

In [6]:

```
# Nomes das colunas
df.columns
```

Out[6]:

```
Int64Index([0, 1, 2, 3, 4], dtype='int64')
```

In [7]:

```
# Tipos das colunas - retorna uma Series
df.dtypes
```

Out[7]:

```
0    float64
1    float64
2    float64
3    float64
4     object
dtype: object
```

Obs: o tipo `object` pode guardar tipos variados de objetos do Python, incluindo strings.

Podemos renomear os índices de linhas ou colunas do `DataFrame` com o método `rename`

In [8]:

```
df.rename(columns={
    0: 'sepal_len', # antes: depois
    1: 'sepal_wid',
    2: 'petal_len',
    3: 'petal_wid',
    4: 'species'
}, inplace=True)

df.head()
```

Out[8]:

	sepal_len	sepal_wid	petal_len	petal_wid	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Acessando dados de um `DataFrame`

Há várias maneiras de acessar o conteúde de um `DataFrame`. Os mais simples são aqueles que usam a notação de colchetes. Primeiramente, podemos acessar uma coluna através do seu índice, retornando uma `Series`

In [9]:

```
df['petal_len'].head()
```

Out[9]:

```
0    1.4
1    1.4
2    1.3
3    1.5
4    1.4
Name: petal_len, dtype: float64
```

Por outro lado, se dentro dos colchetes passamos uma lista de nomes de coluna, o reultado é outro `DataFrame` contendo aquelas colunas. Isso vale inclusive para uma coluna simples:

In [10]:

```
# Mostra apenas a coluna petal_len
df[['petal_len']].head()
```

Out[10]:

	petal_len
0	1.4
1	1.4
2	1.3
3	1.5
4	1.4

In [11]:

```
# Mostra as colunas petal_len e petal_wid
df[['petal_len', 'petal_wid']].head()
```

Out[11]:

	petal_len	petal_wid
0	1.4	0.2
1	1.4	0.2
2	1.3	0.2
3	1.5	0.2
4	1.4	0.2

Caso dentro dos colchetes usamos a notação de slices, obtemos um `DataFrame` contendo o intervalo das observações (linhas) requisitadas.

In [12]:

```
# Mostra da quinta à decima observações
df[4:10]
```

Out[12]:

	sepal_len	sepal_wid	petal_len	petal_wid	species
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa

Podemos também usar as propriedades `loc` e `iloc` para selecionar qualquer trecho de dados que queiramos, de modo otimizado.

- `loc` é baseado em rótulos, de forma que precisamos especificar os rótulos das linhas e colunas que queremos capturar
- `iloc` é baseado em inteiros, de forma que precsamos especificar os índices numéricos

In [13]:

```
# Print out first 5 observations for sepal_wid
df.iloc[:5, 2]
```

Out[13]:

```
0    1.4
1    1.4
2    1.3
3    1.5
4    1.4
Name: petal_len, dtype: float64
```

In [14]:

```
# Print out observation for petal_wid
df.loc[:,'petal_wid']
```

Out[14]:

```
0      0.2
1      0.2
2      0.2
3      0.2
4      0.2
5      0.4
6      0.3
7      0.2
8      0.2
9      0.1
10     0.2
11     0.2
12     0.1
13     0.1
14     0.2
15     0.4
16     0.4
17     0.3
18     0.3
19     0.3
20     0.2
21     0.4
22     0.2
23     0.5
24     0.2
25     0.2
26     0.4
27     0.2
28     0.2
29     0.2
      ...
120    2.3
121    2.0
122    2.0
123    1.8
124    2.1
125    1.8
126    1.8
127    1.8
128    2.1
129    1.6
130    1.9
131    2.0
132    2.2
133    1.5
134    1.4
135    2.3
136    2.4
137    1.8
138    1.8
139    2.1
140    2.4
141    2.3
142    1.9
143    2.3
144    2.5
145    2.3
146    1.9
147    2.0
```

```
148    2.3
149    1.8
Name: petal_wid, Length: 150, dtype: float64
```

Exercícios:

1. Importar os dados do dataset *Breast Cancer Data Set*
2. Renomear as colunas de acordo com o arquivo breast-cancer.names
3. Mostrar o conteúdo das colunas age, menopause e tumor-size