Introdução ao módulo ***pandas***

Prof. Elias Paulino Medeiros

Instituto Federal do Ceará (IFCE)

Principais Características do **pandas**

É um pacote necessário para pesquisa científica em python;

Trabalha com arrays multidimensionais

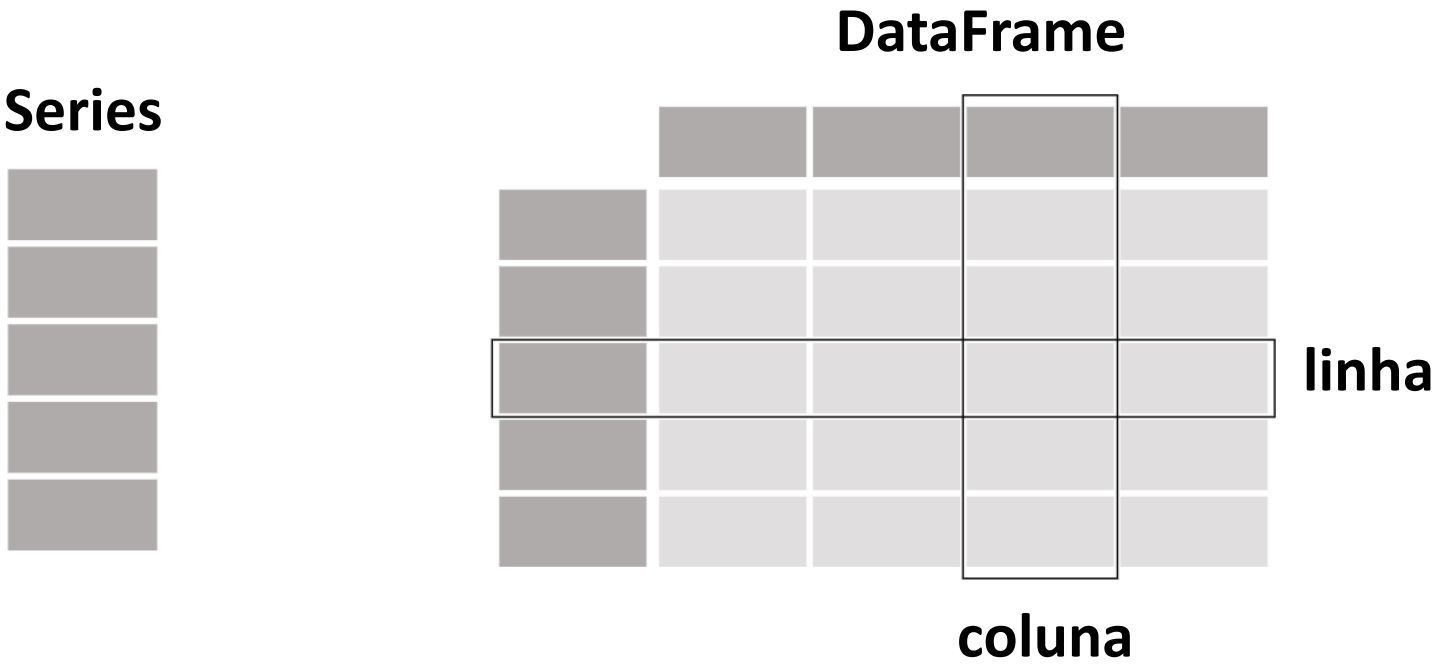
Armazenam valores de diferentes tipos

Utilização de rótulos nas colunas

Fácil manuseio de dados ausentes

Estrutura de dados

Dimensão	Nome	Descrição
1	Series	1-Dimensão; Tipo de dados homogêneo
2	DataFrame	Geralmente 2-Dimensões; Tipo de dados heterogêneo



Criando um DataFrame

```
In [1]: import pandas as pd
```

```
In [2]: df = pd.DataFrame(
   ...:     {
   ...:         "Name": [
   ...:             "Braund, Mr. Owen Harris",
   ...:             "Allen, Mr. William Henry",
   ...:             "Bonnell, Miss. Elizabeth",
   ...:         ],
   ...:         "Age": [22, 35, 58],
   ...:         "Sex": ["male", "male", "female"],
   ...:     }
   ...: )
   ...:
```

```
In [3]: df
Out[3]:
                        Name  Age     Sex
0   Braund, Mr. Owen Harris   22    male
1  Allen, Mr. William Henry   35    male
2  Bonnell, Miss. Elizabeth   58  female
```

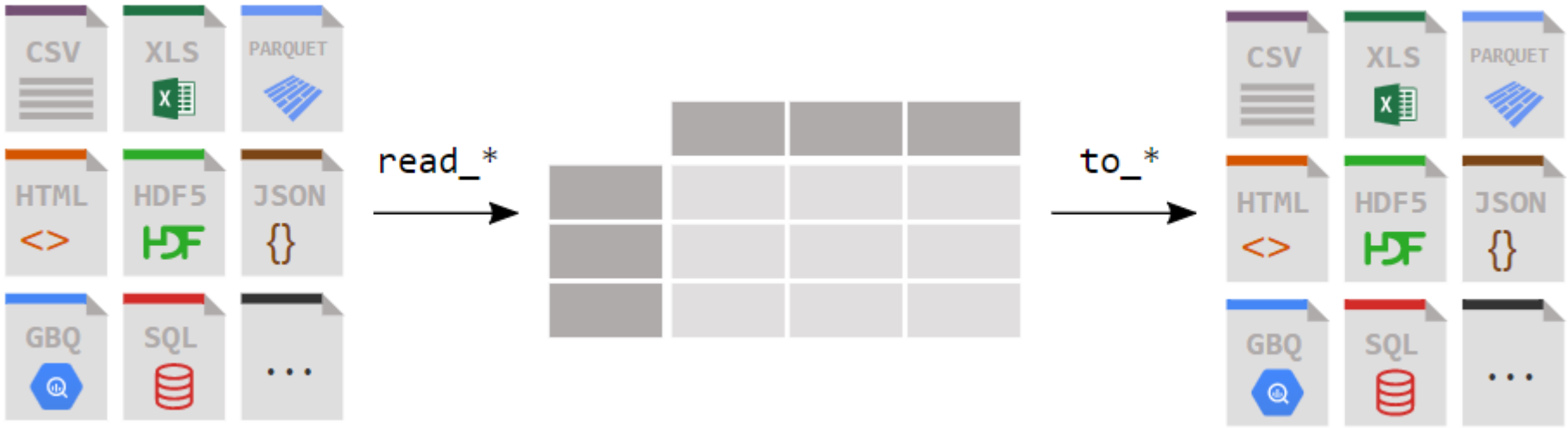
Criando uma Series

```
In [4]: df["Age"]
Out[4]:
0    22
1    35
2    58
Name: Age, dtype: int64
```

```
In [5]: ages = pd.Series([22, 35, 58], name="Age")

In [6]: ages
Out[6]:
0    22
1    35
2    58
Name: Age, dtype: int64
```

Lendo e escrevendo dados tabulares



Lendo dados tabulares

```
In [2]: titanic = pd.read_csv("data/titanic.csv")

In [3]: titanic

In [4]: titanic.head(8)

In [5]: titanic.dtypes
Out[5]:
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

Escrevendo dados tabulares

```
In [6]: titanic.to_excel("titanic.xlsx", sheet_name="passengers", index=False)
In [7]: titanic = pd.read_excel("titanic.xlsx", sheet_name="passengers")
```

```
In [9]: titanic.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

Fatiando DataFrames

Fatiando DataFrames

```
In [4]: ages = titanic["Age"]

In [5]: ages.head()
Out[5]:
0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
Name: Age, dtype: float64

In [6]: type(titanic["Age"])
Out[6]: pandas.core.series.Series

In [7]: titanic["Age"].shape
Out[7]: (891,)
```

```
In [8]: age_sex = titanic[["Age", "Sex"]]

In [9]: age_sex.head()
Out[9]:
    Age     Sex
0  22.0    male
1  38.0  female
2  26.0  female
3  35.0  female
4  35.0    male

In [10]: type(titanic[["Age", "Sex"]])
Out[10]: pandas.core.frame.DataFrame

In [11]: titanic[["Age", "Sex"]].shape
Out[11]: (891, 2)
```

Filtrando DataFrames

```
In [12]: above_35 = titanic[titanic["Age"] > 35]

In [13]: above_35.head()
```

```
In [14]: titanic["Age"] > 35
Out[14]:
0      False
1       True
2      False
3      False
4      False
       ...
886    False
887    False
888    False
889    False
890    False
Name: Age, Length: 891, dtype: bool
```

```
In [16]: class_23 = titanic[titanic["Pclass"].isin([2, 3])]

In [17]: class_23.head()
```

Filtrando DataFrames

```
In [16]: class_23 = titanic[titanic["Pclass"].isin([2, 3])]

In [17]: class_23.head()
In [18]: class_23 = titanic[(titanic["Pclass"] == 2) | (titanic["Pclass"] == 3)]
In [20]: age_no_na = titanic[titanic["Age"].notna()]
In [23]: adult_names = titanic.loc[titanic["Age"] > 35, "Name"]

In [24]: adult_names.head()
Out[24]:
1     Cumings, Mrs. John Bradley (Florence Briggs Th...
6                               McCarthy, Mr. Timothy J
11                             Bonnell, Miss. Elizabeth
13                          Andersson, Mr. Anders Johan
15                    Hewlett, Mrs. (Mary D Kingcome)
Name: Name, dtype: object
```

Filtrando DataFrames

```
In [25]: titanic.iloc[9:25, 2:5]
Out[25]:
    Pclass                                             Name     Sex
9        2  Nasser, Mrs. Nicholas (Adele Achem)  female
10       3      Sandstrom, Miss. Marguerite Rut  female
11       1             Bonnell, Miss. Elizabeth  female
12       3       Saundercock, Mr. William Henry    male
13       3          Andersson, Mr. Anders Johan    male
..     ...                                   ...     ...
20       2                Fynney, Mr. Joseph J    male
21       2               Beesley, Mr. Lawrence    male
22       3          McGowan, Miss. Anna "Annie"  female
23       1         Sloper, Mr. William Thompson    male
24       3        Palsson, Miss. Torborg Danira  female

[16 rows x 3 columns]
```

```
In [26]: titanic.iloc[0:3, 3] = "anonymous"
```

Criando novas a partir de colunas existentes

```
In [2]: air_quality = pd.read_csv("data/air_quality_no2.csv", index_col=0, parse_dates=True)

In [4]: air_quality["london_mg_per_cubic"] = air_quality["station_london"] * 1.882

In [6]: air_quality["ratio_paris_antwerp"] = (
   ...:     air_quality["station_paris"] / air_quality["station_antwerp"]
   ...: )
   ...:

In [8]: air_quality_renamed = air_quality.rename(
   ...:     columns={
   ...:         "station_antwerp": "BETR801",
   ...:         "station_paris": "FR04014",
   ...:         "station_london": "London Westminster",
   ...:     }
   ...: )
   ...:
```

Algumas funções

```
In [4]: titanic["Age"].mean()
Out[4]: 29.69911764705882
```

```
In [5]: titanic[["Age", "Fare"]].median()
Out[5]:
Age     28.0000
Fare    14.4542
dtype: float64
```

```
In [6]: titanic[["Age", "Fare"]].describe()
Out[6]:
              Age        Fare
count  714.000000  891.000000
mean    29.699118   32.204208
std     14.526497   49.693429
min      0.420000    0.000000
25%     20.125000    7.910400
50%     28.000000   14.454200
75%     38.000000   31.000000
max     80.000000  512.329200
```

Algumas funções

```
In [7]: titanic.agg(
   ...:     {
   ...:         "Age": ["min", "max", "median", "skew"],
   ...:         "Fare": ["min", "max", "median", "mean"],
   ...:     }
   ...: )
   ...:
Out[7]:
                Age         Fare
min        0.420000     0.000000
max       80.000000   512.329200
median    28.000000    14.454200
skew       0.389108          NaN
mean            NaN    32.204208
```

Agrupamentos

```
In [8]: titanic[["Sex", "Age"]].groupby("Sex").mean()
Out[8]:
              Age
Sex
female  27.915709
male    30.726645
```

```
In [9]: titanic.groupby("Sex").mean()
Out[9]:
        PassengerId  Survived    Pclass        Age     SibSp     Parch        Fare
Sex
female   431.028662  0.742038  2.159236  27.915709  0.694268  0.649682  44.479818
male     454.147314  0.188908  2.389948  30.726645  0.429809  0.235702  25.523893
```

```
In [10]: titanic.groupby("Sex")["Age"].mean()
Out[10]:
Sex
female    27.915709
male      30.726645
Name: Age, dtype: float64
```

Agrupamentos

```
In [11]: titanic.groupby(["Sex", "Pclass"])["Fare"].mean()
Out[11]:
Sex     Pclass
female  1         106.125798
        2          21.970121
        3          16.118810
male    1          67.226127
        2          19.741782
        3          12.661633
Name: Fare, dtype: float64
```

```
In [12]: titanic["Pclass"].value_counts()
Out[12]:
3    491
1    216
2    184
Name: Pclass, dtype: int64
```

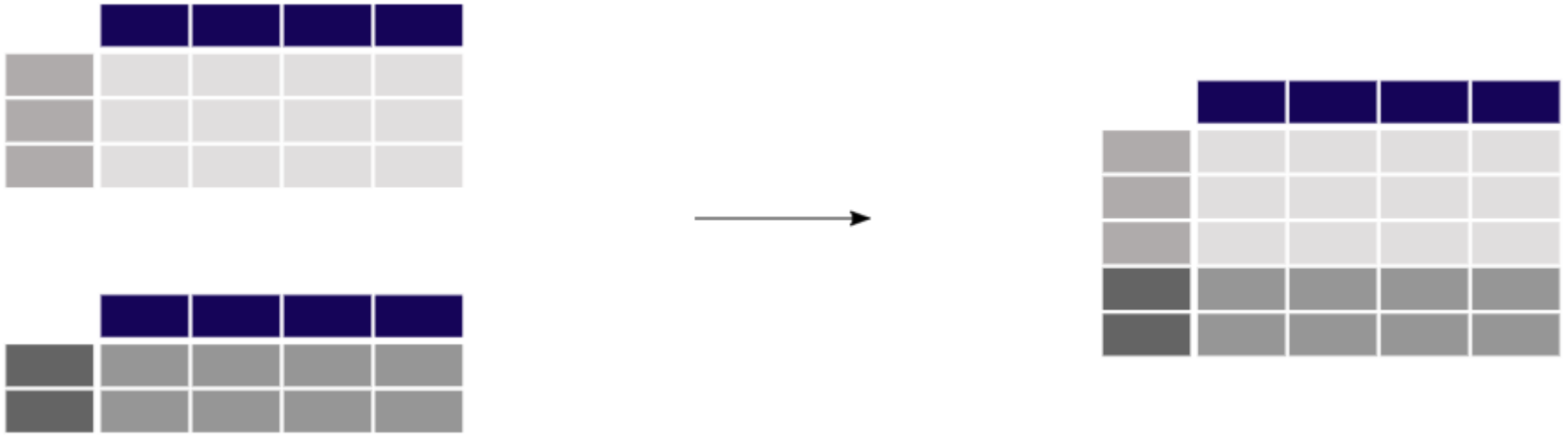
Agrupamentos

```
In [13]: titanic.groupby("Pclass")["Pclass"].count()
Out[13]:
Pclass
1    216
2    184
3    491
Name: Pclass, dtype: int64
```

Ordenando linhas do DataFrame

```
In [6]: titanic.sort_values(by="Age").head()
In [7]: titanic.sort_values(by=['Pclass', 'Age'], ascending=False).head()
```

Combinando DataFrames



Combinando DataFrames

```
In [2]: air_quality_no2 = pd.read_csv("data/air_quality_no2_long.csv",
   ...:                               parse_dates=True)
   ...:

In [3]: air_quality_no2 = air_quality_no2[["date.utc", "location",
   ...:                                    "parameter", "value"]]
   ...:

In [4]: air_quality_no2.head()
Out[4]:
                  date.utc location parameter  value
0  2019-06-21 00:00:00+00:00  FR04014       no2   20.0
1  2019-06-20 23:00:00+00:00  FR04014       no2   21.8
2  2019-06-20 22:00:00+00:00  FR04014       no2   26.5
3  2019-06-20 21:00:00+00:00  FR04014       no2   24.9
4  2019-06-20 20:00:00+00:00  FR04014       no2   21.4
```

Combinando DataFrames

```
In [5]: air_quality_pm25 = pd.read_csv("data/air_quality_pm25_long.csv",
   ...:                                parse_dates=True)
   ...:

In [6]: air_quality_pm25 = air_quality_pm25[["date.utc", "location",
   ...:                                      "parameter", "value"]]
   ...:

In [7]: air_quality_pm25.head()
Out[7]:
                  date.utc location parameter  value
0  2019-06-18 06:00:00+00:00  BETR801      pm25   18.0
1  2019-06-17 08:00:00+00:00  BETR801      pm25    6.5
2  2019-06-17 07:00:00+00:00  BETR801      pm25   18.5
3  2019-06-17 06:00:00+00:00  BETR801      pm25   16.0
4  2019-06-17 05:00:00+00:00  BETR801      pm25    7.5
```

Combinando DataFrames

```
In [8]: air_quality = pd.concat([air_quality_pm25, air_quality_no2], axis=0)

In [9]: air_quality.head()
Out[9]:
                   date.utc location parameter  value
0  2019-06-18 06:00:00+00:00  BETR801      pm25   18.0
1  2019-06-17 08:00:00+00:00  BETR801      pm25    6.5
2  2019-06-17 07:00:00+00:00  BETR801      pm25   18.5
3  2019-06-17 06:00:00+00:00  BETR801      pm25   16.0
4  2019-06-17 05:00:00+00:00  BETR801      pm25    7.5
```

Referências

- https://pandas.pydata.org/docs/getting_started/intro_tutorials/index.html
- https://github.com/pandas-dev/pandas/tree/master/doc/data