Introduction to Data Science

with Python

Alexis Bogroff

May 15, 2022

Presenter

Alexis Bogroff
Lecturer and Mentor in Data Science
at Paris 1 Panthéon-Sorbonne, ESILV,
Openclassrooms, EM-Lyon

Presenter

Alexis Bogroff
Lecturer and Mentor in Data Science
at Paris 1 Panthéon-Sorbonne, ESILV,
Openclassrooms, EM-Lyon

- 4 years Teaching Assistant and lecturer in VBA, Python for finance, SQL, Data Analysis and Data Science
- 9 months Researcher Assistant at Paris 1 Panthéon-Sorbonne within H2020 European Project
- 1 year Data Scientist at Pléiade Asset Management

Pandas

- Objects
  - pd.DataFrame
  - pd.Series
- Masks / Filters
- Basic methods (max, info, describe)
- Apply, vectorial operations
- Other useful methods
  - str
  - drop
  - sort_values
  - value_counts
  - groupby
  - .isna, .fillna
  - Graphs (.plot, .scatter.plot, .plot.bar, .hist)
- merging DataFrames the right method (outer, indicator=True)
- Pandas profiling

Pandas - Series

- Advanced named list
- Named index
- Useful for Time Series

```
s = pd.Series({'a': [1.5, 2.2, 4.9, 3.4, 5.9]})
s
```

✓ 0.4s

```
a    [1.5, 2.2, 4.9, 3.4, 5.9]
dtype: object
```

Pandas - DataFrame

- DataFrame $\pm$ Table
- Advanced dictionnary
- Columns: Series

```
students = {
    'names': ['ali', 'zoe', 'moh'],
    'age': [10, 12, 9],
}

df = pd.DataFrame(students)
```

✓ 0.3s

```
df
```

✓ 0.2s

	names	age
0	ali	10
1	zoe	12
2	moh	9

Pandas - DataFrame

- Extract column

```
df['age']
✓ 0.3s
```

```
stu_1    10
stu_2    12
stu_3     9
Name: age, dtype: int64
```

```
df['names']
✓ 0.2s
```

```
stu_1    ali
stu_2    zoe
stu_3    moh
Name: names, dtype: object
```

Pandas - DataFrame

- Extract specific row
- iloc vs loc

Add index names for the purpose of this exercise

```
df.index = ['stu_1', 'stu_2', 'stu_3']
```

```
df
```

✓ 0.2s

	names	age
stu_1	ali	10
stu_2	zoe	12
stu_3	moh	9

```
df.iloc[0]
```

✓ 0.2s

```
names    ali
age       10
Name: stu_1, dtype: object
```

```
df.iloc[-1]
```

✓ 0.3s

```
names    moh
age        9
Name: stu_3, dtype: object
```

```
df.loc['stu_2']
```

✓ 0.2s

```
names    zoe
age       12
Name: stu_2, dtype: object
```

Pandas - DataFrame

- Extract multiple rows
- Masks (filters)
- Arrays of booleans

```
df
```
✓ 0.2s

	names	age
stu_1	ali	10
stu_2	zoe	12
stu_3	moh	9

```
mask_age = df['age'] >= 10
mask_age
```
✓ 0.3s

```
stu_1     True
stu_2     True
stu_3    False
Name: age, dtype: bool
```

```
df[mask_age]
```
✓ 0.2s

	names	age
stu_1	ali	10
stu_2	zoe	12

Pandas - DataFrame basic methods

- Methods
  - Max

```
df
```
✓ 0.2s

	names	age
stu_1	ali	10
stu_2	zoe	12
stu_3	moh	9

```
df.max()
```
✓ 0.3s

```
names    zoe
age       12
dtype: object
```

```
df['age'].max()
```
✓ 0.2s

```
12
```

Pandas - DataFrame basic methods

- Methods
  - Shape

```
df
```

✓ 0.2s

	names	age
stu_1	ali	10
stu_2	zoe	12
stu_3	moh	9

```
df.shape
```

✓ 0.3s

```
(3, 2)
```

Pandas - DataFrame basic methods

- Methods
  - Info

```
df
```

✓ 0.2s

	names	age
stu_1	ali	10
stu_2	zoe	12
stu_3	moh	9

```
df.info()
```

✓ 0.2s

```
<class 'pandas.core.frame.DataFrame'>
Index: 3 entries, stu_1 to stu_3
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   names   3 non-null      object
 1   age     3 non-null      int64
dtypes: int64(1), object(1)
memory usage: 180.0+ bytes
```

Pandas - DataFrame basic methods

- Methods
  - Describe

```
df
```

✓ 0.2s

	names	age
stu_1	ali	10
stu_2	zoe	12
stu_3	moh	9

```
df.describe()
```

✓ 0.2s

	age
count	3.000000
mean	10.333333
std	1.527525
min	9.000000
25%	9.500000
50%	10.000000
75%	11.000000
max	12.000000

Pandas - DataFrame "apply" method

- Apply
  - Use apply, do not use loops
  - It applies a function to each row
  - All DataFrame or specific columns
  - Use lambda to create temporary functions

```
def double(x):
    return x * 2
```
✓ 0.3s

```
df['age'].apply(double)
```
✓ 0.3s

```
stu_1    20
stu_2    24
stu_3    18
Name: age, dtype: int64
```

```
df
```
✓ 0.2s

	names	age
stu_1	ali	10
stu_2	zoe	12
stu_3	moh	9

```
df.apply(lambda x: x*2)
```
✓ 0.3s

	names	age
stu_1	aliali	20
stu_2	zoezoe	24
stu_3	mohmoh	18

```
df['age'].apply(lambda x: x*2)
```
✓ 0.2s

```
stu_1    20
stu_2    24
stu_3    18
Name: age, dtype: int64
```

Pandas - DataFrame useful methods

- Modify text
  - Access strings methods .str

```
df
```

✓ 0.2s

	names	age
stu_1	ali	10
stu_2	zoe	12
stu_3	moh	9

```
df['names'].str.upper()
```

✓ 0.2s

```
stu_1    ALI
stu_2    ZOE
stu_3    MOH
Name: names, dtype: object
```

Pandas - DataFrame useful methods

- Drop column
  - No effect on current DataFrame
  - *"View"* of the DataFrame

```
df
✓ 0.2s
```

	names	age
stu_1	ali	10
stu_2	zoe	12
stu_3	moh	9

```
df.drop(columns='names')
[338] ✓ 0.3s
```

	age
stu_1	10
stu_2	12
stu_3	9

```
df
[339] ✓ 0.2s
```

	names	age
stu_1	ali	10
stu_2	zoe	12
stu_3	moh	9

Pandas - DataFrame useful methods

- Drop column
  - Modify current DataFrame (method 1)
  - Standard variable assignment

```
df
```
✓ 0.2s

	names	age
stu_1	ali	10
stu_2	zoe	12
stu_3	moh	9

```
df_temp = df.copy()
```
✓ 0.2s

```
df_temp = df_temp.drop(columns='names')
```
✓ 0.2s

```
df_temp
```
✓ 0.2s

	age
stu_1	10
stu_2	12
stu_3	9

Pandas - DataFrame useful methods

- Drop column
  - Modify current DataFrame (method 2)
  - Inplace parameter

```
df
```
✓ 0.2s

	names	age
stu_1	ali	10
stu_2	zoe	12
stu_3	moh	9

```
df_temp = df.copy()
```
✓ 0.2s

```
df_temp.drop(columns='names', inplace=True)
```
✓ 0.2s

```
df_temp
```
✓ 0.2s

	age
stu_1	10
stu_2	12
stu_3	9

Pandas - DataFrame useful methods

- Sort values
  - Select the key column with "*by*" parameter
  - Choose ordering direction with "*ascending*" parameter

```
df
```
✓ 0.2s

	names	age
stu_1	ali	10
stu_2	zoe	12
stu_3	moh	9

```
df.sort_values(by='age')
```
✓ 0.7s

	names	age
stu_3	moh	9
stu_1	ali	10
stu_2	zoe	12

```
df.sort_values(by='age', ascending=False)
```
✓ 0.4s

	names	age
stu_2	zoe	12
stu_1	ali	10
stu_3	moh	9

Pandas - DataFrame useful methods

- Generate table of values counts

```
df_temp
```
✓ 0.4s

	names	age
stu_1	ali	10
stu_2	zoe	12
stu_3	moh	9
stu_4	jul	10
stu_5	lipo	10
stu_6	chen	12

```
df_temp['age'].value_counts()
```
✓ 0.1s

```
10    3
12    2
9     1
Name: age, dtype: int64
```

Pandas - DataFrame useful methods

- Group values using groupby
- Define an aggregation method
  - mean
  - median
  - max
  - etc.

```
df_temp
```
✓ 0.6s

	names	age	centroid
stu_1	ali	10	1
stu_2	zoe	12	3
stu_3	moh	9	3
stu_4	jul	10	2
stu_5	lipo	10	3
stu_6	chen	12	1

```
df_temp.groupby('centroid').mean()
```
✓ 0.4s

	age
centroid	
1	11.000000
2	10.000000
3	10.333333

Pandas - DataFrame useful methods

- Detect NA values
- Replace (impute) NA values
- Presence of NA casts column as float
- Methods:
  - bfill, ffill
  - constant (median, etc.)

```
df_temp.isna()
```
✓ 0.3s

	names	age	centroid
stu_1	False	False	False
stu_2	False	True	False
stu_3	False	False	False
stu_4	False	True	False
stu_5	False	False	False
stu_6	False	False	False

```
df_temp['age'].fillna(10)
```
✓ 0.5s

```
stu_1    10.0
stu_2    10.0
stu_3     9.0
stu_4    10.0
stu_5    10.0
stu_6    12.0
Name: age, dtype: float64
```

```
df_temp
```
✓ 0.5s

	names	age	centroid
stu_1	ali	10.0	1
stu_2	zoe	NaN	3
stu_3	moh	9.0	3
stu_4	jul	NaN	2
stu_5	lipo	10.0	3
stu_6	chen	12.0	1

Pandas - Graphs

- Generate graphs
  - Line chart

```
df_graph['size'] = [100, 140, 70]
```
✓ 0.2s

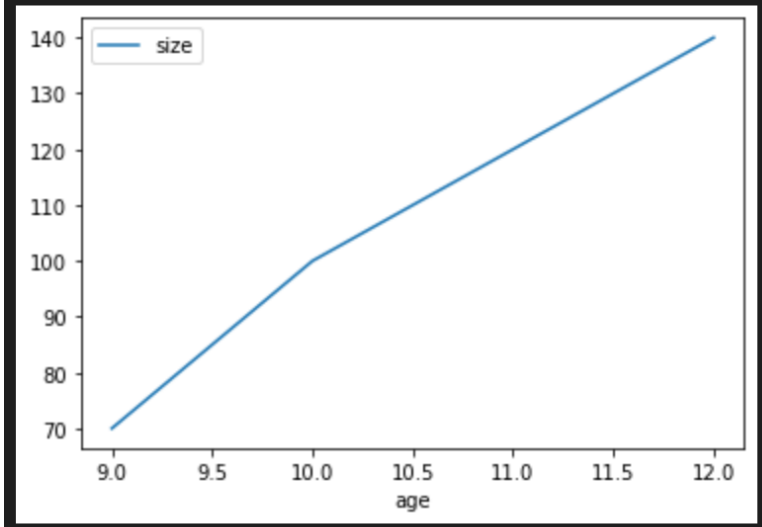
```
df_graph.sort_values('age', inplace=True)
```
✓ 0.2s

```
df_graph
```
✓ 0.2s

	names	age	size
0	moh	9	70
1	ali	10	100
2	zoe	12	140
3	mat	12	100

```
df_graph.plot(x='age', y='size')
```
✓ 0.9s

```
<AxesSubplot:xlabel='age'>
```



Pandas - Graphs

- Generate graphs
  - Scatter plot

```
df_graph
```

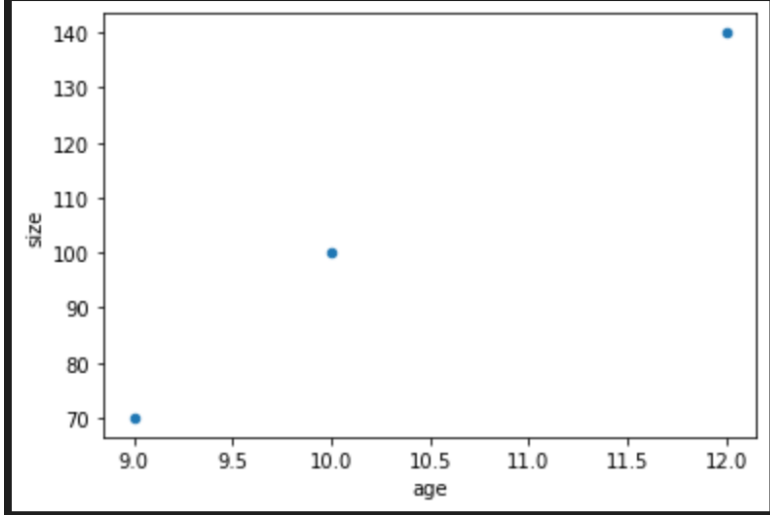
✓ 0.2s

	names	age	size
0	moh	9	70
1	ali	10	100
2	zoe	12	140
3	mat	12	100

```
df_graph.plot.scatter(x='age', y='size')
```

✓ 0.8s

```
<AxesSubplot:xlabel='age', ylabel='size'>
```



Pandas - Graphs

- Generate graphs
  - Histogram

```
df_graph
```

✓ 0.2s

	names	age	size
0	moh	9	70
1	ali	10	100
2	zoe	12	140
3	mat	12	100

```
stu_4 = {'names': 'mat', 'age': 12, 'size': 100}
df_graph = df_graph.append(stu_4, ignore_index=True)
```

✓ 0.2s

```
df_graph['age'].hist()
```

✓ 0.8s

```
<AxesSubplot:>
```