

Data Science & ML Course

Lesson #4 [Part #2] Introduction to Pandas

Ivanovitch Silva
October, 2018



Agenda

- About the two core pandas types: **dataframes** and **series**
- How to **select** data using row and column labels
- A variety of methods for **exploring data** with pandas
- How to **assign** data using various techniques in pandas
- How to use **boolean indexing** with pandas for selection and assignment

Update from repository

```
git clone https://github.com/ivanovitchm/datascience2machinelearning.git
```

Or

```
git pull
```



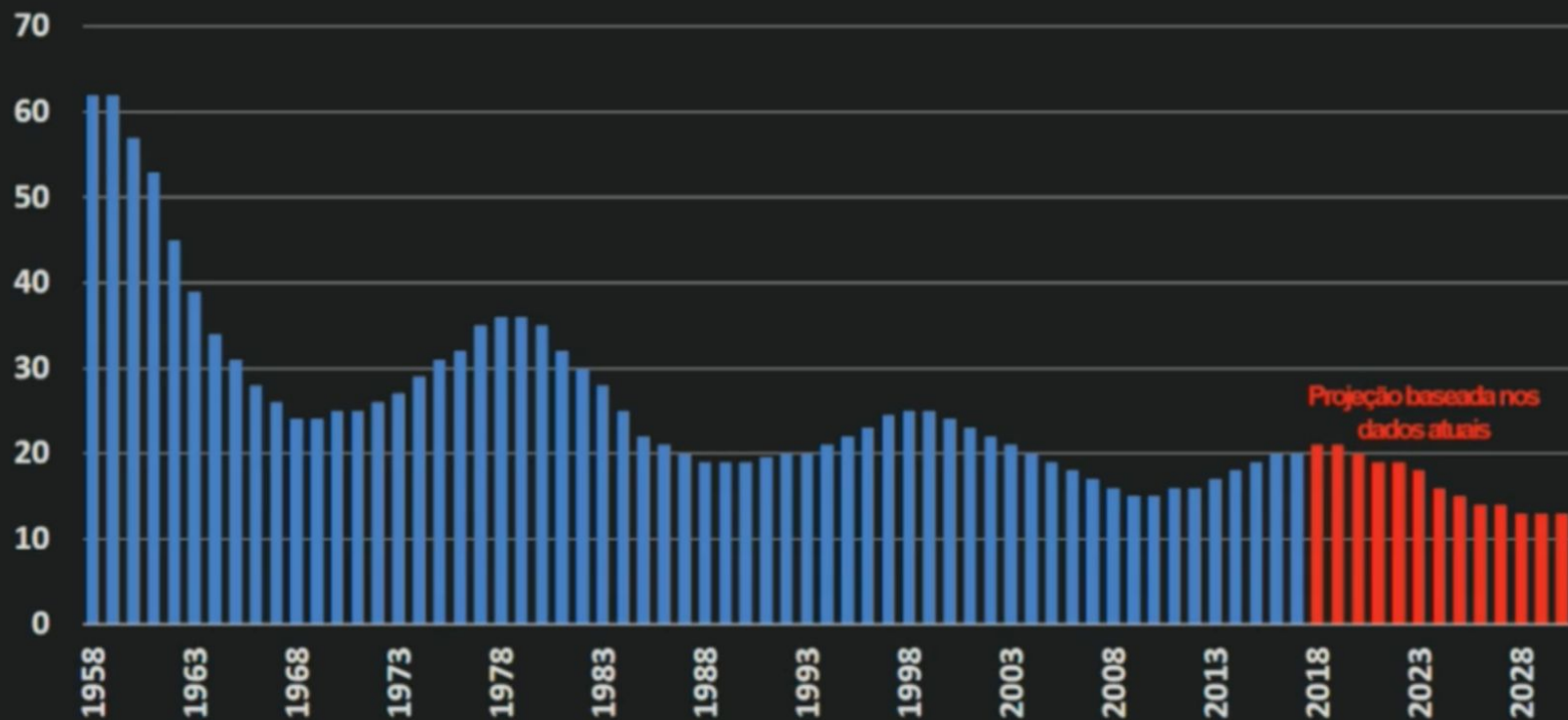
Understanding Pandas & Numpy

- Numpy
 - a. Lack support for column names
 - b. Support for only one data type per ndarray
 - c. There are lots of low level methods, however there are many common analysis patterns that don't have pre-built methods.

The pandas library provides solutions to all of these pain points and more. Pandas is not so much a replacement for NumPy as an **extension of NumPy**.



Tempo médio de permanência de uma empresa no S&P 500 (em anos)



Fonte: INNOSIGHT, Richard N. Foster, Standard & Poor's

UMA EMPRESA DO S&P 500 ESTÁ SENDO SUBSTITUÍDA A CADA DUAS SEMANAS

Richard Foster

The dataset

	rank	revenues	revenue_change	profits	assets	profit_change	ceo	industry	sector	previous_rank
Walmart	1	485873	0.8	13643.0	198825	-7.2	C. Douglas McMillon	General Merchandisers	Retailing	1
State Grid	2	315199	-4.4	9571.3	489838	-6.2	Kou Wei	Utilities	Energy	2
Sinopec Group	3	267518	-9.1	1257.9	310726	-65.0	Wang Yupu	Petroleum Refining	Energy	4
China National Petroleum	4	262573	-12.3	1867.5	585619	-73.7	Zhang Jianhua	Petroleum Refining	Energy	3
Toyota Motor	5	254694	7.7	16899.3	437575	-12.3	Akio Toyoda	Motor Vehicles and Parts	Motor Vehicles & Parts	8

```
import pandas as pd
f500 = pd.read_csv("f500.csv", index_col=0)
f500.index.name = None
```


Introducing Dataframes

The diagram illustrates a DataFrame structure with the following components:

- Column Labels:** rank, revenues, profits, country
- Row Labels:** Walmart, State Grid, Sinopec Group, China Natural Petroleum, Toyota Motor
- Index Axis:** Indicated by a red arrow pointing downwards.
- Column Axis:** Indicated by a red arrow pointing to the right.
- Data Types:**
 - Integer Type:** Indicated by green arrows pointing to the 'rank' and 'revenues' columns.
 - Float Type:** Indicated by a green arrow pointing to the 'profits' column.
 - String Type:** Indicated by green arrows pointing to the 'country' and 'Row Labels'.

	rank	revenues	profits	country
Walmart	1	485873	13643.0	USA
State Grid	2	315199	9571.3	China
Sinopec Group	3	267518	1257.9	China
China Natural Petroleum	4	262573	1867.5	China
Toyota Motor	5	254694	16899.3	Japan

Introducing Dataframes

```
# put your code here  
f500.info(,)
```

```
<class 'pandas.core.frame.DataFrame'>  
Index: 500 entries, Walmart to AutoNation  
Data columns (total 16 columns):  
rank                500 non-null int64  
revenues            500 non-null int64  
revenue_change      498 non-null float64  
profits             499 non-null float64  
assets              500 non-null int64  
profit_change       436 non-null float64  
ceo                  500 non-null object  
industry            500 non-null object  
sector              500 non-null object  
previous_rank       500 non-null int64  
country             500 non-null object  
hq_location         500 non-null object  
website             500 non-null object  
years_on_global_500_list 500 non-null int64  
employees           500 non-null int64  
total_stockholder_equity 500 non-null int64  
dtypes: float64(3), int64(7), object(6)  
memory usage: 66.4+ KB
```

f500.head()
f500.tail()

Selecting Columns From a Dataframe by label

		rank	revenues	profits	country
	Walmart	1	485873	13643.0	USA
	State Grid	2	315199	9571.3	China
f500_selection	Sinopec Group	3	267518	1257.9	China
	China Natural Petroleum	4	262573	1867.5	China
	Toyota Motor	5	254694	16899.3	Japan

```
f500_selection.loc[:, "rank"]
```

Walmart	1
State Grid	2
Sinopec Group	3
China Natural Petroleum	4
Toyota Motor	5

Selecting Columns From a Dataframe by label

`f500_selection`

	rank	revenues	profits	country
Walmart	1	485873	13643.0	USA
State Grid	2	315199	9571.3	China
Sinopec Group	3	267518	1257.9	China
China Natural Petroleum	4	262573	1867.5	China
Toyota Motor	5	254694	16899.3	Japan

`f500_selection.loc[:, ["country", "rank"]]`

	country	rank
Walmart	USA	1
State Grid	China	2
Sinopec Group	China	3
China Natural Petroleum	China	4
Toyota Motor	Japan	5

Selecting Columns From a Dataframe by label

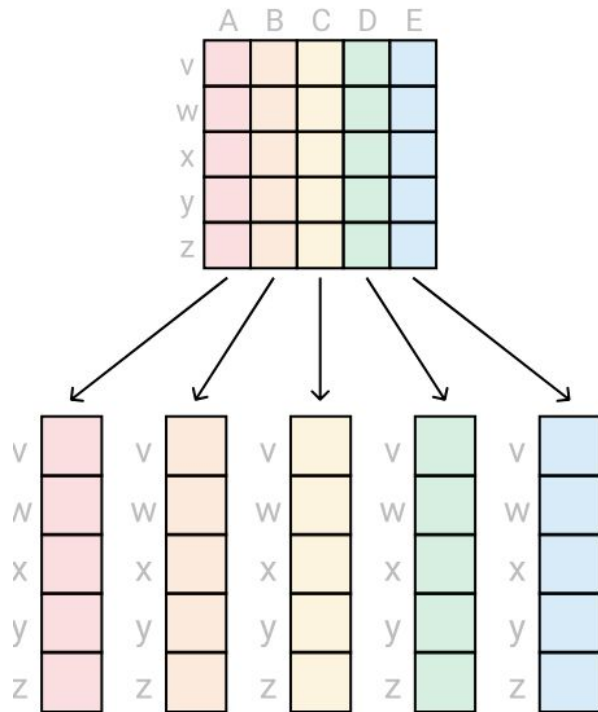
		rank	revenues	profits	country
f500_selection	Walmart	1	485873	13643.0	USA
	State Grid	2	315199	9571.3	China
	Sinopec Group	3	267518	1257.9	China
	China Natural Petroleum	4	262573	1867.5	China
	Toyota Motor	5	254694	16899.3	Japan

		rank	revenues	profits
f500_selection.loc[:, "rank": "profits"]	Walmart	1	485873	13643.0
	State Grid	2	315199	9571.3
	Sinopec Group	3	267518	1257.9
	China Natural Petroleum	4	262573	1867.5
	Toyota Motor	5	254694	16899.3

Column selection shortcuts

Select by Label	Explicit Syntax	Common Shorthand	Other Shorthand
Single column	<code>df.loc[:, "col1"]</code>	<code>df["col1"]</code>	<code>df.col1</code>
List of columns	<code>df.loc[:, ["col1", "col7"]]</code>	<code>df[["col1", "col7"]]</code>	
Slice of columns	<code>df.loc[:, "col1":"col4"]</code>		

Selecting Items from a Series by Label



Original
Dataframe

	A	B	C	D	E
V					
W					
X					
Y					
Z					

Code

```
single_col = df["D"]
```

Result

V	
W	
X	
Y	
Z	

single_col is a
series object

	A	B	C	D	E
V					
W					
X					
Y					
Z					

```
single_row = df.head(1)
```

A	
B	
C	
D	
E	

single_row is a
series object

Original
Dataframe

	A	B	C	D	E
V					
W					
X					
Y					
Z					

Code

```
multi_cols = df[["A", "C", "D"]]
```

Result

	A	C	D
V			
W			
X			
Y			
Z			

multi_cols is a
dataframe object

	A	B	C	D	E
V					
W					
X					
Y					
Z					

```
multi_rows = df.head(3)
```

	A	B	C	D	E
V					
W					
X					

multi_rows is a
dataframe object

Dataframe vs Series

	Series	DataFrame
Dimensions	One	Two
Has 'index' axis	Yes	Yes
Has 'columns' axis	No	Yes
Number of dtypes	One	Many (one per column)

Series and Dataframe Describe Methods

```
revs = f500["revenues"]  
print(revs.describe())
```

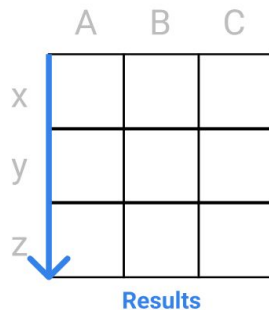
```
count      500.000000  
mean       55416.358000  
std        45725.478963  
min        21609.000000  
25%        29003.000000  
50%        40236.000000  
75%        63926.750000  
max        485873.000000  
Name: revenues, dtype: float64
```

```
print(f500["assets"].describe())
```

```
count      5.000000e+02  
mean       2.436323e+05  
std        4.851937e+05  
min        3.717000e+03  
25%        3.658850e+04  
50%        7.326150e+04  
75%        1.805640e+05  
max        3.473238e+06  
Name: assets, dtype: float64
```

`DataFrame.method(axis=0)`
or
`DataFrame.method(axis="index")`

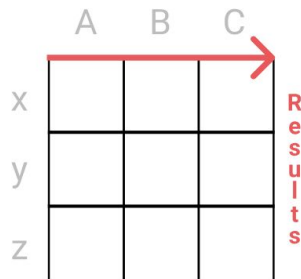
Calculates along the **row** axis



Calculates result for
each **column**.

`DataFrame.method(axis=1)`
or
`DataFrame.method(axis="column")`

Calculates along the **column** axis



Calculates result for
each **row**.

More data exploration methods

```
medians = f500[["revenues", "profits"]].median(axis=0)
# we could also use .median(axis="index")
print(medians)
```

```
revenues    40236.0
profits      1761.6
dtype: float64
```

```
>>> print(top5_rank_revenue)
```

	rank	revenues
Walmart	1	485873
State Grid	2	315199
Sinopec Group	3	267518
China National Petroleum	4	262573
Toyota Motor	5	254694

```
>>> top5_rank_revenue["revenues"] = 0
```

```
>>> print(top5_rank_revenue)
```

	rank	revenues
Walmart	1	0
State Grid	2	0
Sinopec Group	3	0
China National Petroleum	4	0
Toyota Motor	5	0

Assignment with Pandas

Assignment with Pandas

```
>>> top5_rank_revenue.loc["Sinopec Group", "revenues"] = 999
```

```
>>> print(top5_rank_revenue)
```

	rank	revenues
Walmart	1	0
State Grid	2	0
Sinopec Group	3	999
China National Petroleum	4	0
Toyota Motor	5	0

Add a new column

```
>>> top5_rank_revenue["year_founded"] = 0
```

```
>>> print(top5_rank_revenue)
```

	rank	revenues	year_founded
Walmart	1	0	0
State Grid	2	0	0
Sinopec Group	3	999	0
China National Petroleum	4	0	0
Toyota Motor	5	0	0

Add a new row

```
>>> top5_rank_revenue.loc["My New Company"] = 555
```

```
>>> print(top5_rank_revenue)
```

	rank	revenues	year_founded
Walmart	1	0	0
State Grid	2	0	0
Sinopec Group	3	999	0
China National Petroleum	4	0	0
Toyota Motor	5	0	0
My New Company	555	555	555

Using boolean indexing with pandas objects

w	2
x	4
y	6
z	8

pandas series



w	2	< 5
x	4	< 5
y	6	< 5
z	8	< 5

Vectorized
boolean operation



w	True
x	True
y	False
z	False

Boolean
pandas series

	A	B
w	2	3
x	4	6
y	6	9
z	8	12

pandas DataFrame



	A	B	
w	2	3	< 5
x	4	6	< 5
y	6	9	< 5
z	8	12	< 5

Vectorized
boolean operation



	A	B
w	True	True
x	True	False
y	False	False
z	False	False

Boolean
pandas DataFrame

Using boolean indexing with pandas objects

```
result = df.loc[num_bool, "name"]
```

		name		
False	w	Kylie	12	
True	→ x	Rahul	8	→ x
False	y	Michael	5	z
True	→ z	Sarah	8	↗
				result
				Rahul
				Sarah

```
result = df[num_bool]
```

		name	num	
False	w	Kylie	12	
True	→ x	Rahul	8	→ x
False	y	Michael	5	z
True	→ z	Sarah	8	↗
				result
				name num
				Rahul 8
				Sarah 8

Using boolean arrays to assign values

```
f500.loc[f500["sector"] == "Motor Vehicles & Parts", "sector"] = "Motor Vehicles and Parts"
```

Challenge

Finding top performers by country

```
>>> top_3_countries = f500["country"].value_counts().head(3)
```

```
>>> print(top_3_countries)
```

```
USA      132
```

```
China    109
```

```
Japan     51
```

```
Name: country, dtype: int64
```

Lesson_04 Part #2

Section 1.12



Exploring Data With Pandas

Agenda

- Select columns, rows and individual items using their integer location.
- Work with integer axis labels.
- How to use pandas methods to produce boolean arrays.
- Use boolean operators to combine boolean comparisons to perform more complex analysis.
- Use index labels to align data.
- Use aggregation to perform advanced analysis using loops.

Introduction (Pandas vs Numpy)


```
ndarray[2,0]
```

located at row 2, column 0


```
ndarray[1]
```

located at row 1

Numpy

	A	B	C
x			
y			
z			

```
df.loc["z","A"]
```

*located at row with label z,
column with label A*

	A	B	C
x			
y			
z			

```
df.loc["y"]
```

located at row with label y

Pandas

Using iloc to select by integer position

	A	B	C
x			
y			
z			

```
df.iloc[2,0]
```

	A	B	C
x			
y			
z			

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

```
first_column = f500.iloc[:,0]
print(first_column)
```

```
0           Walmart
1       State Grid
2       Sinopec Group
...
497  Wm. Morrison Supermarkets
498           TUI
499       AutoNation
Name: company, dtype: object
```

Slicing with iloc

With `loc[]`, the ending slice is **included**.

With `iloc[]`, the ending slice is **not included**.

```
1 | f500[1:4].
```

	rank	revenues
State Grid	2	315199
Sinopec Group	3	267518
China National Petroleum	4	262573

```
1 | f500.iloc[1:4].
```

	rank	revenues
State Grid	2	315199
Sinopec Group	3	267518
China National Petroleum	4	262573

Loc vs iLoc

`df.iloc[1]`

`iloc[1]` uses the integer position of the row to select the second row

	A	B	C
0			
1			
2			

`df.iloc[1]`

`iloc[1]` uses the integer position of the row to select the second row

	A	B	C
0			
2			
1			

`df.loc[1]`

`loc[1]` uses the label of the row to select the row with an axis label of `1`.

	A	B	C
0			
1			
2			

`df.loc[1]`

`loc[1]` uses the label of the row to select the row with an axis label of `1`.

	A	B	C
0			
2			
1			

Using pandas methods to create boolean masks

```
>>> is_california = usa["hq_location"].str.endswith("CA")
```

```
>>> print(is_california.head())
```

```
0      False
```

```
7      False
```

```
8       True
```

```
9      False
```

```
10     True
```

```
Name: hq_location, dtype: bool
```

```
0      Bentonville, AR
```

```
7      Omaha, NE
```

```
8      Cupertino, CA
```

```
9      Irving, TX
```

```
10     San Francisco, CA
```

```
Name: hq_location, dtype: object
```

Using boolean operators to select items

	company	revenues	country
0	Walmart	485873	USA
1	State Grid	315199	China
2	Sinopec Group	267518	China
3	China Nation...	262573	China
4	Toyota Motor	254694	Japan

f500_sel

```
over_265 = f500_sel["revenues"] > 265000
china = f500_sel["country"] == "China"
```

0	True
1	True
2	True
3	False
4	False

over_265

0	False
1	True
2	True
3	True
4	False

china

```
combined = over_265 & china
```

0	True	&	0	False	=	0	False
1	True	&	1	True	=	1	True
2	True	&	2	True	=	2	True
3	False	&	3	True	=	3	False
4	False	&	4	False	=	4	False

over_265 china combined

Using boolean operators to select items

```
final_cols = ["company", "revenues"]  
result = f500_sel.loc[combined, final_cols]
```

		company	revenues	country			company	revenues
0	False	0	Walmart	485873	USA			
1	True →	1	State Grid	315199	China	→ 1	State Grid	315199
2	True →	2	Sinopec Group	267518	China	→ 2	Sinopec Group	267518
3	False	3	China Nation...	262573	China			
4	False	4	Toyota Motor	254694	Japan			

combined **f500_sel** **result**

Pandas Index Alignment

	fruit_veg	qty		
tomato	fruit	4	corn	yellow
carrot	veg	2	carrot	orange
lime	fruit	4	tomato	red
corn	veg	1	lime	green
eggplant	veg	2	eggplant	purple
food			colors	

	fruit_veg	qty	color
tomato	fruit	4	red
carrot	veg	2	orange
lime	fruit	4	green
corn	veg	1	yellow
eggplant	veg	2	purple
food			

```
food["color"] = colors
```

	fruit_veg	qty		
tomato	fruit	4	corn	yellow
carrot	veg	2	carrot	orange
lime	fruit	4	tomato	red
corn	veg	1	lime	green
eggplant	veg	2	eggplant	purple
food			colors	

Pandas Index Alignment

arugula	rocket
eggplant	aubergine
corn	maize

alt_name

```
food["alt_name"] = alt_name
```

	fruit_veg	qty	color	alt_name
tomato	fruit	4	red	NaN
carrot	veg	2	orange	NaN
lime	fruit	4	green	NaN
corn	veg	1	yellow	maize
eggplant	veg	2	purple	aubergine

food

Using Loops in Pandas

```
>>> print(df)
```

	A	B	C
x	6	1	0
y	1	8	8
z	3	8	7

```
>>> for i in df:  
    print(i)
```

A
B
C

Because one of the key benefits of pandas is that it has vectorized methods to work with data more efficiently, **we want to avoid using loops wherever we can**

Challenge: calculating return on assets by sector

```
{ 'Aerospace & Defense': 'Lockheed Martin',
  'Apparel': 'Nike',
  'Business Services': 'Adecco Group',
  'Chemicals': 'LyondellBasell Industries',
  'Energy': 'National Grid',
  'Engineering & Construction': 'Pacific Construction Group',
  'Financials': 'Berkshire Hathaway',
  'Food & Drug Stores': 'Publix Super Markets',
  'Food, Beverages & Tobacco': 'Philip Morris International',
  'Health Care': 'Gilead Sciences',
  'Hotels, Restaurants & Leisure': 'McDonald\xe2\x80\x99s',
  'Household Products': 'Unilever',
  'Industrials': '3M',
  'Materials': 'CRH',
  'Media': 'Disney',
  'Motor Vehicles & Parts': 'Subaru',
  'Retailing': 'H & M Hennes & Mauritz',
  'Technology': 'Accenture',
  'Telecommunications': 'KDDI',
  'Transportation': 'Delta Air Lines',
  'Wholesalers': 'McKesson' }
```

$$\text{return on assets} = \frac{\text{profits}}{\text{assets}}$$

Lesson_04 Part #2

Section 2.10

