Data Science and Visualization (DSV, F23)

2. Exploratory Data Analysis

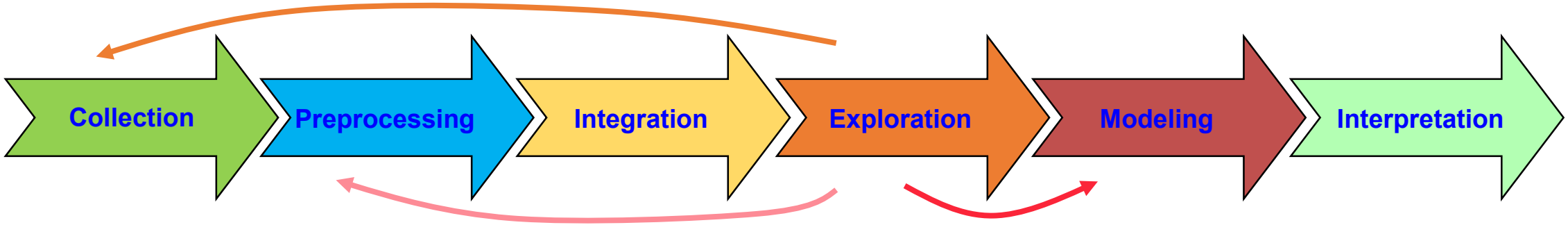
Hua Lu
https://luhua.ruc.dk; luhua@ruc.dk
PLIS, IMT, RUC

Agenda

- **Exploratory Data Analysis**
  - Overview
  - Series in pandas
  - DataFrame in pandas
- Missing Values

Exploratory Data Analysis (EDA)

- Examining a dataset to discover its underlying characteristics with an emphasis on visualization.
  - A term coined by the statistician John Tukey in his book in 1997
- EDA helps you during analysis design to
  - determine if you should gather more data,
  - suggest hypotheses to test, and
  - identify models to develop.
- We will also involve preprocessing while doing EDA in this lecture



What do we do in EDA?

- Analyzing a single data column
  - Data type
  - Basic statistical description of your data
    - Mean, media, mode
    - Distribution
    - Plot, e.g., histogram, boxplot
- Analyzing two dimensions together
  - Correlation?
- Exploring multiple column simultaneously
  - (col1, col2): correlation?

- We can ask questions, and try to answer them by exploring the data.
- Such a question may involve one, two, or more dimensions.
- Data processing and calculation may also be necessary for answering a question.
- The answer of a question may be visualized in a graph.
  - Bar charts
  - Line charts
  - Scatterplots
  - Histograms
  - Boxplot
  - Bubble, pie and radar

Loading Data in Text Format

Panda has many functions to parse and read raw data into *tabular* form as **DataFrame** object(s).

- **read_csv** Load delimited data from a file, URL, or file-like object; use comma as default delimiter
- **read_table** Load delimited data from a file, URL, or file-like object; use tab ('\t') as default delimiter
- **read_fwf** Read data in fixed-width column format (i.e., no delimiters)
- **read_clipboard** Version of read_table that reads data from the clipboard; useful for converting tables from web pages
- **read_excel** Read tabular data from an Excel XLS or XLSX file
- **read_hdf** Read HDF5 files written by pandas
- **read_html** Read all tables found in the given HTML document
- … …

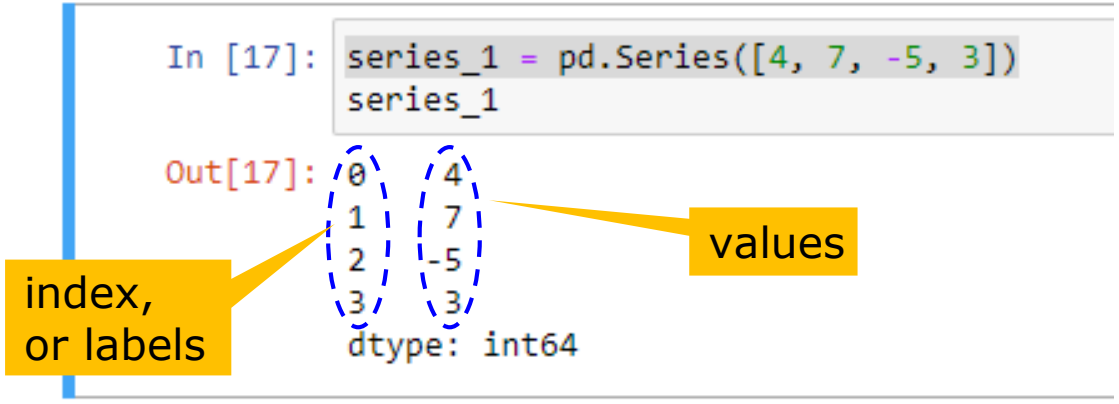
Loading Data in Text Format, cont.

Panda has many functions to parse and read raw data into *tabular* form as **DataFrame** object(s).

- … …
- **read_json** Read data from a JSON (JavaScript Object Notation) string representation
- **read_msgpack** Read pandas data encoded using the MessagePack binary format
- **read_pickle** Read an arbitrary object stored in Python pickle format
- **read_sas** Read a SAS dataset stored in one of the SAS system's custom storage formats
- **read_sql** Read the results of a SQL query (using SQLAlchemy) as a pandas DataFrame
- **read_stata** Read a dataset from Stata file format
- **read_feather** Read the Feather binary file format

Series in pandas

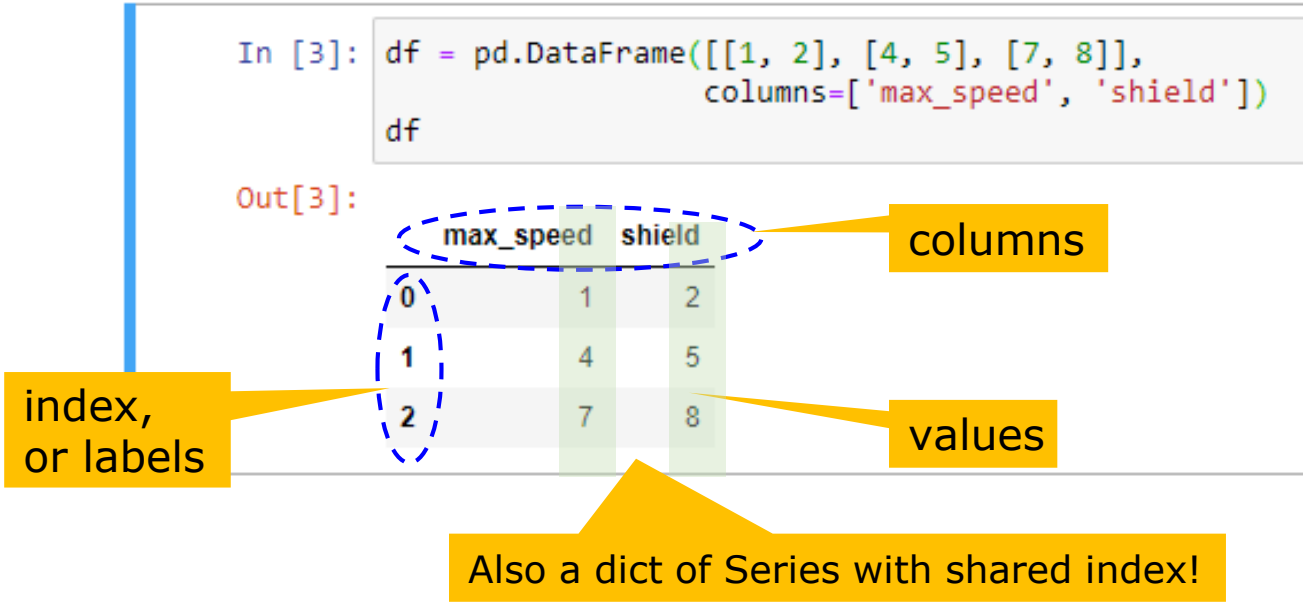
- A **Series** is a 1-d array-like object that contains
  - a sequence of *values* and
  - an associated array of data labels, called its *index*.
- In a Series object
  - All values in its 'array' can be heterogeneous (i.e., of different data types)
  - Its 'array' size is mutable (i.e., can be changed)
- Example



Lecture2-1_Series.ipynb

DataFrame in pandas

- A **DataFrame** is a 2-d size-mutable table with a set of (heterogeneous) dimensions
  - Like Series, it also has index.
  - Each index element corresponds to a set of values (vector) for all the dimensions.
- Example



Lecture2-2_DataFrame.ipynb

Basic Descriptions & Statistics in DataFrame

- shape: (#rows, #columns)
- info(): data type of each column
- describe(): statistics of all numeric columns
  - count, mean, std
  - min, 25%, 50%, 75%, max (percentiles)
- describe(include='all): incl. statistics of non-numeric columns
  - count, unique
  - top, freq
- nlargest() and nsmallest() of a column
  - Largest and smallest values
  - Most frequent and least frequent values

DEMO

Lecture2-2_DataFrame.ipynb

groupby in DataFrame

- df.groupby(column):
  - split the data into several groups based on the different values of the column
- It should be used together with an aggregate operation, e.g.,
  - count(), max(), min(), mean(), sum()

df

	Animal	Max Speed
0	Falcon	380.0
1	Falcon	370.0
2	Parrot	24.0
3	Parrot	26.0

df.groupby(['Animal']).mean()

Animal	Max Speed
Falcon	375.0
Parrot	25.0

df.groupby(['Animal']).count()

Animal	Max Speed
Falcon	2
Parrot	2

df.groupby(['Animal']).max()

Animal	Max Speed
Falcon	380.0
Parrot	26.0

df.groupby(['Animal']).sum()

Animal	Max Speed
Falcon	750.0
Parrot	50.0

Agenda

- Exploratory Data Analysis
- **Missing Values**
  - Types
  - Detection
  - Removal
  - Fill-in

Missing Value Types

- np.nan, np.**NaN**, and np.NAN
  - The same: NaN stands for *Not a Number*.
  - Missing value for a **numeric** column.
- pd.**NaT**
  - NaT stands for *Not a Time*.
  - Missing value for a **DateTime** column.
- Python's **None**
  - None defines a *null* value, or *no value* at all.
  - It's not the same as 0, False (or Boolean), or an empty string.
  - It's a data type of its own (**NoneType**) and only None can be None.
- Data type
  - A pandas *object* dtype column (mainly strings) can hold None, NaN, NaT or all three at the same time!

Consider **NULL** values if you know SQL
- Empty values
- Missing values
- Unknown values
- Values cannot be compared

Missing Value Manipulation in DataFrame

- Functions for detection
  - isnull(), notnull()
  - isna(), notna()
- Statistics of missing values
  - isnull().sum(axis=0 or 1): 0 for columns, 1 for rows
  - Missing rate: isnull().mean()
- Removing missing values
  - dropna(): drop selected rows, columns with NaN in a DataFrame
- Filling for missing values
  - fillna()

Lecture2-3_MissingValues.ipynb

Special ‘Missing’ Values

- They are not NULL values like NaN, NaT or None
- They may be represented as a different special value to indicate something is missing.
- We identify such a special value, replace it with np.NaN, and then use fillna().
- We want to replace the NaN with some value derived from existing values of the same dimension.
  - E.g., for a missing salary value, we may use the average salary of the same age group.
- To this end, we may need a combination:
  - **groupby + transform**

groupby + transform

- df.groupby('A')['C'].transform('mean')

Equivalent to

1. Get the mean for each group in ‘A’: df.groupby('A').mean()
2. Replace each 'C' value by the mean from its corresponding ‘A’ group.

```
df = pd.DataFrame({'A' : ['foo', 'bar', 'foo', 'bar',
                          'foo', 'bar'],
                   'B' : ['one', 'one', 'two', 'three',
                          'two', 'two'],
                   'C' : [1, 5, 5, 2, 5, 5]})
df
```

	A	B	C
0	foo	one	1
1	bar	one	5
2	foo	two	5
3	bar	three	2
4	foo	two	5
5	bar	two	5

```
df.groupby('A').mean()
```

A	C
bar	4.000000
foo	3.666667

This can be combined with fillna()

```
df.groupby('A')['C'].transform('mean')
```

```
0    3.666667
1    4.000000
2    3.666667
3    4.000000
4    3.666667
5    4.000000
Name: C, dtype: float64
```

Special ‘Missing’ Values Example

- Lecture2-4_Fortune500.ipynb
  - We apply **groupby + transform** to a dimenion’s ‘missing values’ only, not to all its values.

Summary

- Series in pandas
- DataFrame in pandas
  - Relation with Series
  - Selection, .loc
  - groupby
- Missing value handling
  - NaN (for all data types) vs. ‘N. A.’ (in strings)
  - Detection
  - Removal
  - Fill-in

References

- Wes McKinney: Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, 2nd edition, O'Reilly Medi
  - Chapter 5. Getting Started with pandas
- GroupBy in Pandas
  - https://www.analyticsvidhya.com/blog/2020/03/groupby-pandas-aggregating-data-python/
- Documentation of pandas
  - https://pandas.pydata.org/docs/getting_started/index.html

Exercises (1)

1. Refer to the table below: write your code in a Jupyter Notebook
    1. Create a DataFrame object for this table, use *None* for NULL values.
    2. Show the DataFrame object.
    3. Show the index of the DataFrame object.
    4. Change the index to ['A', 'B', 'C', 'D', 'E', 'F']
    5. Find the most frequent value of the height column
    6. Save the data to a .csv file and load/show the data again from the file.

name	gender	age	height	weight
John	Male	3	96	15
Kate	Female	4	100	17
Sebastian	Male	5	110	19
Mads	Male	3	100	NULL
Emil	Male	5	NULL	16
Kelly	Female	4	100	15

Exercises (2)

2. Titanic dataset: write your code in a Jupyter Notebook
    1. Get the average age value for each sex group.
    2. Fill in the missing age values as the average age of the corresponding sex group.
3. Fortunate500 dataset: write your code in a Jupyter Notebook
    1. Get the maximum revenue value for each year (*tip*: using groupby).
    2. Find the 10 companies that appear in Fortunate 500 *least* frequently.
    3. Fill in the missing profit values using the minimum profit value of the corresponding year. (remember to convert the profit type to float64)

NB: Refer to Lecture 1 for the datasets