INT 303 BIG DATA ANALYTICS

Lecture: Data Grammar

Jia WANG

Jia.wang02@xjtlu.edu.cn

Xi'an Jiaotong-Liverpool University

西交利物浦大學

OUTLINE

- The basic EDA workflow
- What is the Grammar of Data?
- How is this grammar implemented in Pandas?

The Basic EDA Workflow

THE BASIC EDA WORKFLOW[1]

1. **Build** a DataFrame from the data (ideally, put all data in this object)
2. **Clean** the DataFrame. It should have the following properties:
   - Each row describes a single object
   - Each column describes a property of that object
   - Columns are numeric whenever appropriate
   - Columns contain atomic properties that cannot be further decomposed
3. **Explore global properties**. Use histograms, scatter plots, and aggregation functions to summarize the data.
4. **Explore group properties**. Use groupby, queries, and small multiples to compare subsets of the data.

[1]enunciated in this form by Chris Beaumont for cs109

BUILDING A DATAFRAME

- The easiest way to build a dataframe is simply to read in a CSV file.
- We WILL see an example of this here, and we shall see more examples in labs.
- We'll also see how we may combine multiple data sources into a larger dataframe.

CLEANING DATA

- Dealing with missing values
- Transforming types appropriately
- Taking care of data integrity

WHY DATA CLEANING IS ESSENTIAL?

1. Error-Free Data
2. Data Quality
3. Accurate and Efficient Data
4. Complete Data
5. Maintains Data Consistency

DATA CLEANING CYCLE

IMPORT DATASET

```
#importing the dataset by reading the csv file
data = pd.read_csv(/content/Iris.csv)
```

```
#displaying the first five rows of dataset
data.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

DISPLAY FIRST FIVE ROWS OF DATASET

In [49]:
```
dfcwci.head()
```

Out[49]:

	id	last_name	first_name	middle_name	street_1	street_2	city	state	zip	amount	date	candidate_id
0	NaN	Agee	Steven	NaN	549 Laurel Branch Road	NaN	Floyd	VA	24091	500.0	2007-06-30	16
1	NaN	Ahrens	Don	NaN	4034 Rennellwood Way	NaN	Pleasanton	CA	94566	250.0	2007-05-16	16
2	NaN	Ahrens	Don	NaN	4034 Rennellwood Way	NaN	Pleasanton	CA	94566	50.0	2007-06-18	16
3	NaN	Ahrens	Don	NaN	4034 Rennellwood Way	NaN	Pleasanton	CA	94566	100.0	2007-06-21	16
4	NaN	Akin	Charles	NaN	10187 Sugar Creek Road	NaN	Bentonville	AR	72712	100.0	2007-06-16	16

In [22]:
```
del dfcwci['id']
dfcwci.head()
```

Out[22]:

	last_name	first_name	middle_name	street_1	street_2	city	state	zip	amount	date	candidate_id
0	Agee	Steven	NaN	549 Laurel Branch Road	NaN	Floyd	VA	24091	500.0	2007-06-30	16
1	Ahrens	Don	NaN	4034 Rennellwood Way	NaN	Pleasanton	CA	94566	250.0	2007-05-16	16
2	Ahrens	Don	NaN	4034 Rennellwood Way	NaN	Pleasanton	CA	94566	50.0	2007-06-18	16
3	Ahrens	Don	NaN	4034 Rennellwood Way	NaN	Pleasanton	CA	94566	100.0	2007-06-21	16
4	Akin	Charles	NaN	10187 Sugar Creek Road	NaN	Bentonville	AR	72712	100.0	2007-06-16	16

MERGE DATASET

- Merging the dataset is the process of combining two datasets in one.

pandas.DataFrame.merge ¶

```
DataFrame.merge(right, how='inner', on=None, left_on=None, right_on=None,
left_index=False, right_index=False, sort=False, suffixes=('_x', '_y'), copy=True,
indicator=False, validate=None)                                          [source]
```

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.merge.html

REBUILD MISSING DATA (1)

- To find and fill the missing data in the dataset we will use another function.
- **Using isnull() /isna() function:**

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False

REBUILD MISSING DATA (2)

- Using isna(). sum()

```
data.isna().sum()
```

```
Id              0
SepalLengthCm   0
SepalWidthCm    0
PetalLengthCm   0
PetalWidthCm    0
Species         0
dtype: int64
```

REBUILD MISSING DATA (3)

- **De-Duplicate**
- De-Duplicate means remove all duplicate values

data.duplicated()

```
0      False
1      False
2      False
3      False
4      False
       ...
145    False
146    False
147    False
148    False
149    False
Length: 150, dtype: bool
```

REBUILD MISSING DATA (4)

- **DataFrame.fillna()**
- Fill NA/NaN values using the specified method.

pandas.DataFrame.fillna

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None,
downcast=None)                                                          [source]
```

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.fillna.html

REBUILD MISSING DATA (5)

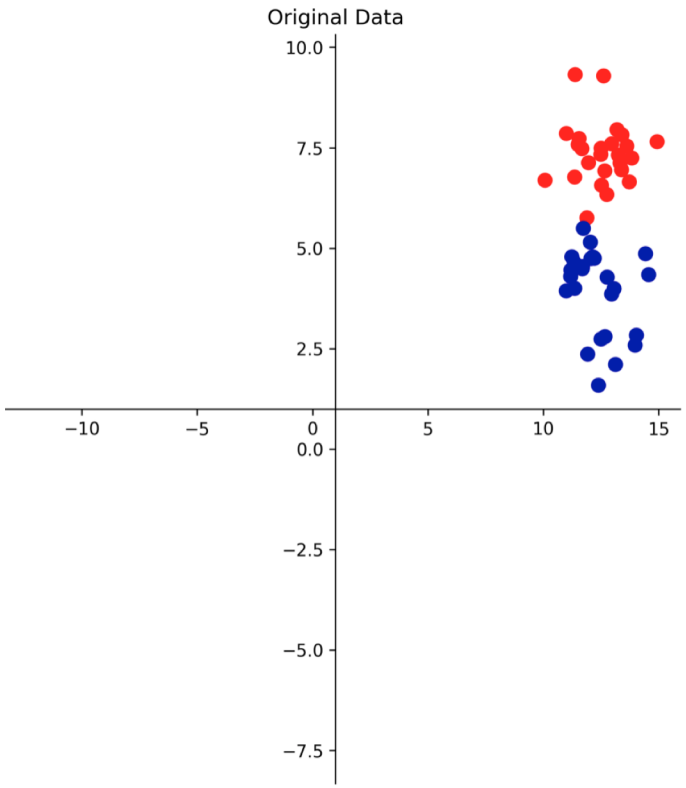
- If a dataset contains duplicate values it can be removed using the drop_duplicates() function.

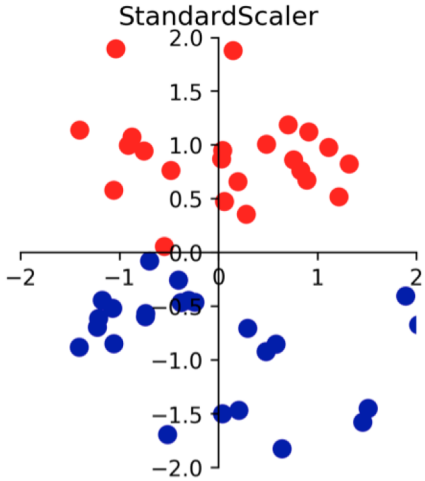
pandas.DataFrame.drop_duplicates

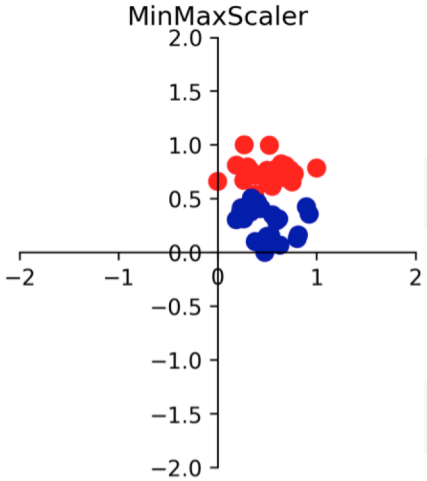
```
DataFrame.drop_duplicates(subset=None, keep='first', inplace=False,
ignore_index=False)
```

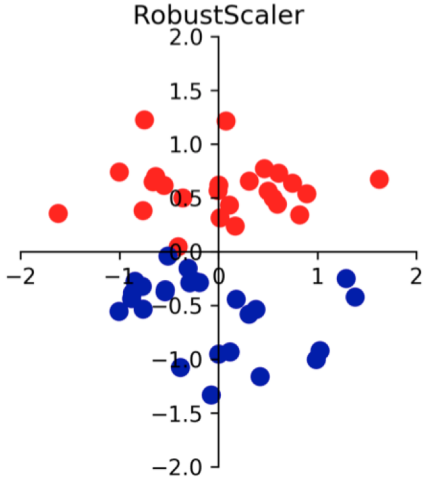
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop_duplicates.html

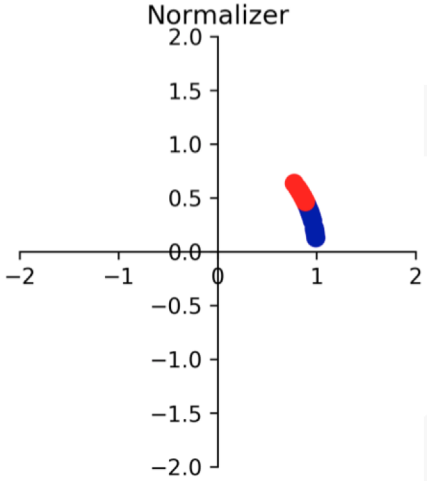
STANDARDIZATION AND NORMALIZATION

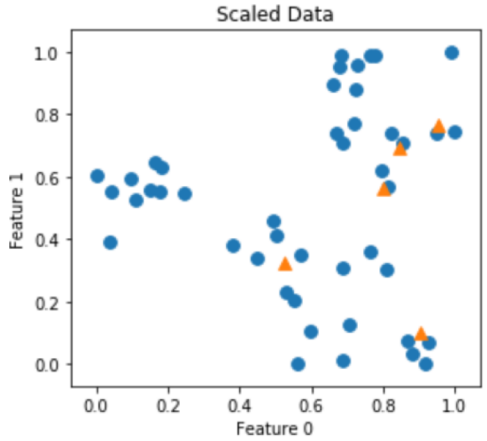


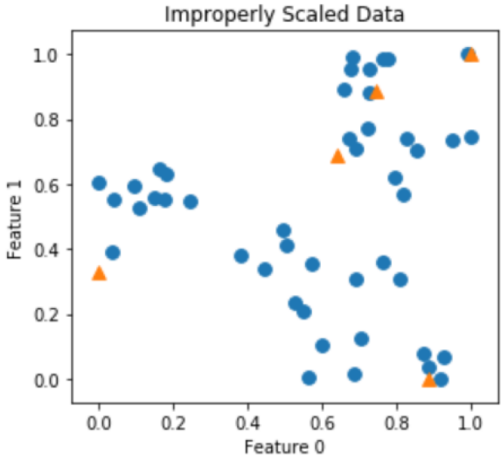












Case A:

1. Import Data
2. Split Data into training and test sets
3. Scale the training and test sets together using MinMaxScaler.
4. Visualization

Case B:

1. Import Data
2. Split Data into training and test sets
3. Scale training set using MinMaxScaler.
4. Rescale the test set separately using MinMaxScaler.
5. Visualization

VERIFY AND ENRICH

- We should verify the dataset and validate its accuracy.
- We have to check that the data cleaned so far is making any sense.
- If the data is incomplete we have to enrich the data.
- approaching the clients again, re-interviewing people, etc.

DATA TRANSFORMATION

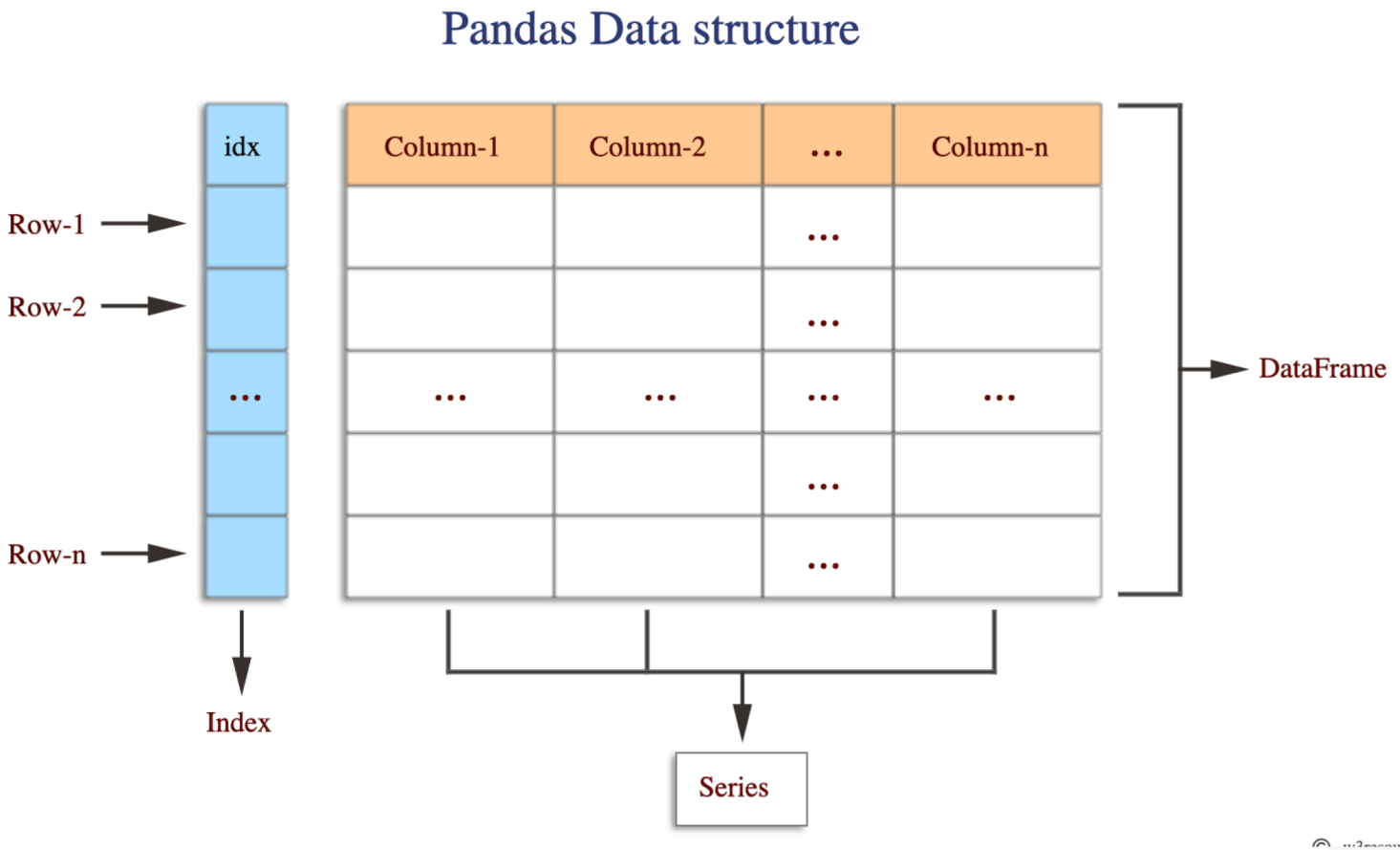
- Query
- Sort
- Select Columns
- Select Distinct
- Assign
- Group by
- Joint

Grammar of Data

PANDAS

- Pandas is well suited for many different kinds of data:
- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
- Ordered and unordered (not necessarily fixed-frequency) time series data.
- Arbitrary matrix data with row and column labels
- Any other form of observational / statistical data sets.

PANDAS


Pandas Data structure
idx
Column-1
Column-2
...
Column-n
Row-1
Row-2
Row-n
DataFrame
Index
Series

PANDAS

Pandas Basic commands:

Imports the following commands to start:

```
import pandas as pd
import numpy as np
```

Pandas version:

```
import pandas as pd
print(pd.__version__)
```

GRAMMAR OF DATA

- If you need to find a Data Related work!!!

- https://www.w3resource.com/python-exercises/pandas/index.php

GRAMMAR OF DATA

- Why bother?
- learn how to do core data manipulations, no matter what the system is.
- one off questions: google, stack-overflow, http://chrisalbon.com

BEST PRACTICE

- Go to notebook:

grammarofdata.ipynb

HOW TO CREATE A SERIES FROM A LIST, NUMPY ARRAY AND DICT?

```
# Input
import numpy as np
a_list = list("abcdefg")
numpy_array = np.arange(1, 10)
dictionary = {"A":  0, "B":1, "C":2, "D":3, "E":5}
```

```
series1 = pd.Series(a_list)
print(series1)
series2 = pd.Series(numpy_array)
print(series2)
series3 = pd.Series(dictionary)
print(series3)
```

```
0    a
1    b
2    c
3    d
4    e
5    f
6    g
dtype: object
0    1
1    2
2    3
3    4
4    5
5    6
6    7
7    8
8    9
dtype: int64
A    0
B    1
C    2
D    3
E    5
dtype: int64
```

HOW TO COMBINE MANY SERIES TO FORM A DATAFRAME?

```
# input
ser1 = pd.Series(list('abcedfghijklmnopqrstuvwxyz'))
ser2 = pd.Series(np.arange(26))
```

```
# using pandas DataFrame
ser_df = pd.DataFrame(ser1, ser2).reset_index()
ser_df.head()
```

	index	0
0	0	a
1	1	b
2	2	c
3	3	e
4	4	d

HOW TO GET USEFUL INFOS

```
# input
state = np.random.RandomState(100)
ser = pd.Series(state.normal(10, 5, 25))
```

```
# using pandas
ser.describe()
```

```
count    25.000000
mean     10.435437
std       4.253118
min       1.251173
25%       7.709865
50%      10.922593
75%      13.363604
max      18.094908
dtype: float64
```

GROUPBY(1)

- Example: Candidates

```
In [8]: dfcand=pd.read_csv("../data/candidates.txt", sep='|')
        dfcand.head(10)
```

Out[8]:

	id	first_name	last_name	middle_name	party
0	33	Joseph	Biden	NaN	D
1	36	Samuel	Brownback	NaN	R
2	34	Hillary	Clinton	R.	D
3	39	Christopher	Dodd	J.	D
4	26	John	Edwards	NaN	D
5	22	Rudolph	Giuliani	NaN	R
6	24	Mike	Gravel	NaN	D
7	16	Mike	Huckabee	NaN	R
8	30	Duncan	Hunter	NaN	R
9	31	Dennis	Kucinich	NaN	D

GROUPBY(2)

- Contributors

```
dfcwci=pd.read_csv("../data/contributors_with_candidate_id.txt", sep="|")
dfcwci.head()
```

	id	last_name	first_name	middle_name	street_1	street_2	city	state	zip	amount	date	candidate_id
0	NaN	Agee	Steven	NaN	549 Laurel Branch Road	NaN	Floyd	VA	24091	500.0	2007-06-30	16
1	NaN	Ahrens	Don	NaN	4034 Rennellwood Way	NaN	Pleasanton	CA	94566	250.0	2007-05-16	16
2	NaN	Ahrens	Don	NaN	4034 Rennellwood Way	NaN	Pleasanton	CA	94566	50.0	2007-06-18	16
3	NaN	Ahrens	Don	NaN	4034 Rennellwood Way	NaN	Pleasanton	CA	94566	100.0	2007-06-21	16
4	NaN	Akin	Charles	NaN	10187 Sugar Creek Road	NaN	Bentonville	AR	72712	100.0	2007-06-16	16

GROUPBY(3)

- Groupby:
  - Splitting the data into groups based on some criteria
  - Applying a function to each group independently
  - Combining the results into a data structure

```
In [28]: dfcwci.groupby("state").sum()
```

Out[28]:

state	zip	amount	candidate_id
AK	2985459621	1210.00	111
AR	864790	14200.00	192
AZ	860011121	120.00	37
CA	14736360720	-5013.73	600
CO	2405477834	-5823.00	111
CT	68901376	2300.00	35
DC	800341853	-1549.91	102
FL	8970626520	-4050.00	803

MERGE

- Merge:
- Combine tables on a common key-value

```
In [40]: cols_wanted=['last_name_x', 'first_name_x', 'candidate_id', 'id', 'last_name_y']
         dfcwci.merge(dfcand, left_on="candidate_id", right_on="id")[cols_wanted]
```

Out[40]:

	last_name_x	first_name_x	candidate_id	id	last_name_y
0	Agee	Steven	16	16	Huckabee
1	Akin	Charles	16	16	Huckabee
2	Akin	Mike	16	16	Huckabee
3	Akin	Rebecca	16	16	Huckabee
4	Aldridge	Brittni	16	16	Huckabee

QUESTION ---MAPPING

Following is a preview of the DataFrame `df`:

x	y	z
1	NaN	1
2	NaN	2
NaN	1	3

Match the commands to their expected outputs:

1. `df.notna().sum()` B
2. `df.isna().any()` C
3. `df['z'].isna()` A

```
# A
0    False
1    False
2    False
Name: z, dtype: bool

# B
x    2
y    1
z    3
dtype: int64

# C
x     True
y     True
z    False
dtype: bool
```

QUESTION

- How many of the given statements are the correct reason why data cleansing is critical：

5, all of this

Error-Free Data

Data Quality

Accurate and Efficient Data

Complete Data

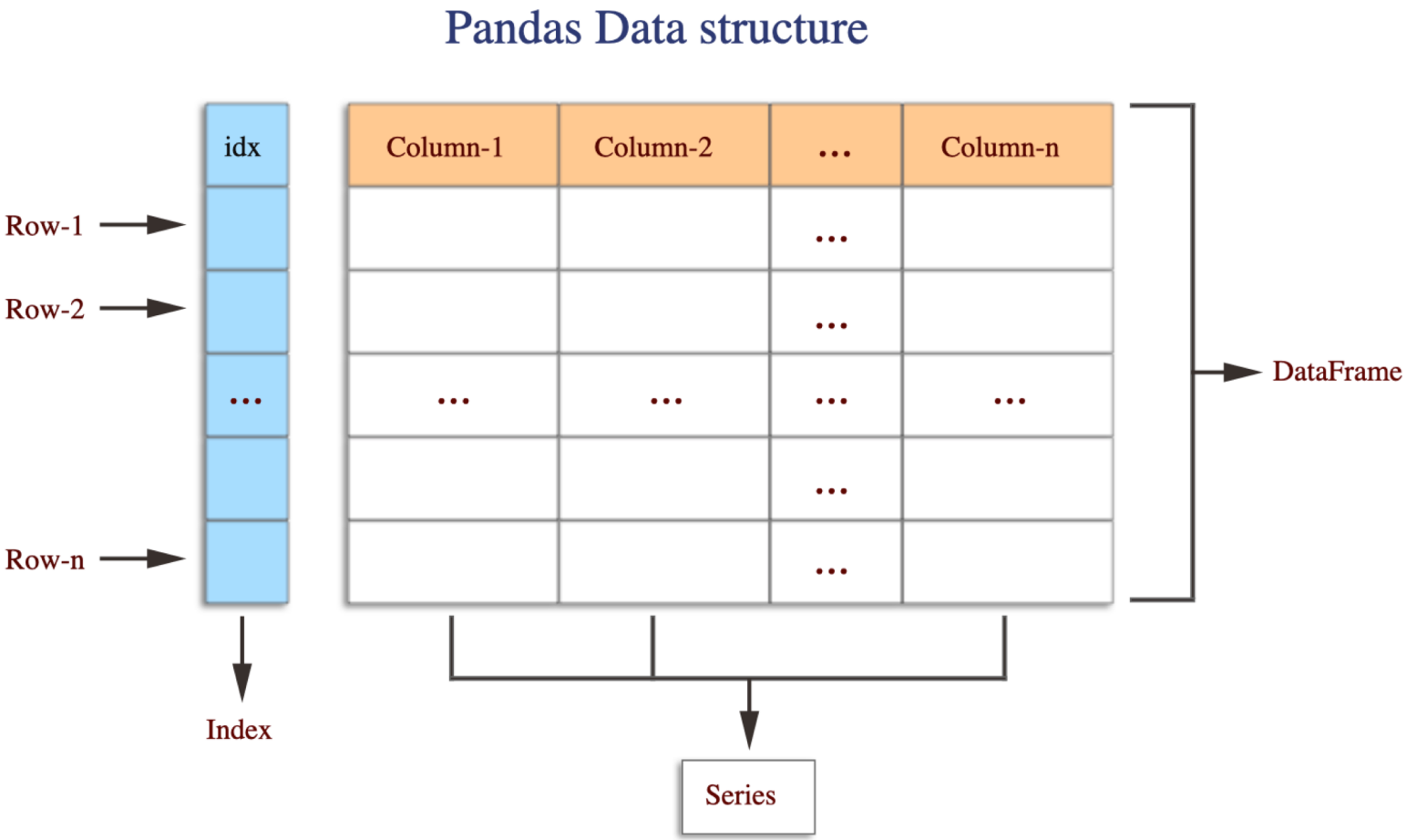
Maintains Data Consistency

QUESTION --- FILL IN A BLANK

- How to get useful infos of a Dataframe SER

- SER. describe()

QUESTION


Pandas Data structure
idx
Column-1
Column-2
...
Column-n
Row-1
Row-2
Row-n
DataFrame
Index
Series

THANK YOU

VISIT US

WWW.XJTLU.EDU.CN

FOLLOW US

@XJTLU

Xi'an Jiaotong-Liverpool University
西交利物浦大学