Predictive Analytics Foundations

Lecture 7 - 8

Lecture 7-8

- **Introduction to Exploratory Data Analysis**
- Intro to Pandas
- Series, DataFrames, and Indices
- Slicing with loc, iloc, and []
- Demo

slido

Audience Q&A Session

ⓘ Start presenting to display the audience questions on this slide.

John Tukey on EDA

John Tukey (1915-2000) was a Princeton Mathematician & Statistician and an **Early Data Scientist**.

Coined/Introduced:

- Fast Fourier Transform algorithm
- Box plot
- "Bit" : binary digit
- **Exploratory Data Analysis**

[Data Analysis & Statistics, Tukey 1965; Image from LIFE Magazine]

John Tukey on EDA

EDA is like **detective work**:

> Exploratory data analysis is an attitude, a state of flexibility, a willingness to look for those things that we believe are not there, as well as those that we believe to be there.

Exploratory Data Analysis (EDA)

"Getting to know and understand the data"

Exploratory Data Analysis (EDA)

"The process of **transforming**, **visualizing**, and **summarizing** data to:

- Build/confirm understanding of the data and its **provenance**
- Identify and address potential issues in the data.
- Inform the subsequent analysis.
- Discover *potential* hypotheses … (be careful…)

Provenance: origin of data; methodology by which data were produced

Exploratory Data Analysis

EDA is an open-ended analysis.

- Informal, no specific idea of what we are looking for.
- Be willing to find something surprising!

Contrast with **confirmatory analysis**:

- Questions are fixed in advance.
- Allows for more rigorous statistical analysis.

Lecture 7-8

- Introduction to Exploratory Data Analysis
- **Intro to Pandas**
- Series, DataFrames, and Indices
- Slicing with loc, iloc, and []
- Demo

Pandas

- We will use pandas to:
  - Read in data from Excel.
  - Manipulate data in spreadsheet.
  - Visualize data (we will also use another Python package called ggplot to do this).
  - Filter and aggregate data from spreadsheet using SQL

Reading Data From Excel

I have the following data saved in the file “grades.csv”:

	A	B	C	D	E
1	Student Name	Math Grade	Science Grade	History Grade	English Grade
2	John Doe	85	92	78	88
3	Jane Smith	91	88	75	90
4	Michael Johnson	78	85	82	86
5	Emily Davis	95	89	92	94
6	David Wilson	88	93	80	89
7	Sarah Brown	76	84	79	87
8	Christopher Lee	90	91	88	92
9	Lisa Anderson	87	82	75	86
10	Kevin Martin	82	78	76	80
11	Amanda White	89	90	85	91
12					

Before you use pandas you must import it. Anytime you use pandas put this line as the top of your code.

```
import pandas as pd
df_grades = pd.read_csv("grades.csv")
```

Built in read_csv method

Path to file

Reading Data From Excel

So, what is df_grades and how does it store the data?

```
import pandas as pd
df_grades = pd.read_csv("grades.csv")
df_grades
```

Typing the name of any variable at the end of a code cell will display the contents of the variable.

Reading Data From Excel

So, what is df_grades and how does it store the data?

```
import pandas as pd
df_grades = pd.read_csv("grades.csv")
df_grades
```

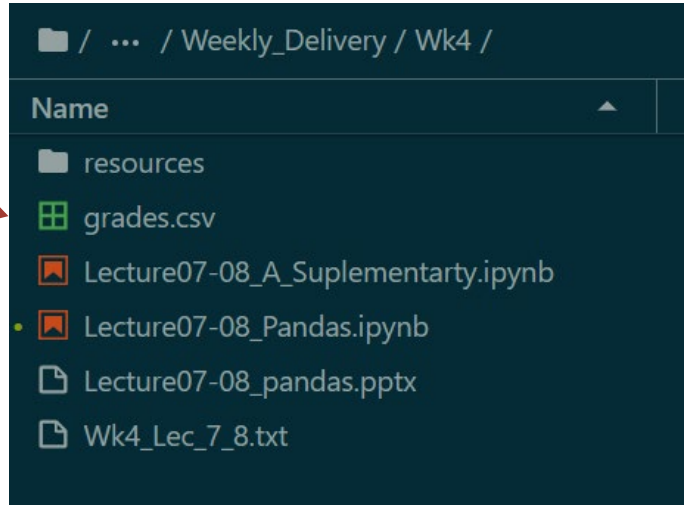
	Student Name	Math Grade	Science Grade	History Grade	English Grade
0	John Doe	85	92	78	88
1	Jane Smith	91	88	75	90
2	Michael Johnson	78	85	82	86
3	Emily Davis	95	89	92	94
4	David Wilson	88	93	80	89
5	Sarah Brown	76	84	79	87
6	Christopher Lee	90	91	88	92
7	Lisa Anderson	87	82	75	86
8	Kevin Martin	82	78	76	80
9	Amanda White	89	90	85	91

- df_grades is a pandas **dataframe.**
- The data is stored in a tabular format very similar to excel.

Reading Data From Excel

Data file

Jupyter notebook

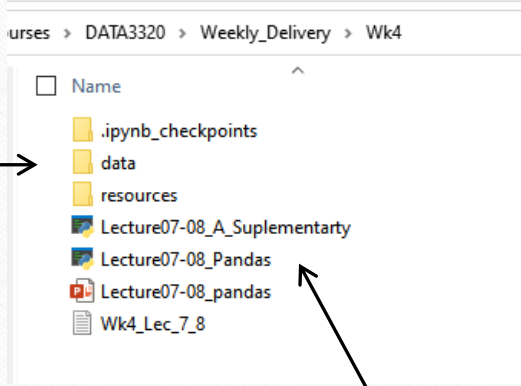



```python
# relative file path
import pandas as pd
df_grades = pd.read_csv("grades.csv")
df_grades.head()
```

	Student Name	Math Grade	Science Grade	History Grade	English Grade
0	John Doe	85	92	78	88
1	Jane Smith	91	88	75	90
2	Michael Johnson	78	85	82	86
3	Emily Davis	95	89	92	94
4	David Wilson	88	93	80	89

Reading in Data From Excel

Now grades.csv is in data Folder



Jupyter notebook

Reading Data From Excel

urses › DATA3320 › Weekly_Delivery › Wk4

Name

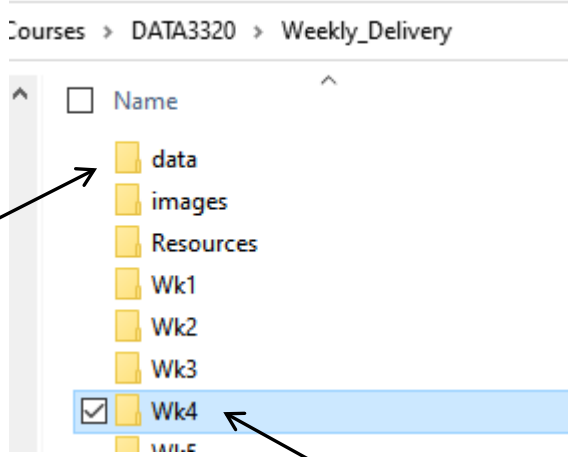
- .ipynb_checkpoints
- data
- resources
- Lecture07-08_A_Suplementarty
- Lecture07-08_Pandas
- Lecture07-08_pandas
- Wk4_Lec_7_8

"/" separates directories

```python
# relative file path
import pandas as pd
df_grades = pd.read_csv("data/grades.csv")
df_grades.head()
```

	Student Name	Math Grade	Science Grade	History Grade	English Grade
0	John Doe	85	92	78	88
1	Jane Smith	91	88	75	90
2	Michael Johnson	78	85	82	86
3	Emily Davis	95	89	92	94
4	David Wilson	88	93	80	89

Reading Data From Excel



Now grades.csv is in Data Folder

Jupyter notebook in Wk4 folder

"../" = go back one directory

```python
# relative file path
import pandas as pd
df_grades = pd.read_csv("../data/grades.csv")
df_grades.head()
```

	Student Name	Math Grade	Science Grade	History Grade	English Grade
0	John Doe	85	92	78	88
1	Jane Smith	91	88	75	90
2	Michael Johnson	78	85	82	86
3	Emily Davis	95	89	92	94
4	David Wilson	88	93	80	89

The head() Method

- If the data is really large you don't want to print out the entire dataframe to your output.
- The **head(n)** method outputs the first n rows of the data frame. If n is not supplied, the default is the first 5 rows.
- I like to run the head() method after I read in the dataframe to check that everything got read in correctly.
- There is also a **tail(n)** method that returns the last n rows of the dataframe

```python
# relative file path
import pandas as pd
df_grades = pd.read_csv("../data/grades.csv")
df_grades.head()
```

	Student Name	Math Grade	Science Grade	History Grade	English Grade
0	John Doe	85	92	78	88
1	Jane Smith	91	88	75	90
2	Michael Johnson	78	85	82	86
3	Emily Davis	95	89	92	94
4	David Wilson	88	93	80	89

Lecture 7-8

- Introduction to Exploratory Data Analysis
- Intro to Pandas
- **Series, DataFrames, and Indices**
- Slicing with loc, iloc, and []
- Demo

What is a Pandas Series?

A Pandas Series is a one-dimensional labeled array data structure provided by the Python library, Pandas.

It is a fundamental building block for data manipulation and analysis in Pandas and is often used to store and manipulate data in various forms.

What is a Pandas Series?

Creating a simple Series from a list.

```
data = [10, 20, 30, 40, 50]
series = pd.Series(data)
print(series)
```

```
0    10
1    20
2    30
3    40
4    50
```

Indexing and Labeling

Index labels are unique identifiers associated with each element in a Pandas Series. They can be either explicitly specified or automatically generated.

•**Automatic Indexing:** By default, when you create a Series, Pandas assigns integer-based labels starting from 0 to the elements in the Series. These labels are similar to row numbers in a spreadsheet.

•**Custom Indexing:** You can also specify custom labels for the Series by providing an index argument when creating the Series.

•**Data Alignment:** Index labels are used to align data during operations between multiple Series objects, ensuring that data is matched correctly based on their labels.

Indexing and Labeling

You can use square brackets **[]** to access elements by their labels.

```
import pandas as pd

data = [10, 20, 30, 40, 50]
index = ['A', 'B', 'C', 'D', 'E']
series = pd.Series(data, index=index)

print(series['B'])  # Accessing by label 'B'
```

```
20
```

Accessing Data

Accessing elements by label and position.

```
import pandas as pd
data = [1, 2, 3, 4, 5]
index = ['A', 'B', 'C', 'D', 'E']
series = pd.Series(data, index=index)
print(series['B'])  # Access by label
print(series[2])    # Access by position
```

```
2
3
```

Operations on Series

Adding two Series together.

```python
import pandas as pd
data = [1, 2, 3, 4, 5]
index = ['A', 'B', 'C', 'D', 'E']
series1 = pd.Series(data, index=index)

data2 = [5, 10, 15, 20, 25]
series2 = pd.Series(data2, index=index)
result = series + series2
print(result)
```

```
A     6
B    12
C    18
D    24
E    30
```

What is dataframe?

Creating a DataFrame in Pandas is a fundamental step in data analysis and manipulation.

A DataFrame is a two-dimensional, **labeled** data structure that resembles a table or spreadsheet.

It is capable of holding data in various formats, making it a versatile tool for working with structured data.

Creating Dataframes

From dictionary

```
import pandas as pd

data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 22],
    'City': ['New York', 'San Francisco', 'Los Angeles']
}

df = pd.DataFrame(data)
print(df)
```

```
      Name  Age           City
0    Alice   25       New York
1      Bob   30  San Francisco
2  Charlie   22    Los Angeles
```

Dataframe Operations

```python
# Selecting a single column
names = df['Name']

# Filtering rows based on a condition
young_people = df[df['Age'] < 30]

# Adding a new column
df['Salary'] = [60000, 70000, 55000]

# Summarizing data
summary = df.describe()

# Saving to a CSV file
df.to_csv('people.csv', index=False)
```

Lecture 7-8

- Introduction to Exploratory Data Analysis
- Intro to Pandas
- Series, DataFrames, and Indices
- **Slicing with loc, iloc, and []**
- Demo

Indexing and Slicing Dataframe

Slicing in Pandas is a powerful technique for selecting specific rows and columns from a DataFrame. You can achieve this using different methods:

1. [ ]
2. .loc[ ]
3. .iloc[ ]

Slicing with [ ]

You can use square brackets **[]** for basic slicing by specifying both row and column selections using labels or integer positions.

```python
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],
    'Age': [25, 30, 22, 28, 35],
    'City': ['New York', 'San Francisco', 'Los Angeles', 'Chicago', 'Miami']
}

df = pd.DataFrame(data)
# Select rows 1 to 3 (inclusive) and the 'Name' and 'Age' columns using []
selected_data = df[1:4][['Name', 'Age']]

print(selected_data)
```

```
      Name  Age
1      Bob   30
2  Charlie   22
3    David   28
```

Slicing with loc (label-based indexing)

- .loc[ ] is primarily used for label-based indexing, which means you select rows and columns by specifying their labels (names).

```python
# Create a sample DataFrame
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],
    'Age': [25, 30, 22, 28, 35],
    'City': ['New York', 'San Francisco', 'Los Angeles', 'Chicago', 'Miami']
}

df = pd.DataFrame(data)

# Select rows with labels 1 to 3 (inclusive) and columns 'Name' and 'Age'
selected_data = df.loc[1:3, ['Name', 'Age']]

print(selected_data)
```

```
      Name  Age
1      Bob   30
2  Charlie   22
3    David   28
```

Slicing with iloc (integer-based indexing)

- .iloc[ ] is used for integer-based indexing, where you select rows and columns by specifying their integer positions (0-based).

```
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],
    'Age': [25, 30, 22, 28, 35],
    'City': ['New York', 'San Francisco', 'Los Angeles', 'Chicago', 'Miami']
}

df = pd.DataFrame(data)
# Select rows from position 1 to 3 (exclusive) and columns from position 0 to 2 (exclusive)
selected_data = df.iloc[1:3, 0:2]

print(selected_data)
```

```
      Name  Age
1      Bob   30
2  Charlie   22
```

Indexing and Slicing Dataframe

It's important to note the key differences between these methods:

- **loc** uses label-based indexing for both rows and columns.
- **iloc** uses integer-based indexing for both rows and columns.
- **[]** can be used for basic slicing with labels or integers but is less versatile than **loc** and **iloc**.

Your choice of slicing method depends on whether you prefer label-based or integer-based indexing and the complexity of your data selection requirements. Each method has its strengths and use cases, so it's helpful to understand all of them for effective data manipulation in Pandas.

Lecture 7-8

- Introduction to Exploratory Data Analysis
- Intro to Pandas
- Series, DataFrames, and Indices
- Slicing with loc, iloc, and []
- **Demo**

Demo