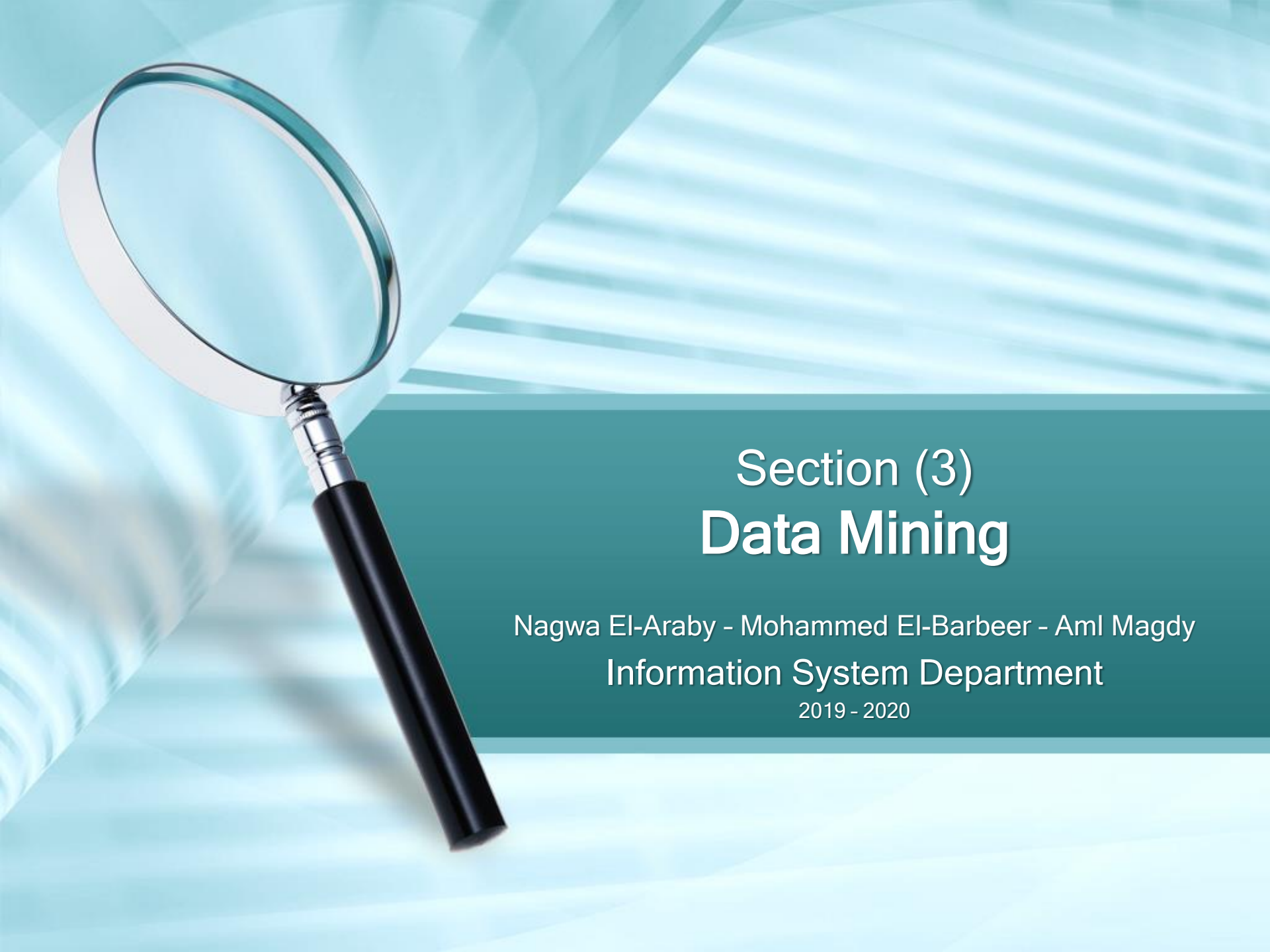

Section (3)
Data Mining

Nagwa El-Araby - Mohammed El-Barbeer - Aml Magdy
Information System Department
2019 - 2020

Importing Data Into Python.

There are various methods to read data in Python. Data can be in any of the popular formats - CSV, TXT, XLS/XLSX (Excel), SAS, Stata, Rdata (R) etc. Loading data in python environment is the most initial step of analyzing data

- ***While importing external files, we need to check the following points –***
- Check whether header row exists or not
- Treatment of special values as missing values
- Consistent data type in a variable (column)
- Date Type variable in consistent date format.
- No truncation of rows while reading external data

.

Importing Data Into Python.

Import CSV files

How to read a CSV file in python using read_csv function of pandas package ?

Without use of read_csv function, it is not straight forward to import CSV file with python object-oriented programming.

Pandas is an awesome powerful python package for data manipulation and supports various functions to load and import data from various formats

Importing Data Into Python.

Install and Load Pandas Package

Make sure you have pandas package already installed on your system. If you set up python using Anaconda, it comes with pandas package so you don't need to install it again. Otherwise you can install it by using command pip install pandas. Next step is to load the package by running the following command. pd is an alias of pandas package. We will use it instead of full name "pandas".

```
import pandas as pd
```

Importing Data Into Python.

Create Sample Data for Import

The program below creates a sample pandas data frame which can be used further

```
dt = {'ID': [11, 12, 13, 14, 15],
'first_name': ['David', 'Jamie', 'Steve', 'Stevart', 'John'],
'company': ['Aon', 'TCS', 'Google', 'RBS', '.'],
'salary': [74, 76, 96, 71, 78]}
 mydt = pd.DataFrame(dt, columns = ['ID', 'first_name', 'company', 'salary'])
```

Importing Data Into Python.

Save data as CSV in the working directory

Check working directory before you save your data file.

```
import os
 os.getcwd()
```

Incase you want to change the working directory, you can specify it in under

os.chdir( ) function.

Single backslash does not work in Python so use 2 backslashes while specifying file location.os.chdir("C:\\Users\\DELL\\Documents\\")

The following command tells python to write data in CSV format in your working directory

.mydt.to_csv('workingfile.csv', index=False)

Importing Data Into Python.

Example 1 : Read CSV file with header row

It's the basic syntax of read_csv() function. You just need to mention the filename. It assumes you have column names in first row of your CSV file.

```
mydata = pd.read_csv("workingfile.csv")
```

It stores the data the way It should be as we have headers in the first row of our datafile. It is important to highlight that header=0 is the default value. Hence we don't need to mention the **header=** parameter. It means header starts from first row as indexing in python starts from **0**. The above code is equivalent to this line of code. pd.read_csv("workingfile.csv", header=0)

Importing Data Into Python.

- **Inspect data after importing**

- mydata.shape
- mydata.columns
- mydata.dtypes

Importing Data Into Python.

Example 2 : Read CSV file with header in second row

Suppose you have column or variable names in second row. To read this kind of CSV file, you can submit the following command.

```
mydata = pd.read_csv("workingfile.csv", header = 1)
```

header=1 tells python to pick header from second row. It's setting second row as header. It's not a realistic example. I just used it for illustration so that you get an idea how to solve it. To make it practical, you can add random values in first row in CSV file and then import it again.

Output:

	11	David	Aon	74
0	12	Jamie	TCS	76
1	13	Steve	Google	96
2	14	Stevart	RBS	71
3	15	John	.	78

Importing Data Into Python.

Define your own column names instead of header row from CSV file

```
mydata0 = pd.read_csv("workingfile.csv", skiprows=1, names=['CustID', 'Name', 'Companies', 'Income'])
```

skiprows = 1 means we are ignoring first row and **names=** option is used to assign variable names manually.

	CustID	Name	Companies	Income
0	11	David	Aon	74
1	12	Jamie	TCS	76
2	13	Steve	Google	96
3	14	Stevart	RBS	71
4	15	John	.	78

Importing Data Into Python.

- **Example 3 : Skip rows but keep header**
- mydata = pd.read_csv("workingfile.csv", skiprows=[1,2])
- In this case, we are skipping **second** and **third** rows while importing. Don't forget index starts from 0 in python so 0 refers to first row and 1 refers to second row and 2 implies third row

	ID	first_name	company	salary
0	13	Steve	Google	96
1	14	Stevart	RBS	71
2	15	John	.	78

- Instead of **[1,2]** you can also write range(1,3). Both means the same thing but range( ) function is very useful when you want to skip many rows so it saves time of manually defining row position.

Importing Data Into Python.

- **Hidden secret of skiprows option**
- When **skiprows = 4**, it means skipping four rows from top. **skiprows=[1,2,3,4]** means skipping rows from second through fifth. It is because when list is specified in skiprows= option, it skips rows at index positions. When a single integer value is specified in the option, it considers skip those rows from top

Importing Data Into Python.

- **Example 4 : Read CSV file without header row**
- If you specify **"header = None",** python would assign a series of numbers starting from 0 to (number of columns - 1) as column names. In this datafile, we have column names in first row.
- mydata0 = pd.read_csv("workingfile.csv", header = None)

Index	0	1	2	3
0	ID	first_name	company	salary
1	11	David	Aon	74
2	12	Jamie	TCS	76
3	13	Steve	Google	96
4	14	Stevart	RBS	71
5	15	John	.	78

Output

Importing Data Into Python.

- **Add prefix to column names**

- mydata0 = pd.read_csv("workingfile.csv", header = None, **prefix="var"**)

- In this case, we are setting var as prefix which tells python to include this keyword before each column name.

Importing Data Into Python.

- **Example 5 : Specify missing values**
- The na_values= options is used to set some values as blank / missing values while importing CSV file.
- mydata00 = pd.read_csv("workingfile.csv", **na_values=['.']**)

	ID	first_name	company	salary
0	11	David	Aon	74
1	12	Jamie	TCS	76
2	13	Steve	Google	96
3	14	Stevart	RBS	71
4	15	John	**NaN**	78

Importing Data Into Python.

- **Example 6 : Read CSV File from External URL**
- You can directly read data from the CSV file that is stored on a web link. It is very handy when you need to load publicly available datasets from github, kaggle and other websites.

- mydata02 =pd.read_csv("http://winterolympicsmedals.com/medals.csv")

- Note:
- This DataFrame contains 2311 rows and 8 columns. Using mydata02.shape, you can generate this summary

-

Importing Data Into Python.

- **Example 7 : Skip Last 5 Rows While Importing CSV**
- mydata04 = pd.read_csv ("http://winterolympicsmedals.com/medals.csv", skip_footer=5)
- *in the above code, we are excluding bottom 5 rows using **skip_footer=** parameter.*

- **Example 8: Read only first 5 rows**
- mydata05 = pd.read_csv ("http://winterolympicsmedals.com/medals.csv", nrows=5)
- Using **nrows=** option, you can load top K number of rows.

Importing Data Into Python.

- **Example 9 :Interpreting "," as thousands separator**
- mydata06 = pd.read_csv ("http://winterolympicsmedals.com/medals.csv", thousands=",")

- **Example 10 : Read only specific columns**
- mydata07 = pd.read_csv("http://winterolympicsmedals.com/medals.csv", usecols=[1,5,7])
- The above code reads only columns based on index positions which are second, sixth and eighth position.

Importing Data Into Python.

- **Example 11: Read some rows and columns**
- mydata08 = pd.read_csv("http://winterolympicsmedals.com/medals.csv", usecols=[1,5,7], nrows=5)
- In the above command, we have combined usecols= and nrows= options. It will select only first 5 rows and selected columns.
- **Example 12: Read file with semi colon delimiter**
- mydata09 = pd.read_csv("file_path", sep = ';')
- Using **sep=** parameter in read_csv( ) function,
- you can import file with any delimiter other than default comma. In this case, we are using semi-colon as a separator.

Importing Data Into Python.

- **Example 13 : Change column type while importing CSV**
- Suppose you want to change column format from int64 to float64 while loading CSV file into Python.
- We can use **dtype =** option for the same.
- mydf = pd.read_csv("workingfile.csv", dtype = {"salary" : "float64"})

- **Example 14 : Measure time taken to import big CSV file**
- With the use of verbose=True, you can capture time taken for Tokenization, conversion and Parser memory cleanup.
- mydf = pd.read_csv("workingfile.csv", verbose=True)

Importing Data Into Python.

- **Example 15 : How to read CSV file without using Pandas package**
- To import CSV file with pure python way,
- you can submit the following command :

```
import csv
with open("C:/Users/DELL/Downloads/nycflights.csv") as f:
  d = DictReader(f)
  l=list(d)
```

Importing Data Into Python.

- You can also download and load CSV file from URL or external webpage.

```
import csv
import requests

response = requests.get('https://dyurovsky.github.io/psyc201/data/lab2/nycflights.csv').text
lines = response.splitlines()
d = csv.DictReader(lines)
l = list(d)
```

Importing Data Into Python.

- **2. Import File from URL**
- You don't need to perform additional steps to fetch data from URL. Simply put **URL** in read_csv() function (applicable only for CSV files stored in URL).

- *mydata = pd.read_csv("http://winterolympicsmedals.com/medals.csv")*

Importing Data Into Python.

- **3. Read Text File**

- We can use read_table() function to pull data from text file. We can also use read_csv() with sep= "\t" to read data from tab-separated file.

```
mydata = pd.read_table("C:\\Users\\Deepanshu\\Desktop\\example2.txt")
mydata = pd.read_csv("C:\\Users\\Deepanshu\\Desktop\\example2.txt", sep ="\t")
```

Importing Data Into Python.

- **4. Read Excel File**

- The read_excel() function can be used to import excel data into Python.

```
mydata =
pd.read_excel("https://www.eia.gov/dnav/pet/hist_xls/RBRTEd.xls",sheetname="D
ata 1", skiprows=2)
```

- *if you do not specify name of sheet in **sheetname= option**, it would take by default first sheet.*

Importing Data Into Python.

- **5. Read delimited file**
- Suppose you need to import a file that is separated with white spaces.

```
mydata2 =
pd.read_table("http://www.ssc.wisc.edu/~bhansen/econometrics/invest.dat",
sep="\s+", header = None)
```

- To include variable names, use the names= option like below -

```
mydata3 =
pd.read_table("http://www.ssc.wisc.edu/~bhansen/econometrics/invest.dat",
sep="\s+", names=['a', 'b', 'c', 'd'])
```

Importing Data Into Python.

- **Read SAS File**

- We can import SAS data file by using read_sas() function
- mydata4 = pd.read_sas('cars.sas7bdat')

Importing Data Into Python.

- **Read SQL Table**
- We can extract table from SQL database (SQL Server / Teradata).
- See the program below –
- **SQL Server**
- You can read data from tables stored in SQL Server by building a connection. You need to have server, User ID (UID), database details to establish connection.

Importing Data Into Python.

```
import pandas as pd
import pyodbc
conn = pyodbc.connect("Driver={SQL
Server};Server=serverName;UID=UserName;PWD=Password;Database=RCO_DW;")
df = pd.read_sql_query('select * from dbo.Table WHERE ID > 10', conn)
df.head()
```


Thank You