Introduction to Programming and Data Structures

Analytics in Python

Malay Bhattacharyya

Associate Professor

MIU, CAIML, TIH
Indian Statistical Institute, Kolkata

September, 2023

Understanding `DataFrame`

`DataFrame` is a two-dimensional, heterogeneous and size-mutable data structure with entries labeled by rows and columns.

Note: The labels are denoted as index.

Using a DataFrame

```
import pandas as pd
pd.DataFrame(Data, index=<Labels>)
```

Using a DataFrame

```
import pandas as pd
pd.DataFrame(Data, index=<Labels>)
```

A DataFrame can have any of the following inputs:

- A dictionary of 1D ndarrays, lists, dicts, or Series
- 2-D ndarray
- Structured or record ndarray
- A Series
- Another DataFrame

Using a DataFrame

```
import pandas as pd
pd.DataFrame(Data, index=<Labels>)
```

A DataFrame can have any of the following inputs:

- A dictionary of 1D ndarrays, lists, dicts, or Series
- 2-D ndarray
- Structured or record ndarray
- A Series
- Another DataFrame

Note: If no <Labels> are passed, one will be created having values [0, ..., len(Data) - 1].

Defining DataFrame with a dictionary of 1D ndarrays/lists

```
import pandas as pd
s = pd.DataFrame({'col1': [1, 2, 3, 4], 'col2': [5, 6,
7, 8]}, index=['a', 'b', 'c', 'd'])
print(s)
```

Defining DataFrame with a dictionary of 1D ndarrays/lists

```
import pandas as pd
s = pd.DataFrame({'col1': [1, 2, 3, 4], 'col2': [5, 6,
7, 8]}, index=['a', 'b', 'c', 'd'])
print(s)
```

Output:

```
   col1 col2
a     1    5
b     2    6
c     3    7
d     4    8
dtype: int64
```

Defining DataFrame with a dictionary of dicts/series

```
import pandas as pd
s = pd.DataFrame({'col1': pd.Series([.1, .2, .3, .4,
.5], index=['a', 'b', 'c', 'd', 'e']), 'col2':
pd.Series([1., 2., 3.], index=['a', 'b', 'd'])})
print(s)
```

Defining DataFrame with a dictionary of dicts/series

```
import pandas as pd
s = pd.DataFrame({'col1': pd.Series([.1, .2, .3, .4,
.5], index=['a', 'b', 'c', 'd', 'e']), 'col2':
pd.Series([1., 2., 3.], index=['a', 'b', 'd'])})
print(s)
```

Output:

```
   col1  col2
a   0.1   1.0
b   0.2   2.0
c   0.3   NaN
d   0.4   3.0
e   0.5   NaN
```

Defining DataFrame with a structured or record ndarray

```
import numpy as np
import pandas as pd
data = np.zeros((2, ), dtype=[('Year', 'i4'),
('Company', 'a12'), ('Share/Mutual Fund', 'f4')])
print(data)
data[:] = [(1907, 'TATA STEEL', 344.00), (2019, "TATA
DIGITAL INDIA FUND", '19.14')]
s = pd.DataFrame(data)
print(s)
```

Defining DataFrame with a structured or record ndarray

```
import numpy as np
import pandas as pd
data = np.zeros((2, ), dtype=[('Year', 'i4'),
('Company', 'a12'), ('Share/Mutual Fund', 'f4')])
print(data)
data[:] = [(1907, 'TATA STEEL', 344.00), (2019, "TATA
DIGITAL INDIA FUND", '19.14')]
s = pd.DataFrame(data)
print(s)
```

Output:

```
[(0, b'', 0.) (0, b'', 0.)]
   Year          Company  Share/Mutual Fund
0  1907    b'TATA STEEL'         344.000000
1  2019  b'TATA DIGITAL'          19.139999
```

Let's consider a real-world data

"ManHourData.xlsx":

1	Process	Complexity	Domain	ManHour
2	Release Management	Medium	App Support	55
3	Release Management	Medium	App Support	55
4	Service Request	Medium	Database Support	72
5	Release Management	Medium	App Support	55
6	Release Management	Medium	App Support	55
7	Release Management	Medium	App Support	55
8	Incident Management	Simple	Database Support	12
9	Release Management	Medium	App Support	55
10	Service Request	Simple	App Support	50
11	Service Request	Simple	Database Support	140
12	Release Management	Medium	App Support	55
13	Release Management	Medium	App Support	9
14	Service Request	Simple	App Support	103

⋮

Reading the data within dataframe

```
import pandas as pd
ED = pd.read_excel("ManHourData.xlsx", engine='openpyxl')
print(ED)
```

Output:

```
          Process Complexity Domain  ManHour
0     Release Management     Medium          App Support       55
1     Release Management     Medium          App Support       55
2        Service Request     Medium    Database Support       72
3     Release Management     Medium          App Support       55
4     Release Management     Medium          App Support       55
...                  ...        ...      ...                  ...
6455     Service Request     Medium    Database Support       49
6456     Service Request     Medium    Database Support       94
6457     Service Request     Medium          App Support       94
6458  Release Management     Medium          App Support       76
6459  Release Management     Medium          App Support       89

[6460 rows x 4 columns]
```

Accessing a column in a dataframe

Accessing a column by its name:

```
print(ED.loc[:, "ManHour"])
```

Accessing a column by its index:

```
print(ED.iloc[:, 2])
```

Output:

```
0        55
1        55
2        72
3        55
4        55
        ..
6455     49
6456     94
6457     94
6458     76
6459     89
Name: ManHour, Length: 6460, dtype: int64
```

Unique values of a column in a dataframe

Accessing the number of unique values of a column:

```
print(ED.loc[:,"Process"].nunique())
```

Accessing the count of each unique value of a column:

```
print(ED.loc[:,"Process"].value_counts())
```

Output:

```
3
Incident Management    3115
Service Request        2847
Release Management      498
Name: Process, dtype: int64
```

Unique values of a column in a dataframe

Accessing the number of unique values of a column:

```
print(ED.loc[:,"Complexity"].nunique())
```

Accessing the count of each unique value of a column:

```
print(ED.loc[:,"Complexity"].value_counts())
```

Output:

```
3
Simple     4026
Medium     2306
Complex     128
Name: Complexity, dtype: int64
```

Unique values of a column in a dataframe

Accessing the number of unique values of a column:

```
print(ED.loc[:,"Domain"].nunique())
```

Accessing the count of each unique value of a column:

```
print(ED.loc[:,"Domain"].value_counts())
```

Output:

```
2
Database Support    4080
App Support          2380
Name: Domain, dtype: int64
```

Selecting rows based on column values in a dataframe

Selecting rows based on one column condition:

```
ED1 = ED.loc[ED["Process"] == "Service Request"]
print(ED1)
```

Output:

```
      Process Complexity  ManHour Domain
2     Service Request     Medium       72    Database Support
8     Service Request     Simple       50           App Support
9     Service Request     Simple      140    Database Support
12    Service Request     Simple      103           App Support
14    Service Request     Simple       98    Database Support
...                 ...        ...      ...                 ...
6452  Service Request     Simple       40    Database Support
6454  Service Request     Simple       47    Database Support
6455  Service Request     Medium       49    Database Support
6456  Service Request     Medium       94    Database Support
6457  Service Request     Medium       94           App Support

[2847 rows x 4 columns]
```

Selecting rows based on column values in a dataframe

Selecting rows based on multiple column conditions:

```
ED2 = ED.loc[(ED["Complexity"] == "Simple") &
(ED["ManHour"] > 50)]
print(ED2)
```

Output:

```
          Process Complexity  ManHour Domain
9         Service Request     Simple     140    Database Support
12        Service Request     Simple     103          App Support
14        Service Request     Simple      98    Database Support
18    Incident Management     Simple      51          App Support
28        Service Request     Simple      60    Database Support
...                   ...        ...     ...                ...
6442      Service Request     Simple     206    Database Support
6447      Service Request     Simple     163    Database Support
6449      Service Request     Simple     145    Database Support
6450      Service Request     Simple      55    Database Support
6451      Service Request     Simple      55    Database Support

[1685 rows x 4 columns]
```

Selecting rows based on column values in a dataframe

Selecting rows based on a list of column values:

```
ED3 = ED.loc[ED["Complexity"].isin(["Simple", "Medium"])]
print(ED3)
```

Output:

```
          Process Complexity  ManHour Domain
0     Release Management     Medium       55          App Support
1     Release Management     Medium       55          App Support
2        Service Request     Medium       72   Database Support
3     Release Management     Medium       55          App Support
4     Release Management     Medium       55          App Support
...                   ...        ...      ...                  ...
6455     Service Request     Medium       49   Database Support
6456     Service Request     Medium       94   Database Support
6457     Service Request     Medium       94          App Support
6458  Release Management     Medium       76          App Support
6459  Release Management     Medium       89          App Support

[6332 rows x 4 columns]
```

Finding the mean

What about the mean ManHour across the entire data?

```
print("Mean:", ED.loc[:,"ManHour"].mean())
```

Output:

```
Mean: 81.26377708978328
```

Finding the mean – Let's go deeper

What about the mean ManHour across different Process?

```
ED1 = ED.loc[ED["Process"]=="Release Management"]
print("Mean (Process = Release Management):",
ED1.loc[:,"ManHour"].mean())
ED2 = ED.loc[ED["Process"]=="Service Request"]
print("Mean (Process = Service Request):",
ED2.loc[:,"ManHour"].mean())
ED3 = ED.loc[ED["Process"]=="Incident Management"]
print("Mean (Process = Incident Management):",
ED3.loc[:,"ManHour"].mean())
```

Output:

```
Mean (Process = Release Management): 115.33935742971887
Mean (Process = Service Request): 86.21074815595364
Mean (Process = Incident Management): 71.29470304975924
```

Finding the mean – Let's go deeper

What about the mean ManHour across different Complexity?

```
ED4 = ED.loc[ED["Complexity"]=="Simple"]
print("Mean (Complexity = Simple):",
ED4.loc[:,"ManHour"].mean())
ED5 = ED.loc[ED["Complexity"]=="Medium"]
print("Mean (Complexity = Medium):",
ED5.loc[:,"ManHour"].mean())
ED6 = ED.loc[ED["Complexity"]=="Complex"]
print("Mean (Complexity = Complex):",
ED6.loc[:,"ManHour"].mean())
```

Output:

```
Mean (Complexity = Simple): 62.36860407352211
Mean (Complexity = Medium): 110.50650477016478
Mean (Complexity = Complex): 148.75
```

Finding the mean – Let's go deeper

What about the mean ManHour across different Domain?

```
ED7 = ED.loc[ED["Domain"]=="App Support"]
print("Mean (Domain = App Support):",
ED7.loc[:,"ManHour"].mean())
ED8 = ED.loc[ED["Domain"]=="Database Support"]
print("Mean (Domain = Database Support):",
ED8.loc[:,"ManHour"].mean())
```

Output:

```
Mean (Domain = App Support): 96.7373949579832
Mean (Domain = Database Support): 72.2375
```

Finding the median

What about the median ManHour across the entire data?

```
print("Median:", ED.loc[:,"ManHour"].median())
```

Output:

```
Median: 52.0
```

Finding the median – Let's go deeper

What about the median ManHour across different `Process`?

```
ED1 = ED.loc[ED["Process"]=="Release Management"]
print("Median (Process = Release Management):",
ED1.loc[:,"ManHour"].median())
ED2 = ED.loc[ED["Process"]=="Service Request"]
print("Median (Process = Service Request):",
ED2.loc[:,"ManHour"].median())
ED3 = ED.loc[ED["Process"]=="Incident Management"]
print("Median (Process = Incident Management):",
ED3.loc[:,"ManHour"].median())
```

Output:

```
Median (Process = Release Management): 78.0
Median (Process = Service Request): 59.0
Median (Process = Incident Management): 41.0
```

Finding the median – Let's go deeper

What about the median ManHour across different Complexity?

```
ED4 = ED.loc[ED["Complexity"]=="Simple"]
print("Median (Complexity = Simple):",
ED4.loc[:,"ManHour"].median())
ED5 = ED.loc[ED["Complexity"]=="Medium"]
print("Median (Complexity = Medium):",
ED5.loc[:,"ManHour"].median())
ED6 = ED.loc[ED["Complexity"]=="Complex"]
print("Median (Complexity = Complex):",
ED6.loc[:,"ManHour"].median())
```

Output:

```
Median (Complexity = Simple): 42.0
Median (Complexity = Medium): 70.0
Median (Complexity = Complex): 113.0
```

Finding the median – Let's go deeper

What about the median ManHour across different `Domain`?

```
ED7 = ED.loc[ED["Domain"]=="App Support"]
print("Median (Domain = App Support):",
ED7.loc[:,"ManHour"].median())
ED8 = ED.loc[ED["Domain"]=="Database Support"]
print("Median (Domain = Database Support):",
ED8.loc[:,"ManHour"].median())
```

Output:

```
Median (Domain = App Support): 60.0
Median (Domain = Database Support): 46.0
```

Finding the mode

What about the mode of ManHour across the entire data?

```
print("Mode:", ED.loc[:,"ManHour"].mode())
```

Output:

```
Mode: 0    30
```

Finding the mode – Let's go deeper

What about the mode of ManHour across different Process?

```
ED1 = ED.loc[ED["Process"]=="Release Management"]
print("Mode (Process = Release Management):",
ED1.loc[:,"ManHour"].mode())
ED2 = ED.loc[ED["Process"]=="Service Request"]
print("Mode (Process = Service Request):",
ED2.loc[:,"ManHour"].mode())
ED3 = ED.loc[ED["Process"]=="Incident Management"]
print("Mode (Process = Incident Management):",
ED3.loc[:,"ManHour"].mode())
```

Output:

```
Mode (Process = Release Management): 0    55
Mode (Process = Service Request): 0    30
Mode (Process = Incident Management): 0    20
```

Finding the mode – Let's go deeper

What about the mode of ManHour across different Complexity?

```
ED4 = ED.loc[ED["Complexity"]=="Simple"]
print("Mode (Complexity = Simple):",
ED4.loc[:,"ManHour"].mode())
ED5 = ED.loc[ED["Complexity"]=="Medium"]
print("Mode (Complexity = Medium):",
ED5.loc[:,"ManHour"].mode())
ED6 = ED.loc[ED["Complexity"]=="Complex"]
print("Mode (Complexity = Complex):",
ED6.loc[:,"ManHour"].mode())
```

Output:

```
Mode (Complexity = Simple): 0    20
Mode (Complexity = Medium): 0    30
Mode (Complexity = Complex): 0    60
```

Finding the mode – Let's go deeper

What about the mode of ManHour across different Domain?

```
ED7 = ED.loc[ED["Domain"]=="App Support"]
print("Mode (Domain = App Support):",
ED7.loc[:,"ManHour"].mode())
ED8 = ED.loc[ED["Domain"]=="Database Support"]
print("Mode (Domain = Database Support):",
ED8.loc[:,"ManHour"].mode())
```

Output:

```
Mode (Domain = App Support): 0    30
Mode (Domain = Database Support): 0    20
```

Finding the standard deviation

What about the standard deviation of ManHour across the entire data?

```
print("SD:", ED.loc[:,"ManHour"].std())
```

Output:

```
SD: 84.60749016388206
```

Note: The value of standard deviation falls within $[0, \infty]$. A higher standard deviation denotes the distribution has larger spread around the mean.

Finding the standard deviation – Let's go deeper

What about the standard deviation of ManHour across different Process?

```
ED1 = ED.loc[ED["Process"]=="Release Management"]
print("SD (Process = Release Management):",
ED1.loc[:,"ManHour"].std())
ED2 = ED.loc[ED["Process"]=="Service Request"]
print("SD (Process = Service Request):",
ED2.loc[:,"ManHour"].std())
ED3 = ED.loc[ED["Process"]=="Incident Management"]
print("SD (Process = Incident Management):",
ED3.loc[:,"ManHour"].std())
```

Output:

```
SD (Process = Release Management): 111.93909712868272
SD (Process = Service Request): 83.98205215828945
SD (Process = Incident Management): 78.06753466532682
```

Finding the standard deviation – Let's go deeper

What about the standard deviation of ManHour across different Complexity?

```
ED4 = ED.loc[ED["Complexity"]=="Simple"]
print("SD (Complexity = Simple):",
ED4.loc[:,"ManHour"].std())
ED5 = ED.loc[ED["Complexity"]=="Medium"]
print("SD (Complexity = Medium):",
ED5.loc[:,"ManHour"].std())
ED6 = ED.loc[ED["Complexity"]=="Complex"]
print("SD (Complexity = Complex):",
ED6.loc[:,"ManHour"].std())
```

Output:

```
SD (Complexity = Simple): 53.70773216955122
SD (Complexity = Medium): 111.93557645602057
SD (Complexity = Complex): 117.49082725127514
```

Finding the standard deviation – Let's go deeper

What about the standard deviation of ManHour across different Domain?

```
ED7 = ED.loc[ED["Domain"]=="App Support"]
print("SD (Domain = App Support):",
ED7.loc[:,"ManHour"].std())
ED8 = ED.loc[ED["Domain"]=="Database Support"]
print("SD (Domain = Database Support):",
ED8.loc[:,"ManHour"].std())
```

Output:

```
SD (Domain = App Support): 100.18053244129945
SD (Domain = Database Support): 72.53009090990354
```

Finding the skewness

What about the skewness of ManHour across the entire data?

```
print("Skewness:", ED.loc[:,"ManHour"].skew())
```

Output:

```
Skewness: 2.625454375023274
```

Note: The value of skewness falls within $[-3, 3]$. A skewness value within $[-0.5, 0.5]$ denotes the distribution is approximately symmetric. A skewness value within $[-1, -0.5]$ or $[0.5, 1]$ denotes the distribution is moderately skewed. A skewness value outside $[-1, 1]$ denotes the distribution is highly skewed.

Finding the skewness – Let's go deeper

What about the skewness of ManHour across different Process?

```
ED1 = ED.loc[ED["Process"]=="Release Management"]
print("Skewness (Process = Release Management):",
ED1.loc[:,"ManHour"].skew())
ED2 = ED.loc[ED["Process"]=="Service Request"]
print("Skewness (Process = Service Request):",
ED2.loc[:,"ManHour"].skew())
ED3 = ED.loc[ED["Process"]=="Incident Management"]
print("Skewness (Process = Incident Management):",
ED3.loc[:,"ManHour"].skew())
```

Output:

```
Skewness (Process = Release Management): 2.002359935148196
Skewness (Process = Service Request): 2.6167777913626815
Skewness (Process = Incident Management): 2.759016977655708 3
```

Finding the skewness – Let's go deeper

What about the skewness of ManHour across different Complexity?

```
ED4 = ED.loc[ED["Complexity"]=="Simple"]
print("Skewness (Complexity = Simple):",
ED4.loc[:,"ManHour"].skew())
ED5 = ED.loc[ED["Complexity"]=="Medium"]
print("Skewness (Complexity = Medium):",
ED5.loc[:,"ManHour"].skew())
ED6 = ED.loc[ED["Complexity"]=="Complex"]
print("Skewness (Complexity = Complex):",
ED6.loc[:,"ManHour"].skew())
```

Output:

```
Skewness (Complexity = Simple): 1.5362123888570327
Skewness (Complexity = Medium): 2.082990062658895
Skewness (Complexity = Complex): 1.3893114388200833
```

Finding the skewness – Let's go deeper

What about the skewness of ManHour across different `Domain`?

```
ED7 = ED.loc[ED["Domain"]=="App Support"]
print("Skewness (Domain = App Support):",
ED7.loc[:,"ManHour"].skew())
ED8 = ED.loc[ED["Domain"]=="Database Support"]
print("Skewness (Domain = Database Support):",
ED8.loc[:,"ManHour"].skew())
```

Output:

```
Skewness (Domain = App Support): 2.3458149538166078
Skewness (Domain = Database Support): 2.6536984813576963
```

Finding the kurtosis

What about the kurtosis of ManHour across the entire data?

```
print("Kurtosis:", ED.loc[:,"ManHour"].kurtosis())
```

Output:

```
Kurtosis: 9.14264020337879
```

Note: The value of kurtosis falls within $[1, \infty]$. A higher kurtosis value denotes the distribution is more peaked and possesses thick tails (i.e., a majority of the values is located in the tails of the distribution instead of around the mean).

Finding the kurtosis – Let's go deeper

What about the kurtosis of ManHour across different Process?

```
ED1 = ED.loc[ED["Process"]=="Release Management"]
print("Kurtosis (Process = Release Management):",
ED1.loc[:,"ManHour"].kurtosis())
ED2 = ED.loc[ED["Process"]=="Service Request"]
print("Kurtosis (Process = Service Request):",
ED2.loc[:,"ManHour"].kurtosis())
ED3 = ED.loc[ED["Process"]=="Incident Management"]
print("Kurtosis (Process = Incident Management):",
ED3.loc[:,"ManHour"].kurtosis())
```

Output:

```
Kurtosis (Process = Release Management): 4.298572255033928
Kurtosis (Process = Service Request): 9.170575675209191
Kurtosis (Process = Incident Management): 10.506586337889791
```

Finding the kurtosis – Let's go deeper

What about the kurtosis of ManHour across different `Complexity`?

```
ED4 = ED.loc[ED["Complexity"]=="Simple"]
print("Kurtosis (Complexity = Simple):",
ED4.loc[:,"ManHour"].kurtosis())
ED5 = ED.loc[ED["Complexity"]=="Medium"]
print("Kurtosis (Complexity = Medium):",
ED5.loc[:,"ManHour"].kurtosis())
ED6 = ED.loc[ED["Complexity"]=="Complex"]
print("Kurtosis (Complexity = Complex):",
ED6.loc[:,"ManHour"].kurtosis())
```

Output:

```
Kurtosis (Complexity = Simple): 1.7339385489802295
Kurtosis (Complexity = Medium): 4.586010129409358
Kurtosis (Complexity = Complex): 1.3237928799690355
```

Finding the kurtosis – Let's go deeper

What about the kurtosis of ManHour across different Domain?

```
ED7 = ED.loc[ED["Domain"]=="App Support"]
print("Kurtosis (Domain = App Support):",
ED7.loc[:,"ManHour"].kurtosis())
ED8 = ED.loc[ED["Domain"]=="Database Support"]
print("Kurtosis (Domain = Database Support):",
ED8.loc[:,"ManHour"].kurtosis())
```

Output:

```
Kurtosis (Domain = App Support): 6.455064295149233
Kurtosis (Domain = Database Support): 10.411423256331128
```

Problems

1. Write a program in Python to calculate values of all sorts of measures of central tendency, dispersion and shape of the man-hours required for all combinations of Process, Complexity and Domain of jobs ($3 \times 3 \times 2 = 18$ types) given in the ManHour data and construct a DataFrame to keep all the said values. The rows and columns of the DataFrame must represent the job type (as a combination of Process, Complexity and Domain) and the values of statistical measures calculated for each of them, respectively.
2. Write a program in Python to explore various statistical properties of any data of your choice given in `https://data.gov.in`.