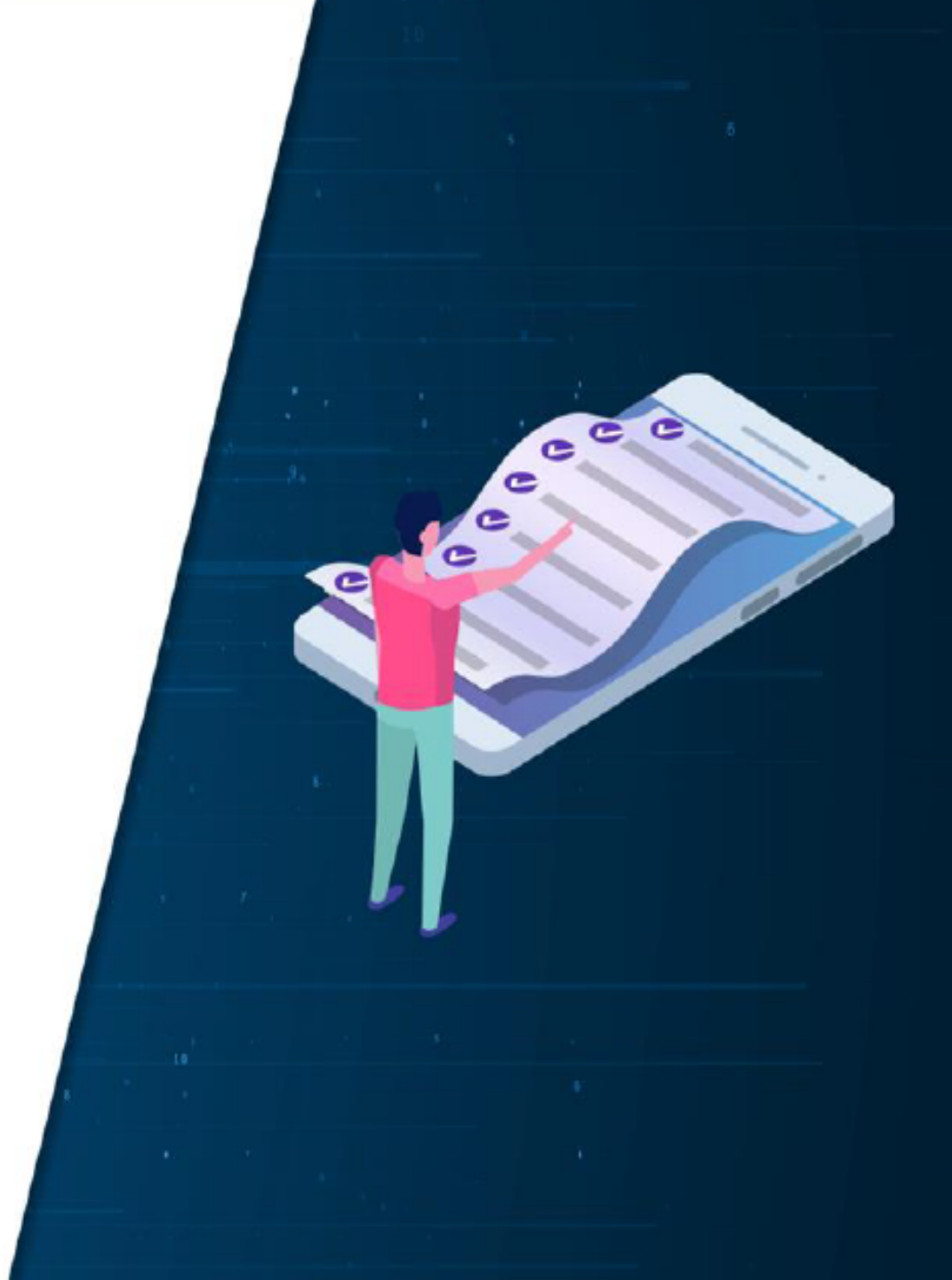
Python for Data Analysis

Learning Objectives

By the end of this lesson, you will be able to:

- Describe Data Analytics process and its steps
- List the skills and tools required for data analysis
- Understand the challenges of the Data Analytics process
- Explain Exploratory data analysis technique
- Illustrate data visualization techniques
- Describe Hypothesis testing



Why Data Analytics

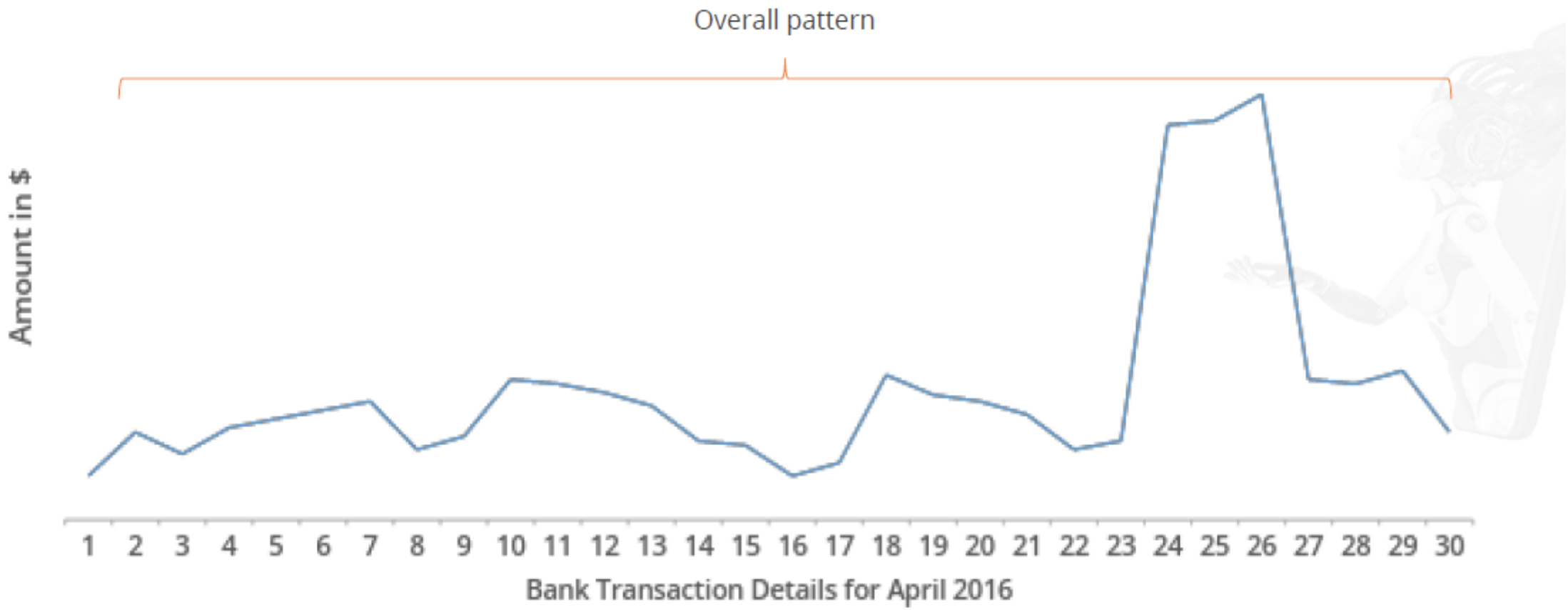
Data by itself is just an information source. But, unless you understand it, you will not be able to use it effectively.

Date	Description	Deposit	Withdrawal	Balance
Apr 1	ATM Post Debit		100	$200,000
Apr 2	PayPal Transfer 231054	200		$202,000
Apr 3	Simplilearn course fee		150	$200,500
Apr 4	Starbucks Café		210	$198,400
Apr 5	Walmart TX		230	$196,100
Apr 6	eBay swiss watch 239		250	$193,600
Apr 7	Caterpillar black boots men		270	$190,900
Apr 8	Halo blue shirt 831		160	$189,300

Information source; overall patterns not clearly visible

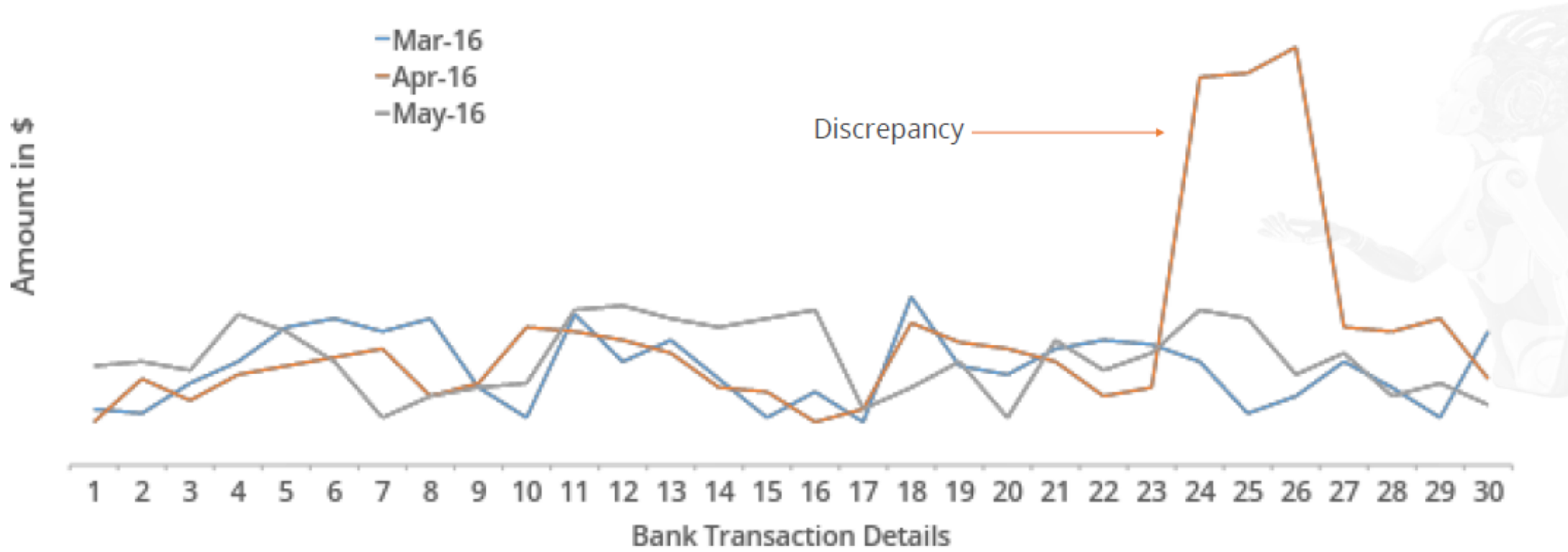
Why Data Analytics

When the transaction details are presented as a line chart, the deposit and withdrawal patterns become apparent.



Why Data Analytics

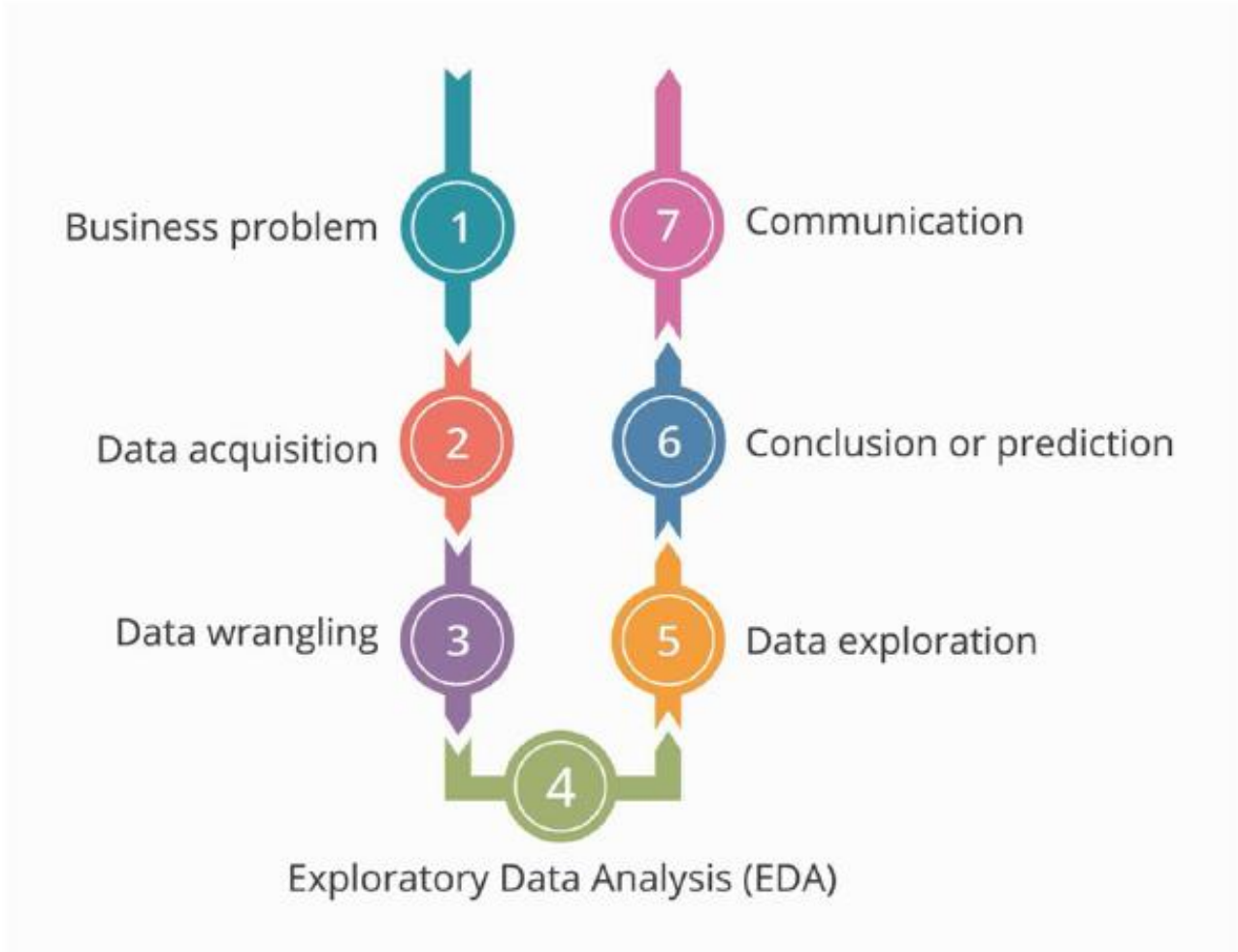
When the transaction details are presented as a line chart, the deposit and withdrawal patterns become apparent. It helps view and analyze general trends and discrepancies.



Introduction to Data Analytics

Data Analytics is a combination of processes to extract information from datasets.

Data Analytics Process



Business Problem

The process of analytics begins with questions or business problems of stakeholders.

Business problems trigger the need to analyze data and find answers.

Data Acquisition

Collect data from various sources for analysis to answer the question raised in step 1.

Data Scientist Expertise:

- File handling
- File formats
- Web scraping

Twitter, Facebook, LinkedIn, and other social media and information sites provide streaming APIs.

Server logs can be extracted from enterprise system servers to analyze and optimize application performance.

Data Wrangling and Exploration

Data wrangling is the most important phase of the data analytic process.

Data cleansing

Data manipulation

Data discovery

Data pattern

Data Wrangling

Data Exploration

Data Wrangling: Challenges

This phase includes data cleansing, data manipulation, data aggregation, data split, and reshaping of data.

Causes of challenges in the data wrangling phase:

- Unexpected data format
- Erroneous data
- Voluminous data to be manipulated
- Classifying data into linear or clustered
- Determining relationship between observation, feature, and response

Data wrangling is the most challenging phase and takes up 70% of the Data Scientist's time.

Data Exploration: Model Selection

This phase includes data cleansing, data manipulation, data aggregation, data split, and reshaping of data.

Model selection

- Based on the overall data analysis process
- Should be accurate to avoid iterations
- Depends on pattern identification and algorithms
- Depends on hypothesis building and testing
- Leads to building mathematical statistical functions

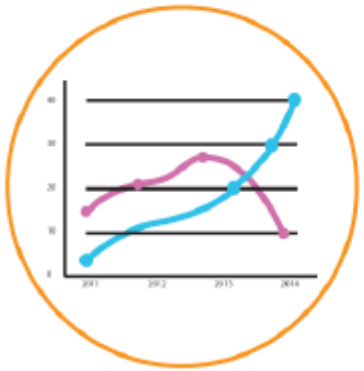
Exploratory Data Analysis (EDA)

APPROACH

EDA approach studies the data to recommend suitable models that best fit the data.

FOCUS

The focus is on data; its structure, outliers, and models suggested by the data.



ASSUMPTIONS

EDA techniques make minimal or no assumptions. They present and show all the underlying data without any data loss.

EDA TECHNIQUES

Quantitative:
Provides numeric outputs for the inputted data
Graphical:
Uses statistical functions for graphical output

EDA: Quantitative Technique

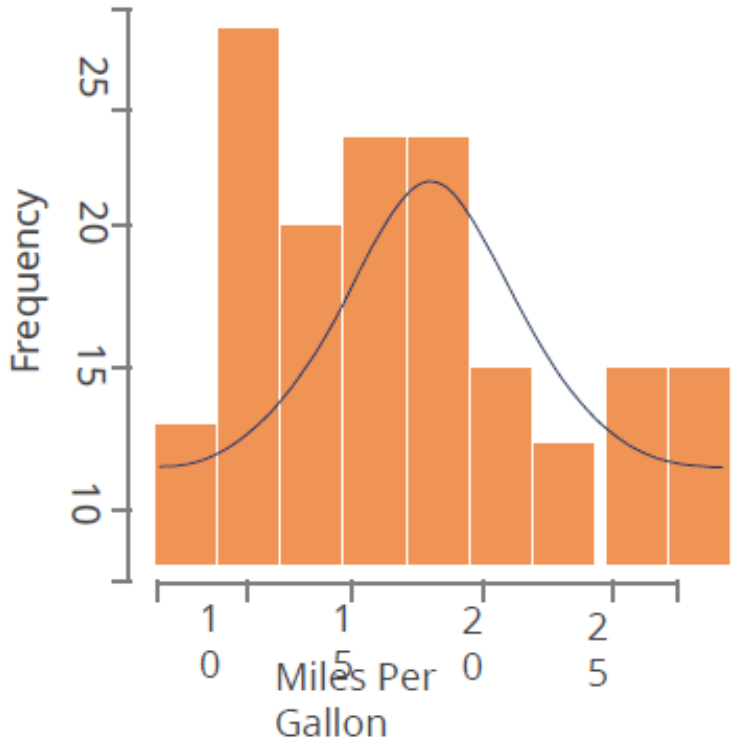
EDA: Quantitative technique has two goals, measurement of central tendency and spread of data.

	Measurement of Central Tendency
Mean	Mean is the point which indicates how centralized the data points are. • Suitable for symmetric distributions
Median	Median is the exact middle value. • Suitable for skewed distributions and for catching outliers in the dataset
Mode	Mode is the most common value in the data (frequency).

	Measurement of Spread
Variance	Variance is approximately the mean of the squares of the deviations.
Standard deviation	Standard deviation is the square root of the variance.
Inter-quartile range	Inter-quartile range is the distance between the 75^{th} and 25^{th} percentile. It's essentially the middle 50% of the data.

EDA: Graphical Technique

Histograms and scatter plots are two popular graphical techniques to depict data.



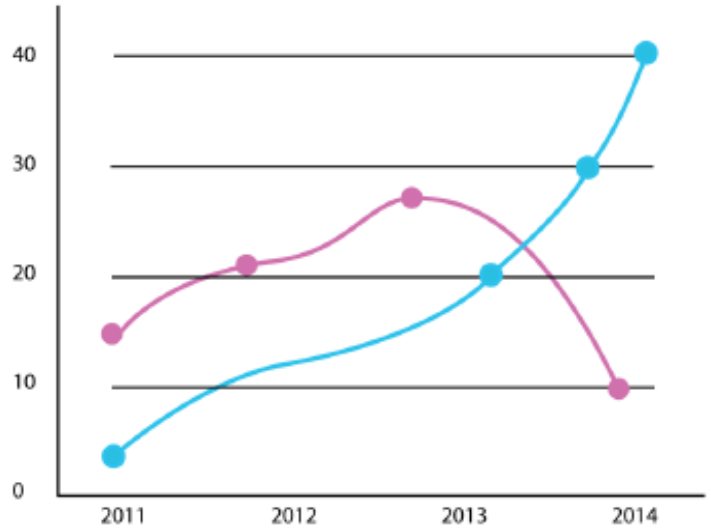
Histogram graphically summarizes the distribution of a univariate dataset.

It shows:

- the center or location of data (mean, median, or mode)
- the spread of data
- the skewness of data
- the presence of outliers
- the presence of multiple modes in the data

EDA: Graphical Technique

Histograms and scatter plots are two popular graphical techniques to depict data.



Scatter plot represents relationships between two variables.
It can answer these questions visually:

- Are variables X and Y related?
- Are variables X and Y linearly related?
- Are variables X and Y non-linearly related?
- Does change in variation of Y depend on X?
- Are there outliers?

Hypothesis

Conclusion or Prediction

This step involves reaching a conclusion and making predictions based on the data analysis.

- Involves heavy use of mathematical and statistical functions
- Requires model selection, training, and testing to help in forecasting
- Is called **machine learning** as data analysis is fully or semi-automated with minimal or no human intervention

Meaning of Hypothesis

Hypothesis is used to establish the relationship between dependent and independent variables.

Hypothesis building begins in the data exploration stage, but becomes more mature in the conclusion or prediction phase.

Data Exploration Stage

Conclusion and Prediction

Key Considerations of Hypothesis Building
Testable explanations of a problem or observation
Used in quantitative and qualitative analyses to provide research solutions
Involves two variables, one dependent on another
Independent variable manipulated by the researcher
Dependent variable changes when the independent variable changes

Hypothesis Building Using a Model

There are three phases to hypothesis building, which are model building, model evaluation, and model deployment.

Phase 1: Model Building
- Identify best input variables
- Evaluate the model's capacity to forecast with these variables

Phase 2: Model Evaluation
- Train and test the model for accuracy
- Optimize model accuracy, performance, and comparisons with other models

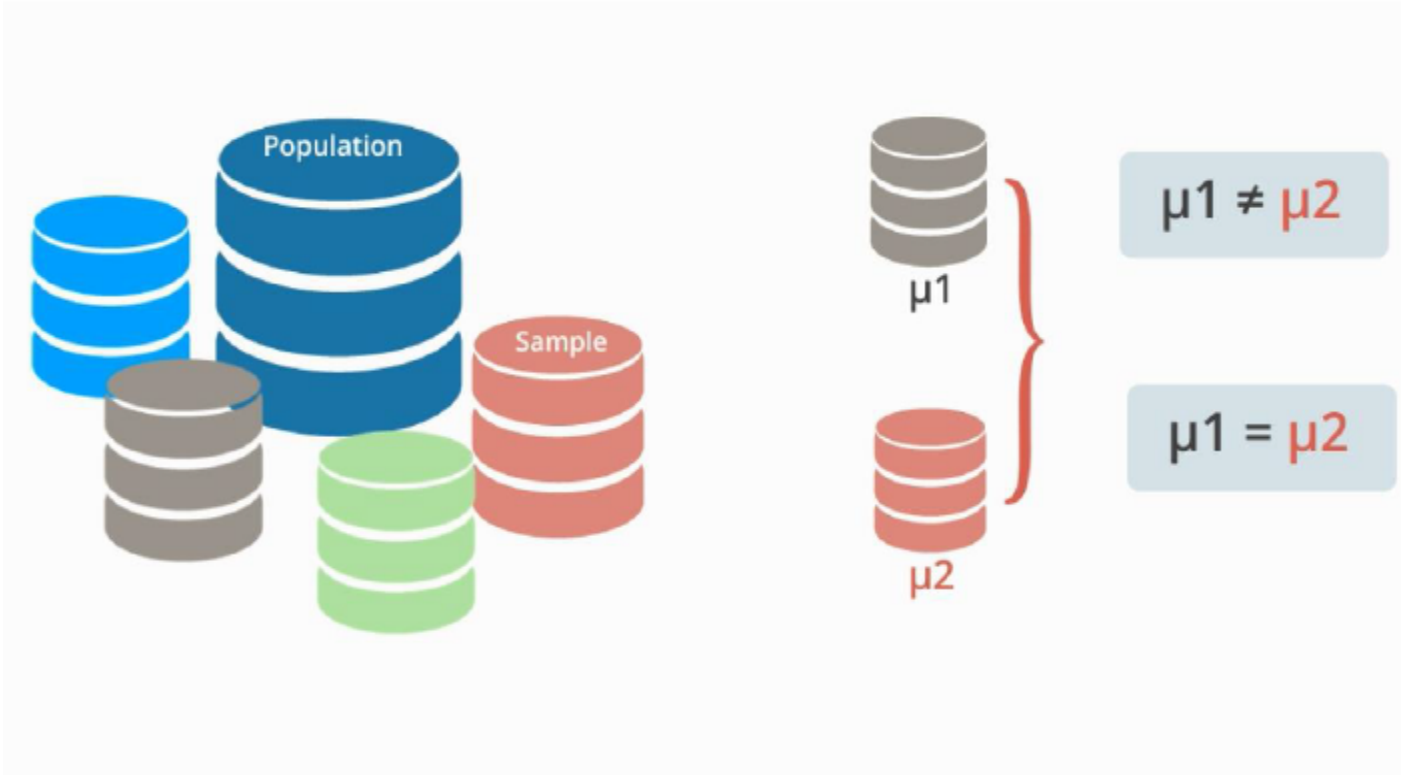
Phase 3: Model Deployment
- Use the model for prediction
- Use the model to compare actual outcome with expectations

Hypothesis Testing

Draw two samples from the population and calculate the difference between their means.

Calculating the difference between the two means is **hypothesis testing.**

Hypothesis Testing



Alternative Hypothesis

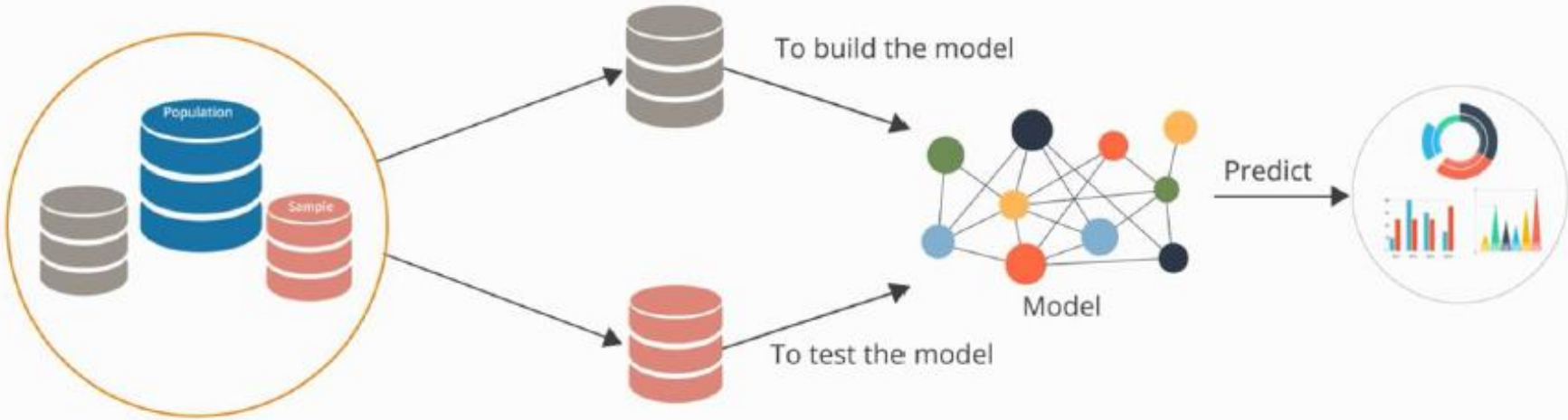
- Proposed model outcome is accurate and matches the data.
- There is a difference between the means of S1 and S2.

Null Hypothesis

- Opposite of the alternative hypothesis.
- There is no difference between the means of S1 and S2.

Hypothesis Testing Process

Choosing the training and test dataset, and evaluating them with the null and alternative hypothesis.



Null Hypothesis
Proposed model does not predict better than the existing model.

Alternative Hypothesis
Proposed model predicts better than the existing model.

Usually the training dataset is between 60% to 80% of the big dataset and the test dataset is between 20% to 40% of the big dataset.

Visualisation

Communication

The last step of data analysis is **communication**, where the analyzed data is formally presented to stakeholders.

Forms of Data analysis presentations:

- Visual graphs
- Plotting maps
- Reports
- Whitepaper reports
- PowerPoint presentations

Data Visualization

Data visualization techniques are used for effective communication of data.




Benefits of data visualization:

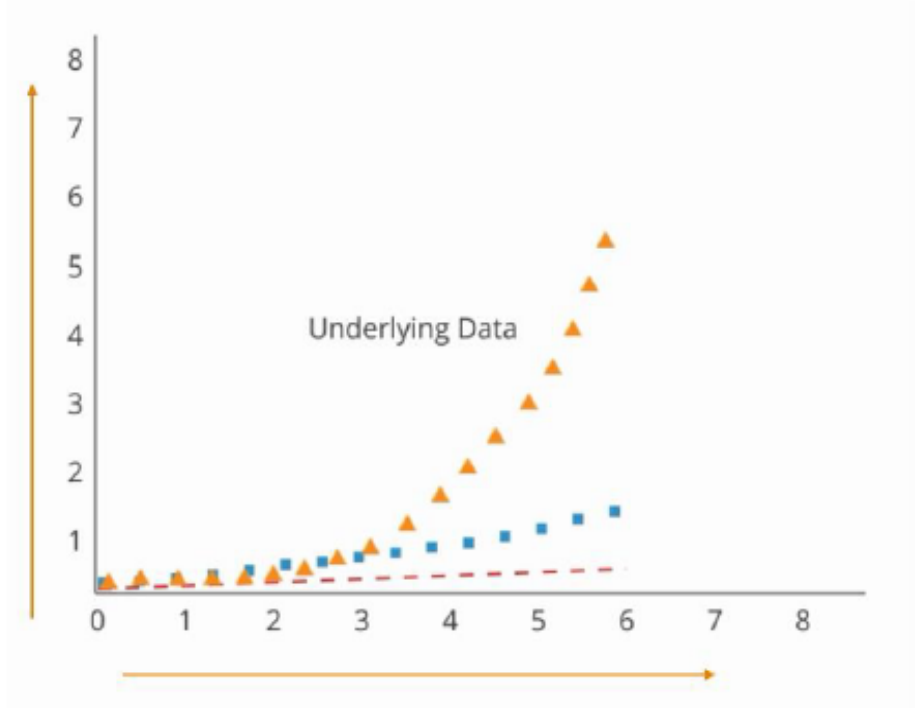
- Simplifies quantitative information through visuals
- Shows the relationship between data points and variables
- Identifies patterns
- Establishes trends

Examples of data visualization:

- Presenting information about new and existing customers on the website and their behavior when they access the website
- Representing web traffic pattern for the website, for example, more activity on the website in the morning than in the evening

Plotting

Plotting is a data visualization technique used to represent underlying data through graphics.



Features of plotting:

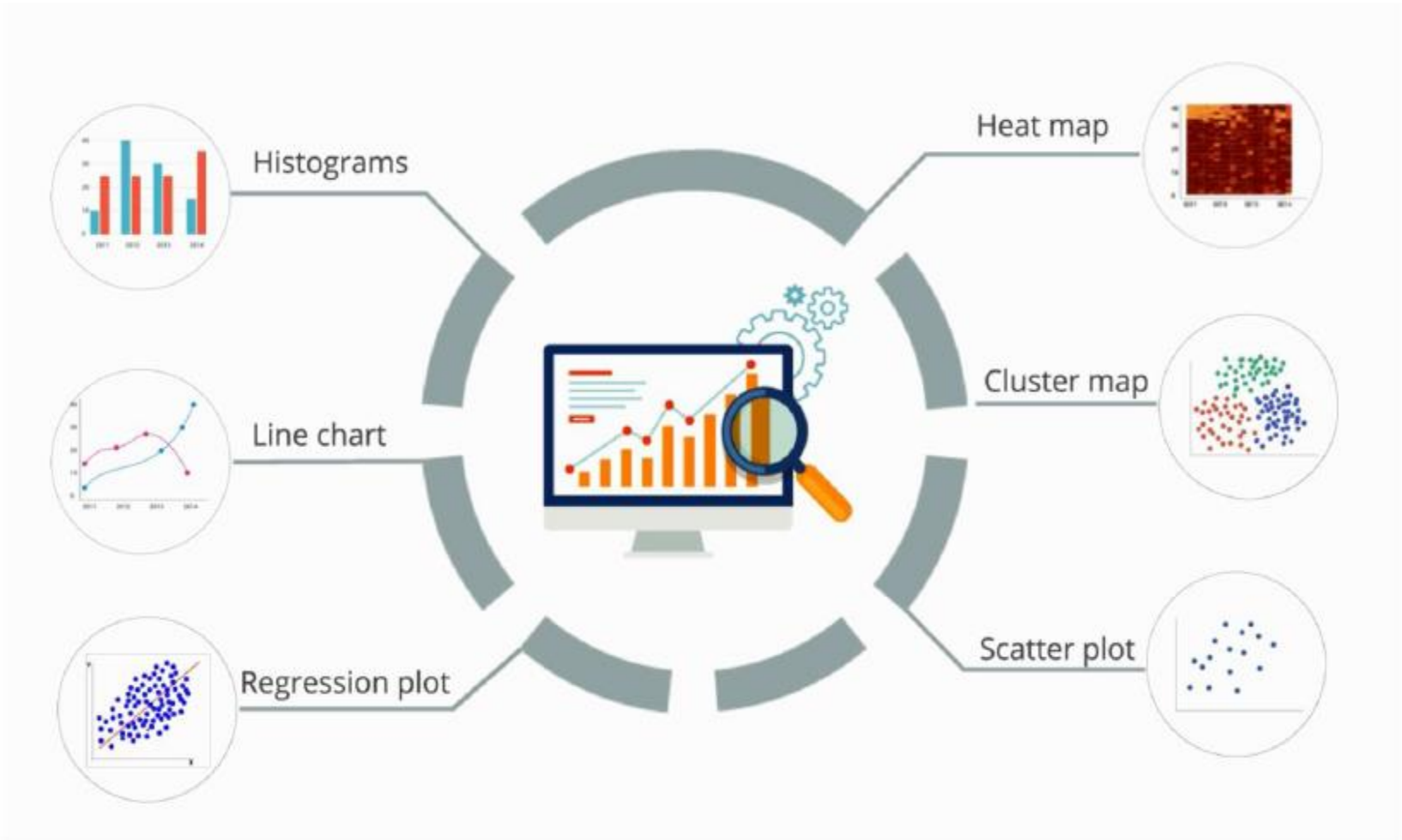
- Plotting is like telling a story about data using different colors, shapes, and sizes.
- Plotting shows the relationship between variables.
- Example:
  - Change in value of Y results in change in value of X
  - X is independent of y

Data Types for Plotting

 Numerical Data	There are two types of numerical data: Discrete Data: Distinct or counted values Example: Number of employees in a company or number of students in a class Continuous Data: Values within a range that can be measured Example: Height can be measured in feet or inches and weight can be measured in pounds or kilograms
 Categorical Data	There are two types of categorical data: Cluster or group: Grouped values Example: Students can be divided into different groups based on height: Tall, Medium, and Short Ordinal data: Grouped values as per ranks Example: A ranking system; a five-point scale with ranks like **Agree**, **Strongly agree**, and **Disagree**
 Time Series	Data is measured in time blocks, such as, date, month, year, and time (hours, minutes, and seconds

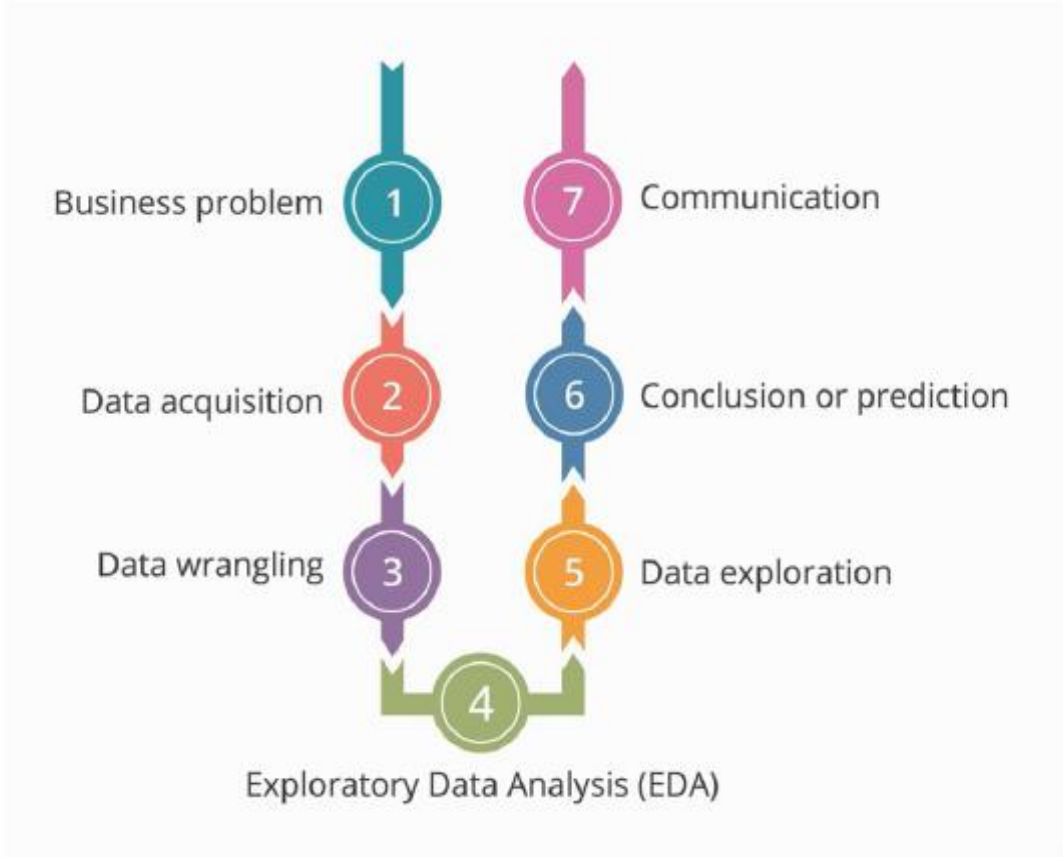
Types of Plot

Different data types can be visualized using various plotting techniques.



Data Analytics: An Iterative Process

Data Analytics is an iterative process involving tracing back the steps, often to ensure that you are on the right track.



Process Result: Question is answered or business problem is solved.

Data Analytics: Skills and Tools

Skills and tools required for each step of the data analysis process.

Question or Business Problem

- Ability to ask appropriate questions and know the business
- Domain knowledge
- Passion for data
- Analytical approach

Data Acquisition

- BeautifulSoup for web scraping
- CSV or other file knowledge
- NumPy
- Pandas
- Database

Data Wrangling

- CSV or other file knowledge
- NumPy
- Pandas
- Database
- SciPy

Data Exploration

- NumPy
- SciPy
- Pandas
- Matplotlib

Conclusion or Predictions

- Scikit-Learn – the main machine learning library
- CSV or other file knowledge
- NumPy
- Pandas
- Database
- SciPy

Communication or Data Visualization

- Pandas
- Database
- Matplotlib
- PPT
- CSV or other file knowledge

Knowledge Check 1

What is the goal of data acquisition?

Select all that apply.

a. Collect data from various data sources

b. Answer business questions through graphics

c. Collect web server logs

d. Scrape the web through web APIs

Knowledge Check 1

What is the goal of data acquisition?

Select all that apply.

a. Collect data from various data sources

b. Answer business questions through graphics

c. Collect web server logs

d. Scrape the web through web APIs

The correct answer is **a, c, d**

Data acquisition is a process to collect data from various data sources, such as RDBMS, No SQL databases, web server logs and also scrape the web through web APIs.

Knowledge Check 2

What is Exploratory data analysis technique?

Select all that apply.

a. Analysis of data using quantitative techniques

b. Conducted only on a small subset of data

c. Analysis of data using graphical techniques

d. Suggests models that best fit the data

Knowledge Check 2

What is Exploratory data analysis technique?

Select all that apply.

a. Analysis of data using quantitative techniques

b. Conducted only on a small subset of data

c. Analysis of data using graphical techniques

d. Suggests models that best fit the data

The correct answer is **a, c, d**

Most EDA techniques are graphical in nature with a few quantitative techniques and also suggest models that best fit the data. They use almost the entire data with minimum and no assumptions.

Knowledge Check 3

Which plotting technique is used for continuous data?

Select all that apply.

a. Regression plot

b. Line chart

c. Histogram

d. Heat map

Knowledge Check 3

Which plotting technique is used for continuous data?

Select all that apply.

a. Regression plot

b. Line chart

c. Histogram

d. Heat map

The correct answer is **b, c**

Line charts and histograms are used to plot continuous data.

Knowledge Check 4

Which Python library is the main machine learning library?

a. Pandas

b. Matplotlib

c. Scikit-learn

d. NumPy

Knowledge Check 4

Which Python library is the main machine learning library?

a. Pandas

b. Matplotlib

c. Scikit-learn

d. NumPy

The correct answer is **c**

SciKit-learn is the main machine learning library in Python.

Knowledge Check 5

Which of the following includes data transformation, merging, aggregation, group by operation, and reshaping?

a. Data acquisition

b. Data visualization

c. Data wrangling

d. Machine learning

Knowledge Check 5	**Which of the following includes data transformation, merging, aggregation, group by operation, and reshaping?**

a. Data acquisition

b. Data visualization

c. Data wrangling

d. Machine learning

The correct answer is **c**

Data wrangling includes data transformation, merging, aggregation, group by operation, and reshaping.

Knowledge Check 6

Which measure of central tendency is used to catch outliers in the data?

a. Mean

b. Median

c. Mode

d. Variance

Knowledge Check 6

Which measure of central tendency is used to catch outliers in the data?

a. Mean

b. Median

c. Mode

d. Variance

The correct answer is **b**

Median is the exact middle value and most suitable to catch outliers.

Knowledge Check 6	In hypothesis testing, the proposed model is built on:

a. Entire dataset

b. Test dataset

c. Small subset

d. Training dataset

Knowledge Check 6

In hypothesis testing, the proposed model is built on:

a. Entire dataset

b. Test dataset

c. Small subset

d. Training dataset

The correct answer is **d**

The proposed model is built on the training dataset in hypothesis testing.

Knowledge Check 7	Beautiful soup library is used for ____.

a. Data wrangling

b. Web scraping

c. Plotting

d. Machine learning

Knowledge Check 7

Beautiful soup library is used for ____.

a. Data wrangling

b. Web scraping

c. Plotting

d. Machine learning

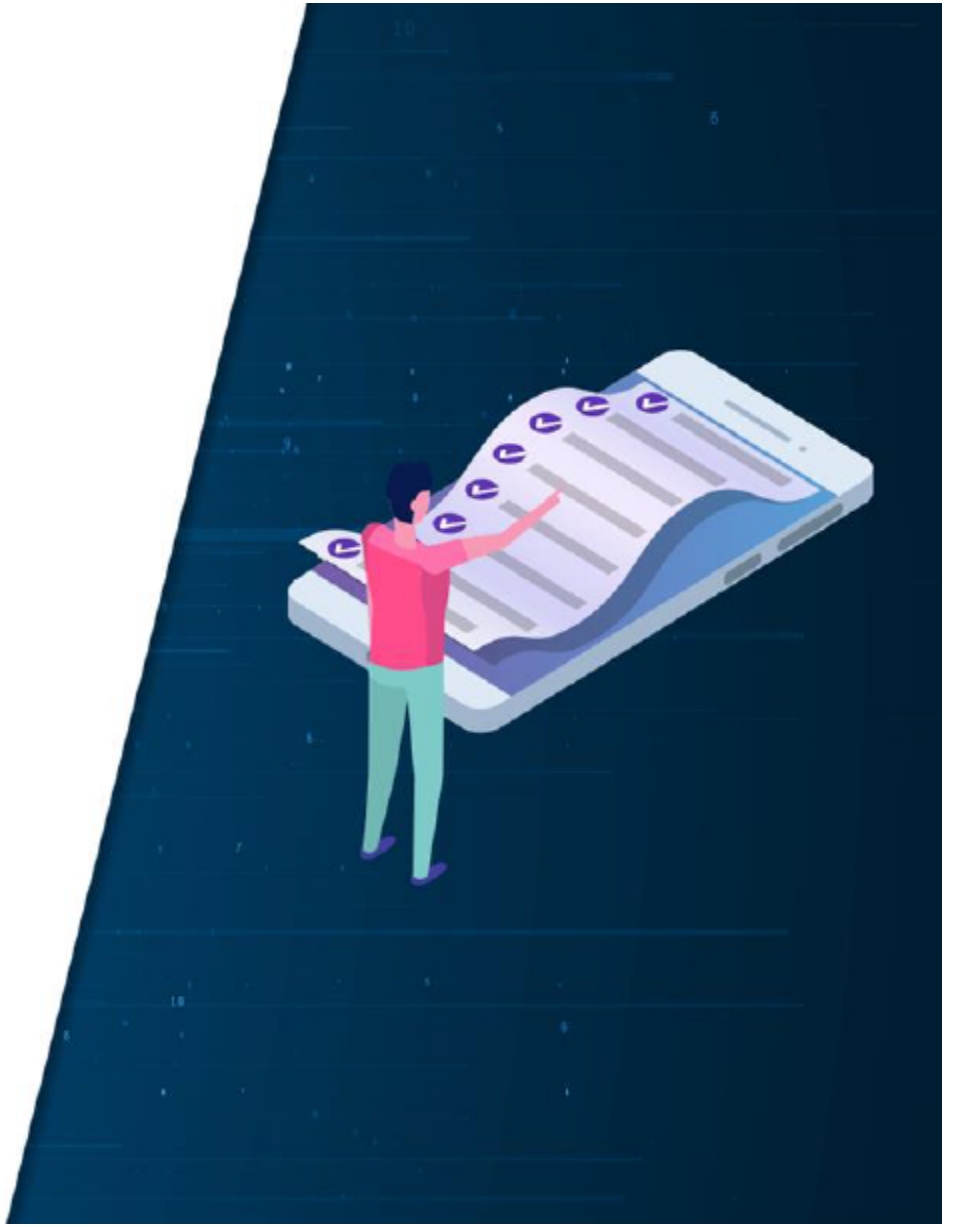
The correct answer is **b**

BeautifulSoup is used for web scraping and mainly used in the data acquisition phase.

Key Takeaways

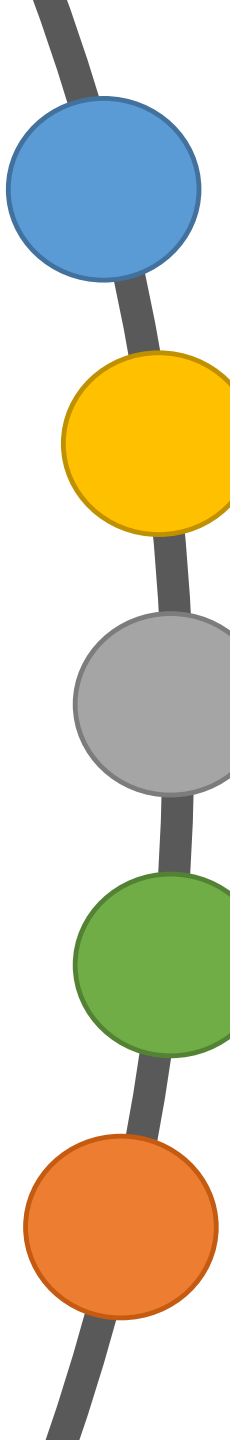
You are now able to:

- Describe Data Analytics process and its steps
- List the skills and tools required for data analysis
- Understand the challenges of the Data Analytics process
- Explain Exploratory data analysis technique
- Illustrate data visualization techniques
- Describe Hypothesis testing



Thank You……!

Tutorial Content



Overview of Python Libraries for Data Scientists

Reading Data; Selecting and Filtering the Data; Data manipulation, sorting, grouping, rearranging

Plotting the data

Descriptive statistics

Inferential statistics

Python Libraries for Data Science

Many popular Python toolboxes/libraries:

- NumPy
- SciPy
- Pandas
- SciKit-Learn

All these libraries are installed on the SCC

Visualization libraries

- matplotlib
- Seaborn

and many more …

Python Libraries for Data Science

NumPy:

- introduces objects for multidimensional arrays and matrices, as well as functions that allow to easily perform advanced mathematical and statistical operations on those objects
- provides vectorization of mathematical operations on arrays and matrices which significantly improves the performance
- many other python libraries are built on NumPy

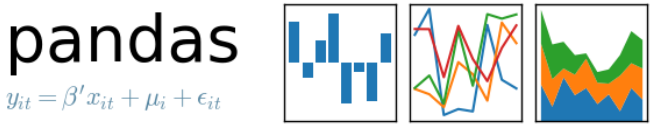
Link: http://www.numpy.org/

Python Libraries for Data Science

SciPy:

- collection of algorithms for linear algebra, differential equations, numerical integration, optimization, statistics and more
- part of SciPy Stack
- built on NumPy

Link: https://www.scipy.org/scipylib/



Python Libraries for Data Science

Pandas:

- adds data structures and tools designed to work with table-like data (similar to Series and Data Frames in R)
- provides tools for data manipulation: reshaping, merging, sorting, slicing, aggregation etc.
- allows handling missing data

Link: http://pandas.pydata.org/

Python Libraries for Data Science

SciKit-Learn:

- provides machine learning algorithms: classification, regression, clustering, model validation etc.

- built on NumPy, SciPy and matplotlib

Link: http://scikit-learn.org/

Python Libraries for Data Science

matplotlib:

- python 2D plotting library which produces publication quality figures in a variety of hardcopy formats
- a set of functionalities similar to those of MATLAB
- line plots, scatter plots, barcharts, histograms, pie charts etc.
- relatively low-level; some effort needed to create advanced visualization

Link: https://matplotlib.org/

Python Libraries for Data Science

Seaborn:

- based on matplotlib
- provides high level interface for drawing attractive statistical graphics
- Similar (in style) to the popular ggplot2 library in R

Link: https://seaborn.pydata.org/

Loading Python Libraries

In [ ]:
```
#Import Python Libraries
import numpy as np
import scipy as sp
import pandas as pd
import matplotlib as mpl
import seaborn as sns
```

Press `Shift+Enter` to execute the *jupyter* cell

Reading data using pandas

```
In [ ]: #Read csv file
        df = pd.read_csv("http://rcs.bu.edu/examples/python/data_analysis/Salaries.csv")
```

Note: The above command has many optional arguments to fine-tune the data import process.

There is a number of pandas commands to read other data formats:

```
pd.read_excel('myfile.xlsx',sheet_name='Sheet1', index_col=None, na_values=['NA'])
pd.read_stata('myfile.dta')
pd.read_sas('myfile.sas7bdat')
pd.read_hdf('myfile.h5','df')
```

Exploring data frames

In [3]:
```
#List first 5 records
df.head()
```

Out[3]:

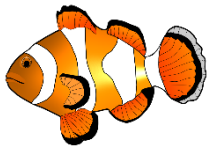
	rank	discipline	phd	service	sex	salary
0	Prof	B	56	49	Male	186960
1	Prof	A	12	6	Male	93000
2	Prof	A	23	20	Male	110515
3	Prof	A	40	31	Male	131205
4	Prof	B	20	18	Male	104800

Hands-on exercises

✓ Try to read the first 10, 20, 50 records;

✓ Can you guess how to view the last few records;

Hint:



Data Frame data types

Pandas Type	Native Python Type	Description
object	string	The most general dtype. Will be assigned to your column if column has mixed types (numbers and strings).
int64	int	Numeric characters. 64 refers to the memory allocated to hold this character.
float64	float	Numeric characters with decimals. If a column contains numbers and NaNs(see below), pandas will default to float64, in case your missing value has a decimal.
datetime64, timedelta[ns]	N/A (but see the datetime module in Python's standard library)	Values meant to hold time data. Look into these for time series experiments.

Data Frame data types

```
In [4]: #Check a particular column type
        df['salary'].dtype
```

```
Out[4]: dtype('int64')
```

```
In [5]: #Check types for all the columns
        df.dtypes
```

```
Out[4]: rank          object
        discipline    object
        phd           int64
        service       int64
        sex           object
        salary        int64
        dtype: object
```

Data Frames attributes

Python objects have *attributes* and *methods*.

df.attribute	description
dtypes	list the types of the columns
columns	list the column names
axes	list the row labels and column names
ndim	number of dimensions
size	number of elements
shape	return a tuple representing the dimensionality
values	numpy representation of the data

Hands-on exercises

- ✓ Find how many records this data frame has;
- ✓ How many elements are there?
- ✓ What are the column names?
- ✓ What types of columns we have in this data frame?

Data Frames methods

Unlike attributes, python methods have *parenthesis.*
All attributes and methods can be listed with a *dir()* function: `dir(df)`

df.method()	description
head([n]), tail([n])	first/last n rows
describe()	generate descriptive statistics (for numeric columns only)
max(), min()	return max/min values for all numeric columns
mean(), median()	return mean/median values for all numeric columns
std()	standard deviation
sample([n])	returns a random sample of the data frame
dropna()	drop all the records with missing values

Hands-on exercises

- ✓ Give the summary for the numeric columns in the dataset
- ✓ Calculate standard deviation for all numeric columns;
- ✓ What are the mean values of the first 50 records in the dataset? ***Hint:*** use head() method to subset the first 50 records and then calculate the mean

Selecting a column in a Data Frame

Method 1: Subset the data frame using column name:

df[‘column_name']

Method 2: Use the column name as an attribute:

df.column_name

Note: there is an attribute *rank* for pandas data frames, so to select a column with a name "rank" we should use method 1.

Hands-on exercises

- ✓ Calculate the basic statistics for the *salary* column;
- ✓ Find how many values in the *salary* column (use *count* method);
- ✓ Calculate the average salary;

Data Frames *groupby* method

Using "group by" method we can:

- Split the data into groups based on some criteria
- Calculate statistics (or apply a function) to each group
- Similar to dplyr() function in R

In [ ]:
```
#Group data using rank
df_rank = df.groupby(['rank'])
```

In [ ]:
```
#Calculate mean value for each numeric column per each group
df_rank.mean()
```

rank	phd	service	salary
AssocProf	15.076923	11.307692	91786.230769
AsstProf	5.052632	2.210526	81362.789474
Prof	27.065217	21.413043	123624.804348

Data Frames *groupby* method

Once groupby object is create we can calculate various statistics for each group:

```
In [ ]: #Calculate mean salary for each professor rank:
        df.groupby('rank')[['salary']].mean()
```

	salary
rank	
AssocProf	91786.230769
AsstProf	81362.789474
Prof	123624.804348

Note: If single brackets are used to specify the column (e.g. salary), then the output is Pandas Series object. When double brackets are used the output is a Data Frame

Data Frames *groupby* method

groupby performance notes:

- no grouping/splitting occurs until it's needed. Creating the *groupby* object only verifies that you have passed a valid mapping
- by default the group keys are sorted during the *groupby* operation. You may want to pass sort=False for potential speedup:

In [ ]:
```
#Calculate mean salary for each professor rank:
df.groupby(['rank'], sort=False)[['salary']].mean()
```

Data Frame: filtering

To subset the data we can apply Boolean indexing. This indexing is commonly known as a filter. For example if we want to subset the rows in which the salary value is greater than $120K:

```
In [ ]: #Calculate mean salary for each professor rank:
        df_sub = df[ df['salary'] > 120000 ]
```

Any Boolean operator can be used to subset the data:

> greater; >= greater or equal;

< less; <= less or equal;

== equal; != not equal;

```
In [ ]: #Select only those rows that contain female professors:
        df_f = df[ df['sex'] == 'Female' ]
```

Data Frames: Slicing

There are a number of ways to subset the Data Frame:

- one or more columns
- one or more rows
- a subset of rows and columns

Rows and columns can be selected by their position or label

Data Frames: Slicing

When selecting one column, it is possible to use single set of brackets, but the resulting object will be a Series (not a DataFrame):

In [ ]:
```
#Select column salary:
df['salary']
```

When we need to select more than one column and/or make the output to be a DataFrame, we should use double brackets:

In [ ]:
```
#Select column salary:
df[['rank','salary']]
```

Data Frames: Selecting rows

If we need to select a range of rows, we can specify the range using ":"

In [ ]:
```
#Select rows by their position:
df[10:20]
```

Notice that the first row has a position 0, and the last value in the range is omitted:
So for 0:10 range the first 10 rows are returned with the positions starting with 0 and ending with 9

Data Frames: method loc

If we need to select a range of rows, using their labels we can use method loc:

In [ ]:
```
#Select rows by their labels:
df_sub.loc[10:20,['rank','sex','salary']]
```

Out[ ]:

	rank	sex	salary
10	Prof	Male	128250
11	Prof	Male	134778
13	Prof	Male	162200
14	Prof	Male	153750
15	Prof	Male	150480
19	Prof	Male	150500

Data Frames: method iloc

If we need to select a range of rows and/or columns, using their positions we can use method iloc:

In [ ]:
```
#Select rows by their labels:
df_sub.iloc[10:20,[0, 3, 4, 5]]
```

Out[ ]:

	rank	service	sex	salary
26	Prof	19	Male	148750
27	Prof	43	Male	155865
29	Prof	20	Male	123683
31	Prof	21	Male	155750
35	Prof	23	Male	126933
36	Prof	45	Male	146856
39	Prof	18	Female	129000
40	Prof	36	Female	137000
44	Prof	19	Female	151768
45	Prof	25	Female	140096

Data Frames: method iloc (summary)

```
df.iloc[0]  # First row of a data frame
df.iloc[i]  #(i+1)th row
df.iloc[-1] # Last row
```

```
df.iloc[:, 0]  # First column
df.iloc[:, -1] # Last column
```

```
df.iloc[0:7]        #First 7 rows
df.iloc[:, 0:2]     #First 2 columns
df.iloc[1:3, 0:2]   #Second through third rows and first 2 columns
df.iloc[[0,5], [1,3]]  #1st and 6th rows and 2nd and 4th columns
```

Data Frames: Sorting

We can sort the data by a value in the column. By default the sorting will occur in ascending order and a new data frame is return.

In [ ]:
```
# Create a new data frame from the original sorted by the column Salary
df_sorted = df.sort_values( by ='service')
df_sorted.head()
```

Out[ ]:

	rank	discipline	phd	service	sex	salary
55	AsstProf	A	2	0	Female	72500
23	AsstProf	A	2	0	Male	85000
43	AsstProf	B	5	0	Female	77000
17	AsstProf	B	4	0	Male	92000
12	AsstProf	B	1	0	Male	88000

Data Frames: Sorting

We can sort the data using 2 or more columns:

In [ ]:
```
df_sorted = df.sort_values( by =['service', 'salary'], ascending = [True, False])
df_sorted.head(10)
```

Out[ ]:

	rank	discipline	phd	service	sex	salary
52	Prof	A	12	0	Female	105000
17	AsstProf	B	4	0	Male	92000
12	AsstProf	B	1	0	Male	88000
23	AsstProf	A	2	0	Male	85000
43	AsstProf	B	5	0	Female	77000
55	AsstProf	A	2	0	Female	72500
57	AsstProf	A	3	1	Female	72500
28	AsstProf	B	7	2	Male	91300
42	AsstProf	B	4	2	Female	80225
68	AsstProf	A	4	2	Female	77500

Missing Values

Missing values are marked as NaN

In [ ]:
```
# Read a dataset with missing values
flights = pd.read_csv("http://rcs.bu.edu/examples/python/data_analysis/flights.csv")
```

In [ ]:
```
# Select the rows that have at least one missing value
flights[flights.isnull().any(axis=1)].head()
```

Out[ ]:

	year	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	tailnum	flight	origin	dest	air_time	distance	hour	minute
330	2013	1	1	1807.0	29.0	2251.0	NaN	UA	N31412	1228	EWR	SAN	NaN	2425	18.0	7.0
403	2013	1	1	NaN	NaN	NaN	NaN	AA	N3EHAA	791	LGA	DFW	NaN	1389	NaN	NaN
404	2013	1	1	NaN	NaN	NaN	NaN	AA	N3EVAA	1925	LGA	MIA	NaN	1096	NaN	NaN
855	2013	1	2	2145.0	16.0	NaN	NaN	UA	N12221	1299	EWR	RSW	NaN	1068	21.0	45.0
858	2013	1	2	NaN	NaN	NaN	NaN	AA	NaN	133	JFK	LAX	NaN	2475	NaN	NaN

Missing Values

There are a number of methods to deal with missing values in the data frame:

df.method()	description
dropna()	Drop missing observations
dropna(how='all')	Drop observations where all cells is NA
dropna(axis=1, how='all')	Drop column if all the values are missing
dropna(thresh = 5)	Drop rows that contain less than 5 non-missing values
fillna(0)	Replace missing values with zeros
isnull()	returns True if the value is missing
notnull()	Returns True for non-missing values

Missing Values

- When summing the data, missing values will be treated as zero
- If all values are missing, the sum will be equal to NaN
- cumsum() and cumprod() methods ignore missing values but preserve them in the resulting arrays
- Missing values in GroupBy method are excluded (just like in R)
- Many descriptive statistics methods have *skipna* option to control if missing data should be excluded . This value is set to *True* by default (unlike R)

Aggregation Functions in Pandas

Aggregation - computing a summary statistic about each group, i.e.

- compute group sums or means
- compute group sizes/counts

Common aggregation functions:

min, max
count, sum, prod
mean, median, mode, mad
std, var

Aggregation Functions in Pandas

agg() method are useful when multiple statistics are computed per column:

In [ ]:
```
flights[['dep_delay','arr_delay']].agg(['min','mean','max'])
```

Out[ ]:

	dep_delay	arr_delay
min	-16.000000	-62.000000
mean	9.384302	2.298675
max	351.000000	389.000000

Basic Descriptive Statistics

df.method()	description
describe	Basic statistics (count, mean, std, min, quantiles, max)
min, max	Minimum and maximum values
mean, median, mode	Arithmetic average, median and mode
var, std	Variance and standard deviation
sem	Standard error of mean
skew	Sample skewness
kurt	kurtosis

Task to do

- Converting string/int to int/float
- Converting float to int
- Converting a column of mixed data types
- Handling missing values
- Converting a money column to float
- Converting boolean to 0/1
- Converting multiple data columns at once
- Defining data types when reading a CSV file
- Creating a custom function to convert data type
- astype() vs. to_numeric()

- import pandas as pd
  import numpy as np
- def load_df():
  return pd.DataFrame({
  'string_col': ['1','2','3','4'],
  'int_col': [1,2,3,4],
  'float_col': [1.1,1.2,1.3,4.7],
  'mix_col': ['a', 2, 3, 4],
  'missing_col': [1.0, 2, 3, np.nan],
  'money_col': ['£1,000.00','£2,400.00','£2,400.00','£2,400.00'],
  'boolean_col': [True, False, True, True],
  'custom': ['Y', 'Y', 'N', 'N']
  })
- df = load_df()

- **df.dtypes**
- df.**int_col.dtypes**
- **df.info()**

Converting string to int/float

- # string to int
  >>> df['string_col'] = df['string_col']**.astype('int')**
  >>> df.dtypes
- # string to float
  >>> df['string_col'] = df['string_col']**.astype('float')**
  >>> df.dtypes

- **2. Converting float to int**

df['float_col'] = df['float_col']**.astype('int')**

df['float_col'] = df['float_col']**.round(0)**.astype('int')

Getting ValueError
df['mix_col'] = df['mix_col'].astype('int')

- The error shows it’s the problem with the value 'a' as it cannot be converted to an integer. In order to get around this problem, we can use Pandas to_numeric() function with argument errors='coerce'.
- df['mix_col'] = pd.**to_numeric**(df['mix_col'], **errors='coerce'**)
- But when checking the dtypes, you will find it get converted to float64.
- >>> df['mix_col'].dtypes
- dtype('**float64**')

- In some cases, you don’t want to output to be float values you want it to be integers, for instance converting an ID column. We can call astype('Int64').
- Note it has a capital I and is different than Numpy 'int64'. What this does is change Numpy’s NaN to Pandas’ NA and this allows it to be an integer.
- >>> df['mix_col'] = pd.to_numeric(df['mix_col'], errors='coerce').**astype('Int64')**>>> df['mix_col'].dtypes
- Alternatively, we can replace Numpy nan with another value (for example replacing NaN with 0) and call astype('int')
- df['mix_col'] = pd.to_numeric(df['mix_col'], errors='coerce').**fillna(0)**.**astype('int')**

Handling Missing value

Now we should be fully equipped with dealing with the missing values. In Pandas, missing values are given the value NaN, short for “Not a Number”. For technical reasons, these NaN values are always of the float64.

- df.**missing_col.dtypes**
- dtype('**float64**')

To get around the error, we can call astype('Int64') as we did above (Note it is captial I, same as mentioned in the last section). What this does is change Numpy’s NaN to Pandas’ NA and this allows it to be an integer.

- df['missing_col'] = df['missing_col'].**astype('Int64')**

Alternatively, we can replace Numpy NaN with another value (for example replacing NaN with 0) and call astype('int')

- df['mix_col'] = df['missing_col'].**fillna(0).astype('int')**

Converting a money column to numbers

Let’s move on to the money column. The problem is that if we are using the method above we’re going to get all NaN or NA values because they are all strings with symbols £ and ,, and they can’t be converted to numbers. So the first what we have to do is removing all invalid symbols.

- >>> df['money_replace'] = df['money_col'].**str.replace('£', '').str.replace(',','')**
  >>> df['money_replace'] = pd.to_numeric(df['money_replace'])
  >>> df['money_replace']

We chain 2 replace() calls, one for £ and the other for ,, to replace them with an empty string.

If you are familiar with regular expression, we can also replace those symbols with a regular expression.

- >>> df['money_regex'] = df['money_col'].str.replace(**'[\£\,]', '', regex=True**)
  >>> df['money_regex'] = pd.to_numeric(df['money_replace'])
  >>> df['money_regex']

Converting boolean to 0/1

We have True/False, but you can imagine a case in which need these as 0 and 1 , for instance, if you are building a machine learning model and this is one of your input features, you'd need it to be numeric and you would use 0 and 1 to represent False and True. This is actually very simple, you can just call astype('int'):

- df['boolean_col'] = df['boolean_col']**.astype('int')**

Converting multiple column data types at once

So far, we have been converting data type one column at a time. For instance

- # Converting column string_col and int_col one at a time
  df['string_col'] = df['string_col'].astype('float16')
  df['int_col'] = df['int_col'].astype('float16')

There is a DataFrame method also called astype() allows us to convert multiple column data types at once. It is time-saving when you have a bunch of columns you want to change.

- df = df.astype({
  'string_col': 'float16',
  'int_col': 'float16'
  })

Defining the data type of each column when reading a CSV file

If you want to set the data type for each column when reading a CSV file, you can use the argument dtype when loading data with read_csv():

- df = pd.read_csv(
  'dataset.csv',
  dtype={
  'string_col': 'float16',
  'int_col': 'float16'
  }
)

Creating a custom function to convert data to numbers

When data is a bit complex to convert, we can create a custom function and apply it to each value to convert to the appropriate data type.

For instance, the **money_col** column, here is a simple function we can use:

- >>> **def convert_money(value):**
  value = value.replace('£','').replace(',', '')
  return float(value)>>> df['money_col']**.apply(convert_money)**

We can also use a lambda function:

- df['money_col']
  .apply(**lambda v: v.replace('£','').replace(',','')**)
  .astype('float')

Difference between astype() and to_numeric()

- The simplest way to convert data type from one to the other is to use astype() method. The method is supported by both Pandas DataFrame and Series. If you already have a numeric data type (int8, int16, int32, int64, float16, float32, float64, float128, and boolean) you can also use astype() to:
- convert it to another numeric data type (int to float, float to int, etc.)
- use it to downcast to a smaller or upcast to a larger byte size
- However, astype() won't work for a column of mixed types. For instance, the **mixed_col** has a and **missing_col** has NaN. If we try to use astype() we would get a **ValueError**. As of Pandas 0.20.0, this error can be suppressed by setting the argument errors='ignore', but your original data will be returned untouched.

Graphics to explore the data

Seaborn package is built on matplotlib but provides high level interface for drawing attractive statistical graphics, similar to ggplot2 library in R. It specifically targets statistical data visualization

To show graphs within Python notebook include inline directive:

```
In [ ]: %matplotlib inline
```

Graphics

	description
distplot	histogram
barplot	estimate of central tendency for a numeric variable
violinplot	similar to boxplot, also shows the probability density of the data
jointplot	Scatterplot
regplot	Regression plot
pairplot	Pairplot
boxplot	boxplot
swarmplot	categorical scatterplot
factorplot	General categorical plot

Basic statistical Analysis

statsmodel and scikit-learn - both have a number of function for statistical analysis

The first one is mostly used for regular analysis using R style formulas, while scikit-learn is more tailored for Machine Learning.

statsmodels:

- linear regressions
- ANOVA tests
- hypothesis testings
- many more ...

scikit-learn:

- kmeans
- support vector machines
- random forests
- many more ...

See examples in the Tutorial Notebook