Data Science Tools, Part 2: Pandas, Matplotlib

Upcoming Assignments

- Reading: Chapter 2 Reading Part 2
- DS Lab 3: Intro to Pandas due Jan 24th 11:59 pm

Class #	Week #	Month	Date	Topic	Reading	Labs
1	1	Jan	8	Welcome, Introduction, Course Objectives, DS Lifecycle	Chapter 1 Intro DS	Lab 1: Colab Set Up, GitHub
2	2	Jan	13	Python setup, Google colab, Github		
3	2	Jan	15	NumPy, Vectorization	Chapter 2.1-2.5 Python	Lab 2: Vectorization
4	3	Jan	20	Pandas, Matplotlib, Seaborn		
5	3	Jan	22	Data Cleaning and Preparation	Chapter 2.6-2.8 Python	Lab 3: NumPy, Pandas
6	4	Jan	27	Data Acquisition, ETL, Populations, Sampling	Chapter 3 Data Prep	
7	4	Jan	29	Descriptive Statistics	Chapter 4 Prob & Stat	Lab 4: Data Preparation
8	5	Feb	3	Exploratory Data Analysis (EDA)		
9	5	Feb	5	Principles of Data Visualization	Chapter 5 EDA	
10	6	Feb	10	Data management - databases, SQL queries		
11	6	Feb	12	More SQL Features, Joins	Chapter 7 D[illegible] an	
12	7	Feb	17	**MONDAY SCHEDULE, NO CLASS**		
13	7	Feb	19	SQLite		
14	8	Feb	24	MIDTERM REVIEW		
15	8	Feb	26	MIDTERM		
16	9	Mar	3	Overview of ML		
17	9	Mar	5	Unsupervised Learning- Kmeans	Chapter 8 Unsupervised Learning	
18	10	Mar	10	Unsupervised Learning- Hierarchical, DBSCAN		
19	10	Mar	12	Supervised Learning: Part 1	Chapter 9 Supervised Learn	Lab 8: Cluster Analysis
20	11	Mar	17	Supervised Learning: Part 2		
21	11	Mar	19	Evaluation of models, comparing performance	Chapter 10 Decision Trees	Lab 9: ML Classification/Regression
22	12	Mar	24	Feature Importance with RF and Logistic Regression		
23	12	Mar	26	ANN, Multi-Layer Perceptron, Backpropagation	Chapter 12 Eval	**New Lab?**
24	13	Mar	31	Deep Learning		
25	13	Apr	2	GenAI - Introduction	Chapter 13 ANN	Lab 10: MLP and Backpropagation

Today

- Pandas
- Visualizations
  - Matplotlib
  - Seaborn

Tim Kapp
tkapp@byu.edu
TMCB 2254 — By Appointment

Teaching Assistants:

- Kayla Ou -- ouj22@byu.edu
- Spencer Hales — srhales@byu.edu
- Spencer Marshall — sm892@byu.edu
- Michael Jensen — mikelj01@student.byu.edu
- Toby Alley — talley0@byu.edu

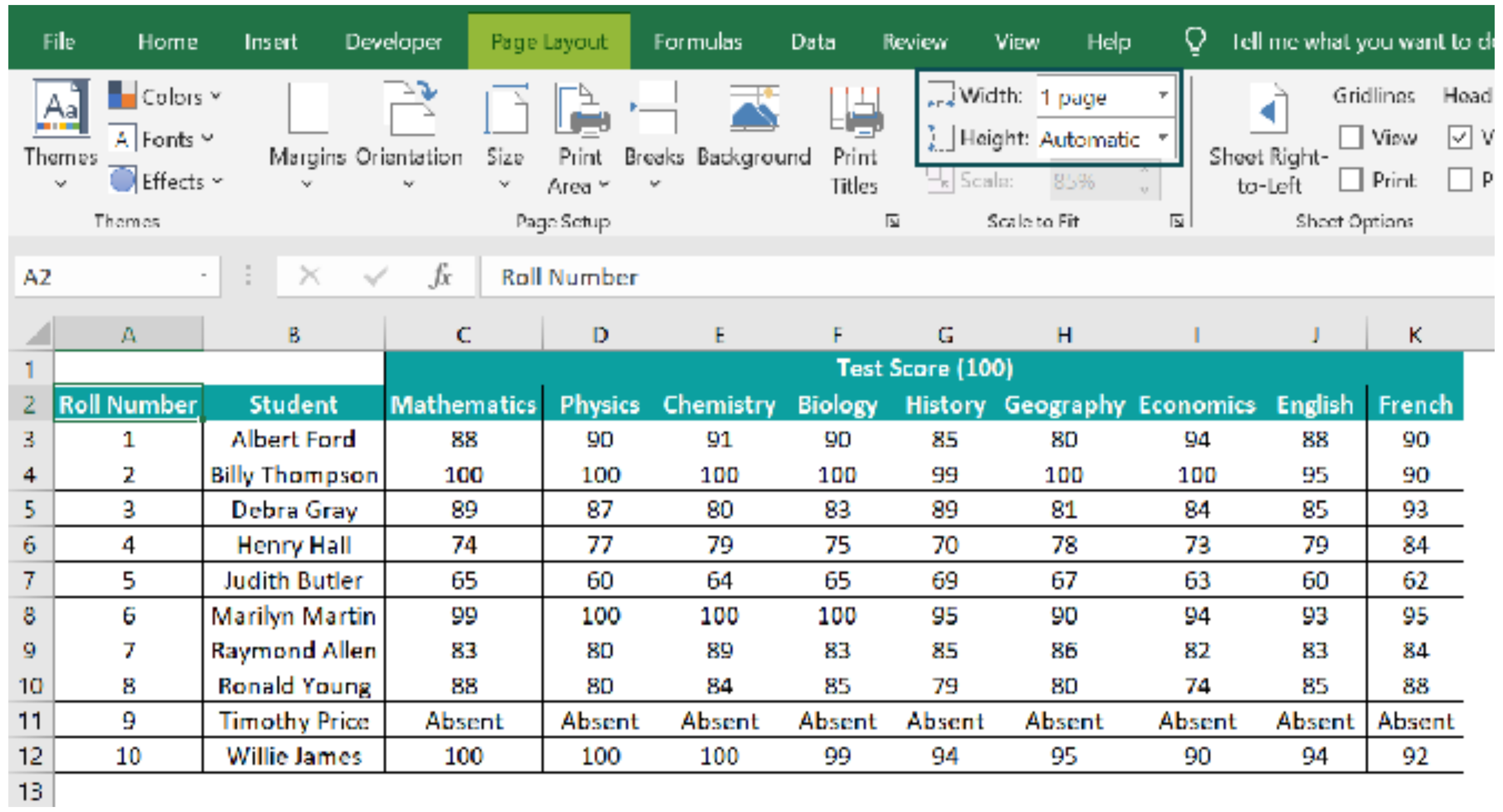
West View Building (WVB 1151)
Office Hours: See Syllabus on Canvas

Pre-Quiz: How much do you already know about Pandas?

- What is python pandas?
- How is a pandas DataFrame different than a NumPy array?
- Name three unique operations (i.e., methods) you can do with pandas?
- How do you read from or write to a csv file using pandas?
- Keep this Quiz and see if you can fill-in any missing questions during the discussion today

What is Pandas?

- Think of **pandas** as **Excel for Python**.
- If you’ve ever used **Excel** or **Google Sheets** — adding columns, filtering rows, making summaries — pandas does the same, but with **code**.



		Test Score (100)								
Roll Number	Student	Mathematics	Physics	Chemistry	Biology	History	Geography	Economics	English	French
1	Albert Ford	88	90	91	90	85	80	94	88	90
2	Billy Thompson	100	100	100	100	99	100	100	95	90
3	Debra Gray	89	87	80	83	89	81	84	85	93
4	Henry Hall	74	77	79	75	70	78	73	79	84
5	Judith Butler	65	60	64	65	69	67	63	60	62
6	Marilyn Martin	99	100	100	100	95	90	94	93	95
7	Raymond Allen	83	80	89	83	85	86	82	83	84
8	Ronald Young	88	80	84	85	79	80	74	85	88
9	Timothy Price	Absent	Absent	Absent	Absent	Absent	Absent	Absent	Absent	Absent
10	Willie James	100	100	100	99	94	95	90	94	92

```
df.head()
```

	Job #	Doc #	Borough	Initial Cost	Total Est. Fee
0	121577873	2	MANHATTAN	$75000.00	$986.00
1	520129502	1	STATEN ISLAND	$0.00	$1144.00
2	121601560	1	MANHATTAN	$30000.00	$522.50
3	121601203	1	MANHATTAN	$1500.00	$225.00
4	121601338	1	MANHATTAN	$19500.00	$389.50

Pandas Definition

- Open-Source software library written for Python
- Pandas derived from the term “**pan**el **da**ta” from econometrics
- Data structures and operations for manipulating numerical tables and time series
- DataFrame is a 2-Dimensional structure built as a combination of Series arrays with a shared index
- Built on NumPy
- Originally released 11 Jan 2008. The current stable release is **version 2.3.2**, released August 21, 2025.

https://en.wikipedia.org/wiki/Pandas_(software)

Pandas Series

A **Pandas Series** is a one-dimensional labeled array, capable of holding data of any type (integers, floats, strings, Python objects, etc.).

Key features of a pandas Series:

- **Indexing**: Each element in the Series has a corresponding index, which allows for easy access and manipulation of data. Default is numeric indexing.
- **Homogeneous**: The Series can hold elements of the same type (though mixed types are possible, but uncommon and not recommended practice).

python

```python
import pandas as pd

# Create a pandas Series
data = pd.Series([10, 20, 30, 40], index=['a', 'b', 'c', 'd'])

print(data)
```

This will output:

```
a    10
b    20
c    30
d    40
dtype: int64
```

Pandas Dataframe

- Defined as multiple **Series** objects that share an index
- A **Pandas DataFrame** is a two-dimensional, labeled data structure in Python, similar to a table or spreadsheet, that stores data in rows and columns. Each column in a DataFrame can have a different data type (e.g., integers, floats, strings, etc.).

Key features of a DataFrame:

- **Rows and Columns:** Like a table, with rows representing individual records and columns representing variables or features.
- **Labels:** Rows and columns can have labels (names), making it easy to access, slice, or manipulate data. Index and Column for row/column labels
- **Data Handling:** It can handle missing data and supports arithmetic operations, data filtering, aggregation, and transformation.
- **Data Input/Output:** Can read and write data from various file formats (e.g., CSV, Excel, SQL, etc.).

Example:

python

```python
import pandas as pd

# Create a DataFrame
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'City': ['New York', 'San Francisco', 'Los Angeles']
}

df = pd.DataFrame(data)

print(df)
```

This code would output:

```
      Name  Age           City
0    Alice   25       New York
1      Bob   30  San Francisco
2  Charlie   35    Los Angeles
```

Subsetting Data

- ***Subsetting data*** involves choosing specific rows and columns from a dataframe according to labels, indices, and slices.
- A single column can be selected by using the label of the desired column. Ex: Using the country dataset assigned to the variable country, the Population column can be selected using the country['Population'] or country.Population. Multiple columns can also be selected by using an array of strings. Ex: country[['Name', 'Population']]
- The iloc(x,y) method for a dataframe is used to select an individual element using an index location, where x is the row and y is the column. Ex: country.iloc[0,1] returns the element in row 0 and column 1. The colon character : is used in slice notation to select multiple rows or columns. Ex: country.iloc[:5,1:3] returns rows before row 5 and columns 1 thru 2.
- The loc(x,y) method can also be used to subset data, but y, in this case, is an array of column labels, instead of an integer or a range of integers. Ex: Both country.iloc[:7,1:3] and country.loc[:6,['Continent','Population']] give the same results.

Conditional Filtering

- Comparison and logical operators can be used to subset data. When these operators are used, only rows for which the expression is true will be returned. Ex: country[country['Population'] > 100000] will display rows whose 'Population' column values are greater than 100,000.

Missing Values

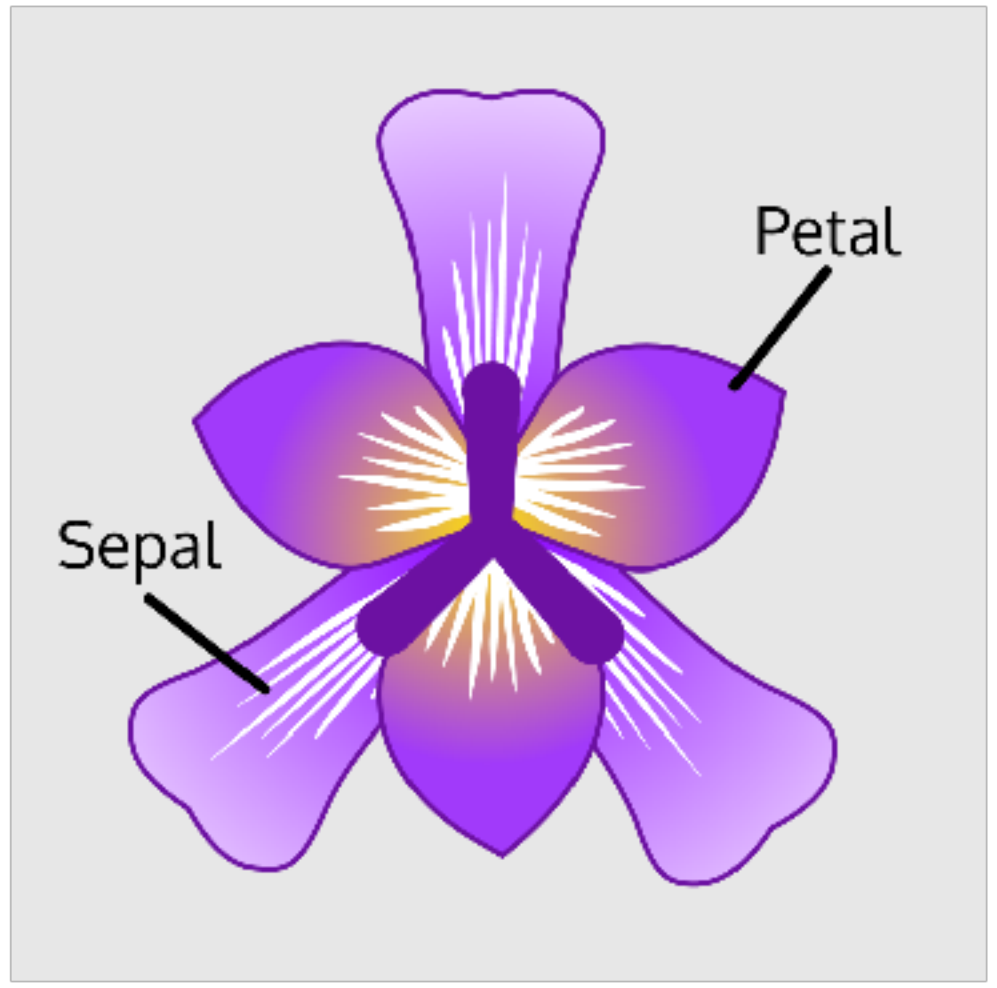
There are several methods to deal with missing values in the data frame:

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

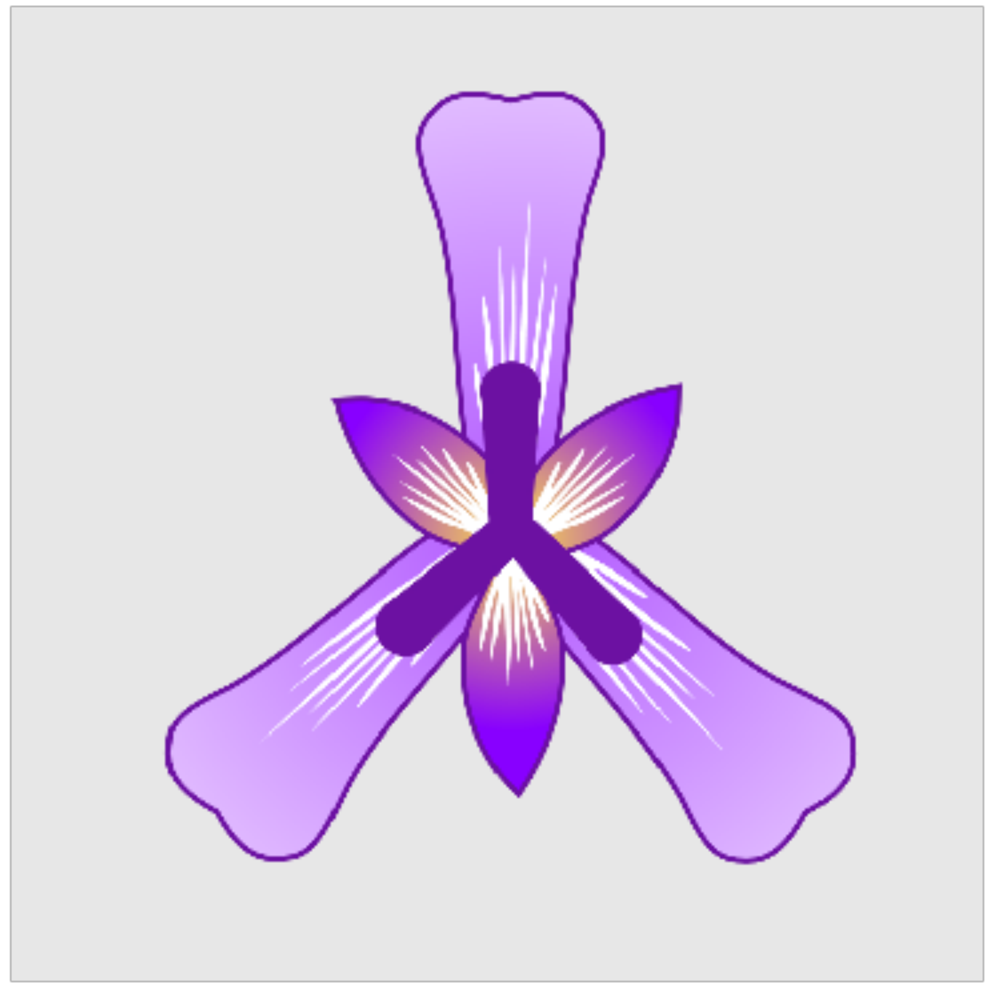
Iris Flower Data Set

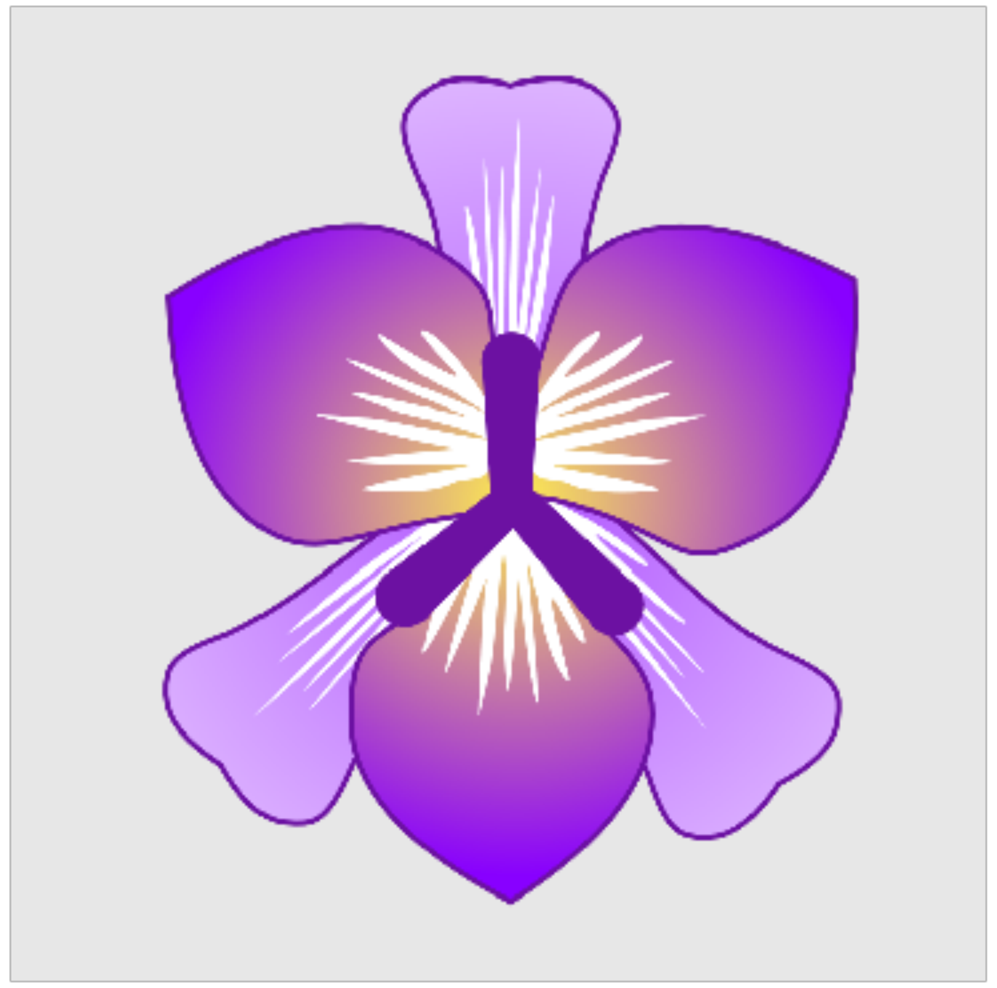
- The ***Iris* flower data set** was made famous by the British statistician and biologist Ronald Fisher in 1936.
- It is sometimes called Anderson's Iris data set because Edgar Anderson collected the data to quantify the morphologic variation of Iris flowers of three related species.
- The dataset contains a set of 150 records with five attributes: sepal length, sepal width, petal length, petal width and species.
- The iris data set is widely used for teaching machine learning. The dataset is included in Python in the machine learning library scikit-learn.

https://en.wikipedia.org/wiki/Iris_flower_data_set#External_links



Iris Versicolor


Iris Setosa


Iris Virginica

```
from sklearn.datasets import load_iris

iris = load_iris()
iris
```

This code gives:

```
{'data': array([[5.1, 3.5, 1.4, 0.2],
               [4.9, 3., 1.4, 0.2],
               [4.7, 3.2, 1.3, 0.2],
               [4.6, 3.1, 1.5, 0.2],...
'target': array([0, 0, 0, ... 1, 1, 1, ... 2, 2, 2, ...
'target_names': array(['setosa', 'versicolor', 'virginica'],
dtype='<U10'),
...}
```

Groupby General Concept

Student	Subject	Score
John	Math	80
Alice	Science	90
Bob	Science	70
John	Math	75
Bob	Science	95
Alice	Math	85

groupby('Student')

John	Math	80
John	Math	75

Alice	Science	90
Alice	Math	85

Bob	Science	70
Bob	Science	95

Aggregate Function
.mean()

Student	
Alice	87.5
Bob	82.5
John	77.5

Quiz: How much do you now know about Pandas?

- What is Python Pandas?
- How is a Pandas DataFrame different than a NumPy array?
- Name three unique operations (i.e., methods) you can do with Pandas?
- How do you read from or write to a csv file using Pandas?

Matplotlib

Matplotlib is a popular Python library used for creating static, animated, and interactive visualizations. Matplotlib can produce a variety of plots, such as:

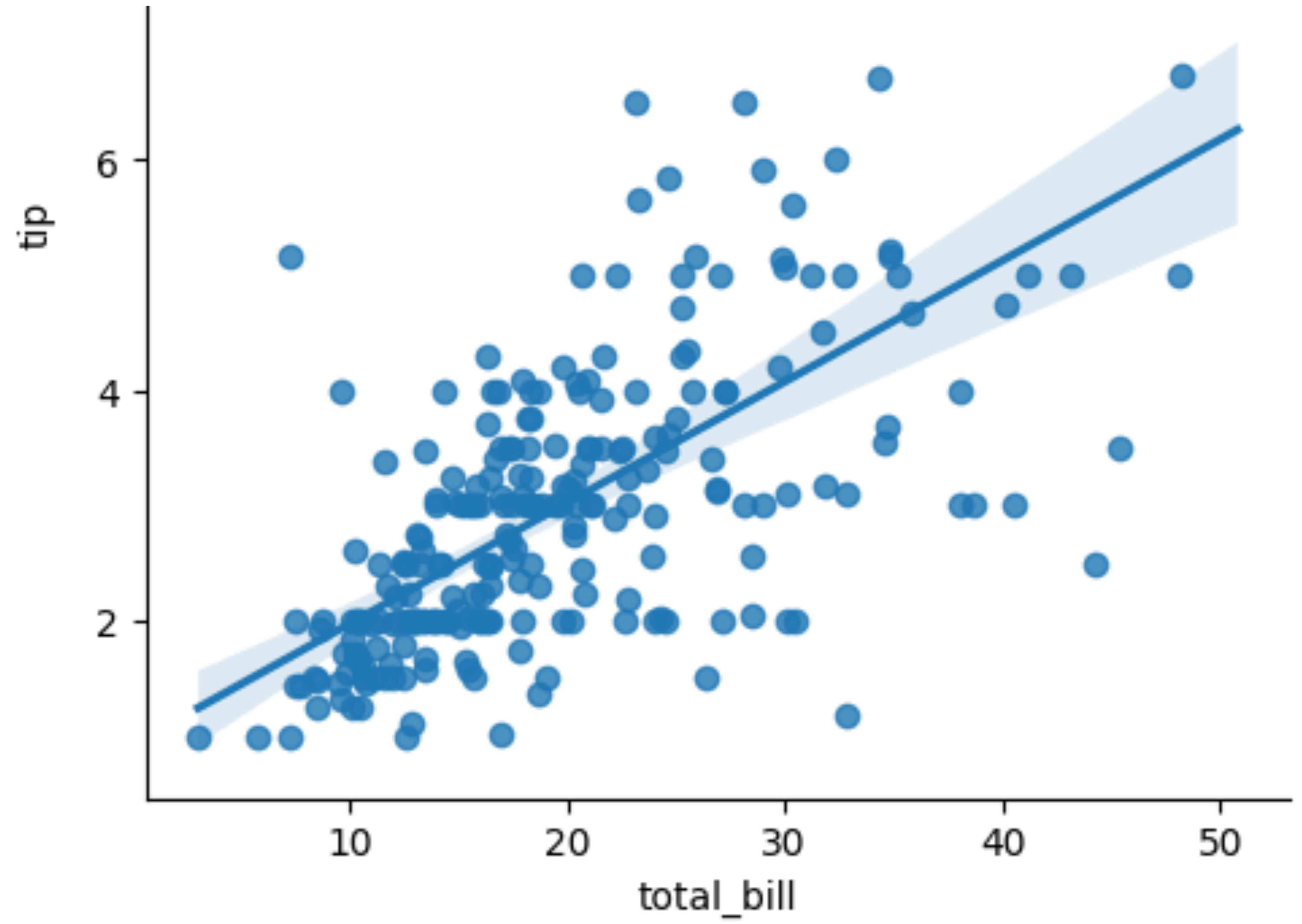
- Line plots
- Bar charts
- Scatter plots
- Histograms
- Pie charts
- 3D plots

It works closely with other libraries like NumPy for numerical computations and Pandas for handling data structures.

seaborn

- Seaborn is a Python data visualization library built on top of Matplotlib, designed to make it easier to create informative and aesthetically pleasing statistical graphics.
- Seaborn integrates closely with Pandas data structures, which makes it especially powerful for working with data frames and structured data.
- **Plot Types**: Seaborn supports many types of plots, such as:
  - Line plots
  - Bar plots
  - Scatter plots
  - Heatmaps
  - Pair plots
  - Box plots
  - Violin plots

```
import seaborn as sns
import matplotlib.pyplot as plt

# Load an example dataset from Seaborn
tips = sns.load_dataset("tips")

# Create a scatter plot with a linear fit
sns.lmplot(x="total_bill", y="tip", data=tips)

# Display the plot
plt.show()
```



Box and Whisker Plot

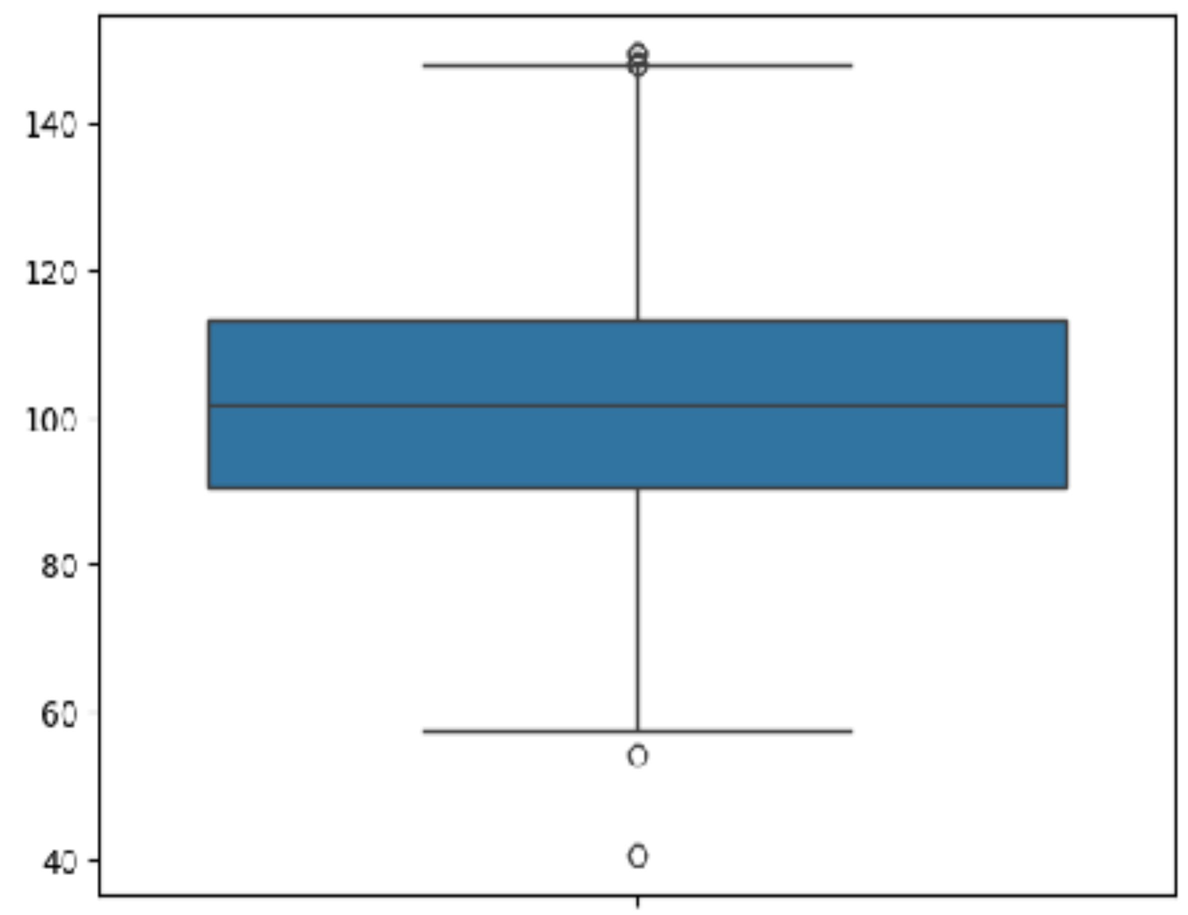
A **boxplot** is a standardized way of displaying the distribution of data based on a five-number summary: minimum, first quartile (Q1), median, third quartile (Q3), and maximum. It helps to show the spread and skewness of the data, highlighting potential outliers.

Key Elements :

1. **Box**: Spans from Q1 to Q3 (the interquartile range, IQR).
2. **Median**: A line inside the box showing the middle of the dataset.
3. **Whiskers**: Lines extending from Q1 to the minimum and Q3 to the maximum values within 1.5 times the IQR.
4. **Outliers**: Data points that fall outside the whiskers.

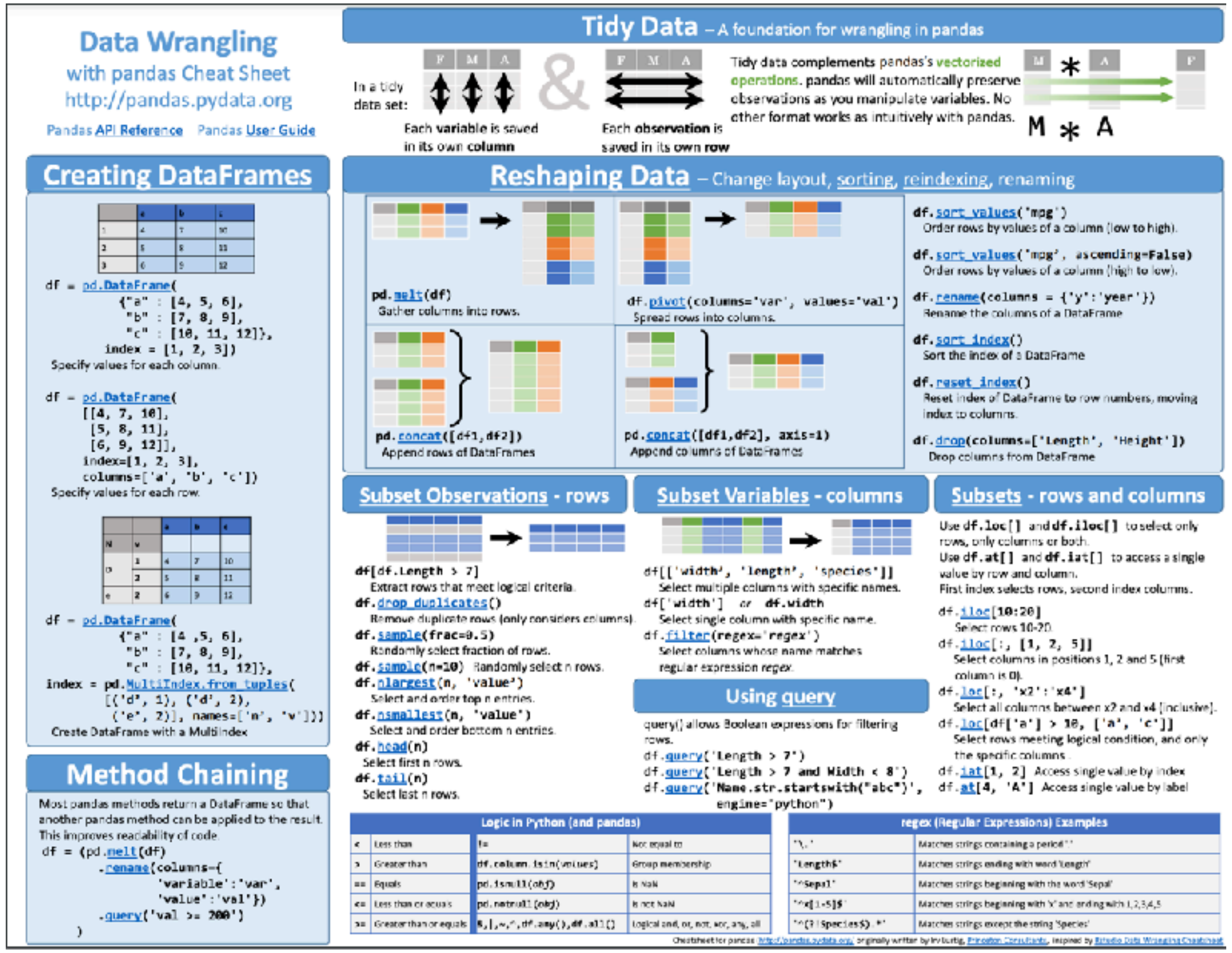
By combining these elements, a box plot quickly provides insights into the data's **distribution, variability, and any unusual observations** (like outliers).

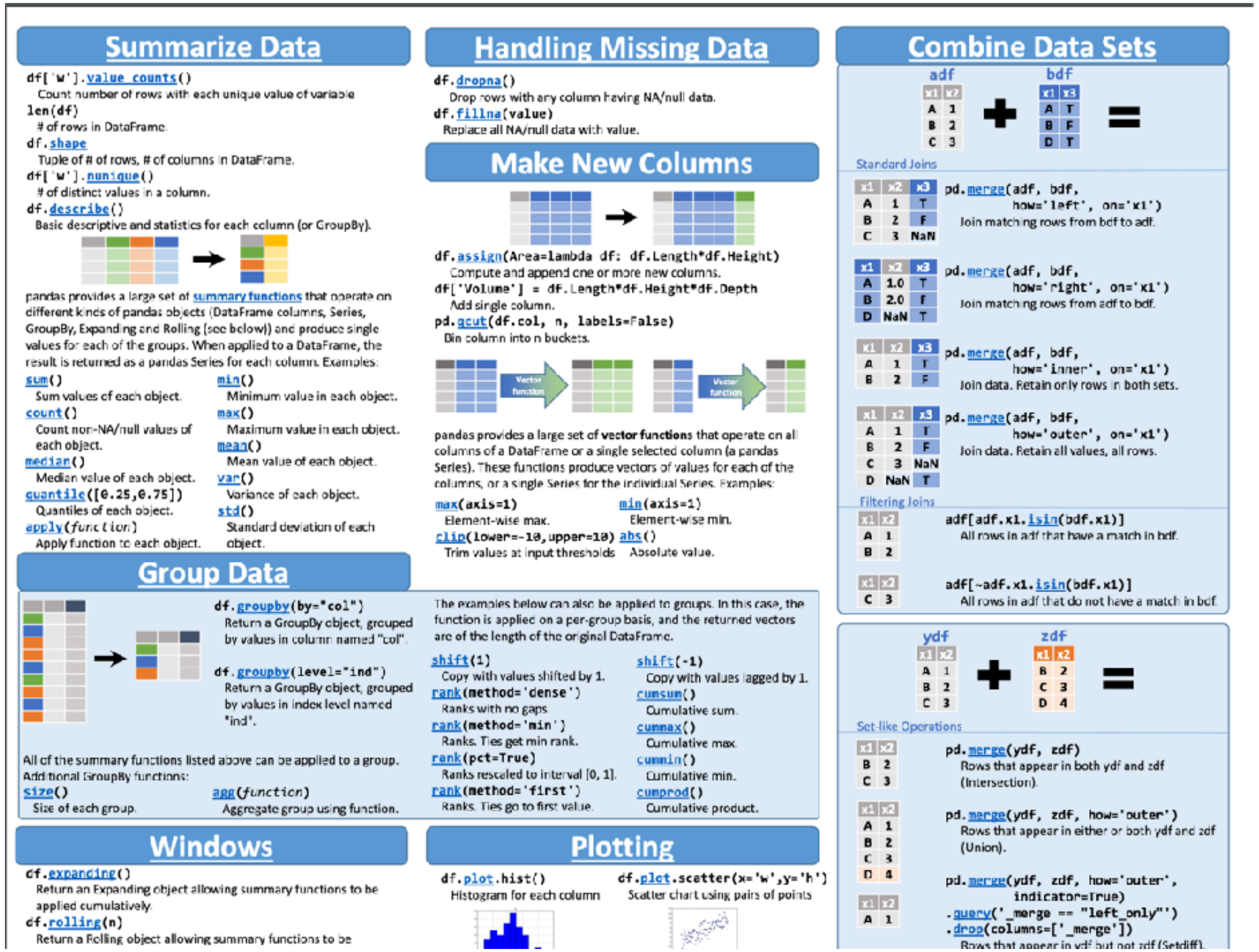
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Generate random data
np.random.seed(10)
data = np.random.normal(100, 20, 200)

# Create a boxplot using seaborn
sns.boxplot(data=data)

# Display the plot
plt.show()
```



Appendix

Most commonly used Pandas Operations





https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf