Practical Data Analysis and Visualization.
A hands - on appraoch to Processing, Analyzing, and Visualizing Structured Datasets.

Prepared By: Siman Giri, Instructor: Ronit and Shiv for Herald Center for AI.

Summer, 2025

1 Learning Objectives.

- Utilize Pandas as the primary library for processing structured data in Python, with an emphasis on handling CSV files and leveraging advanced features to analyze time series data effectively.
- Develop robust data preprocessing skills by appropriately managing edge cases, including identification and treatment of missing or incomplete data.
- Implement user-friendly error handling to ensure reliable and maintainable data processing workflows.
- Employ NumPy for efficient numerical computations, including array manipulations and vectorized operations to support data analysis tasks.
- Create insightful visualizations using Matplotlib and/or Seaborn to explore and investigate specific data phenomena.
- Independently consult plotting library documentation and utilize example code to design and customize more sophisticated visualizations, enhancing interpretability and presentation of data insights.

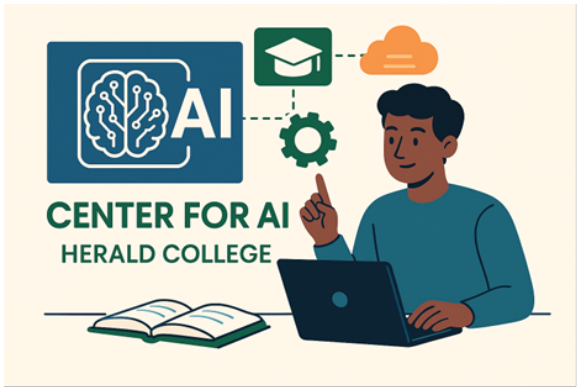


image generated via copilot.

2 Worksheet Overview:

This document, Worksheet 0, is intended to reinforce and consolidate the fundamental skills acquired in the course 5CS037. The exercises presented herein are closely aligned with the content covered in the Week 2 and Week 3 workshops of the 2024 academic session. The primary objective is to facilitate your familiarity and proficiency with key Python libraries, namely Pandas, Matplotlib, and NumPy.
A concise reference cheat sheet detailing essential commands and functions for these libraries is provided in the appendix to support your learning process. You are expected to complete the assigned tasks independently, demonstrating adherence to best coding practices, including the development of clear, maintainable, and well-documented code.
Kindly ensure that your completed worksheet is submitted by the conclusion of your Week 2 workshop.
This worksheet is structured into three main sections:

- **Exercise on Pandas:** Data manipulation and analysis.
- **Exercise on NumPy:** Numerical computations and array operations.
- **Exercise on Visualization:** Creating visual insights using Matplotlib and or Seaborn.

3 Exercises with Pandas:

Data Analysis with Pandas using California Housing Dataset:

1. Dataset Setup:

The dataset is accessible via `scikit-learn`'s `fetch_california_housing` method. Load it as follows in Python:

Access via:

```python
from sklearn.datasets import fetch_california_housing
data = fetch_california_housing(as_frame=True)
df = data.frame
```

The dataset contains the following columns:

- `MedInc` - Median income in block group
- `HouseAge` - Median house age in block group
- `AveRooms` - Average rooms per household
- `AveBedrms` - Average bedrooms per household
- `Population` - Population per block group
- `AveOccup` - Average occupants per household
- `Latitude` - Block group latitude
- `Longitude` - Block group longitude
- `MedHouseVal` - Median house value for California districts

2. Warm - Up Exercises:

Common Setup

- Load the dataset into a Pandas DataFrame.
- Inspect the dataset using `df.info()` and `df.describe()`.

Problem 1 – Sorting

1. Create a DataFrame `med_income` containing only the `MedInc` column. Display the first 5 rows.
2. Create a DataFrame `pop_lat` with columns `Population` and `Latitude` (in that order). Display the first 5 rows.
3. Create a DataFrame `house_age_rooms` with columns `HouseAge` and `AveRooms`. Display the first 5 rows.

Problem 2 – Subsetting

Subsetting Rows:

1. Filter houses where `MedInc > 8.0`, save as `high_income`. Display the result.
2. Filter houses where `Latitude > 37`, save as `north_california`. Display the result.
3. Filter houses where `AveRooms > 6.0` and `AveOccup < 2.0`, save as `spacious_low_occupancy`. Display the result.

Subsetting Categorical Equivalents:

1. Create a new column `Region` based on `Latitude` values:
   - `'North'` if `Latitude > 37`
   - `'Central'` if `35 < Latitude` $\leq$ `37`
   - `'South'` otherwise
2. Filter houses where `Region` is `'North'` or `'Central'`, save as `north_central_region`. Display the result.

Problem – 3 Exploratory Data Analysis:

Q1. Which house has the highest value per room?

Hint:

1. Create a new column `value_per_room` = `MedHouseVal / AveRooms`.
2. Filter rows where `value_per_room > 1`, save as `high_vpr`.
3. Sort `high_vpr` by descending `value_per_room`, save as `high_vpr_sorted`.
4. Display the top 5 rows with columns `MedHouseVal`, `AveRooms`, and `value_per_room`.

Q2. Among high-population areas (`Population > 5000`), which have the highest median income per person?

Hint:

1. Create a column `income_per_person` = `MedInc / Population`.
2. Filter rows where `Population > 5000`, save as `dense_areas`.
3. Sort `dense_areas` by descending `income_per_person`, save as `rich_dense_areas`.
4. Display the top 5 rows with `MedInc`, `Population`, and `income_per_person`.

Problem – 4 Group By Exercises:

Q1. What percent of total house value comes from each `Region`?

Hint:

1. Calculate total `MedHouseVal` for all houses.
2. Group by `Region` and sum `MedHouseVal`.
3. Divide each region's total by the overall total to get percentage contributions.

Q2. What percent of total houses belong to different age groups?

Hint:

1. Define `AgeGroup` based on `HouseAge`:
   - `'New'`: `HouseAge < 20`
   - `'Mid'`: 20 $\leq$ `HouseAge < 40`
   - `'Old'`: `HouseAge` $\geq$ 40
2. Count total houses.
3. Group by `AgeGroup` and count.
4. Compute percentage shares for each group.

3. Advance Exercises:

1. Correlation Analysis:

- Compute Pearson correlation coefficients between `MedHouseVal` and all other numerical features.
- Identify which features have the strongest positive and negative correlations with house value.
- Interpret these relationships.

2. Handling Missing Data:

- Randomly set 5% of `AveRooms` values to `NaN` (simulate missingness).
- Impute missing values using median imputation.
- Visualize and compare distributions of `AveRooms` before and after imputation using histograms or boxplots.
- Discuss the effect of imputation on data distribution.

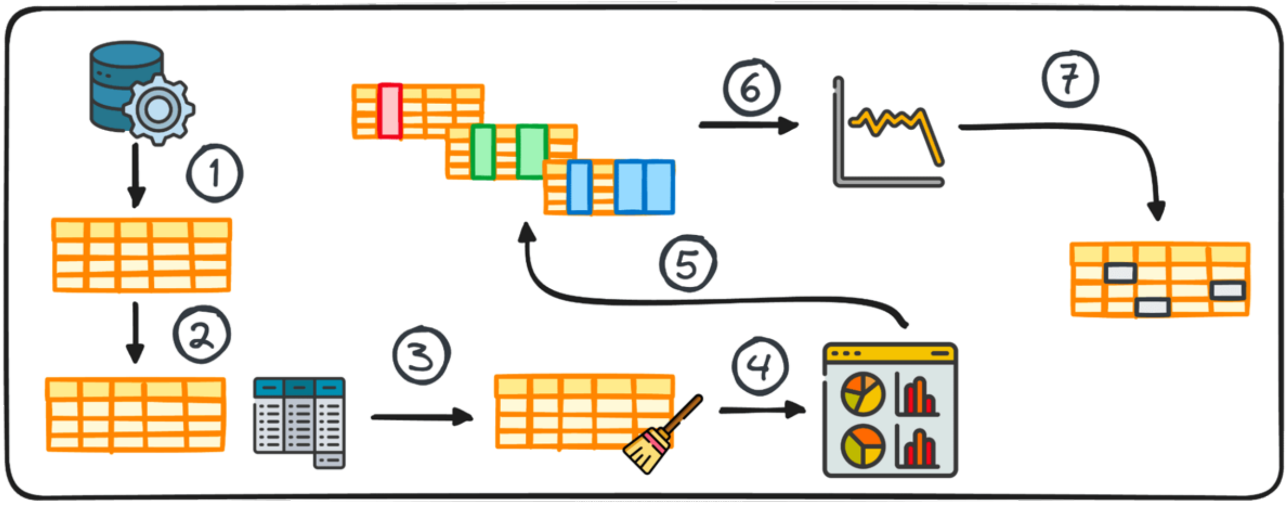


Image by : J. Ferrer – 7 steps to mastering Data Analysis {kDnuggets}

4 Exercises on Numpy:

1. Numpy Foundations - Warm Up Exercises:

Problem 1 – Array Creation:

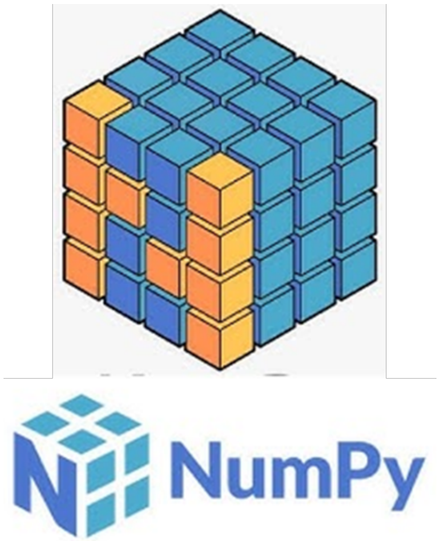
1. Create a 1D NumPy array containing integers from 0 to 19.
2. Reshape it into a 4x5 matrix.
3. Generate a 5x5 identity matrix and a 3x3 matrix filled with 7.

Problem 2 – Basic Operations:

1. Create two 3x3 matrices `A` and `B` with random integers (0–9).
2. Perform:
   - Element-wise addition, multiplication, and division.
   - Matrix multiplication (`A @ B`).
3. Compute mean, median, standard deviation, and sum for each matrix.

Problem 3 – Indexing and Slicing:

1. Slice the first two rows of matrix `A`.
2. Select elements greater than 5.
3. Replace all even numbers in `A` with -1.



2. Numpy: Advanced Exercises:

1. Broadcasting Challenge

- Create a 3x1 column vector and a 1x4 row vector.
- Use broadcasting to generate a 3x4 multiplication table.

2. Vectorization vs Loops

- Write a function to compute element-wise square of an array using:
  - a for-loop
  - NumPy vectorized operation
- Compare their execution time using `%%timeit` or `time` module.

3. Simulation Task

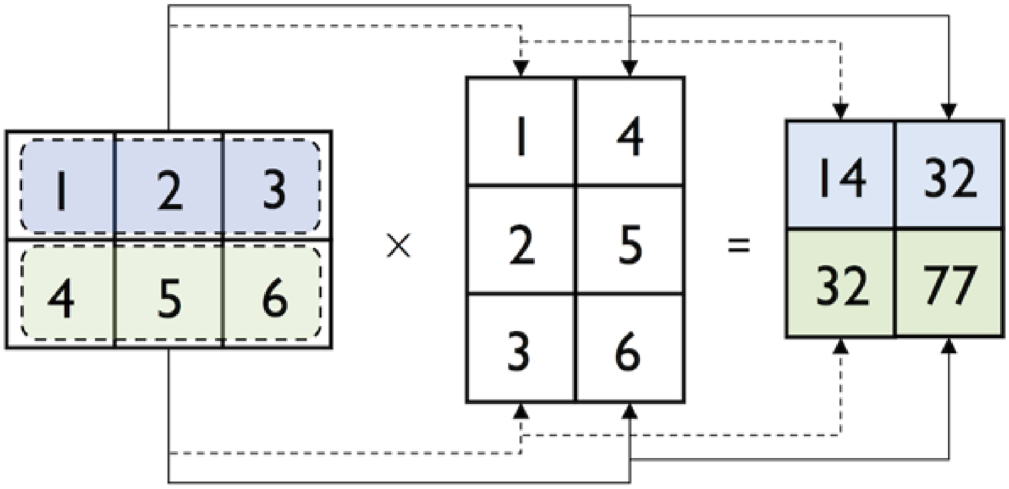
- Simulate 1000 random coin tosses and calculate proportion of heads.
- Simulate 1000 dice rolls and plot histogram of outcomes.

4. Solving Systems of Equations

- Solve the system:

$$3x + y = 9x + 2y = 8$$

- Use `np.linalg.solve` to find the solution.



5 Exercises on Visualization with Matplotlib or Seaborn:

1. Warm - Up Exercises:

Problem 1 – Basic Plotting with Matplotlib

1. Generate a line plot of the function $y = \sin(x)$ over the interval $[0, 2\pi]$.

2. Customize the plot with title, axis labels, and grid.

3. Save the plot to a file.

Problem 2 – Histograms and Bar Plots

1. Plot a histogram of the `MedHouseVal` column from the California dataset.

2. Create a bar chart comparing average `MedInc` across `Region`.

Problem 3 – Scatter Plots

1. Create a scatter plot of `MedInc` vs. `MedHouseVal`.

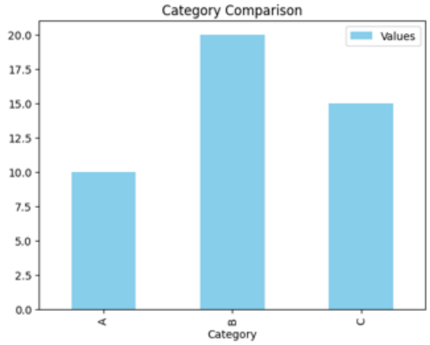
2. Color the points by `Region` and add transparency.

3. Add a regression line using `Seaborn`'s `regplot`.

Problem 4 – Subplots

1. Create a 2x2 subplot grid showing:

   - Line plot of sine
   - Histogram of income
   - Bar chart of region-wise population
   - Boxplot of house value grouped by age group



2. Advanced Exercise: Visualization

1. Heatmaps

- Compute the correlation matrix of the California dataset.
- Plot a heatmap using `sns.heatmap` with annotations.

2. Pairplot

- Use `Seaborn`'s `pairplot` to show pairwise relationships between `MedInc`, `MedHouseVal`, `HouseAge`, and `AveRooms`.
- Color points by `Region`.

3. Distribution Analysis

- Use `Seaborn`'s `distplot` or `displot` to visualize:
  - Distribution of `MedHouseVal`
  - Log-transformed version to see skewness reduction

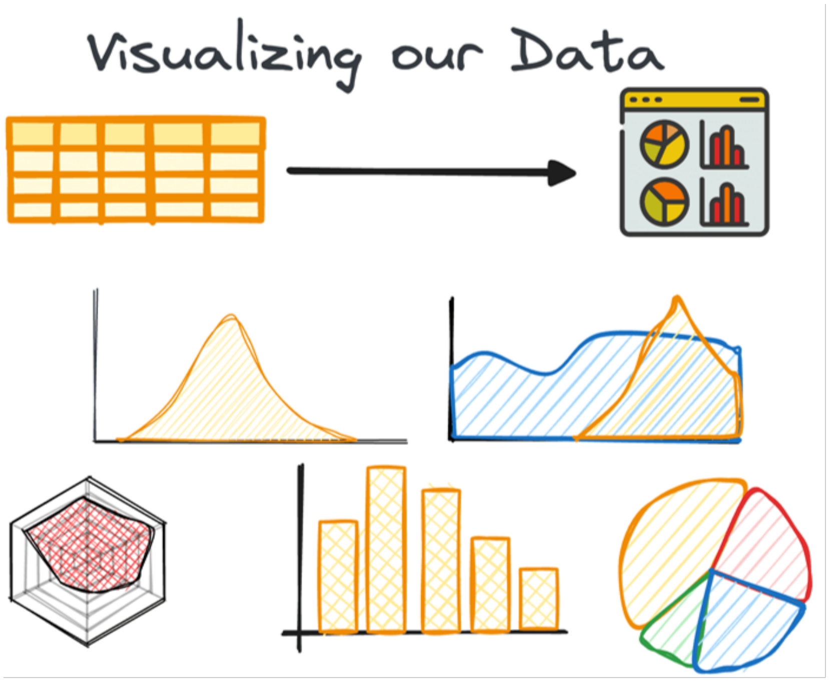


Image by : J. Ferrer – 7 steps to mastering Data Analysis {kDnuggets}

6 Appendix.

Pandas, Numpy and Visualization Cheat Sheet for Data Analytics:

1. Pandas Cheat Sheet:

1. Core Data Structures

Command	Description
`pd.Series([1,2,3])`	1D labeled array
`pd.DataFrame(dict)`	2D labeled data structure
`df.index`	Access row labels
`df.columns`	Access column labels

2. Data Loading & Inspection

Command	Description
`pd.read_csv('file.csv')`	Load CSV file
`pd.read_excel('file.xlsx')`	Load Excel file
`df.head(n)`	First n rows
`df.tail(n)`	Last n rows
`df.info()`	Data types and memory
`df.describe()`	Summary statistics
`df.shape`	(rows, columns) tuple

3. Data Selection

`Command`	**Description**
`df[’col’]`	Select column
`df[[’col1’,’col2’]]`	Select multiple columns
`df.loc[row_label]`	Select by label
`df.iloc[row_index]`	Select by position
`df.query(’a > b’)`	Boolean selection
`df.sample(n=5)`	Random sample

4. Data Cleaning

`Command`	**Description**
`df.isna().sum()`	Count missing values
`df.dropna()`	Remove missing values
`df.fillna(value)`	Replace missing values
`df.duplicated()`	Find duplicates
`df.drop_duplicates()`	Remove duplicates
`df.astype(’category’)`	Change data type

5. Data Transformation

`Command`	**Description**
`df.sort_values(’col’)`	Sort by column
`df.rename(columns={})`	Rename columns
`df.assign(new=expr)`	Add new column
`df.pivot_table()`	Create pivot table
`pd.get_dummies(df)`	One-hot encoding
`df.groupby(’col’).mean()`	Groupby operations

6. Merging & Joining

Command	Description
pd.concat([df1,df2])	Concatenate DataFrames
pd.merge(df1,df2)	Database-style join
df1.join(df2)	Join on index

7. Time Series

Command	Description
pd.to_datetime()	Convert to datetime
df.resample('D').mean()	Resample time series
df.rolling(7).mean()	Rolling window

8. Visualization

Command	Description
df.plot.line()	Line plot
df.plot.bar()	Bar plot
df.plot.hist()	Histogram
df.plot.scatter()	Scatter plot
df.plot.box()	Box plot

2. NumPy Cheat Sheet:

Array Creation	
`np.array([1,2,3])`	Create 1D array
`np.zeros((3,4))`	Array of zeros
`np.ones((2,2))`	Array of ones
`np.arange(0,10,2)`	Range with step
`np.linspace(0,1,5)`	Evenly spaced numbers
`np.random.rand(2,2)`	Random values in [0,1)
Array Properties and Indexing	
`a.shape`	Dimensions of array
`a.size`	Total number of elements
`a.dtype`	Data type
`a[1,2]`	Access element
`a[:,1]`	Access column
`a[1,:]`	Access row
Mathematical Operations	
`a + b`	Elementwise addition
`a * b`	Elementwise multiplication
`np.dot(a, b)`	Matrix multiplication
`np.sum(a)`	Sum of elements
`np.mean(a)`	Mean value
`np.std(a)`	Standard deviation
Reshaping and Manipulation	

`a.reshape((3,2))`	Reshape array
`a.flatten()`	Flatten to 1D
`a.T`	Transpose
`np.concatenate([a,b], axis=0)`	Combine arrays
`np.split(a, 2)`	Split into sub-arrays
Boolean Indexing & Filtering	
`a[a > 5]`	Filter values
`np.where(a > 5, 1, 0)`	Conditional replace
`np.any(a > 5)`	Check if any True
`np.all(a > 0)`	Check if all True

3. Visualization Cheat Sheet (Matplotlib & Seaborn)

1. Matplotlib Basics

`Command`	**Description**
`plt.plot(x, y)`	Line plot
`plt.bar(x, y)`	Bar plot
`plt.hist(data)`	Histogram
`plt.scatter(x, y)`	Scatter plot
`plt.title('Title')`	Plot title
`plt.xlabel('x')`	X-axis label
`plt.ylabel('y')`	Y-axis label
`plt.legend()`	Show legend
`plt.show()`	Display plot

2. Seaborn Basics

`Command`	**Description**
`sns.lineplot(x, y)`	Line plot
`sns.barplot(x, y)`	Bar chart with confidence intervals
`sns.histplot(data)`	Histogram
`sns.boxplot(x, y)`	Box plot
`sns.scatterplot(x, y)`	Scatter plot
`sns.heatmap(data)`	Heatmap
`sns.pairplot(df)`	Pairwise relationships

3. Plot Customization

Command	Description
`plt.grid(True)`	Show grid
`plt.xlim([0,10])`	Set x-axis range
`plt.ylim([0,5])`	Set y-axis range
`plt.xticks(rotation=45)`	Rotate x-ticks
`sns.set(style='whitegrid')`	Set plot style

The - End