

Shri Ramdeobaba College of Engineering & Management, Nagpur.

RCOEM

Shri Ramdeobaba College of
Engineering and Management, Nagpur

TOOLS FOR DATA SCIENCE.

IV SEMESTER 2023-24

TA-01

Name: Vinit Kadam.

Branch/Roll No: ECS/B61.

1. Data Analysis with Pandas and Matplotlib:

- **Objective:** Perform data analysis on a given dataset using Pandas and visualize the results using Matplotlib.
- **Requirements:**
Choose a dataset (e.g., CSV, Excel, or any other format) related to a topic of interest (e.g., finance, sports, health).
Use Pandas to load and clean the data.
Perform basic statistical analysis (mean, median, standard deviation).
Create meaningful visualizations using Matplotlib (e.g., bar chart, line plot, scatter plot).
Provide insights or conclusions based on the analysis.

```

In [3]: import pandas as pd
import matplotlib.pyplot as plt

In [9]: #Load the dataset
ds=pd.read_csv('state - state.csv')

In [10]: #Display Structure of the data
print(ds.head())

  state  population  murder rate
0    MH  1025479666      0.5
1    MP  2541676563      0.4
2    UP  1254789654      0.8
3    AP  2541676783      0.3

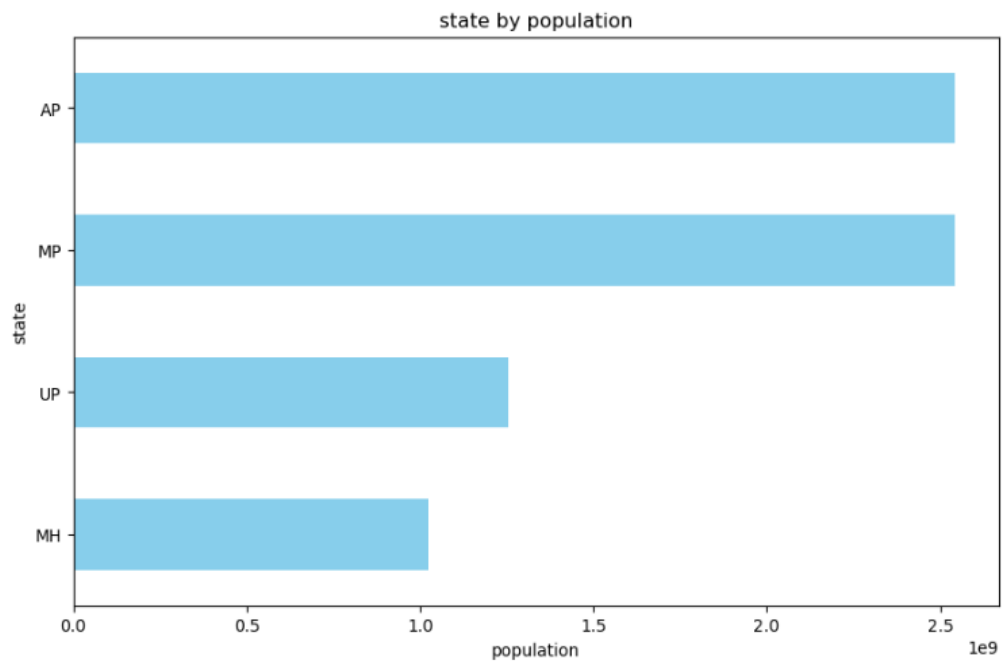
In [11]: mean_population = ds['population'].mean()
median_murder_rate = ds['murder_rate'].median()
std_deviation = ds['population'].std()

In [12]: print(mean_population)
print(median_murder_rate)
print(std_deviation)

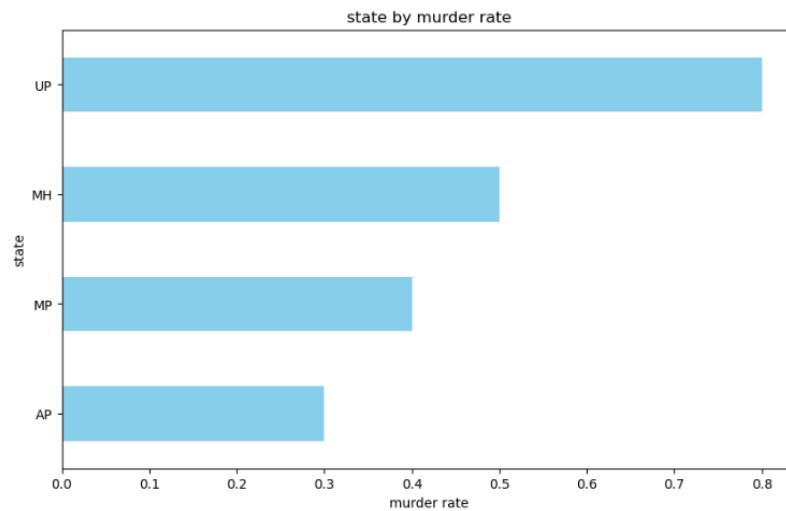
1840905666.5
0.45
814577917.1862354

In [15]: # Bar chart
plt.figure(figsize=(10, 6))
ds.groupby('state')['population'].sum().sort_values().plot(kind='barh', color='skyblue')
plt.title('state by population')
plt.xlabel('population')
plt.ylabel('state')
plt.show()

```



```
In [16]: plt.figure(figsize=(10, 6))
ds.groupby('state')['murder rate'].sum().sort_values().plot(kind='barh', color='skyblue')
plt.title('state by murder rate')
plt.xlabel('murder rate')
plt.ylabel('state')
plt.show()
```



- **Conclusions:**
 - Based on the bar charts, it seems that AP has the highest total population and UP has the highest murder rate.

2. Statistical Analysis with R:

- **Objective:** Perform statistical analysis on a dataset using R's built-in statistical functions.
- **Requirements:**
 - Choose a dataset suitable for statistical analysis (e.g., survey data, experiment results).
 - Calculate descriptive statistics (mean, median, standard deviation) for relevant variables.
 - Conduct hypothesis testing or create confidence intervals for specific hypotheses.
 - Visualize the results using appropriate plots (e.g., histograms, violin plots).
 - Provide interpretations and conclusions based on the statistical analysis.

```
1 data <- read.csv("weight-height.csv")
2 head(data)
3 mean_h <- mean(data$Height)
4 median_h <- median(data$Height)
5 sd_h <- sd(data$Height)
6 print(mean_h)
7 print(median_h)
8 print(sd_h)
9 test_result <- t.test(data$Height, mu = 300)
10 print(test_result)
11 hist(data$Height, main = "Height Distribution", xlab = "Height", col = "lightblue", border = "black")
12 abline(v = mean_h, col = "red", lwd = 2)
13 legend("topright", legend = c("Height Distribution", "Mean Height"), fill = c("lightblue", "red"))
```

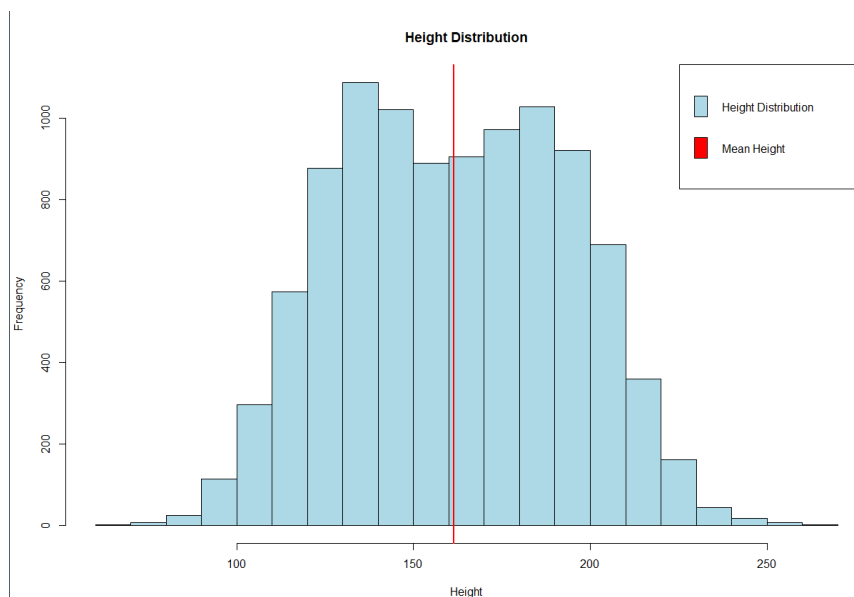
```

> data <- read.csv("weight-height.csv")
> head(data)
  Gender  Weight  Height
1  Male  73.84702 241.8936
2  Male  68.78190 162.3105
3  Male  74.11011 212.7409
4  Male  71.73098 220.0425
5  Male  69.88180 206.3498
6  Male  67.25302 152.2122
> mean_h <- mean(data$Height)
> median_h <- median(data$Height)
> sd_h <- sd(data$Height)
> print(mean_h)
[1] 161.4404
> print(median_h)
[1] 161.2129
> print(sd_h)
[1] 32.10844
> test_result <- t.test(data$Height, mu = 300)
> print(test_result)

      One Sample t-test

data:  data$Height
t = -431.54, df = 9999, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 300
95 percent confidence interval:
 160.8110 162.0697
sample estimates:
mean of x
161.4404

```



- **Conclusion:**
 - The mean height of the individuals in the dataset is 161.4404 cm.
 - The median height is 161.2126 cm, indicating the central tendency.
 - The standard deviation of height is 32.108 cm, reflecting the spread of the data.

3. Title: Data Analysis with Pandas and NumPy.

◆ Problem Statement:

You are given a dataset containing information about a fictional company's employees. The dataset (employee_data.csv) has the following columns:

Employee_ID: Unique identifier for each employee.

First_Name: First name of the employee.

Last_Name: Last name of the employee.

Department: Department in which the employee works.

Salary: Salary of the employee.

Joining_Date: Date when the employee joined the company.

• Tasks:

○ Data Loading:

Load the dataset (employee_data.csv) into a Pandas DataFrame.

Display the first 5 rows to get an overview of the data.

○ Data Cleaning:

Check for and handle any missing values in the dataset.

Convert the Joining_Date column to a datetime format.

○ Data Exploration:

Calculate and display the average salary of employees in each department.

Identify the employee with the highest salary and display their information.

○ Time-based Analysis:

Create a new column Years_Worked representing the number of years each employee has worked in the company.

Calculate the average salary for employees based on the number of years they have worked (grouped by years).

○ Data Visualization:

Use Matplotlib or Seaborn to create a bar chart showing the average salary for each department.

Create a histogram of the distribution of employee salaries.

```
In [6]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Data Loading
df = pd.read_csv('employee_data.csv')

# Display the first 5 rows
print("First 5 rows of the dataset:")
print(df.head())
```

First 5 rows of the dataset:

	Employee_ID	First_Name	Last_Name	Department	Joining_Date	Salary
0	E02387	Emily	Davis	IT	2016	\$141,604
1	E04105	Theodore	Dinh	IT	1997	\$99,975
2	E02572	Luna	Sanders	Finance	2006	\$163,099
3	E02832	Penelope	Jordan	IT	2019	\$84,913
4	E01639	Austin	Vo	Finance	1995	\$95,409

```
In [7]: # Data Cleaning
# Check for missing values
print("\nChecking for missing values:")
print(df.isnull().sum())

# Handle missing values
df.dropna(inplace=True)
```

Checking for missing values:

	Employee_ID	First_Name	Last_Name	Department	Joining_Date	Salary
	0	0	0	0	0	0

dtype: int64

```
In [8]: # Convert Joining_Date to datetime format
df['Joining_Date'] = pd.to_datetime(df['Joining_Date'])

# Clean the 'Salary' column
df['Salary'] = df['Salary'].replace(['$', ','], '', regex=True).astype(float)

# Data Exploration
# Calculate and display the average salary of employees in each department
average_salary_by_department = df.groupby('Department')['Salary'].mean()
print("\nAverage Salary by Department:")
print(average_salary_by_department)
```

Average Salary by Department:

Department	Average Salary
Finance	129254.0
IT	111559.5
Sales	50994.0

Name: Salary, dtype: float64

```
In [9]: # Identify the employee with the highest salary and display their information
highest_salary_employee = df[df['Salary'] == df['Salary'].max()]
print("\nEmployee with the highest salary:")
print(highest_salary_employee)
```

Employee with the highest salary:

	Employee_ID	First_Name	Last_Name	Department	Joining_Date	Salary
2	E02572	Luna	Sanders	Finance	1970-01-01 00:00:00.000000	163099.0

```
In [10]: # Time-based Analysis
# Create a new column Years_Worked
current_year = pd.to_datetime('now').year
df['Years_Worked'] = current_year - df['Joining_Date'].dt.year

# Calculate average salary based on the number of years worked
average_salary_by_years = df.groupby('Years_Worked')['Salary'].mean()
print("\nAverage Salary by Years Worked:")
print(average_salary_by_years)
```

```
Average Salary by Years Worked:
Years_Worked
54    107962.857143
Name: Salary, dtype: float64
```

```
In [11]: # Data Visualization
# Bar chart showing average salary for each department
plt.figure(figsize=(10, 6))
sns.barplot(x='Department', y='Salary', data=df, ci=None)
plt.title('Average Salary by Department')
plt.xlabel('Department')
plt.ylabel('Average Salary')
plt.show()

# Histogram of the distribution of employee salaries
plt.figure(figsize=(10, 6))
sns.histplot(df['Salary'], bins=20, kde=True)
plt.title('Distribution of Employee Salaries')
plt.xlabel('Salary')
plt.ylabel('Frequency')
plt.show()
```

