MEDTOUREASY

Final Report of Traineeship Program 2023

Analyze the Death Age Difference of Right-Handers with Left-Handers

26th October 2023

Arnab Paul

CONTENTS

ACKNOWLEDGEMENT

The traineeship opportunity that I had with MedTourEasy was a great chance for learning and understand the intricacies of Data analysis by using Python in Data Analytics; and also, for personal and professional development. I am very obliged to have a chance to interact with so many professionals who guided me throughout the traineeship project and made it a great learning curve for me. Firstly, I express my deepest gratitude and special thanks to the Training & Development Team of MedTourEasy who gave me an opportunity to carry out my traineeship at their esteemed organization. Also, I express my thanks to Coursera, for making me understand the details of the Data Analytics profile and training me in the same so that I can carry out the project properly and with maximum client satisfaction and for spending his valuable time in spite of his busy schedule.

I would also like to thank the team of MedTourEasy who made the working environment productive and very conducive.

INTRODUCTION

➢ ABOUT THE COMPANY

MedTourEasy MedTourEasy, a global healthcare company, provides you with the informational resources needed to evaluate your global options. MedTourEasy provides analytical solutions to our partner healthcare providers globally.

➢ ABOUT THE PROJECT

A National Geographic survey in 1986 resulted in over a million responses that included age, sex, and hand preference for throwing and writing. Researchers Avery Gilbert and Charles Wysocki analysed this data and noticed that rates of left-handedness were around 13% for people younger than 40 but decreased with age to about 5% by the age of 80. They concluded based on analysis of a subgroup of people who throw left-handed but write right-handed that this age-dependence was primarily due to the changing social acceptability of left-handedness. This means that the rates aren't a factor of age specifically but rather of the year you were born, and if the same study was done today, we should expect a shifted version of the same distribution as a function of age. Ultimately, we'll see what effect this changing rate has on the apparent mean age of death of left-handed people, but let's start by plotting the rates of left-handedness as a function of age.

➢ OBJECTIVES & DELIVERABLES

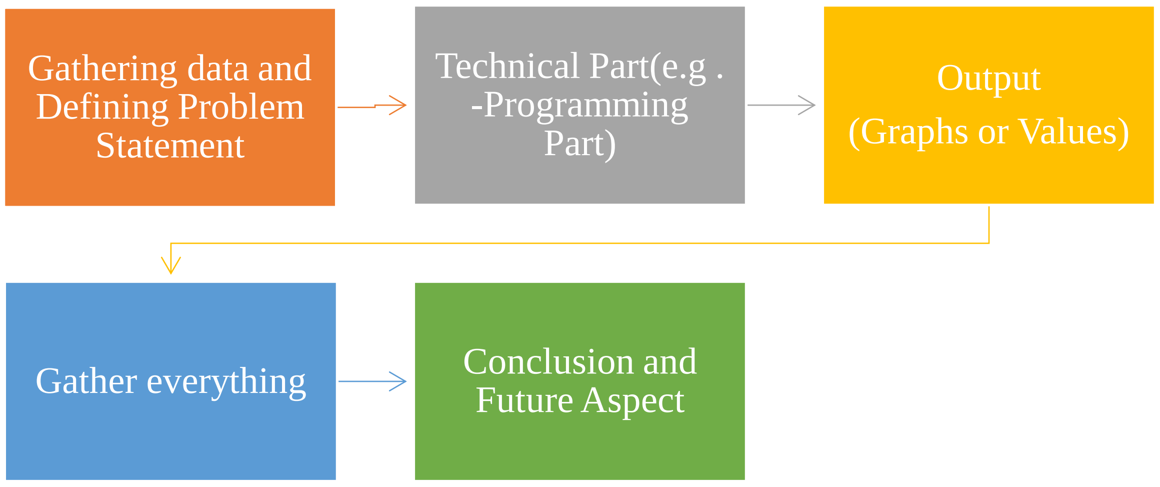
- **Age Distribution Data:** The project uses age distribution data to explore whether there is a difference in the average age at death for left-handed individuals compared to right-handed individuals.

- **Changing Rates of Left-Handedness Over Time**: It seeks to understand if the observed differences in age at death are primarily influenced by the changing rates of left-handedness over time, specifically related to the year people were born.

- **Analysis of Mean Age at Death**: Ultimately, the project aims to assess how the changing rate of left-handedness over time affects the apparent mean age of death among left-handed people. In other words, it seeks to determine whether any observed differences in life expectancy are linked to changing societal attitudes towards left-handedness.

METHODOLOGY

➢ FLOW OF THE PROJECT


Gathering data and Defining Problem Statement
Technical Part(e.g . -Programming Part)
Output
(Graphs or Values)
Gather everything
Conclusion and Future Aspect

➢ LANGUAGE AND PLATFORM USED

➢ PYTHON

Python is a high-level, versatile, and widely used programming language known for its simplicity and readability. Here's a proper summarization of Python:

1. **General-Purpose Language**: Python is a general-purpose programming language, meaning it can be used for a wide range of applications, from web development and data analysis to scientific computing and artificial intelligence.

2. **Readability and Simplicity**: Python is celebrated for its clear and concise syntax, making it easy to read and write. It emphasizes readability, which makes it an excellent choice for beginners and experienced developers alike.

3. **Interpreted Language**: Python is an interpreted language, which means we do not need to compile your code before running it. This speeds up development and makes it suitable for scripting and rapid application development.

4. **Cross-Platform Compatibility**: Python is available on various platforms (Windows, macOS, Linux) and supports a wide range of libraries and frameworks.

5. **Extensive Standard Library**: Python comes with a robust standard library that provides modules and functions for various tasks, reducing the need for writing code from scratch.

6. **Dynamic Typing**: Python uses dynamic typing, allowing variables to change their data type during runtime. This offers flexibility but requires careful variable management.

7. **High-Level Language**: Python abstracts many low-level details, simplifying complex tasks and enabling rapid development.

8. **Object-Oriented**: Python supports object-oriented programming (OOP) principles, including classes and objects.

9. **Large Ecosystem**: Python has a vast ecosystem of third-party libraries and frameworks. Notably, it's popular in data science and machine learning with libraries like NumPy, Pandas, and TensorFlow.

10. **Community Support**: Python boasts a large and active community. This means extensive online resources, tutorials, and an abundance of third-party packages.

11. **Indentation Matters**: Python uses indentation (whitespace) to define code blocks, replacing traditional braces or brackets. Proper indentation is essential for the code to run correctly.

12. **Dynamically Typed**: Python is dynamically typed, meaning you don't need to declare variable types explicitly. However, you must be mindful of data types to avoid errors.

13. **Open Source**: Python is open source, which means it's freely available and can be modified and redistributed.

14. **Popular Use Cases**: Python is widely used in web development (Django, Flask), scientific computing (SciPy, Matplotlib), data analysis (Pandas), machine learning (Scikit-learn), and more.

15. **Versatility**: Python can be used for a broad range of applications, from small scripts to large-scale applications. Its versatility and extensive library support make it a popular choice across different domains.

In summary, Python is a powerful, user-friendly, and versatile programming language with a strong emphasis on simplicity and readability. It's suitable for a wide range of applications, and its extensive ecosystem and community support make it a valuable tool for developers, scientists, and data analysts.

- PROGRAMMING PLATFORM – **GOOGLE COLAB**

Google Colab, short for Google Collaboratory, is a cloud-based integrated development environment (IDE) designed for Python programming. Here is a proper summarization of Google Colab:

1. **Cloud-Based Python Environment**: Google Colab provides a web-based platform for running Python code without the need for local installations. It's hosted on Google's servers and allows users to access and work on Python projects from anywhere with an internet connection.

2. **Free of Charge**: One of its key advantages is that it's entirely free to use. Users can run Python code and leverage its features without incurring any cost.

3. **Jupyter Notebook Integration:** Google Colab is built on top of Jupyter Notebooks, which is a popular interactive computing environment. It supports Jupyter's interactive and literate programming style, making it well-suited for data analysis, research, and collaborative work.

4. **Pre-Installed Libraries and Dependencies**: Google Colab comes with many pre-installed Python libraries and essential dependencies

commonly used in data science and machine learning. This saves users time and effort compared to setting up these libraries locally.

5. **Hardware Acceleration**: Colab offers access to powerful GPU (Graphics Processing Unit) and TPU (Tensor Processing Unit) resources. This is especially beneficial for machine learning and deep learning tasks, as it accelerates the training of neural networks.

6. **Collaboration Features**: Users can easily share Colab notebooks with others, enabling real-time collaboration. It supports collaborative editing, comments, and sharing via a link.

7. **Cloud Storage Integration**: Colab integrates with Google Drive, allowing users to save and access their notebooks directly from their Google Drive accounts. This simplifies data storage and sharing.

8. **Rich Text and Multimedia Support**: Colab notebooks can include rich text, images, and even embedded videos, making it a versatile platform for creating interactive reports and presentations.

9. **Version Control**: Google Colab is compatible with version control systems like Git, which allows users to manage and track changes to their code.

10. **No Local Setup Required**: Users don't need to set up Python or data science libraries on their local machines, making it a convenient choice for those who want to work with Python without dealing with installation and compatibility issues.

11. **Execution Environment Customization**: Users can specify the hardware accelerator (CPU, GPU, TPU) and modify runtime settings to suit the requirements of their projects.

12. **Code Export and Download**: Colab allows users to export their notebooks in various formats, including Jupyter Notebook files, PDFs, and more.

In summary, Google Colab is a cloud-based, free-to-use Python development environment that integrates Jupyter Notebooks. It offers powerful hardware resources, collaboration features, and pre-installed libraries, making it a popular choice for data science, machine learning, and collaborative coding projects that require minimal local setup and can benefit from cloud-based resources.

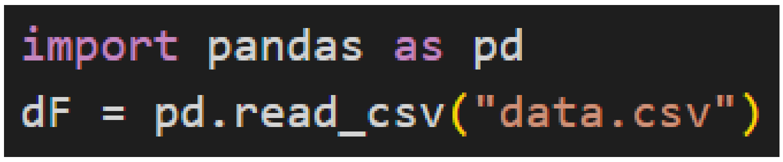
IMPLEMENTATION

➢ GATHERING REQUIREMENTS AND DEFINING PROBLEM STATEMENT

This is the first step wherein the requirements are collected from the clients to understand the deliverables and goals to be achieved after which a problem statement is defined which has to be adhered to while developing the project.

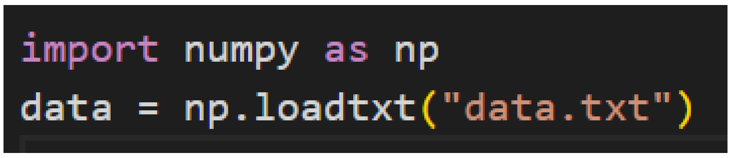
➢ DATA COLLECTION AND IMPORTING

- **Data Collection**: Data collection is the process of gathering data from various sources, such as databases, web APIs, files, or manual input. Common sources of data include databases, web scraping, API requests, and sensors.
- **Data Importing**: Data importing is the process of loading collected data into Python for analysis. Python provides several libraries and tools for data importing, such as **Pandas,** and **NumPy**, and built-in functions like **open()** for working with files.
  - ❖ Pandas: Pandas is a powerful library for data manipulation and analysis. It can import data from various formats, such as CSV, Excel, SQL databases, and more.



```
import pandas as pd
dF = pd.read_csv("data.csv")
```

  - ❖ Numpy: NumPy is a library for numerical operations in Python. It can be used for importing data from text files as arrays.



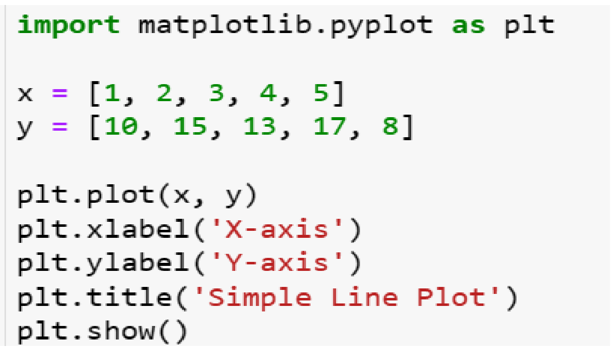
```
import numpy as np
data = np.loadtxt("data.txt")
```

- **Data Sources**: Common data sources include CSV files, Excel spreadsheets, JSON files, SQL databases, web APIs, and HTML pages. In this project, we basically used two datasets,

1. Death distribution data for the United States from the year 1999. (Link)
2. Rates of left-handedness digitized from a figure in this 1992 paper by Gilbert and Wysocki. (Link)

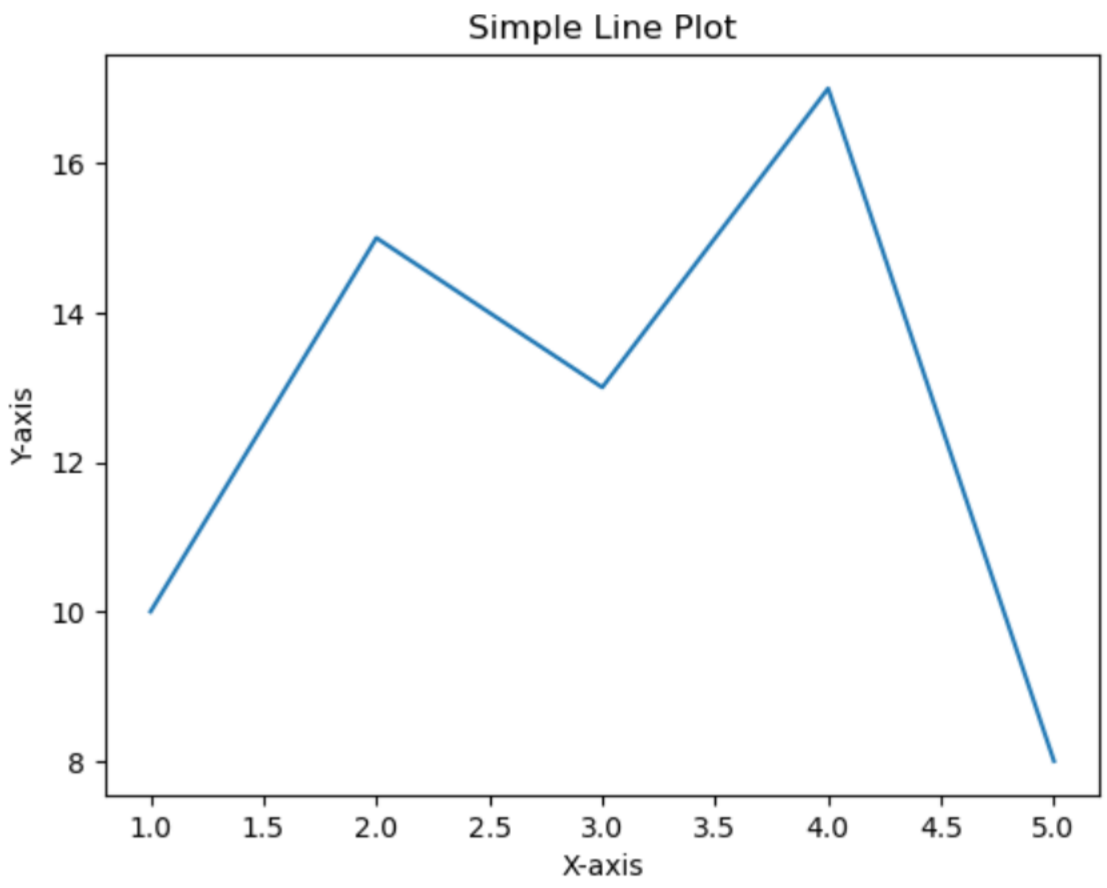
➢ DATA VISUALIZATION

- We use popular libraries like
  - **Matplotlib - Matplotlib** is a widely used 2D plotting library that provides a high degree of customization for creating static, publication-quality charts and graphs. It is the foundation for many other Python plotting libraries.



```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [10, 15, 13, 17, 8]

plt.plot(x, y)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Simple Line Plot')
plt.show()
```

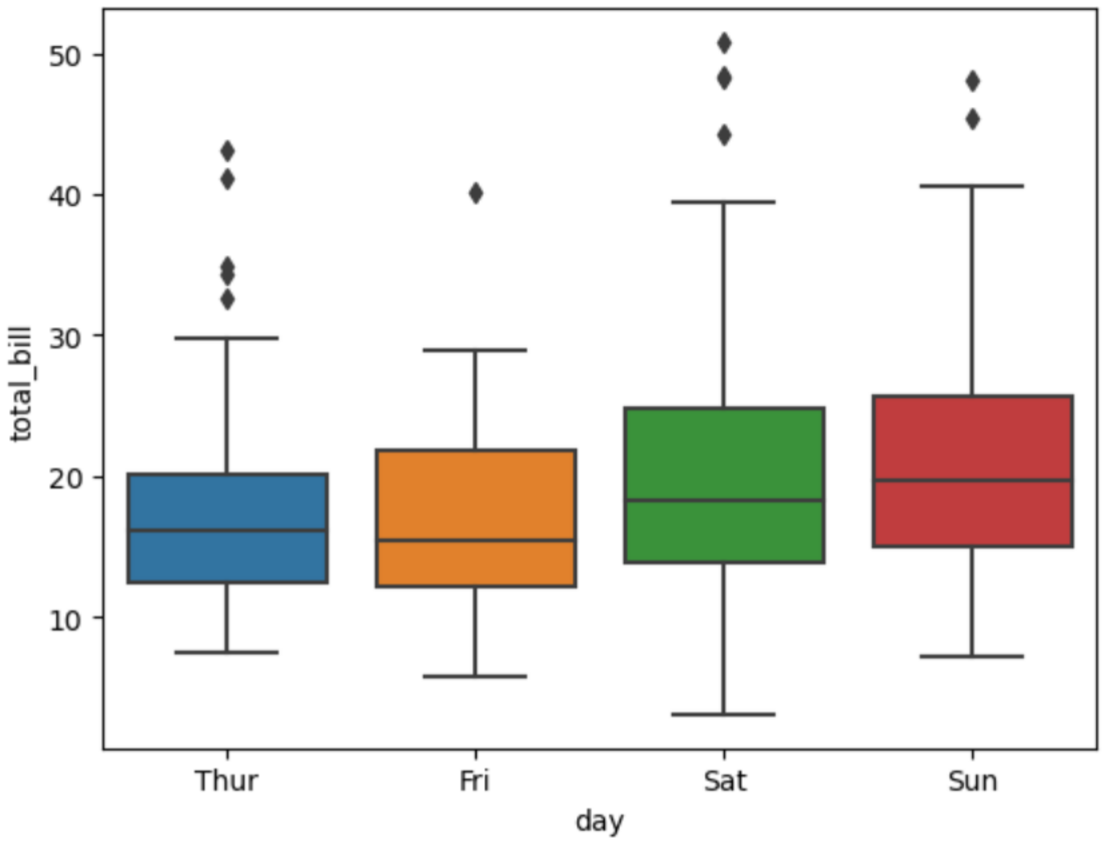
The output will be



- **Seaborn: Seaborn** is built on top of Matplotlib and provides a higher-level interface for creating attractive and informative statistical graphics. It simplifies many common data visualization tasks.

```python
import seaborn as sns

tips = sns.load_dataset("tips")
sns.boxplot(x="day", y="total_bill", data=tips)
```

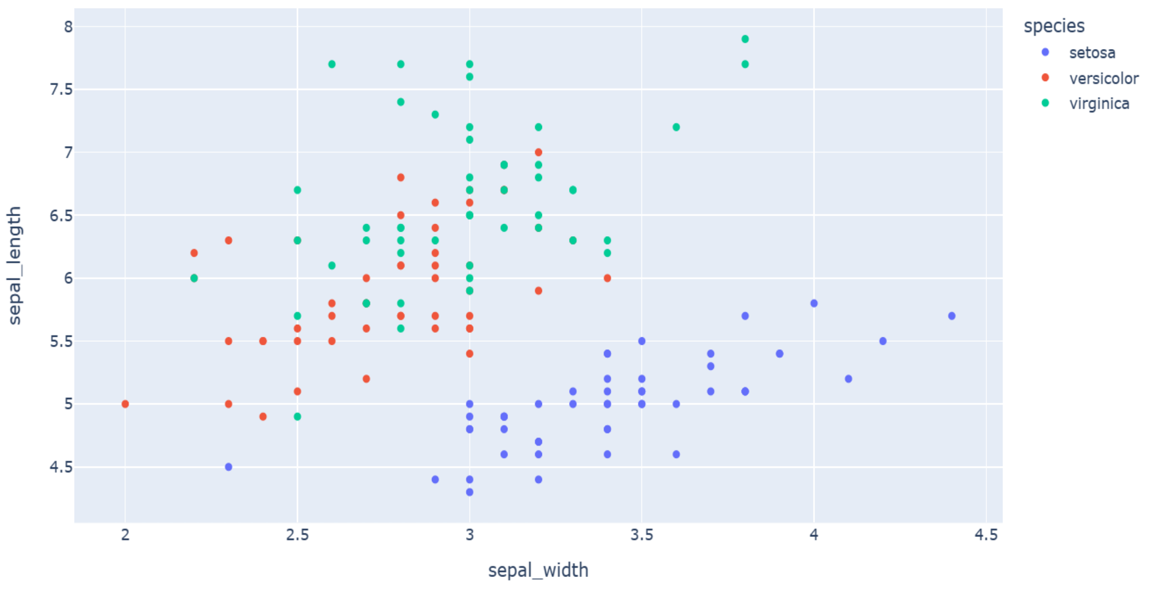
The output will be



- **Plotly: Plotly** is a powerful library for creating interactive and web-based visualizations. It supports a wide range of chart types, including scatter plots, bar charts, and 3D visualizations.

```python
import plotly.express as px

df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length", color="species")
fig.show()
```

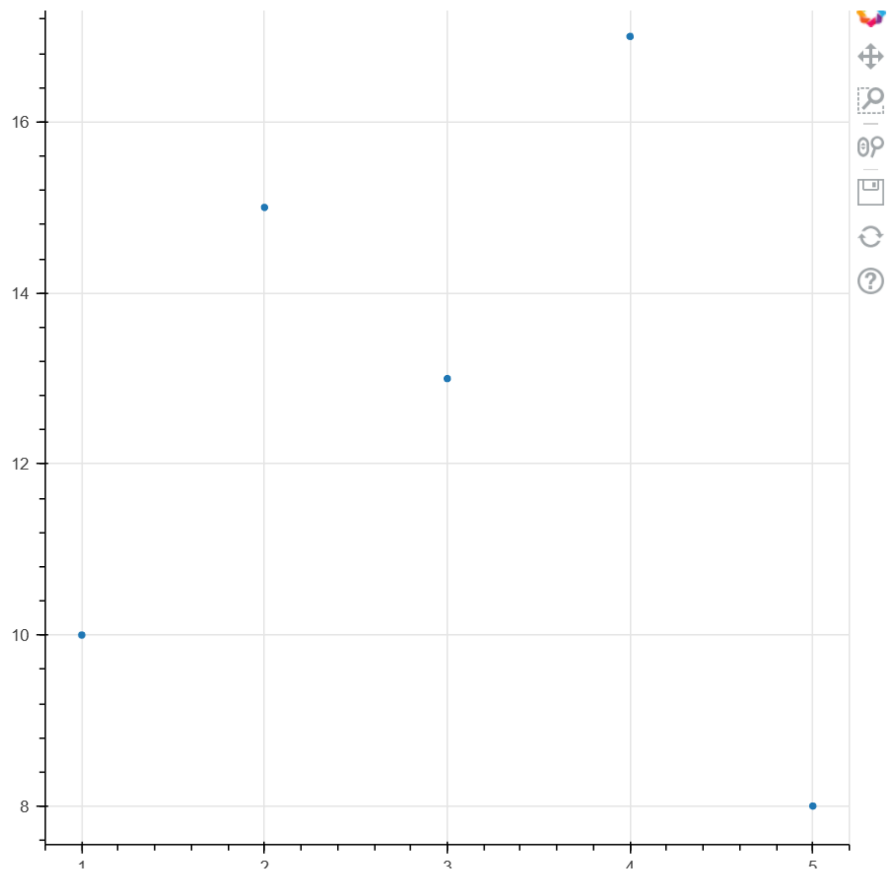
The output will be



- **Bokeh: Bokeh** is another library for creating interactive, web-based visualizations. It is particularly well-suited for building interactive dashboards and applications

```
from bokeh.plotting import figure, show
from bokeh.io import output_notebook

p = figure()
p.circle(x=[1, 2, 3, 4, 5], y=[10, 15, 13, 17, 8])
show(p)
```

The output will be



Here the blue points are our main result
(x,y) = {(1,10), (2,16), (3,13), (4,17), (5,8)}

➢ BAYES’ RULE

- **Bayes' rule**, also known as Bayes' theorem, is a fundamental concept in probability theory and statistics. It describes how to update the probability of a hypothesis or event based on new evidence or information. The formula for Bayes' rule is as follows:

$$P(A|B) = [P(B|A) \cdot P(A)] / P(B)$$

- **P(A|B)** is the posterior probability, which represents the probability of event or hypothesis A occurring given that event or evidence B has occurred.

- **P(B|A)** is the likelihood, representing the probability of evidence B occurring given that event or hypothesis A is true.

- **P(A)** is the prior probability, which is the initial or prior probability of event or hypothesis A occurring before considering the new evidence B.

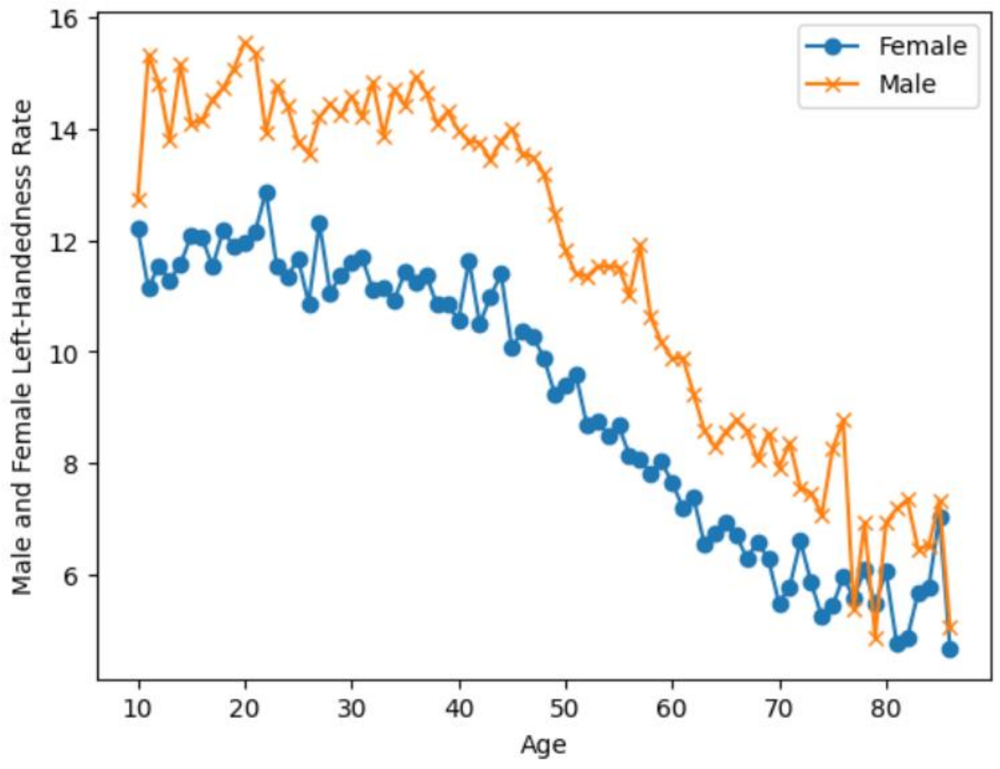
- **P(B)** is the probability of the evidence or event B occurring, regardless of whether event A is true or not.

The main idea behind Bayes' rule is to update our beliefs about the probability of A based on new information in the form of evidence B. It is particularly useful in scenarios such as:

  - **Bayesian Inference**: In statistics, Bayes' rule is used to estimate parameters or make predictions in a Bayesian framework. It's essential for tasks like parameter estimation in Bayesian statistics.
  - **Machine Learning**: In machine learning, Bayes' theorem is used in Bayesian classifiers (e.g., Naive Bayes) for classification tasks.
  - **Medical Diagnostics**: Bayes' rule is used in medical diagnostics to update the probability of a disease or condition given the results of a diagnostic test.
  - **Spam Filtering**: It's employed in spam filters to classify emails as spam or not based on the presence of certain words or patterns.
  - **Natural Language Processing**: In NLP, Bayes' rule can be used for tasks like text classification and sentiment analysis.

SAMPLE SCREENSHOTS AND OBSERVATIONS

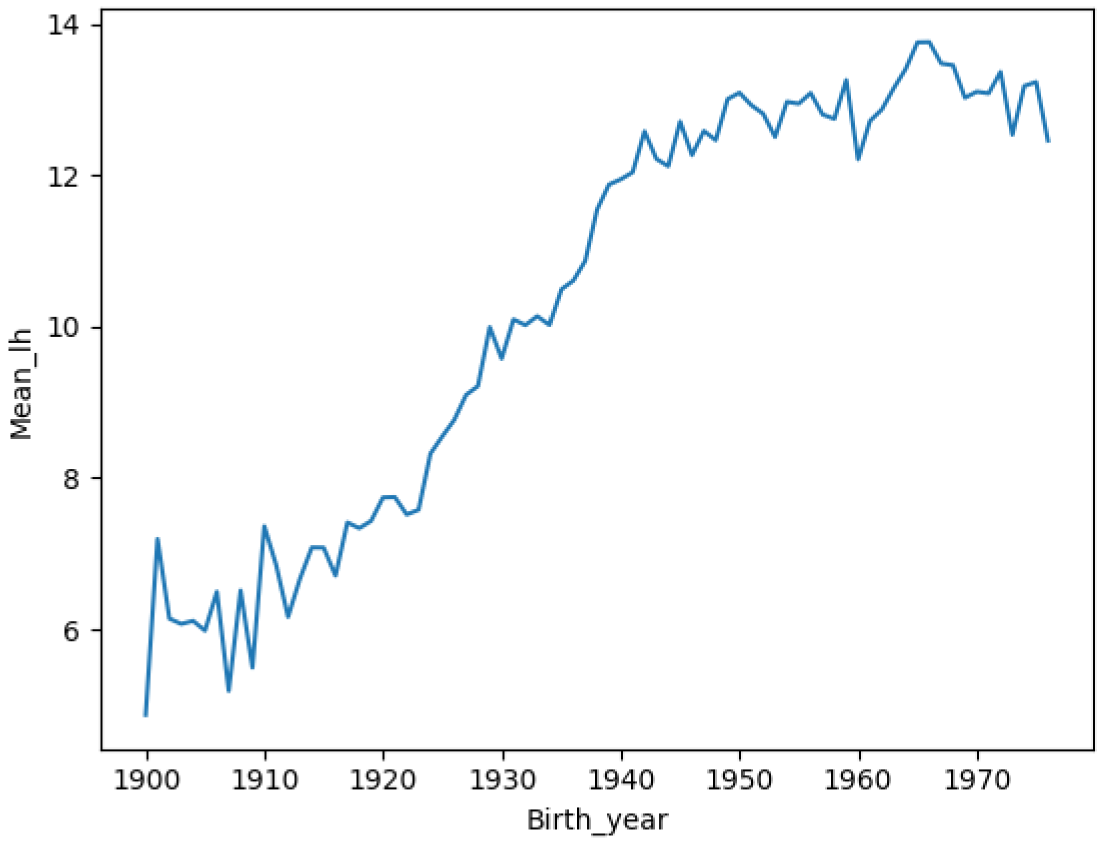
➢ TASK 1: Rates of left-handedness as a function of age



the graph gives you an initial view of how left-handedness rates vary with age and gender. The specific insights you can gain from the graph will depend on the characteristics of the dataset and the nature of left-handedness in the population it represents.

1. We can clearly see Male and Female Left-Handedness is inversely proportional to age, which means left-handedness will decrease as a person gets older.
2. Initially the rate of left-handedness rate is higher for males and lower for females (this mainly happens between ages 0 – 75), But later on we get more than 5 intersection points, where the rate of left-handedness is almost the same for both males and females (between ages 75- 80)

➢ **TASK 2: Rates of left-handedness over time, converting this data into a plot of the rates of left-handedness as a function of the year of birth, and average over males and females to get a single rate for both sexes**

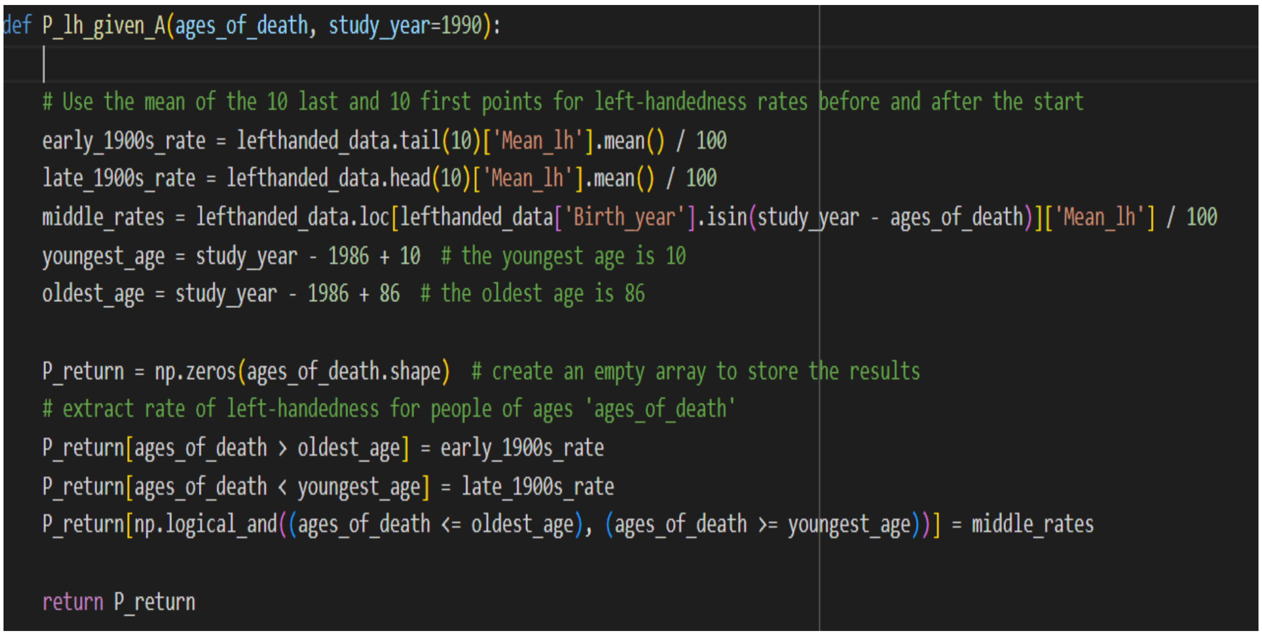


This graph gives you insights into the mean left-handedness with respect to the birth year, the Mean_lh is calculated by calculating the average of lefthanded data of males and females.

1. We can clearly see that as the birth year increases and the mean left-handed data also increases, it's a directly proportional relationship.
2. If we do a rough estimation, it is certain that from 1900 to the close of 1915 there were a few ups and downs of Mean_lh.
3. From 1920 to 1950 the Mean_lh is gradually increasing.
4. From 1950 to post 1970 the Mean_lh almost in a constant rate.
5. In 1964 the Mean_lh is the highest

➢ **TASK 3: Calculating the probability of dying at age A given that the person is left-handed, its denoted by P(LH/A)**

- To calculate P(LH/A) for ages that might fall outside the original data, we will need to extrapolate the data to earlier and later years. Since the data is flattened out in the early 1900s and late 1900s. Here are using data until 1910 as the data looks kind of flattened.



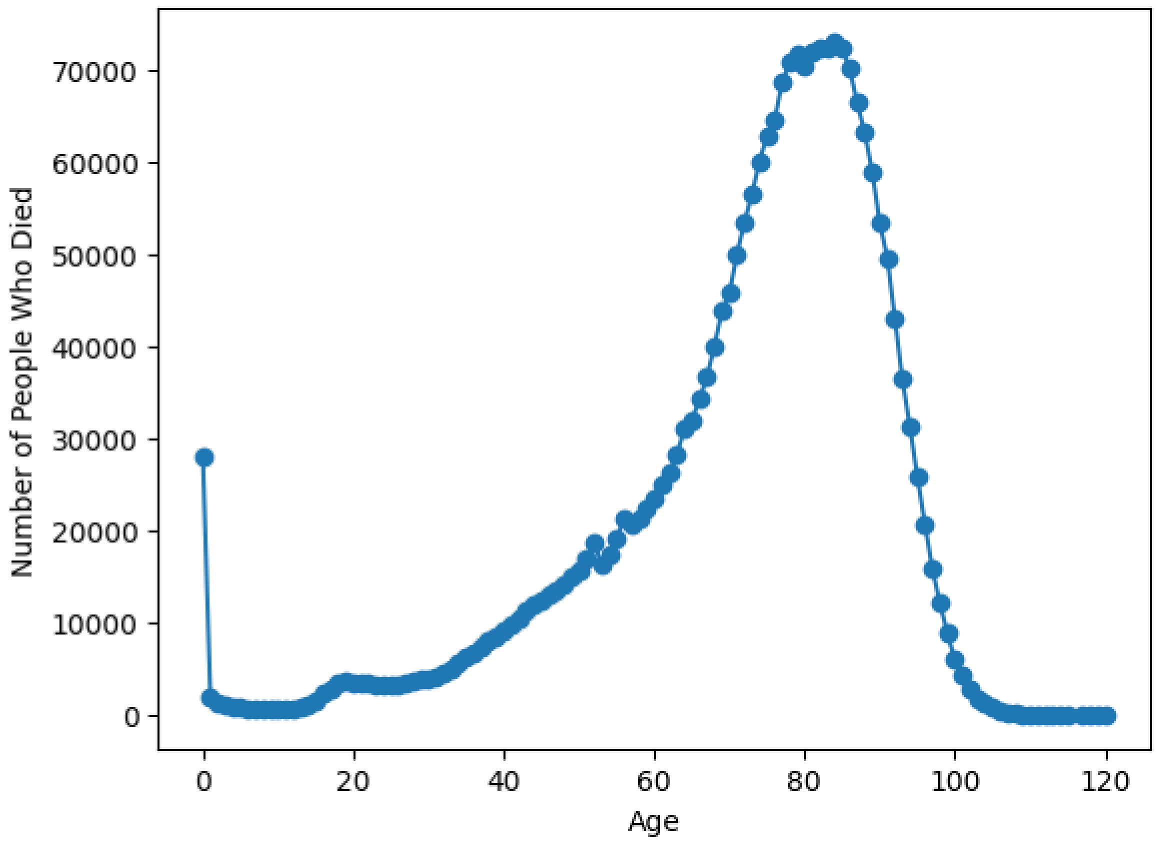
```
def P_lh_given_A(ages_of_death, study_year=1990):

    # Use the mean of the 10 last and 10 first points for left-handedness rates before and after the start
    early_1900s_rate = lefthanded_data.tail(10)['Mean_lh'].mean() / 100
    late_1900s_rate = lefthanded_data.head(10)['Mean_lh'].mean() / 100
    middle_rates = lefthanded_data.loc[lefthanded_data['Birth_year'].isin(study_year - ages_of_death)]['Mean_lh'] / 100
    youngest_age = study_year - 1986 + 10  # the youngest age is 10
    oldest_age = study_year - 1986 + 86  # the oldest age is 86

    P_return = np.zeros(ages_of_death.shape)  # create an empty array to store the results
    # extract rate of left-handedness for people of ages 'ages_of_death'
    P_return[ages_of_death > oldest_age] = early_1900s_rate
    P_return[ages_of_death < youngest_age] = late_1900s_rate
    P_return[np.logical_and((ages_of_death <= oldest_age), (ages_of_death >= youngest_age))] = middle_rates

    return P_return
```

We have done these things in our code,

- Imported the numpy package aliased as np.

- Used the last ten Mean_lh data points to get an average rate for the early 1900s. Named the resulting DataFrame early_1900s_rate
- Used the first ten Mean_lh data points to get an average rate for the late 1900s. Named the resulting DataFrame late_1900s_rate.
- For the early 1900s ages, fill in P_return with the appropriate left-handedness rates for ages_of_death. That is, input early_1900s_rate as a fraction, i.e., divide by 100.
- For the late 1900s ages, filled in P_return with the appropriate left-handedness rates for ages_of_death. That is, input late_1900s_rate as a fraction, that is, divided by 100.

- **TASK 4: When do people normally die?**

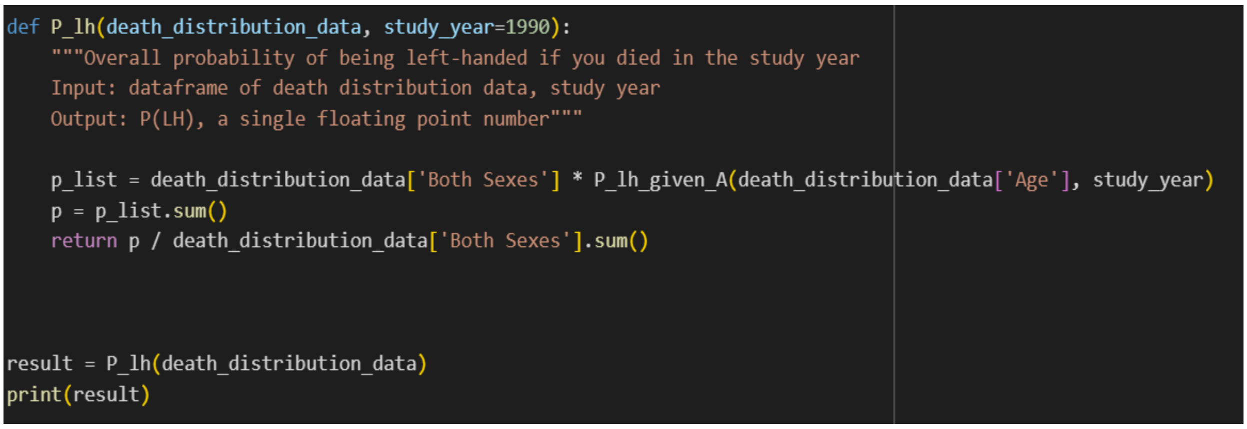


We can see the death data with respect to age. Here are some observations we can clearly highlight.

1. almost 30,000 infants die normally.

2. At the age of 0-20 there was almost a constant rate of death.
3. Between 60 to 80 years old the death rate rapidly increased. And after 80 it decreased very much, and eventually it became 0.

- **TASK 5: Overall probability of left-handedness**



```python
def P_lh(death_distribution_data, study_year=1990):
    """Overall probability of being left-handed if you died in the study year
    Input: dataframe of death distribution data, study year
    Output: P(LH), a single floating point number"""

    p_list = death_distribution_data['Both Sexes'] * P_lh_given_A(death_distribution_data['Age'], study_year)
    p = p_list.sum()
    return p / death_distribution_data['Both Sexes'].sum()



result = P_lh(death_distribution_data)
print(result)
```

  - Created a function called P_lh() which calculates the overall probability of left-handedness in the population for a given study year.

The Output of the following program is 0.07766387615350638 ≈ 0.078.

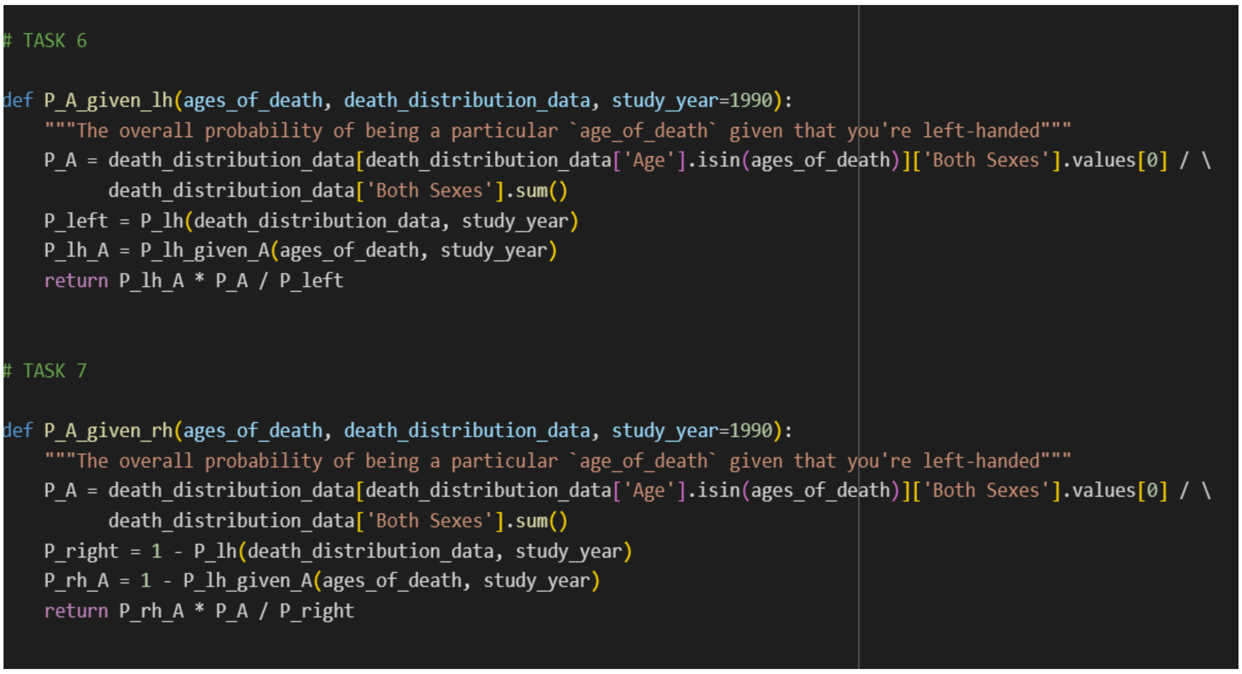
Note – for the function P_lh_given_A(), we have to take reference from **TASK 3**.

- **TASK 6 and TASK 7: Putting it all together: dying while left-handed Now we have the means of calculating all three quantities we need: P(A), P(LH), and P (LH | A). We can combine all three using Bayes' rule to get P (A | LH), the probability of being age A at death (in the study year) given that you're left-handed. To make this answer meaningful,**

though, we also want to compare it to P (A | RH), the probability of being age A at death given that you are right-handed. We are calculating the following quantity twice, once for left-handers and once for right-handers.

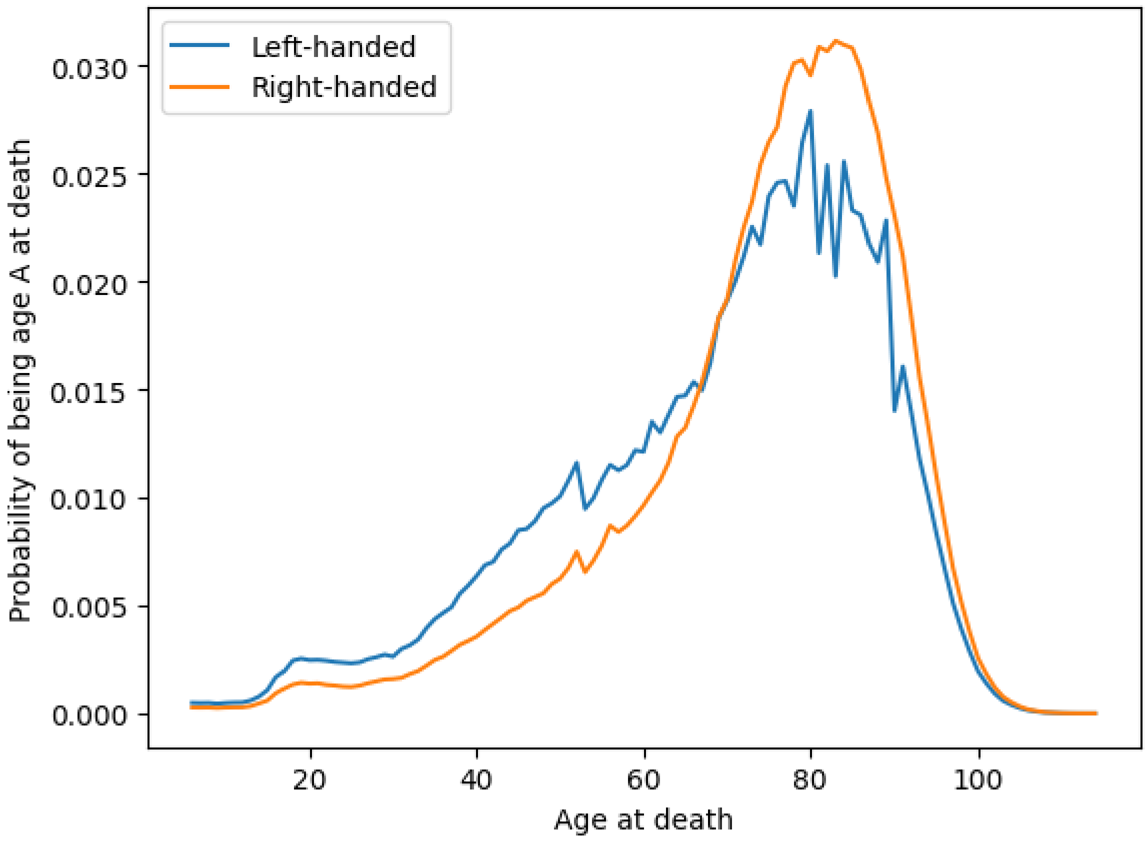
- A function to calculate P_A_given_lh() is called.
- P_A, the overall probability of dying at age A, which is given death_distribution_data at age A divided by the total number of dead people (the sum of the Both Sexes column of death_distribution_data).
- Calculated the overall probability of left-handedness P(LH) using the function defined in Task 5.
- Calculated P(LH | A) using the function defined in **Task 3**

The same task will be continued for P_A_given_rh(). The overall probability of right-handedness is P(RH), which is 1-P(LH). The value of P(RH | A), which is 1 - P(LH | A).



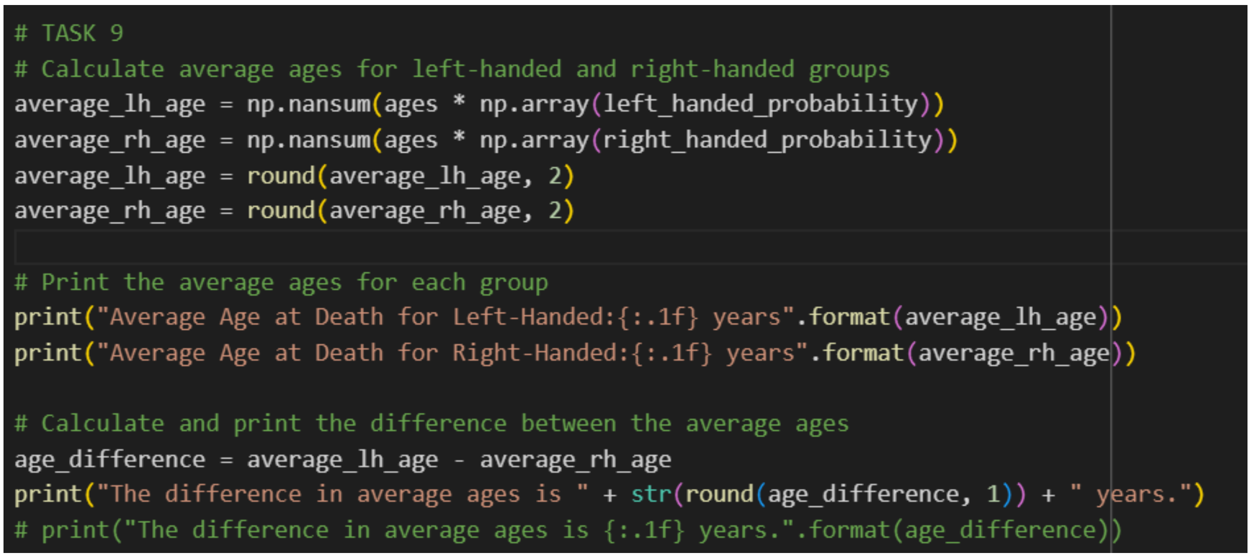
```
# TASK 6

def P_A_given_lh(ages_of_death, death_distribution_data, study_year=1990):
    """The overall probability of being a particular `age_of_death` given that you're left-handed"""
    P_A = death_distribution_data[death_distribution_data['Age'].isin(ages_of_death)]['Both Sexes'].values[0] / \
          death_distribution_data['Both Sexes'].sum()
    P_left = P_lh(death_distribution_data, study_year)
    P_lh_A = P_lh_given_A(ages_of_death, study_year)
    return P_lh_A * P_A / P_left


# TASK 7

def P_A_given_rh(ages_of_death, death_distribution_data, study_year=1990):
    """The overall probability of being a particular `age_of_death` given that you're left-handed"""
    P_A = death_distribution_data[death_distribution_data['Age'].isin(ages_of_death)]['Both Sexes'].values[0] / \
          death_distribution_data['Both Sexes'].sum()
    P_right = 1 - P_lh(death_distribution_data, study_year)
    P_rh_A = 1 - P_lh_given_A(ages_of_death, study_year)
    return P_rh_A * P_A / P_right
```

➢ **TASK 8: Plotting the probability of being a certain age at death given that a person is left- or right-handed for a range of ages, the range is between 6 years to 120 years.**



1. Maximum amount of people who are left-handed die younger.
2. From age 50 to age nearly 80 the probability of being aged at death is significantly increased.
3. For right-handed persons the probability is almost constant.
4. At the age of 65 and 80 the probability of being age A at death is and right-handed people. **(P ≈ 0.00030)**

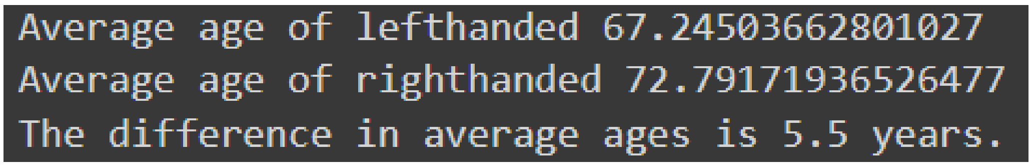
- **TASK 9: Mean age at death for left-handers and right-handers.**



```
# TASK 9
# Calculate average ages for left-handed and right-handed groups
average_lh_age = np.nansum(ages * np.array(left_handed_probability))
average_rh_age = np.nansum(ages * np.array(right_handed_probability))
average_lh_age = round(average_lh_age, 2)
average_rh_age = round(average_rh_age, 2)

# Print the average ages for each group
print("Average Age at Death for Left-Handed:{:.1f} years".format(average_lh_age))
print("Average Age at Death for Right-Handed:{:.1f} years".format(average_rh_age))

# Calculate and print the difference between the average ages
age_difference = average_lh_age - average_rh_age
print("The difference in average ages is " + str(round(age_difference, 1)) + " years.")
# print("The difference in average ages is {:.1f} years.".format(age_difference))
```

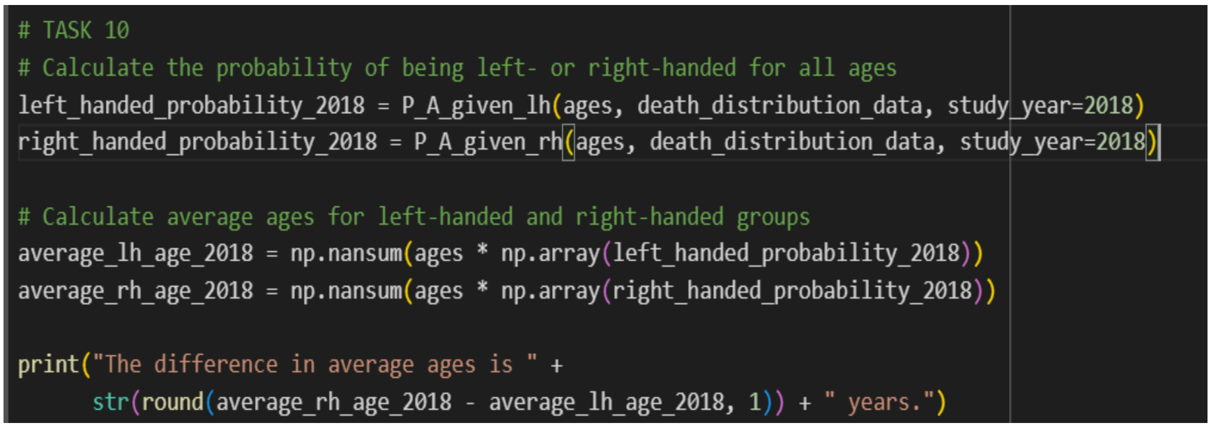
- Multiply the ages list by the left-handed probabilities of being those ages at death, then use **np. nansum** to calculate the sum.
- Assigned the result to average_lh_age.Did the same with the right-handed probabilities to calculate **average_rh_age.**
- Print **average_lh_age** and **average_rh_age**.
- Calculated the difference between the two average ages and printed it.
- To make the printed output prettier, We used the **round()** function to round your results to two decimal places.

OUTPUT of the above code



```
Average age of lefthanded 67.24503662801027
Average age of righthanded 72.79171936526477
The difference in average ages is 5.5 years.
```

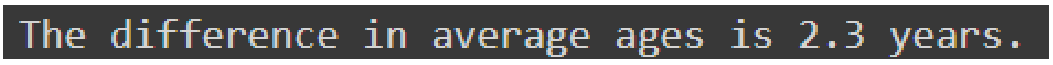
- **TASK 10: The ultimatum: To finish off, calculate the age gap we would expect if we did the study in 2018 instead of in 1990.**



```
# TASK 10
# Calculate the probability of being left- or right-handed for all ages
left_handed_probability_2018 = P_A_given_lh(ages, death_distribution_data, study_year=2018)
right_handed_probability_2018 = P_A_given_rh(ages, death_distribution_data, study_year=2018)

# Calculate average ages for left-handed and right-handed groups
average_lh_age_2018 = np.nansum(ages * np.array(left_handed_probability_2018))
average_rh_age_2018 = np.nansum(ages * np.array(right_handed_probability_2018))

print("The difference in average ages is " +
      str(round(average_rh_age_2018 - average_lh_age_2018, 1)) + " years.")
```

- We almost did the same work as task 9, But we did this analysis for the year 2018.
- The rates of left-handedness are much smaller since left-handedness has not increased for the people born after about 1960.
- It's very interesting that the difference between young and old has a large amount of difference in the probability of death.

The Output of the above code is



```
The difference in average ages is 2.3 years.
```

CONCLUSION AND FUTURE ASPECTS

The conclusion of the project will depend on the results obtained from the data analysis. These tasks involve data manipulation, probability calculations, and visualization using libraries such as pandas, numpy, and matplotlib. The project aims to analyze the relationship between handedness, age at death, and population statistics to uncover insights and patterns within the data. We have a pretty big age gap between left-handed and right-handed people purely because of the changing rates of left-handedness in the population, which is good news for left-handers: someone probably won't die young because of their sinisterness. The reported rates of left-handedness have increased from just 3% in the early 1900s to about 11% today, which means that older people are much more likely to be reported as right-handed than left-handed, and so looking at a sample of recently deceased people will have more old right-handers. Our number is still less than the 9-year gap measured in the study.

1. figure out how much variability we would expect to encounter in the age difference purely because of random sampling:
2. if we take a smaller sample of recently deceased people and assign handedness with the probabilities of the survey, what does that distribution look like?
3. How often would we encounter an age gap of nine years using the same data and assumptions?
4. Gather a larger and more diverse dataset for a more robust analysis.
5. Refine the methodology to account for potential confounding variables.
6. Collaborate with experts in genetics and public health to gain a better understanding of the underlying mechanisms.

REFERENCES

Data Collection and Theory Collection

- https://gist.githubusercontent.com/mbonsma/8da0990b71ba9a09f7de395574e54df1/raw/aec88b30af87fad8d45da7e774223f91dad09e88/lh_data.csv (1992 paper by Gilbert and Wysocki)

- https://gist.githubusercontent.com/mbonsma/2f4076aab6820ca1807f4e29f75f18ec/raw/62f3ec07514c7e31f5979beeca86f19991540796/cdc_vs00199_table310.tsv (death distribution data)

- https://www.medtoureasy.com/about-us/

Programming Concepts

- https://matplotlib.org/

- https://pandas.pydata.org/

- https://numpy.org/