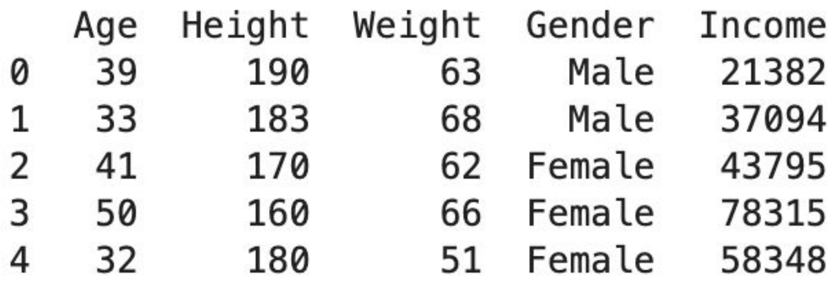
Statistical Analysis & Visualization in Python

By Jalal Haider

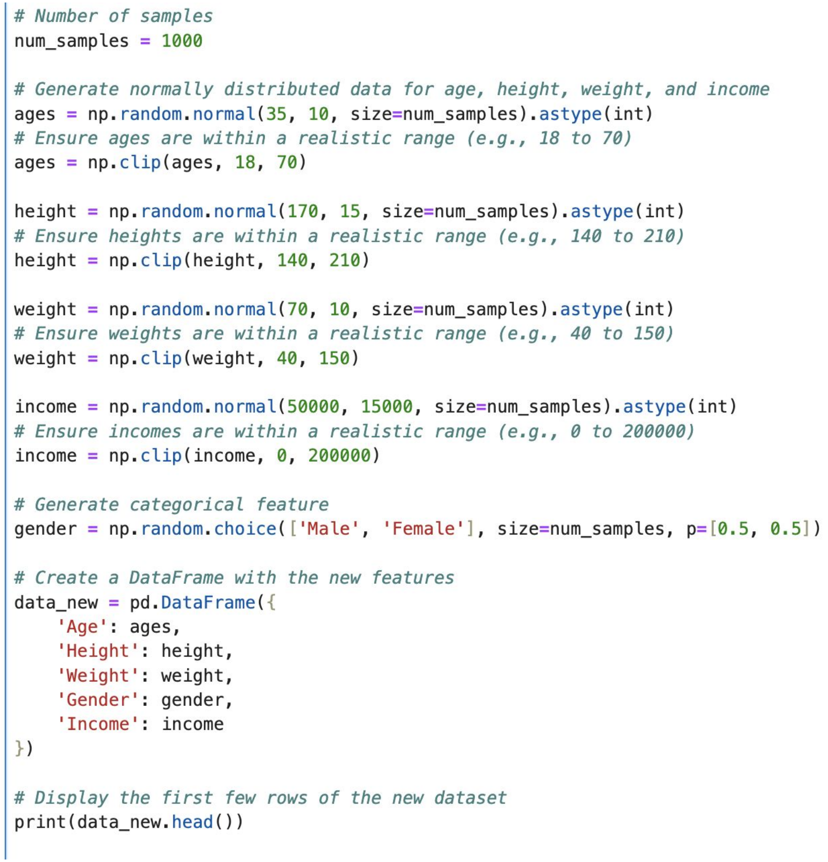
https://github.com/jalalhaider19/Jalal/blob/main/P1-OESON-Jalal.py

Data Generation



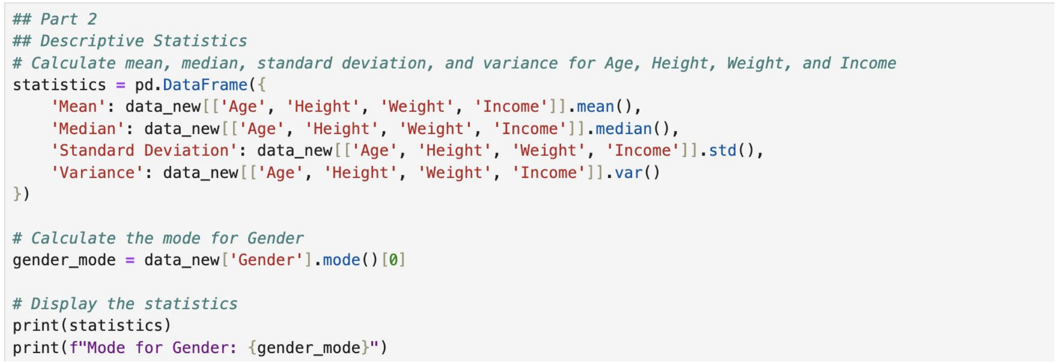
```
   Age  Height  Weight  Gender  Income
0   39     190      63    Male   21382
1   33     183      68    Male   37094
2   41     170      62  Female   43795
3   50     160      66  Female   78315
4   32     180      51  Female   58348
```

The central limit theorem justifies their use, as the sum of many independent variables tends to be normally distributed. This distribution's properties, like the predictable spread of data around the mean, simplify the application of parametric tests and modeling techniques, such as t-tests and linear regression. Additionally, in order to mitigate inaccuracies, e.g: heights and measurement errors, following a normal distribution, makes it a good fit for deriving insights from the data.

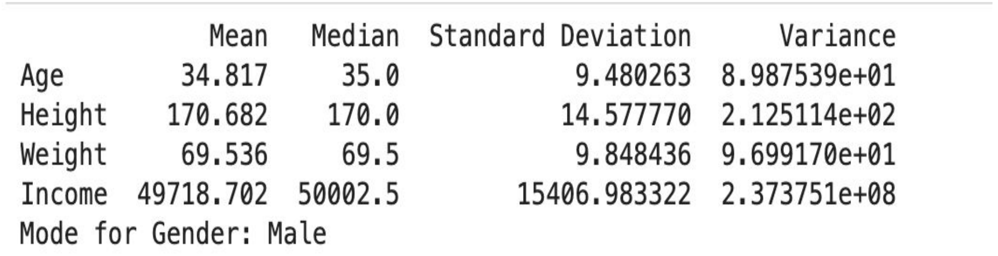


```python
# Number of samples
num_samples = 1000

# Generate normally distributed data for age, height, weight, and income
ages = np.random.normal(35, 10, size=num_samples).astype(int)
# Ensure ages are within a realistic range (e.g., 18 to 70)
ages = np.clip(ages, 18, 70)

height = np.random.normal(170, 15, size=num_samples).astype(int)
# Ensure heights are within a realistic range (e.g., 140 to 210)
height = np.clip(height, 140, 210)

weight = np.random.normal(70, 10, size=num_samples).astype(int)
# Ensure weights are within a realistic range (e.g., 40 to 150)
weight = np.clip(weight, 40, 150)

income = np.random.normal(50000, 15000, size=num_samples).astype(int)
# Ensure incomes are within a realistic range (e.g., 0 to 200000)
income = np.clip(income, 0, 200000)

# Generate categorical feature
gender = np.random.choice(['Male', 'Female'], size=num_samples, p=[0.5, 0.5])

# Create a DataFrame with the new features
data_new = pd.DataFrame({
    'Age': ages,
    'Height': height,
    'Weight': weight,
    'Gender': gender,
    'Income': income
})

# Display the first few rows of the new dataset
print(data_new.head())
```

Descriptive Statistics



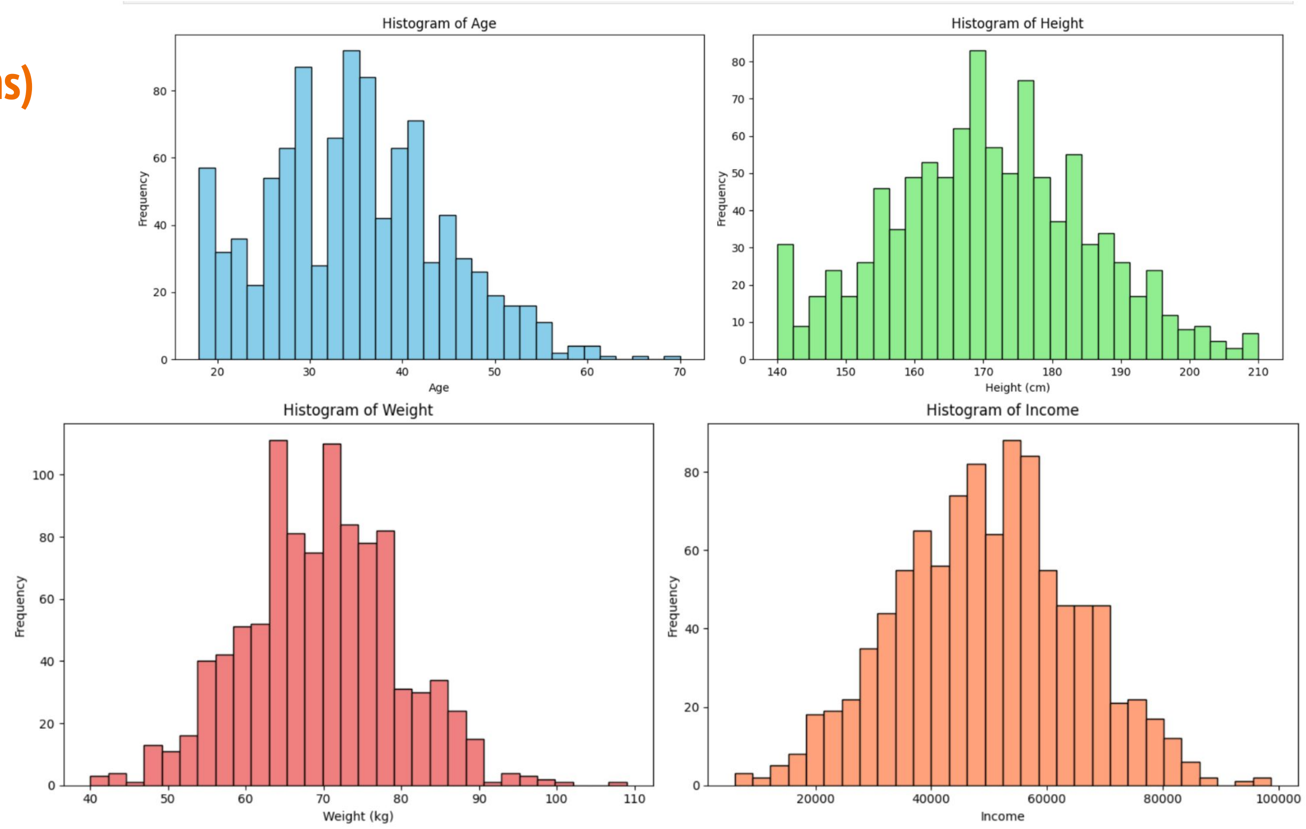
```python
## Part 2
## Descriptive Statistics
# Calculate mean, median, standard deviation, and variance for Age, Height, Weight, and Income
statistics = pd.DataFrame({
    'Mean': data_new[['Age', 'Height', 'Weight', 'Income']].mean(),
    'Median': data_new[['Age', 'Height', 'Weight', 'Income']].median(),
    'Standard Deviation': data_new[['Age', 'Height', 'Weight', 'Income']].std(),
    'Variance': data_new[['Age', 'Height', 'Weight', 'Income']].var()
})

# Calculate the mode for Gender
gender_mode = data_new['Gender'].mode()[0]

# Display the statistics
print(statistics)
print(f"Mode for Gender: {gender_mode}")
```



```
              Mean   Median  Standard Deviation      Variance
Age         34.817     35.0            9.480263  8.987539e+01
Height     170.682    170.0           14.577770  2.125114e+02
Weight      69.536     69.5            9.848436  9.699170e+01
Income   49718.702  50002.5        15406.983322  2.373751e+08
Mode for Gender: Male
```

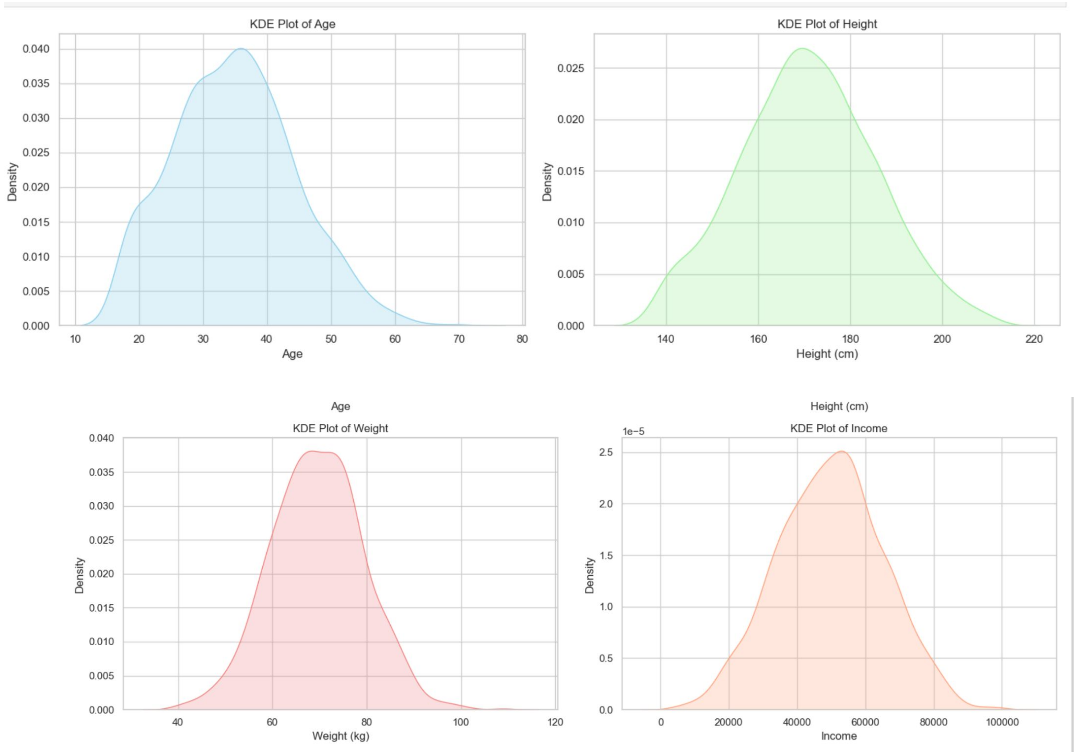
Data Visualization (Histograms)

- Age is rightly skewed with the distribution falling to the left and tail to the right
- Weight has unimodal distribution
- Income is more normally distributed
- Height is also unimodally distributed with one distinct peak



Data Visualization (KDE Plots)

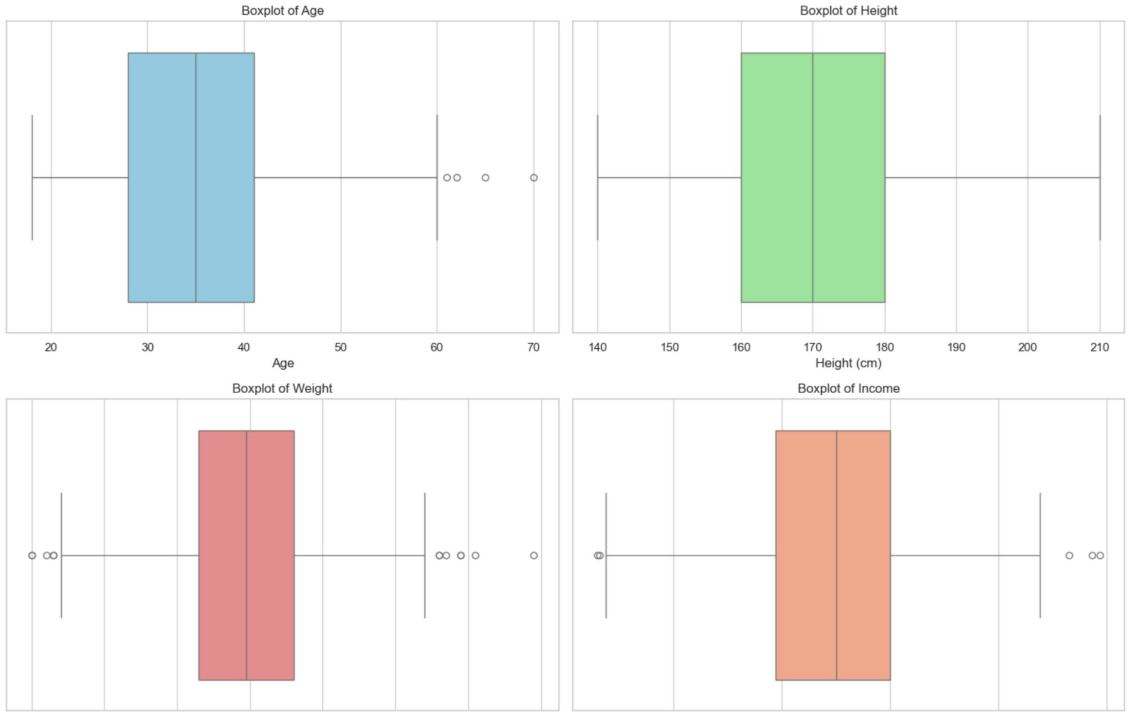
* Also show similar findings to that of histograms



Data Visualization (Box Plots)

Circles represent outliers in the boxplot.

- Height does not have any outliers
- Weight & Income have a few outliers, all out of the lower and upper whiskers
- Age has a few but most outliers, all beyond the upper quartile range, backed by skewness



Correlation Analysis

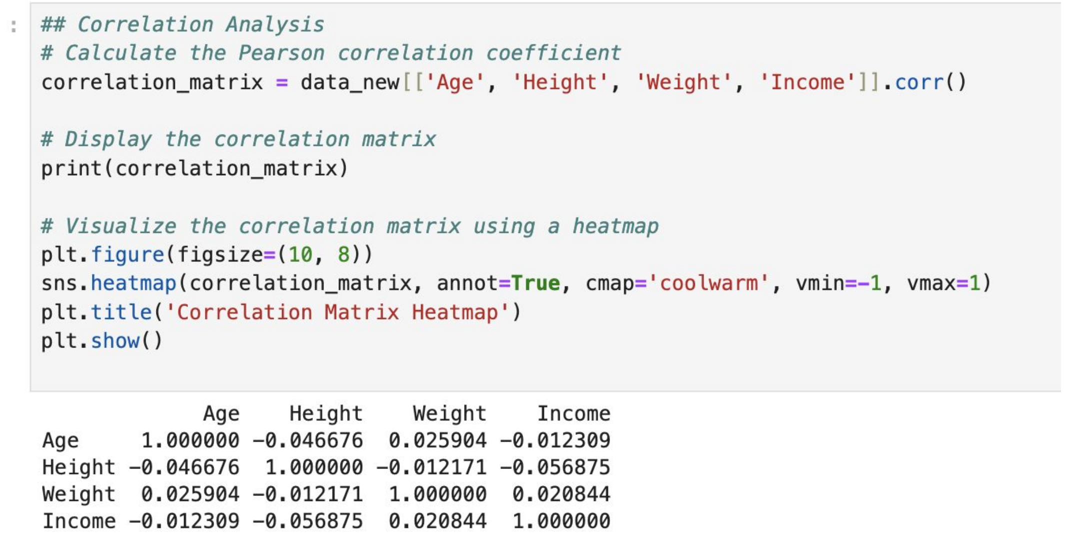
Age & Weight have a positive correlation

Age & Height, Age & Income have a negative correlation

Income & weight have a slight positive correlation (very bizarre)

Age and Income also have a very slight negative correlation.

Height & Weight also have a very slight negative correlation.



```python
## Correlation Analysis
# Calculate the Pearson correlation coefficient
correlation_matrix = data_new[['Age', 'Height', 'Weight', 'Income']].corr()

# Display the correlation matrix
print(correlation_matrix)

# Visualize the correlation matrix using a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Matrix Heatmap')
plt.show()
```

```
             Age    Height    Weight    Income
Age     1.000000 -0.046676  0.025904 -0.012309
Height -0.046676  1.000000 -0.012171 -0.056875
Weight  0.025904 -0.012171  1.000000  0.020844
Income -0.012309 -0.056875  0.020844  1.000000
```

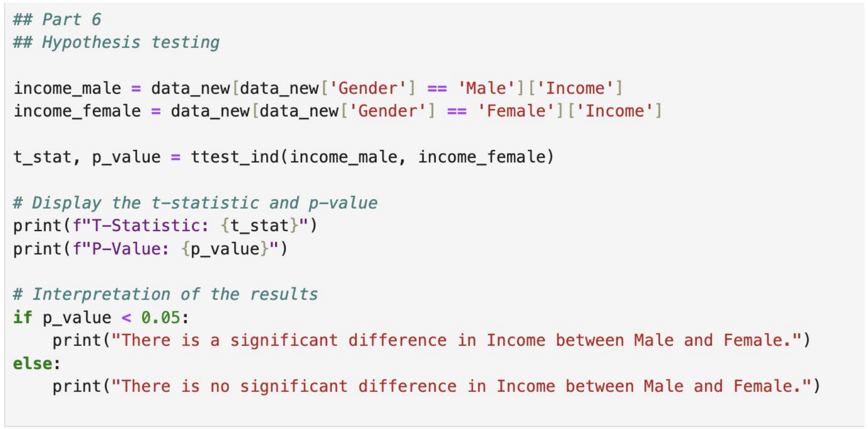
Inferential Statistics

Null hypothesis:

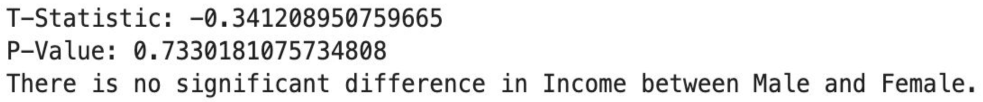
μMale = μFemale

Alternate Hypothesis:

μMale ≠ μFemale



```
## Part 6
## Hypothesis testing

income_male = data_new[data_new['Gender'] == 'Male']['Income']
income_female = data_new[data_new['Gender'] == 'Female']['Income']

t_stat, p_value = ttest_ind(income_male, income_female)

# Display the t-statistic and p-value
print(f"T-Statistic: {t_stat}")
print(f"P-Value: {p_value}")

# Interpretation of the results
if p_value < 0.05:
    print("There is a significant difference in Income between Male and Female.")
else:
    print("There is no significant difference in Income between Male and Female.")
```



```
T-Statistic: -0.341208950759665
P-Value: 0.7330181075734808
There is no significant difference in Income between Male and Female.
```

Key Findings and Conclusions

- Data seems to be have some strange correlations such as Weight and Income, shows that there are definitely some bad
- Considering Age is skewed to the right, it might be better to consider the mean as a better measure for central tendency
- For gender, it is better to consider the mode as it is a nominal data type
- Gender disparity from this dataset has no significant effect on the income
- For normally distributions of weight and income it is also better to use the mean as a measure for central tendency

Thank You!