Lab - Correlation Analysis in Python

Name: John Loyd C. Santiago
Course and Section: CPE 019-CPE32S3
Date of Submission: 7/2/24
Instructor: Engr. Roman Richard

Objectives

- **Part 1: The Dataset**
- **Part 2: Scatterplot Graphs and Correlatable Variables**
- **Part 3: Calculating Correlation with Python**
- **Part 4: Visualizing**

Scenario/Background

Correlation is an important statistical relationship that can indicate whether the variable values are linearly related.

In this lab, you will learn how to use Python to calculate correlation. In Part 1, you will setup the dataset. In Part 2, you will learn how to identify if the variables in a given dataset are correlatable. Finally, in Part 3, you will use Python to calculate the correlation between two sets of variable.

Required Resources

- 1 PC with Internet access
- Raspberry Pi version 2 or higher
- Python libraries: pandas, numpy, matplotlib, seaborn
- Datafiles: brainsize.txt

Part 1: The Dataset

You will use a dataset that contains a sample of 40 right-handed Anglo Introductory Psychology students at a large Southwestern university. Subjects took four subtests (Vocabulary, Similarities, Block Design, and Picture Completion) of the Wechsler (1981) Adult Intelligence Scale-Revised. The researchers used Magnetic Resonance Imaging (MRI) to determine the brain size of the subjects. Information about gender and body size (height and weight) are also included. The researchers withheld the weights of two subjects and the height of one subject for reasons of confidentiality. Two simple modifications were applied to the dataset:

1. Replace the quesion marks used to represent the withheld data points described above by the 'NaN' string. The substitution was done because Pandas does not handle the question marks correctly.
2. Replace all tab characters with commas, converting the dataset into a CSV dataset.

The prepared dataset is saved as `brainsize.txt`.

Step 1: Loading the Dataset From a File.

Before the dataset can be used, it must be loaded onto memory.

In the code below, The first line imports the `pandas` modules and defines `pd` as a descriptor that refers to the module.

The second line loads the dataset CSV file into a variable called `brainFile`.

The third line uses `read_csv()`, a `pandas` method, to convert the CSV dataset stored in `brainFile` into a dataframe. The dataframe is then stored in the `brainFrame` variable.

Run the cell below to execute the described functions.

```
import pandas as pd
brainFile = 'brainsize.txt'
brainFrame = pd.read_csv(brainFile, '\t')
```

```
<ipython-input-31-b6627a3e1c2d>:3: FutureWarning: In a future version of pandas all arguments of read_csv except for the argument 'file
  brainFrame = pd.read_csv(brainFile, '\t')
```

Step 2: Verifying the dataframe.

To make sure the dataframe has been correctly loaded and created, use the `head()` method. Another Pandas method, `head()` displays the first five entries of a dataframe.

```
# Here I just show the first 5 data
brainFrame.head()
```

	Gender	FSIQ	VIQ	PIQ	Weight	Height	MRI_Count
0	Female	133	132	124	118.0	64.5	816932
1	Male	140	150	124	NaN	72.5	1001121
2	Male	139	123	150	143.0	73.3	1038437
3	Male	133	129	128	172.0	68.8	965353
4	Female	137	132	134	147.0	65.0	951545

```
# In here I just show all of the data
brainFrame.head(40)
```

	Gender	FSIQ	VIQ	PIQ	Weight	Height	MRI_Count
0	Female	133	132	124	118.0	64.5	816932
1	Male	140	150	124	NaN	72.5	1001121
2	Male	139	123	150	143.0	73.3	1038437
3	Male	133	129	128	172.0	68.8	965353
4	Female	137	132	134	147.0	65.0	951545
5	Female	99	90	110	146.0	69.0	928799
6	Female	138	136	131	138.0	64.5	991305
7	Female	92	90	98	175.0	66.0	854258
8	Male	89	93	84	134.0	66.3	904858
9	Male	133	114	147	172.0	68.8	955466
10	Female	132	129	124	118.0	64.5	833868
11	Male	141	150	128	151.0	70.0	1079549
12	Male	135	129	124	155.0	69.0	924059
13	Female	140	120	147	155.0	70.5	856472
14	Female	96	100	90	146.0	66.0	878897
15	Female	83	71	96	135.0	68.0	865363
16	Female	132	132	120	127.0	68.5	852244
17	Male	100	96	102	178.0	73.5	945088
18	Female	101	112	84	136.0	66.3	808020
19	Male	80	77	86	180.0	70.0	889083
20	Male	83	83	86	NaN	NaN	892420
21	Male	97	107	84	186.0	76.5	905940
22	Female	135	129	134	122.0	62.0	790619
23	Male	139	145	128	132.0	68.0	955003
24	Female	91	86	102	114.0	63.0	831772
25	Male	141	145	131	171.0	72.0	935494
26	Female	85	90	84	140.0	68.0	798612
27	Male	103	96	110	187.0	77.0	1062462
28	Female	77	83	72	106.0	63.0	793549
29	Female	130	126	124	159.0	66.5	866662
30	Female	133	126	132	127.0	62.5	857782

Part 2: Scatterplot Graphs and Correlatable Variables

Step 1: The pandas `describe()` method.

The pandas module includes the `describe()` method which performs same common calculations against a given dataset. In addition to provide common results including count, mean, standard deviation, minimum, and maximum, `describe()` is also a great way to quickly test the validity of the values in the dataframe.

Run the cell below to output the results computed by `describe()` against the `brainFrame` dataframe.

```
#
brainFrame.describe()
```

	FSIQ	VIQ	PIQ	Weight	Height	MRI_Count
count	40.000000	40.000000	40.00000	38.000000	39.000000	4.000000e+01
mean	113.450000	112.350000	111.02500	151.052632	68.525641	9.087550e+05
std	24.082071	23.616107	22.47105	23.478509	3.994649	7.228205e+04
min	77.000000	71.000000	72.00000	106.000000	62.000000	7.906190e+05
25%	89.750000	90.000000	88.25000	135.250000	66.000000	8.559185e+05
50%	116.500000	113.000000	115.00000	146.500000	68.000000	9.053990e+05
75%	135.500000	129.750000	128.00000	172.000000	70.500000	9.500780e+05
max	144.000000	150.000000	150.00000	192.000000	77.000000	1.079549e+06

Step 2: Scatterplot graphs

Scatterplot graphs are important when working with correlations as they allow for a quick visual verification of the nature of the relationship between the variables. This lab uses the Pearson correlation coefficient, which is sensitive only to a linear relationship between two variables. Other more robust correlation methods exist but are out of the scope of this lab.

a. Load the required modules.

Before graphs can be plotted, it is necessary to import a few modules, namely `numpy` and `matplotlib`. Run the cell below to load these modules.

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

b. Separate the data.

To ensure the results do not get skewed because of the differences in male and female bodies, the dateframe is split into two dataframes: one containing all male entries and another with only female instances.

Running the cell below creates the two new dataframes, menDf and womenDf, each one containing the respective entries.

```
menDf = brainFrame[(brainFrame.Gender == 'Male')]
womenDf = brainFrame[(brainFrame.Gender == 'Female')]
```

c. Plot the graphs.

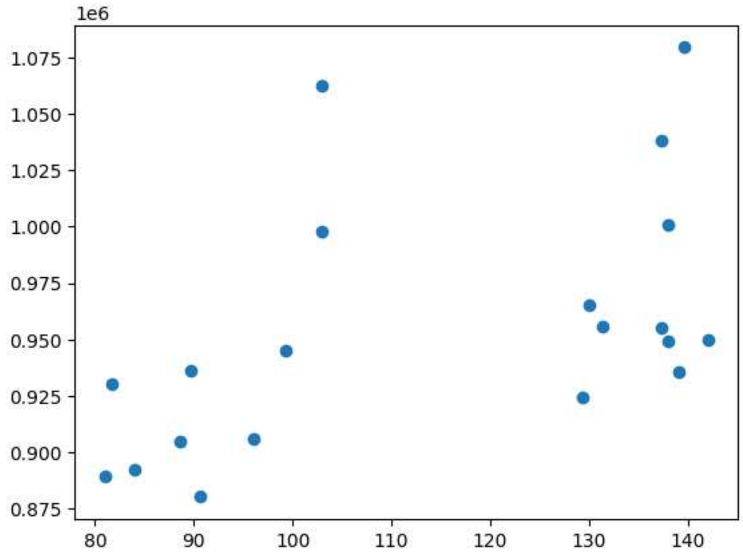
Because the dataset includes three different measures of intelligence (PIQ, FSIQ, and VIQ), the first line below uses Pandas `mean()` method to calculate the mean value between the three and store the result in the `menMeanSmarts` variable. Notice that the first line also refers to the menDf, the filtered dataframe containing only male entries.

The second line uses the `matplotlib` method `scatter()` to create a scatterplot graph between the `menMeanSmarts` variable and the `MRI_Count` attribute. The MRI_Count in this dataset can be thought as of a measure of the physical size of the subjects' brains.

The third line simply displays the graph.

The fourth line is used to ensure the graph will be displayed in this notebook.

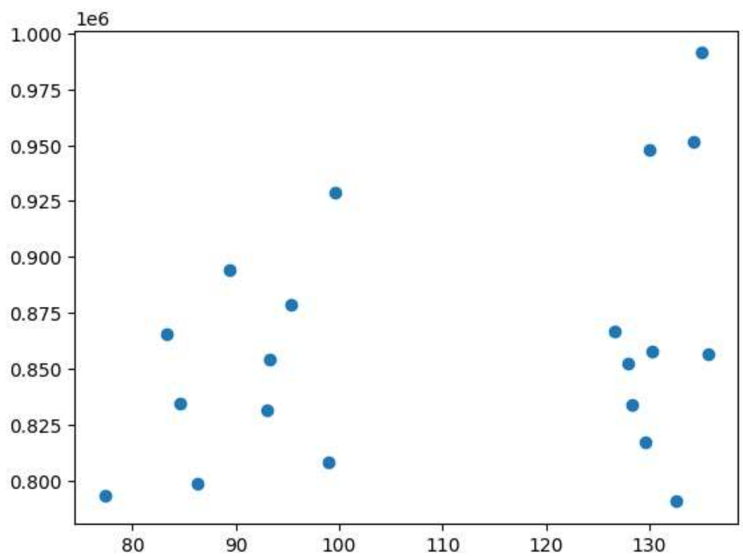
```
menMeanSmarts = menDf[["PIQ", "FSIQ", "VIQ"]].mean(axis=1)
plt.scatter(menMeanSmarts, menDf["MRI_Count"])
plt.show()
%matplotlib inline
```



Similarly, the code below creates a scatterplot graph for the women-only filtered dataframe.

```
womenMeanSmarts = womenDf[["PIQ", "FSIQ", "VIQ"]].mean(axis=1)
plt.scatter(womenMeanSmarts, womenDf["MRI_Count"])

plt.show()
%matplotlib inline
```



Part 3: Calculating Correlation with Python

Step 1: Calculate correlation against brainFrame.

The pandas `corr()` method provides an easy way to calculate correlation against a dataframe. By simply calling the method against a dataframe, one can get the correlation between all variables at the same time.

```
brainFrame.corr(method='pearson')
```

```
<ipython-input-23-4d3089cc6357>:1: FutureWarning: The default value of numeric_only in
  brainFrame.corr(method='pearson')
```

	FSIQ	VIQ	PIQ	Weight	Height	MRI_Count
FSIQ	1.000000	0.946639	0.934125	-0.051483	-0.086002	0.357641
VIQ	0.946639	1.000000	0.778135	-0.076088	-0.071068	0.337478
PIQ	0.934125	0.778135	1.000000	0.002512	-0.076723	0.386817
Weight	-0.051483	-0.076088	0.002512	1.000000	0.699614	0.513378
Height	-0.086002	-0.071068	-0.076723	0.699614	1.000000	0.601712
MRI_Count	0.357641	0.337478	0.386817	0.513378	0.601712	1.000000

Notice at the left-to-right diagonal in the correlation table generated above. Why is the diagonal filled with 1s? Is that a coincidence? Explain.

Because its where the same correlation intersect

Still looking at the correlation table above, notice that the values are mirrored; values below the 1 diagonal have a mirrored counterpart above the 1 diagonal. Is that a coincidence? Explain.

No, Because the they are like that because of their specified correlation

Using the same `corr()` method, it is easy to calculate the correlation of the variables contained in the female-only dataframe:

```
womenDf.corr(method='pearson')
```

```
<ipython-input-24-01fad84dd5db>:1: FutureWarning: The default value of numeric_only in
  womenDf.corr(method='pearson')
```

	FSIQ	VIQ	PIQ	Weight	Height	MRI_Count
FSIQ	1.000000	0.955717	0.939382	0.038192	-0.059011	0.325697
VIQ	0.955717	1.000000	0.802652	-0.021889	-0.146453	0.254933
PIQ	0.939382	0.802652	1.000000	0.113901	-0.001242	0.396157
Weight	0.038192	-0.021889	0.113901	1.000000	0.552357	0.446271
Height	-0.059011	-0.146453	-0.001242	0.552357	1.000000	0.174541
MRI_Count	0.325697	0.254933	0.396157	0.446271	0.174541	1.000000

And the same can be done for the male-only dataframe:

```
menDf.corr(method='pearson')
```

```
<ipython-input-25-4396b7a1db7e>:1: FutureWarning: The default value of numeric_only in
  menDf.corr(method='pearson')
```

	FSIQ	VIQ	PIQ	Weight	Height	MRI_Count
FSIQ	1.000000	0.944400	0.930694	-0.278140	-0.356110	0.498369
VIQ	0.944400	1.000000	0.766021	-0.350453	-0.355588	0.413105
PIQ	0.930694	0.766021	1.000000	-0.156863	-0.287676	0.568237
Weight	-0.278140	-0.350453	-0.156863	1.000000	0.406542	-0.076875
Height	-0.356110	-0.355588	-0.287676	0.406542	1.000000	0.301543
MRI_Count	0.498369	0.413105	0.568237	-0.076875	0.301543	1.000000

Part 4: Visualizing

Step 1: Install Seaborn.

To make it easier to visualize the data correlations, heatmap graphs can be used. Based on colored squares, heatmap graphs can help identify correlations in a glance.

The Python module named `seaborn` makes it very easy to plot heatmap graphs.

First, run the cell below to download and install the `seaborn` module.

```
# Code cell 11
!pip install seaborn
```

```
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.13.1)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/python3.10/dist-packages (from seaborn) (1.23.5)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.10/dist-packages (from seaborn) (1.5.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3.10/dist-packages (from seaborn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.2
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (23.2
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.1
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.2->seaborn) (2023.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4-
```

With this code you can now access seaborn

Step 2: Plot the correlation heatmap.

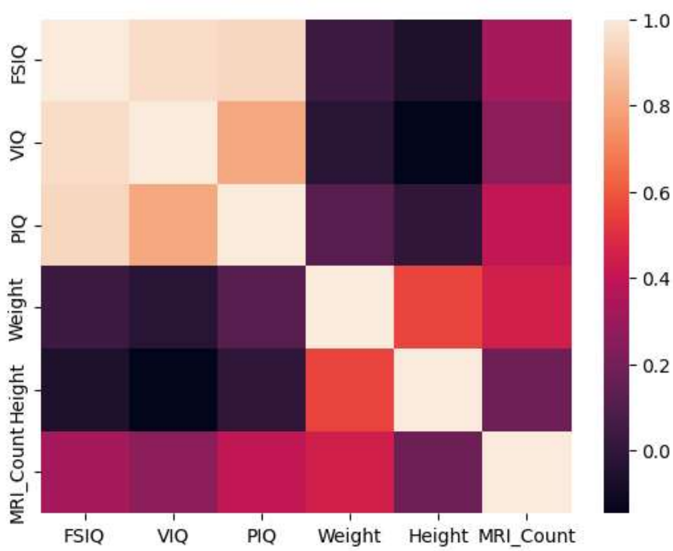
Now that the dataframes are ready, the heatmaps can be plotted. Below is a breakdown of the code in the cell below:

Line 1: Generates a correlation table based on the `womenNoGenderDf` dataframe and stores it on `wcorr`.
Line 2: Uses the `seaborn heatmap()` method to generate and plot the heatmap. Notice that `heatmap()` takes `wcorr` as a parameter.
Line 3: Use to export and save the generated heatmap as a PNG image. While the line 3 is not active (it has the comment `#` character preceding it, forcing the interpreter to ignore it), it was kept for informational purposes.

```
# Code cell 12
import seaborn as sns

wcorr = womenDf.corr()
sns.heatmap(wcorr)
#plt.savefig('attribute_correlations.png', tight_layout=True)
```

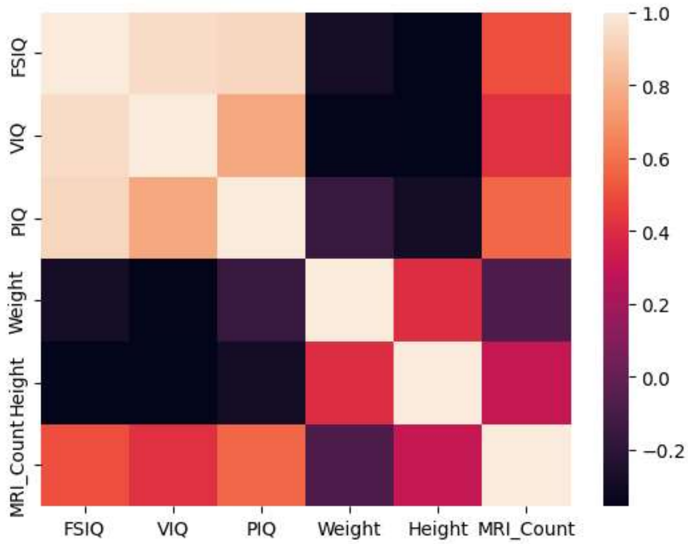
```
<ipython-input-28-4bc71e77167c>:4: FutureWarning: The default value of numeric_only in
  wcorr = womenDf.corr()
<Axes: >
```



Similarly, the code below creates and plots a heatmap for the male-only dataframe.

```
# Code cell 14
mcorr = menDf.corr()
sns.heatmap(mcorr)
#plt.savefig('attribute_correlations.png', tight_layout=True)
```

```
<ipython-input-29-ff3e250059fc>:2: FutureWarning: The default value of numeric_only in
  mcorr = menDf.corr()
<Axes: >
```



- Many variable pairs present correlation close to zero. What does that mean?

It means that alot of the variable are not linearly related

- Why separate the genders?

So that the result of your data is correct, Because if you have the men and woman in the same graph It can effect some variables and the might make a wrong result in your correlation

- What variables have stronger correlation with brain size (MRI_Count)? Is that expected? Explain.

The variables that have a stronger correlation with brain size or MRI_COUNT are FSIQ and PIQ. I think yes, Because a lot of people doesn't like talking in front of others so thats why I think they have lesser correlation than the other two

SUPPLEMENTARY

```
import pandas as pd
NETFile = 'netflix.txt'
stockFrame = pd.read_csv(NETFile, '\t')
```

```
<ipython-input-36-046a614064de>:3: FutureWarning: In a future version of pandas all arguments of read_csv except for the argument 'file
  stockFrame = pd.read_csv(NETFile, '\t')
```

```
stockFrame.head(250)
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	01/02/2023	353.859985	365.390015	349.910004	361.989990	361.989990	8005200
1	02/02/2023	365.160004	368.320007	358.429993	366.890015	366.890015	7857000
2	03/02/2023	359.079987	379.429993	359.000000	365.899994	365.899994	9402000
3	06/02/2023	363.640015	368.450012	360.679993	361.480011	361.480011	4994900
4	07/02/2023	358.510010	364.179993	354.179993	362.950012	362.950012	6289400
...	...	...	...	...	...	...	...
245	24/01/2024	537.750000	562.500000	537.070007	544.869995	544.869995	26432800
246	25/01/2024	551.950012	563.460022	548.460022	562.000000	562.000000	9451900
247	26/01/2024	561.809998	579.640015	558.429993	570.419983	570.419983	12754500
248	29/01/2024	571.349976	578.549988	562.679993	575.789978	575.789978	6905400
249	30/01/2024	567.320007	570.880005	560.820007	562.849976	562.849976	6181800

250 rows × 7 columns

here it show data that have gathered for a year related to Netflix (NFLX) stock prices with various vaiables like Date, Open, High, Close, Adj Close and volume.

```
stockFrame.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	251.000000	251.000000	251.000000	251.000000	251.000000	2.510000e+02
mean	404.181354	409.754742	398.958446	404.270478	404.270478	6.135308e+06
std	60.849872	61.318707	60.722608	61.193722	61.193722	3.814621e+06
min	287.339996	297.450012	285.329987	292.760010	292.760010	1.404700e+06
25%	348.994995	356.860001	344.490005	348.119995	348.119995	3.966000e+06
50%	412.000000	418.839996	407.399994	411.690002	411.690002	5.128900e+06
75%	444.729996	448.574997	439.175003	444.944992	444.944992	6.880600e+06
max	571.349976	579.640015	562.679993	575.789978	575.789978	2.807440e+07

In here I just show the validity of the datas that collected for the dataset and different variables.

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


stocks = stockFrame[["High", "Low", "Close", "Adj Close", "Volume"]].mean(axis=1)
plt.scatter(stocks, stockFrame["Open"])
plt.show()
%matplotlib inline
```

```
<ipython-input-48-1c0042e27081>:1: FutureWarning: Dropping of nuisance columns in DataF
  stocks = stockFrame[["Date","High", "Low", "Close", "Adj Close", "Volume"]].mean(axis
```


```
stockFrame.corr(method='pearson')
```

```
<ipython-input-44-6e756423900a>:1: FutureWarning: The default value of numeric_only in
  stockFrame.corr(method='pearson')
```

	Open	High	Low	Close	Adj Close	Volume
Open	1.000000	0.996777	0.997574	0.993954	0.993954	-0.066355
High	0.996777	1.000000	0.997285	0.997328	0.997328	-0.038919
Low	0.997574	0.997285	1.000000	0.997691	0.997691	-0.080164
Close	0.993954	0.997328	0.997691	1.000000	1.000000	-0.059169
Adj Close	0.993954	0.997328	0.997691	1.000000	1.000000	-0.059169
Volume	-0.066355	-0.038919	-0.080164	-0.059169	-0.059169	1.000000

In here I calculated the correlation of the data where it show that almost all of the variable have very strong positive correlation but not the with volume

```
import seaborn as sns

scorr = stockFrame.corr()
sns.heatmap(scorr)
```