SIT220/731 2022.T1: Task 3P

Working with numpy Matrices (Multidimensional Data)

[redacted]

Undergradute (SIT220)

1. Task:

Importing Libraries:

The purpose of this code is to import libraries. It contains bundles of code that can be used repeatedly in different programs. It makes Python Programming simpler and convenient for the programmer.

In [3]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import sys
np.set_printoptions(threshold=sys.maxsize) #comment out if you don't want to see whole array
```

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Loading files in notebook.

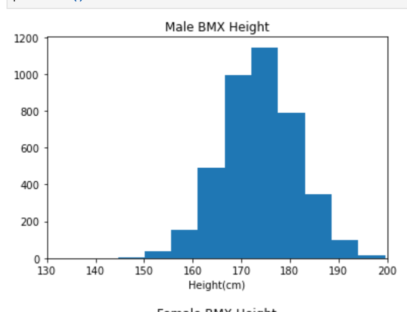
The purpose of this code is to read external files such as csv, txt, etc.

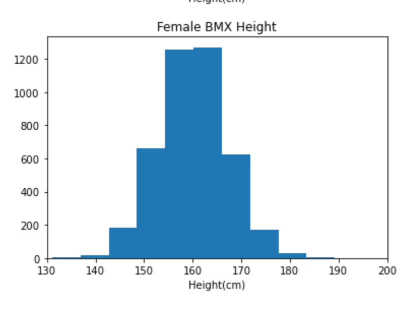
In [4]:
```
maleBMX = pd.read_csv('Data/nhanes_adult_male_bmx_2020.csv', comment = '#')
femaleBMX = pd.read_csv('Data/nhanes_adult_female_bmx_2020.csv', comment = '#')
maleBMXnp = maleBMX.to_numpy()
femaleBMXnp = femaleBMX.to_numpy()
#to_numpy() converts pandas dataframe to numpy array
```

This section shows male and female BMX Height on a graph

In [5]:
```
maleHeight = maleBMXnp[:, 1]
plt.hist(x = maleHeight)
plt.title('Male BMX Height')
plt.xlabel('Height(cm)')
plt.xlim(130, 200)
plt.show()

femaleHeight = femaleBMXnp[:, 1]
plt.hist(x = femaleHeight)
plt.title('Female BMX Height')
plt.xlabel('Height(cm)')
plt.xlim(130, 200)
plt.show()
```



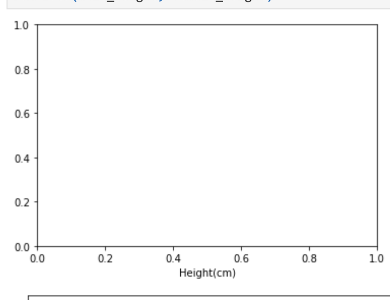


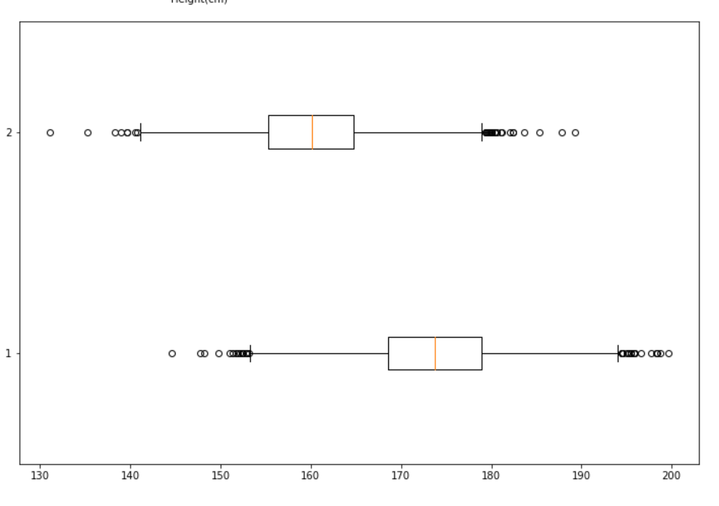
The boxplot version of male and female BMX height.

In [6]:
```
#Choose what data from csv file
male_height = maleBMXnp[:, 1]
female_height = femaleBMXnp[:, 1]

def BoxPlots(vector1, vector2):
    boxes = []
    boxes.append(vector1)
    boxes.append(vector2)
    plt.figure(figsize = (12, 8))
    plt.boxplot(boxes,vert=0)

plt.xlabel('Height(cm)') #The x label will depend on whatever data that is passed in
BoxPlots(male_height, female_height)
```





This section belows shows us the 5 number summary of male and female

[illegible]

Here we create an array of male's BMI

[illegible]

Then we append to the maleBMX file

[illegible]

By using scipy package, we are able to call a function which will calculate the zscore for our maleBMX file instead of manually calculating it ourself.

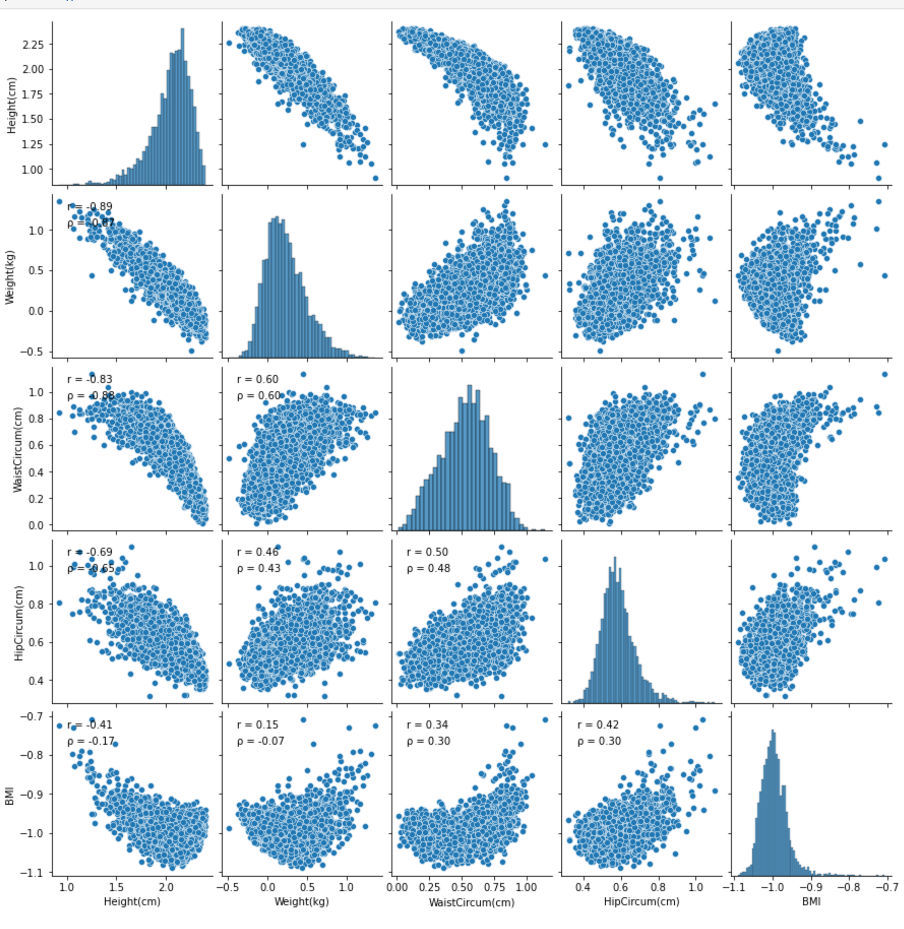
[illegible]

Here we are able to create a function where we can plot the pearsons' and spearmans' correlation on a graph

[illegible]

From the seaborn libary, we call pairplot to see correlation between different variables.

[illegible]



The pearson's and spearman's correlation measures the strength of the linear relationship between two variables. It has a value between -1 to 1, with a value of -1 meaning a total negative linear correlation, 0 being no correlation, and + 1 meaning a total positive correlation. For example, the closer the correlation number is to 1 the stronger the relationship between the variables.

2. Additional Tasks for Postgraduate (SIT731) Students (*)

This is where we two array for waist to height ratio and waist to hip ratio.

In [16]:
```
male_height = newMaleBMX[:, 1]
male_waist = newMaleBMX[:, 6]
male_hip = newMaleBMX[:, 5]

WaistHeightRatioList = []
WaistHipRatioList = []

#calc waist to height ratio
for W, H in zip(male_waist, male_height):
        ratio = W / H
        WaistHeightRatioList.append(ratio)

#calc waist to hip ratio
for W, H in zip(male_waist, male_hip):
        ratio = W / H
        WaistHipRatioList.append(ratio)

WaistHeightRatio = np.array(WaistHeightRatioList) #convert list into an array
WaistHipRatio = np.array(WaistHipRatioList)
```

Then add to the maleBMX file

In [17]:
```
newMaleBMX = np.append(newMaleBMX, WaistHeightRatio[:, None], axis=1) #add to column
newMaleBMX = np.append(newMaleBMX, WaistHipRatio[:, None], axis=1)
print(newMaleBMX[0:5,:])
```

```
[[ 98.8        182.3         42.          40.1         38.2
  108.2        120.4         29.72922633   0.66044981   1.11275416]
 [ 74.3        184.2         41.1         41.          30.2
   94.5         86.8         21.89825769   0.47122693   0.91851852]
 [103.7        185.3         47.          44.          32.
  107.8        109.6         30.20145858   0.59147329   1.01669759]
 [ 86.         167.8         39.5         38.4         29.
  106.4        108.3         30.54320016   0.6454112    1.01785714]
 [ 99.4        181.6         40.4         39.9         36.
  120.2        107.          30.1407945    0.58920705   0.89018303]]
```

In [18]:
```
newMaleBMX.shape
```

Out[18]:
```
(4081, 10)
```

We also do the same thing for female

In [19]:
```
female_height = femaleBMXnp[:, 1]
female_waist = femaleBMXnp[:, 6]
female_hip = femaleBMXnp[:, 5]

WaistHeightRatioList1 = []
WaistHipRatioList1 = []

#calc waist to height ratio
for W, H in zip(female_waist, female_height):
        ratio = W / H
        WaistHeightRatioList1.append(ratio)

#calc waist to hip ratio
for W, H in zip(female_waist, female_hip):
        ratio = W / H
        WaistHipRatioList1.append(ratio)

WaistHeightRatio1 = np.array(WaistHeightRatioList1) #convert list into an array
WaistHipRatio1 = np.array(WaistHipRatioList1)
```

In [21]:
```
newFemaleBMX = np.append(femaleBMXnp, WaistHeightRatio1[:, None], axis=1) #add to column
newFemaleBMX = np.append(newFemaleBMX, WaistHipRatio1[:, None], axis=1)
print(newFemaleBMX[0:5,:])
```

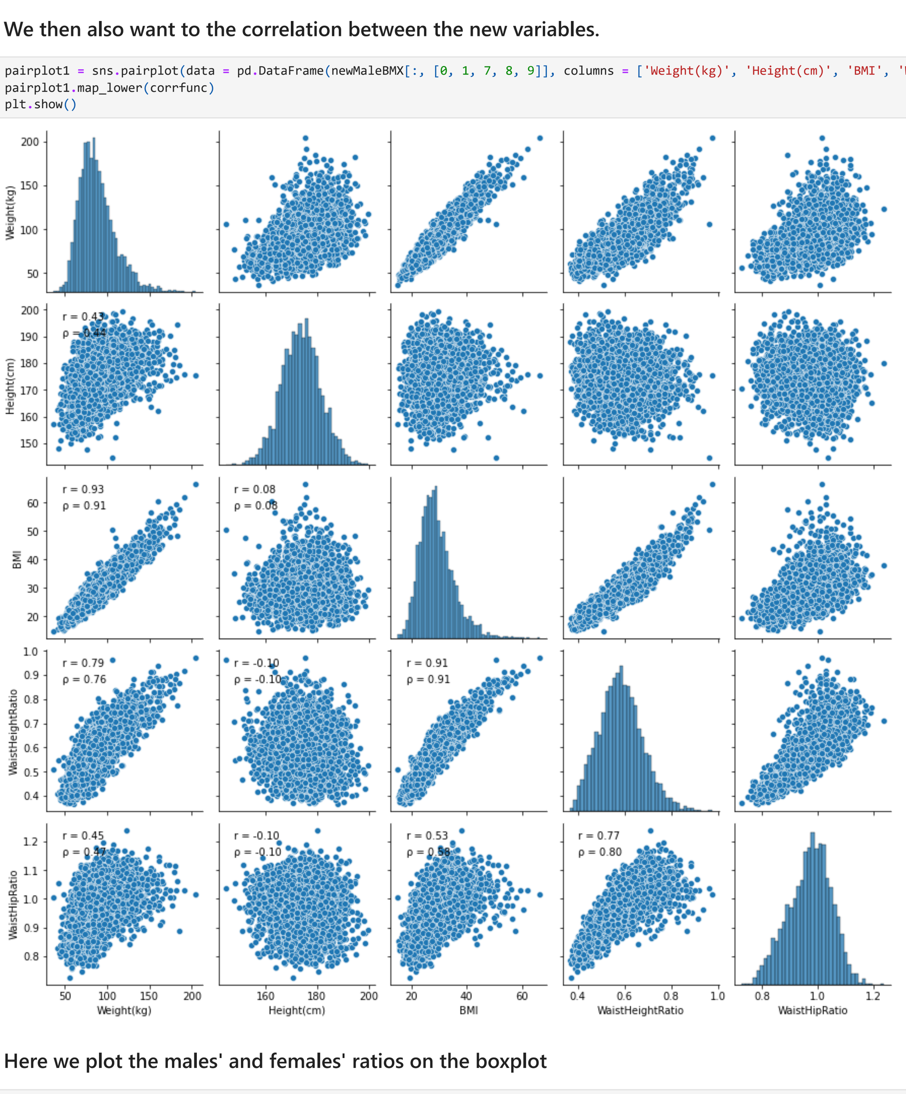
```
[[ 97.1        160.2         34.7         40.8         35.8
  126.1        117.9          0.73595506   0.93497224]
 [ 91.1        152.7         33.5         33.          38.5
  125.5        103.1          0.67518009   0.82151394]
 [ 73.         161.2         37.4         38.          31.8
  106.2         92.           0.5707196    0.86629002]
 [ 61.7        157.4         38.          34.7         29.
  101.          90.5          0.57496823   0.8960396 ]
 [ 55.4        154.6         34.6         34.          28.3
   92.5         73.2          0.47347995   0.79135135]]
```

In [22]:
```
newFemaleBMX.shape
```

Out[22]:
```
(4221, 9)
```

We then also want to the correlation between the new variables.

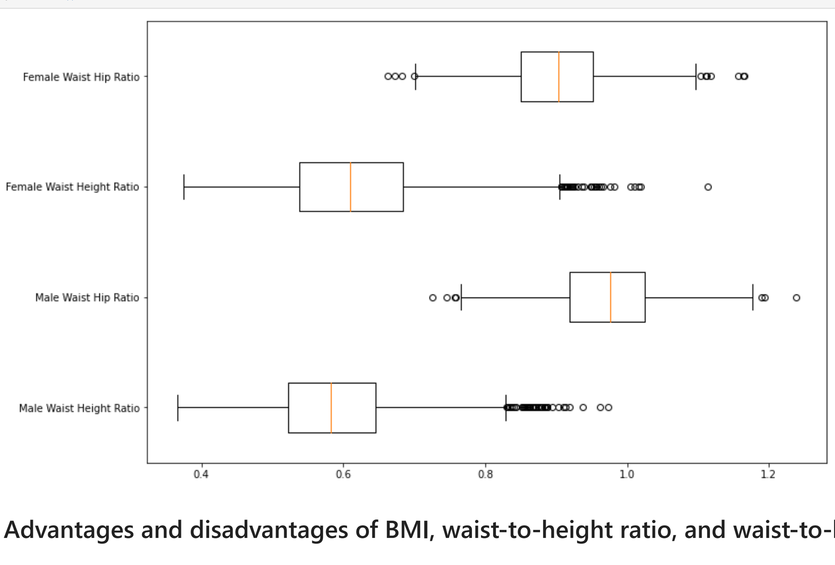
In [23]:
```
pairplot1 = sns.pairplot(data = pd.DataFrame(newMaleBMX[:, [0, 1, 7, 8, 9]], columns = ['Weight(kg)', 'Height(cm)', 'BMI', 'WaistHeightRatio', 'WaistHipRatio']))
pairplot1.map_lower(corrfunc)
plt.show()
```



Here we plot the males' and females' ratios on the boxplot

In [24]:
```
mergeData = [newMaleBMX[:, 8], newMaleBMX[:, 9], newFemaleBMX[:, 7], newFemaleBMX[:, 8]]
```

In [25]:
```
plt.figure(figsize=(12, 8))
plt.boxplot(mergeData, vert = 0, labels = ['Male Waist Height Ratio', 'Male Waist Hip Ratio', 'Female Waist Height Ratio', 'Female Waist Hip Ratio'])
plt.show()
```



Advantages and disadvantages of BMI, waist-to-height ratio, and waist-to-hip ratio.

BMI (Advantages and Disadvantages):

Advantages: Fast and easy to perform and Lower values predicts more disease free intervals

Disadvantages: Does not accurately indicate the distribution of body fat and Affected by changes in lean body mass as well as by changes in body fat

---

Waist to Height Ratio (Advantages and Disadvantages):

Advantages: Correlated with the risk of myocardial infarction

Disadvantages: BMI and WC are better than WHR regarding the cardiovascular and the metabolic risk assessment

---

Waist to Hip Ratio (Advantages and Disadvantages):

Advantages: A fixed value of 0.5 can be applied to all ethnic groups

Disadvantages: New measure and more research is needed for its validation

Optional Features

A way to print out the top 5 lowest and highest persons BMI, is to sort by BMI and then sort the array in ascending or decending order.

In [30]:
```
sortedmatrix = maleZScores[maleZScores[:, 7].argsort()]#sort by bmi ascending order
sortedmatrix[0:5,:]
```

Out[30]:
```
array([[ 0.44103412,  2.06392426, -0.81944375, -0.71390611, -0.83338269,
         0.59635366,  0.35540924, -1.08998873],
       [ 0.37670099,  2.07852336, -0.81599455, -0.62126875, -0.88698833,
         0.66676129,  0.28542327, -1.08315727],
       [ 0.29751831,  2.16157961, -0.77750024, -0.6585176 , -0.80463663,
         0.49373529,  0.37057783, -1.08275658],
       [ 0.6873977 ,  1.97236307, -0.80512448, -0.76361312, -0.90890289,
         0.50437033,  0.39493128, -1.08142188],
       [ 0.29688108,  2.17560304, -0.679432  , -0.65414899, -0.87391667,
         0.52053845,  0.29493623, -1.08046114]])
```

First 5 rows

In [31]:
```
sortedmatrix[0:5,:]
```

Out[31]:
```
array([[ 0.44103412,  2.06392426, -0.81944375, -0.71390611, -0.83338269,
         0.59635366,  0.35540924, -1.08998873],
       [ 0.37670099,  2.07852336, -0.81599455, -0.62126875, -0.88698833,
         0.66676129,  0.28542327, -1.08315727],
       [ 0.29751831,  2.16157961, -0.77750024, -0.6585176 , -0.80463663,
         0.49373529,  0.37057783, -1.08275658],
       [ 0.6873977 ,  1.97236307, -0.80512448, -0.76361312, -0.90890289,
         0.50437033,  0.39493128, -1.08142188],
       [ 0.29688108,  2.17560304, -0.679432  , -0.65414899, -0.87391667,
         0.52053845,  0.29493623, -1.08046114]])
```

Last 5 rows

In [32]:
```
sortedmatrix.shape #4081 - 5 = 4076
```

Out[32]:
```
(4081, 8)
```

In [33]:
```
sortedmatrix[4076:,:] #or you can sort the array in descending order and print the first 5
```

Out[33]:
```
array([[ 1.15858609,  1.14185854, -1.08625078, -1.09796007, -0.9858855 ,
         0.82236238,  0.83574442, -0.78845509],
       [ 0.57555403,  1.48130768, -1.16047381, -0.97069685, -0.96854029,
         0.86884569,  0.94648172, -0.77247816],
       [ 1.01856983,  1.06622111, -1.15574061, -1.10455961, -0.97925438,
         0.99562662,  0.88797001, -0.72883298],
       [ 1.3538228 ,  0.91421385, -1.12123578, -1.1528515 , -0.92250845,
         0.80882814,  0.84496039, -0.72522945],
       [ 0.44264125,  1.25180326, -1.07975119, -1.19028105, -0.88788803,
         1.03491447,  1.13710246, -0.70854118]])
```

In [ ]: