SIT220: Task 3P - Working with numpy Matrices (Multidimensional Data)

George David Nicholson (undergraduate (SIT220))

Student ID: 218403172

Email: gnicholson@deakin.edu.au

In [153…

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import cm
import scipy.stats as stats
import seaborn as sns
```

In [154…

```
# import the csv as a frame via pd.read_csv

female = pd.read_csv("nhanes_adult_female_bmx_2020.csv", comment="#")
male = pd.read_csv("nhanes_adult_male_bmx_2020.csv", comment = "#")
```

Female data

In [155…

```
#convert to numpy matric
female = female.to_numpy()
female[:6 , :]
# 6 rows all columns
```

Out[155…

```
array([[ 97.1, 160.2,  34.7,  40.8,  35.8, 126.1, 117.9],
       [ 91.1, 152.7,  33.5,  33. ,  38.5, 125.5, 103.1],
       [ 73. , 161.2,  37.4,  38. ,  31.8, 106.2,  92. ],
       [ 61.7, 157.4,  38. ,  34.7,  29. , 101. ,  90.5],
       [ 55.4, 154.6,  34.6,  34. ,  28.3,  92.5,  73.2],
       [ 62. , 144.7,  32.5,  34.2,  29.8, 106.7,  84.8]])
```

In [ ]:

```
```

1. weight (kg),
2. standing height (cm),
3. upper arm length (cm),
4. upper leg length (cm),
5. arm circumference (cm),
6. hip circumference (cm),
7. waist circumference (cm)

In [156…

```
type(female)
```

Out[156…

```
numpy.ndarray
```

In [157…

```
female.ndim
```

Out[157…

```
2
```

In [158…

```
# rows , columns
female.shape
```

Out[158…

```
(4221, 7)
```

In [159…

```
femheights = female[ 0: , 1]
print(femheights)
```

```
[160.2 152.7 161.2 ... 159.6 168.5 147.8]
```

In [160…

```
femheights.shape
```

Out[160…

```
(4221,)
```

In [ ]:

```

```

Male data

In [161…

```
male = male.to_numpy()
male[:6 , :]
```

Out[161…

```
array([[ 98.8, 182.3,  42. ,  40.1,  38.2, 108.2, 120.4],
       [ 74.3, 184.2,  41.1,  41. ,  30.2,  94.5,  86.8],
       [103.7, 185.3,  47. ,  44. ,  32. , 107.8, 109.6],
       [ 86. , 167.8,  39.5,  38.4,  29. , 106.4, 108.3],
       [ 99.4, 181.6,  40.4,  39.9,  36. , 120.2, 107. ],
       [ 90.2, 162.5,  38.7,  38. ,  37.3, 110.2, 116.2]])
```

In [162…

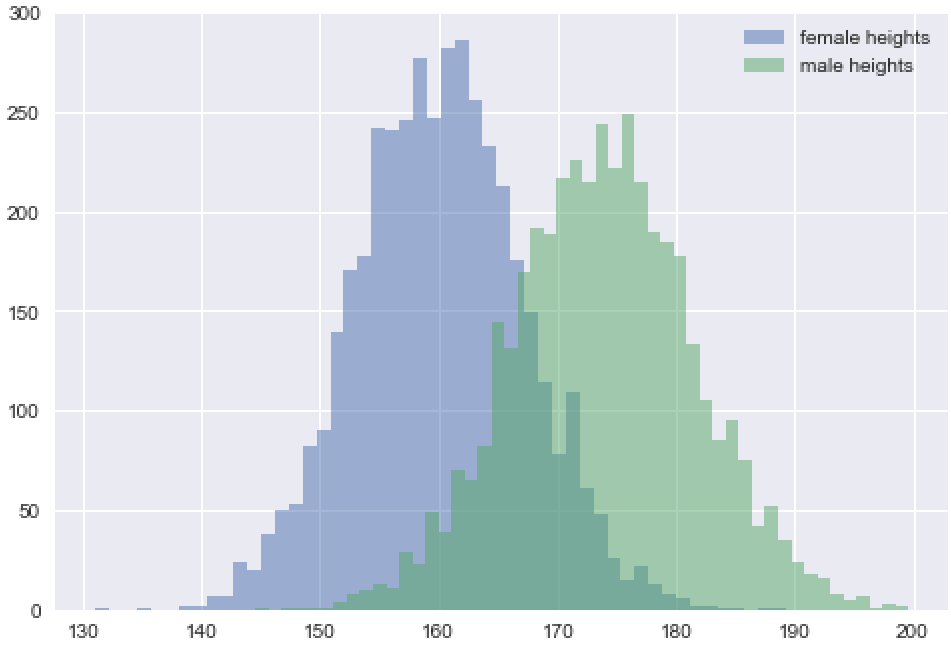
```
maleheights = male[ 0: , 1]
print(maleheights)
```

```
[182.3 184.2 185.3 ... 168.7 176.4 167.5]
```

Trying out different Histogram plotting methods

In [163…

```
# femheights, maleheights = plt.xlim(5)
plt.hist(femheights, bins = 50, label = 'female heights', alpha = 0.5)
plt.hist(maleheights, bins = 50, label = 'male heights', alpha = 0.5)
plt.legend(loc="best")
plt.show()
```

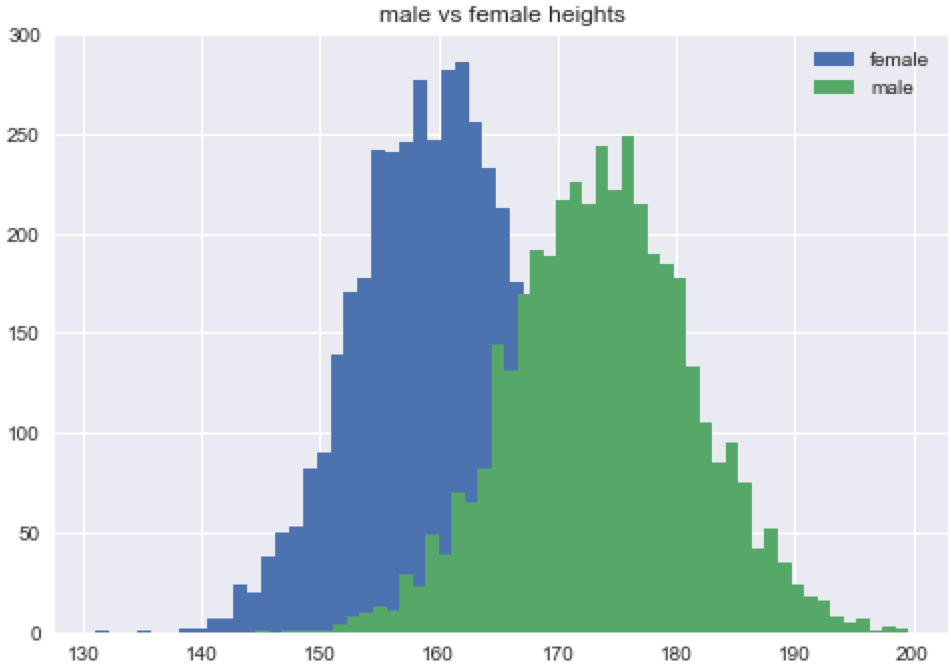


In [164…

```
#this is the one

plt.subplot(1,1,1)
plt.hist(femheights[:], label = 'female' ,bins = 50)
plt.xlim()

plt.subplot(1,1,1)
plt.hist(maleheights[:], label='male', bins = 50)
plt.xlim()

plt.legend()
plt.title('male vs female heights')
```

Out[164…

```
Text(0.5, 1.0, 'male vs female heights')
```



In [ ]:

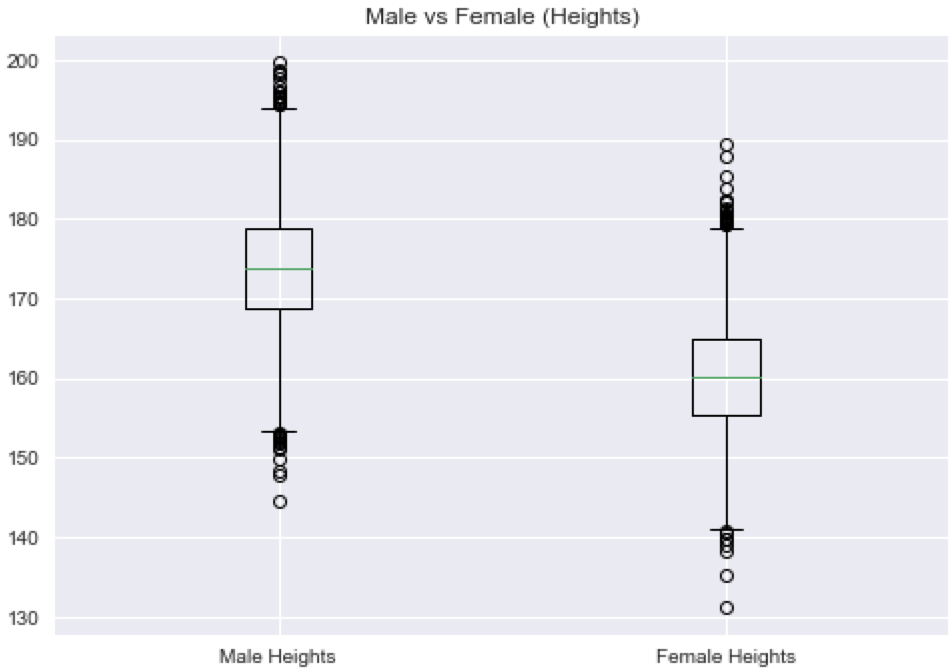
Boxplot of Male vs Female heights

In [165…

```python
plt.boxplot([maleheights, femheights], labels =['Male Heights', 'Female Heights']);
plt.title('Male vs Female (Heights)')
```

Out[165…

```
Text(0.5, 1.0, 'Male vs Female (Heights)')
```



The above box and whisker plot shows that males have a IQR of 170-180 while females have an IQR of 155 - 165

Numerical aggregates

measures of location

In [166…

```python
m = np.mean(maleheights)
mm = np.median(maleheights)
mmale = np.quantile(maleheights, [0, 0.25, 0.5, 0.75, 1])
maleIQR = np.quantile(maleheights, [0.75]) - np.quantile(maleheights, 0.25)


print("the male arithmetic mean: " + str(m))
print("the male arithmetic median: " + str(mm))
print("Male Quantiles: " + str(mmale))
print("Male IQR is: " + str(maleIQR))
```

```
the male arithmetic mean: 173.82702768929187
the male arithmetic median: 173.8
Male Quantiles: [144.6 168.6 173.8 178.9 199.6]
Male IQR is: [10.3]
```

from the male mean and median height calculation we can see that the mean and median are more or less equal when rounded to the nearest decimal place

proving to show a symmetric distribution, this shows to be a more central measurement of central tendency as it looks to be centred when comparing the histogram and calculated mean

In [167…

```
f = np.mean(femheights)
fm = np.median(femheights)
ffemale = np.quantile(femheights, [0, 0.25, 0.5, 0.75, 1])
femIQR = np.quantile(femheights, [0.75]) - np.quantile(femheights, 0.25)


print("the female arithmetic mean: " + str(f))
print("the female arithmetic median: " + str(fm))
print("Female Quantiles: " + str(ffemale))
print("Female IQR is: " + str(femIQR))
```

```
the female arithmetic mean: 160.13679222932953
the female arithmetic median: 160.1
Female Quantiles: [131.1 155.3 160.1 164.8 189.3]
Female IQR is: [9.5]
```

from the measure of location with regards to female heights, it also seems to be symettrically distributed judging from the similarity in the calculated median and mean when rounded to the nearest decimal

From the above results we can see that males have a larger IQR and Arithmetic mean/median then females do thus showing that males on average are taller then females

measures of dispersion

Standard Deviation and IQR

In [168…

```
ss = np.std(maleheights)
print("the average degree of spread from the arithmetic mean calculated is (Standard De
print("Male IQR is: " + str(maleIQR))
print('------------------------------------------------------------------------------')
sts = np.std(femheights)
print("the average degree of spread from the arithmetic mean calculated is (Standard De
print("Female IQR is: " + str(femIQR))
```

```
the average degree of spread from the arithmetic mean calculated is (Standard Deviatio
n): 7.661471130202061
Male IQR is: [10.3]
---------------------------------------------------------------------------
the average degree of spread from the arithmetic mean calculated is (Standard Deviatio
n): 7.062021850008261
Female IQR is: [9.5]
```

The IQR and Standard Deviation show that males are infact taller on average then females, the +0.5 degrees in Standard deviation shows that males have a higher degree of spread from the calculated mean as opposed to the females, the fact that males also have a IQR of +0.8 more then females supports this hypothesis as IQR is considered to involved the median 50 % (0.75 - .025), while also Standard deviation being sensitive to outliers, it also proves the hypothesis on both gender fronts.

measures of shape

In [169…

```
male_skew = stats.skew(maleheights)
female_skew = stats.skew(femheights)
```

In [170…

```
print("Measurement of Male dataframe Shape Skewness: " + str(male_skew))
```

```
Measurement of Male dataframe Shape Skewness: 0.018749535133802897
```

In [171…

```
print("Measurement of Female dataframe Shape Skewness: " + str(female_skew))
```

```
Measurement of Female dataframe Shape Skewness: 0.0811184528074054
```

From the results of skewness it is clear that both dataframes have almost symnetric distribution as can be seen by the low value within the 0 range. however, this being noted, the female value is a total of 332% larger then the males in the positive direction showing that the female heights dataframe is more rightly skewed then the male althought both being near symetric from all calculations till now

In [172…

```
#for i,p in zip(heights, weights):
#    x = p / (i/100) **2
#    bmis.append(x)
```

In [173…

```
maleWeight = np.array([male[0:4081, 0]])
print(maleWeight)
print(maleheights)
print(maleWeight.size)
print(maleheights.size)

malebmi = []
```

```
[[ 98.8  74.3 103.7 ... 108.8  79.5  59.7]]
[182.3 184.2 185.3 ... 168.7 176.4 167.5]
4081
4081
```

In [174…

```
for i,p in zip(maleheights, maleWeight):
    x = p / (i/100) **2
    malebmi.append(x)
```

In [175…

```
print(malebmi)
```

```
[array([29.72922633, 22.35710037, 31.20365152, ..., 32.73825733,
       23.92179649, 17.9639151 ])]
```

In [176…

```
malebmi = np.array(malebmi)
```

In [177…

```
malebmi.shape
```

Out[177…

```
(1, 4081)
```

```
In [ ]:
```

```
In [178…  malebmi.shape = (4081, 1)
```

```
In [179…  malebmi.shape
```

```
Out[179…  (4081, 1)
```

```
In [180…  male = np.append(male, malebmi, axis=1)
```

```
In [181…  male.shape
```

```
Out[181…  (4081, 8)
```

```
In [ ]:
```

```
In [182…  male[:6 , :]
```

```
Out[182…  array([[ 98.8       , 182.3       ,  42.       ,  40.1       ,
                   38.2       , 108.2       , 120.4       ,  29.72922633],
                 [ 74.3       , 184.2       ,  41.1       ,  41.        ,
                   30.2       ,  94.5       ,  86.8       ,  22.35710037],
                 [103.7       , 185.3       ,  47.        ,  44.        ,
                   32.        , 107.8       , 109.6       ,  31.20365152],
                 [ 86.        , 167.8       ,  39.5       ,  38.4       ,
                   29.        , 106.4       , 108.3       ,  25.87766664],
                 [ 99.4       , 181.6       ,  40.4       ,  39.9       ,
                   36.        , 120.2       , 107.        ,  29.90976819],
                 [ 90.2       , 162.5       ,  38.7       ,  38.        ,
                   37.3       , 110.2       , 116.2       ,  27.14145966]])
```

```
In [ ]:
```

```
In [ ]:
```

Create a new matrix zmale being the version of the male dataset with all its columns standardised (by computing the z-scores of each column).

```
In [183…  zmale = np.array(male)
```

```
In [184…  zmale[:6, :]
```

```
Out[184…  array([[ 98.8       , 182.3       ,  42.       ,  40.1       ,
                   38.2       , 108.2       , 120.4       ,  29.72922633],
       [ 74.3       , 184.2       ,  41.1       ,  41.        ,
         30.2       ,  94.5       ,  86.8       ,  22.35710037],
       [103.7       , 185.3       ,  47.        ,  44.        ,
         32.        , 107.8       , 109.6       ,  31.20365152],
       [ 86.        , 167.8       ,  39.5       ,  38.4       ,
         29.        , 106.4       , 108.3       ,  25.87766664],
       [ 99.4       , 181.6       ,  40.4       ,  39.9       ,
         36.        , 120.2       , 107.        ,  29.90976819],
       [ 90.2       , 162.5       ,  38.7       ,  38.        ,
         37.3       , 110.2       , 116.2       ,  27.14145966]])
```

In [185…

```
zmale = stats.zscore(zmale, axis = 1, ddof = 1)
```

In [186…

```
zmale
```

Out[186…

```
array([[ 0.30177076,  1.84444832, -0.74761948, ...,  0.47543746,
         0.70083466, -0.97432423],
       [ 0.04729613,  2.13238699, -0.58259483, ...,  0.43054304,
         0.28445387, -0.93819673],
       [ 0.39212036,  1.90680498, -0.66036271, ...,  0.46822584,
         0.501638  , -0.95357945],
       ...,
       [ 0.50875535,  1.67155958, -0.85399686, ...,  0.68734965,
         0.62328865, -0.9677875 ],
       [ 0.11353375,  2.01122184, -0.66982562, ...,  0.51108863,
         0.45821188, -0.97490892],
       [-0.14385339,  2.08887563, -0.54566177, ...,  0.49406918,
         0.41950681, -1.00828167]])
```

Draw a scatterplot matrix (pairplot) for the standardised versions of height, weight, waist circumference, hip circumference, and BMI of the males (based on zmale). Compute Pearson’s and Spearman’s correlation coefficients for all these pairs of variables. Interpret the obtained results.

1. weight (kg),
2. standing height (cm),
3. upper arm length (cm),
4. upper leg length (cm),
5. arm circumference (cm),
6. hip circumference (cm),
7. waist circumference (cm)
8. BMI

0,2,6,7,8

In [187…

```
plt.style.use("seaborn")
df = pd.DataFrame(zmale[:, [1, 0 , 6 , 5 ,7]],
                 columns =[ "height", "weight", "waist circumference", "hip circumferen
                 )
df
```

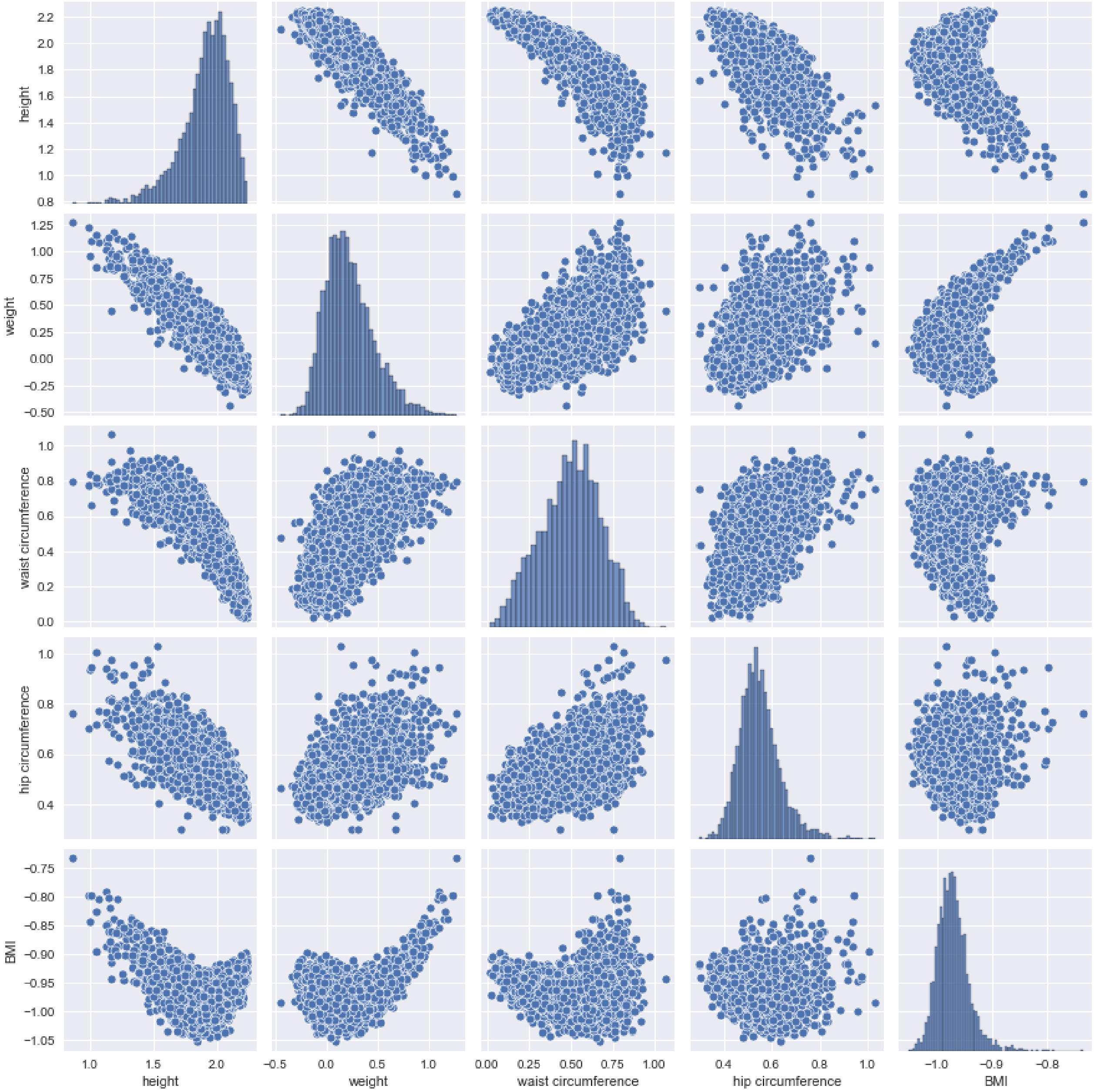
Out[187…

	height	weight	waist circumference	hip circumference	BMI
0	1.844448	0.301771	0.700835	0.475437	-0.974324
1	2.132387	0.047296	0.284454	0.430543	-0.938197
2	1.906805	0.392120	0.501638	0.468226	-0.953579
3	1.829113	0.214033	0.654330	0.616816	-0.973037
4	1.837925	0.324105	0.464068	0.707164	-0.955649
...	...	...	...	...	...
4076	1.675787	0.557058	0.653692	0.614667	-0.927894
4077	1.954590	0.315714	0.412689	0.529059	-0.962898
4078	1.671560	0.508755	0.623289	0.687350	-0.967787
4079	2.011222	0.113534	0.458212	0.511089	-0.974909
4080	2.088876	-0.143853	0.419507	0.494069	-1.008282

4081 rows × 5 columns

In [188…

```
sns.pairplot(df)
plt.show()
```



Compute Pearson’s and Spearman’s correlation coefficients for all these pairs of variables. Interpret the obtained results.

In [189…

```
spear = np.array(df)
spear
```

Out[189…

```
array([[ 1.84444832,  0.30177076,  0.70083466,  0.47543746, -0.97432423],
       [ 2.13238699,  0.04729613,  0.28445387,  0.43054304, -0.93819673],
       [ 1.90680498,  0.39212036,  0.501638  ,  0.46822584, -0.95357945],
       ...,
       [ 1.67155958,  0.50875535,  0.62328865,  0.68734965, -0.9677875 ],
       [ 2.01122184,  0.11353375,  0.45821188,  0.51108863, -0.97490892],
       [ 2.08887563, -0.14385339,  0.41950681,  0.49406918, -1.00828167]])
```

To give the correlation between all variables we tranpose the numpy matrix into the pearson cooefficient

In [190…

```
C = np.corrcoef(spear.T)
```

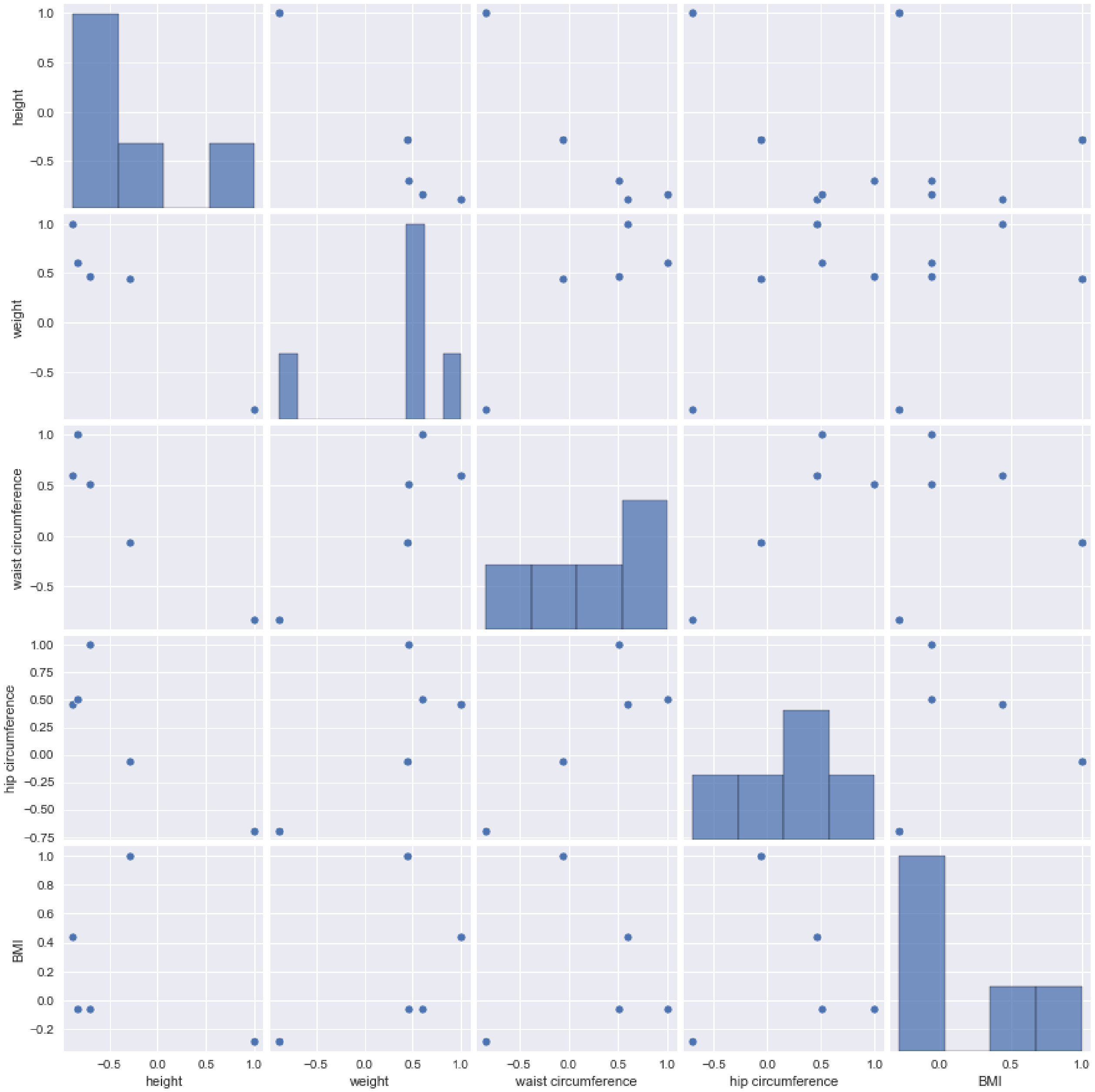
```
C
```

Out[190…

```
array([[ 1.        , -0.88215127, -0.83226326, -0.69651814, -0.28698764],
       [-0.88215127,  1.        ,  0.59952306,  0.46189775,  0.43720816],
       [-0.83226326,  0.59952306,  1.        ,  0.50863846, -0.0633839 ],
       [-0.69651814,  0.46189775,  0.50863846,  1.        , -0.06125073],
       [-0.28698764,  0.43720816, -0.0633839 , -0.06125073,  1.        ]])
```

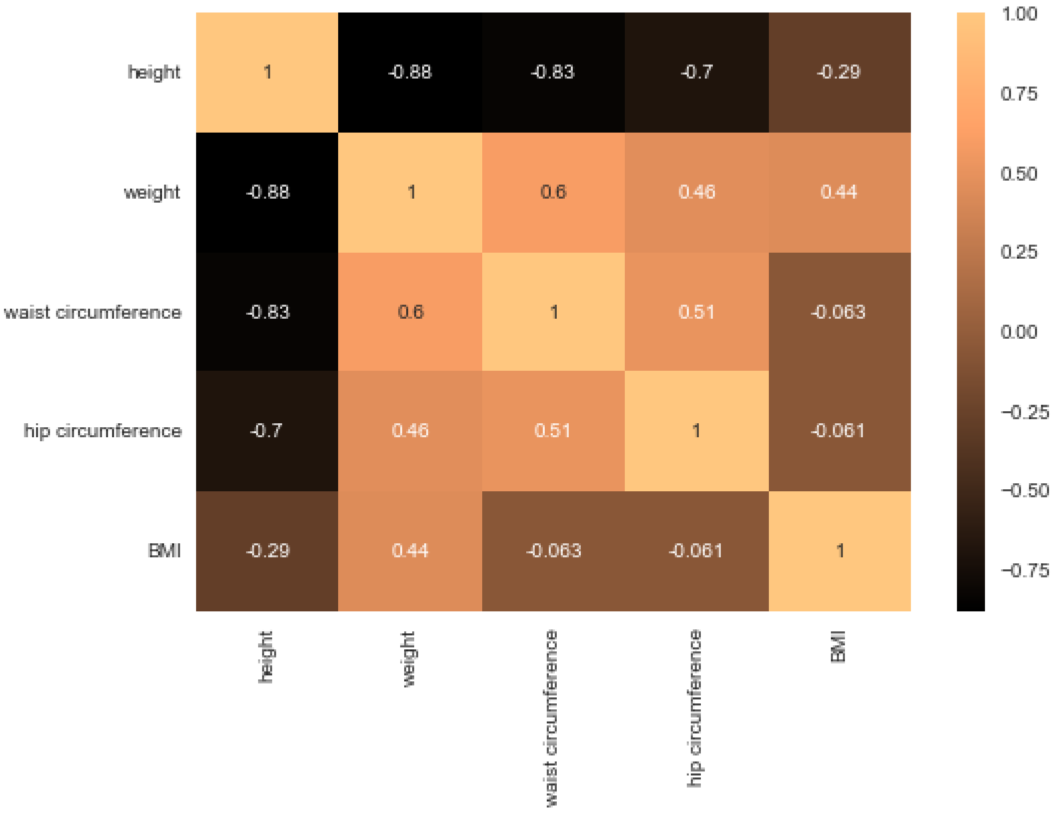
In [191…

```
plt.style.use("seaborn")
df = pd.DataFrame(C [: , [0, 1 , 2, 3 ,4]],
                  columns =[ "height", "weight", "waist circumference", "hip circumferen
                  )
df
sns.pairplot(df)
plt.show()
```



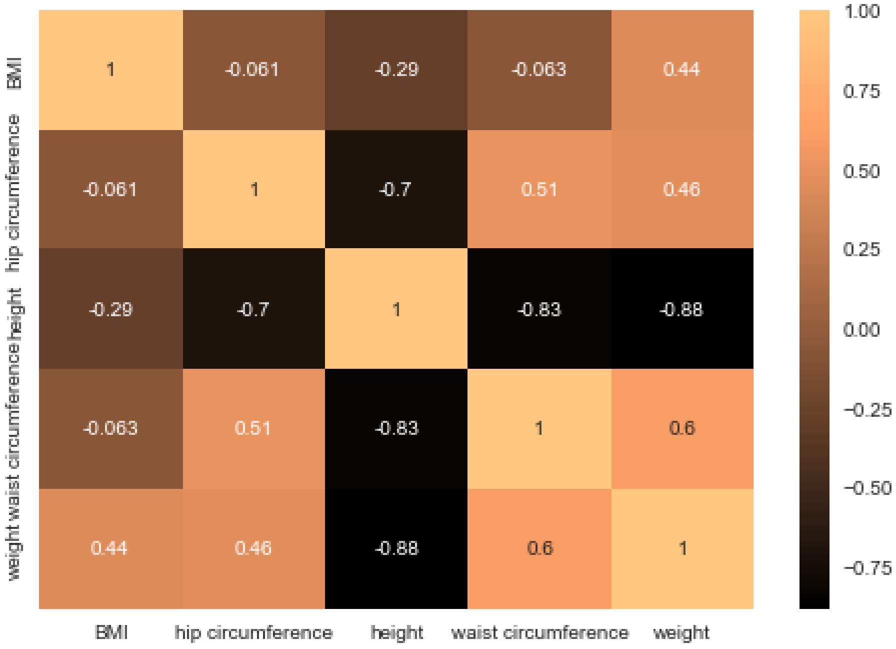
In [192…

```
clr = cm.get_cmap("copper")
cols = np.array([ "height", "weight", "waist circumference", "hip circumference", "BMI"
sns.heatmap(C, xticklabels=cols, yticklabels=cols, annot=True, cmap = clr)
plt.show()
```



In [193…

```
clr = cm.get_cmap("copper")
o = [4, 3, 0, 2, 1,]
cols = np.array([ "height", "weight", "waist circumference", "hip circumference", "BMI"
sns.heatmap(C[np.ix_(o,o)], xticklabels=cols[o], yticklabels=cols[o], annot=True, cmap
plt.show()
```



Spearman’s correlation coefficients

In [194…

```
#Spearmans rho
```

In [195…

```
def plot_corr(x,y):
    r = stats.pearsonr(x,y)[0]
    rho = stats.spearmanr(x,y)[0]
    plt.scatter(x,y, label =f"r    = {r:.3}\nrho =  {rho:.3}")
    plt.legend()
```

In [196…

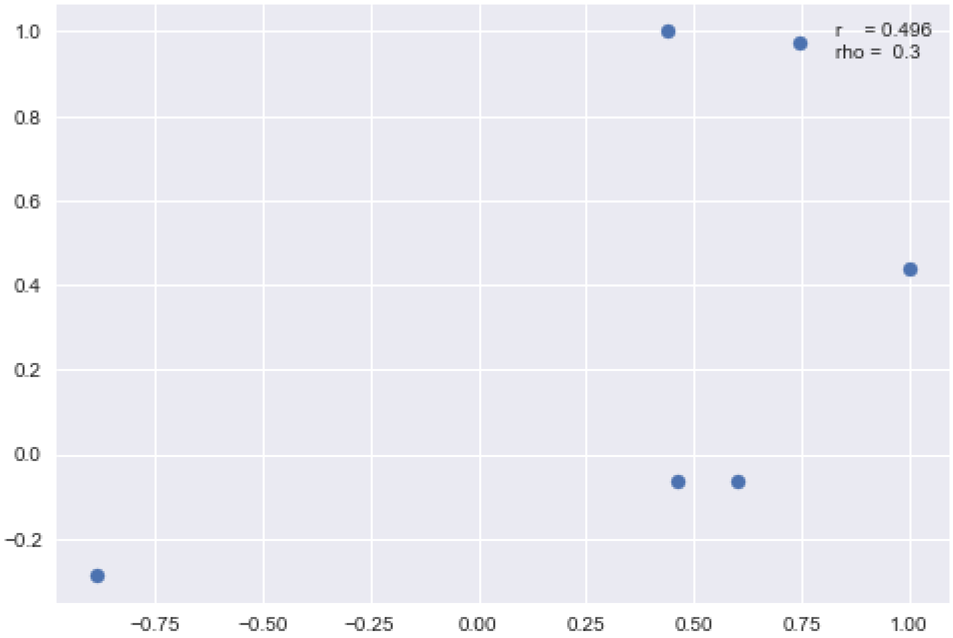
```
x = C[:, 1]
y = C[:, 4]
stats.pearsonr(stats.rankdata(x),stats.rankdata(y))[0]
```

Out[196…

```
0.30000000000000004
```

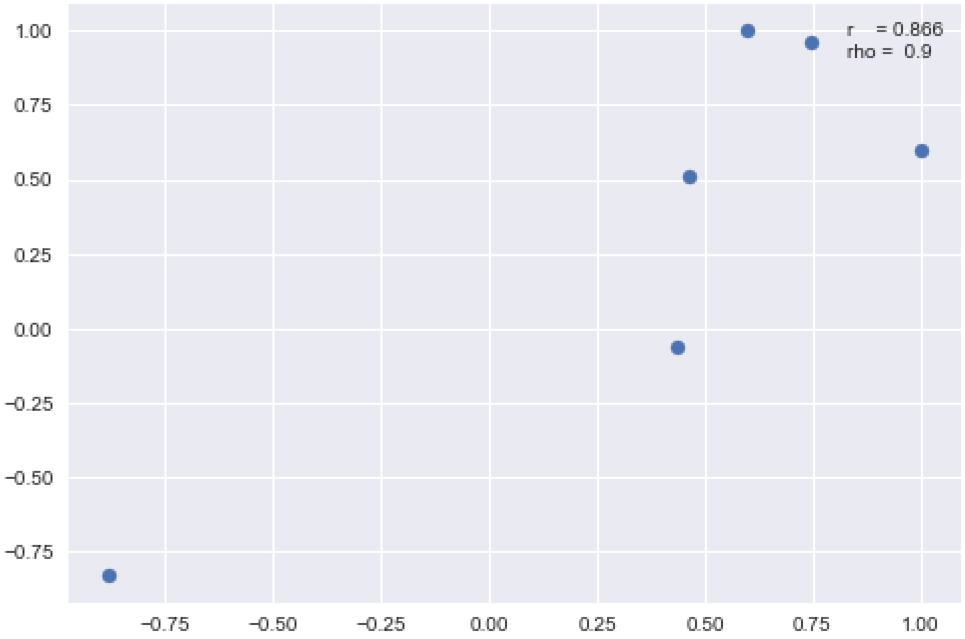
In [197…

```
stats.spearmanr(x,y)[0]
plot_corr(x,y)
```



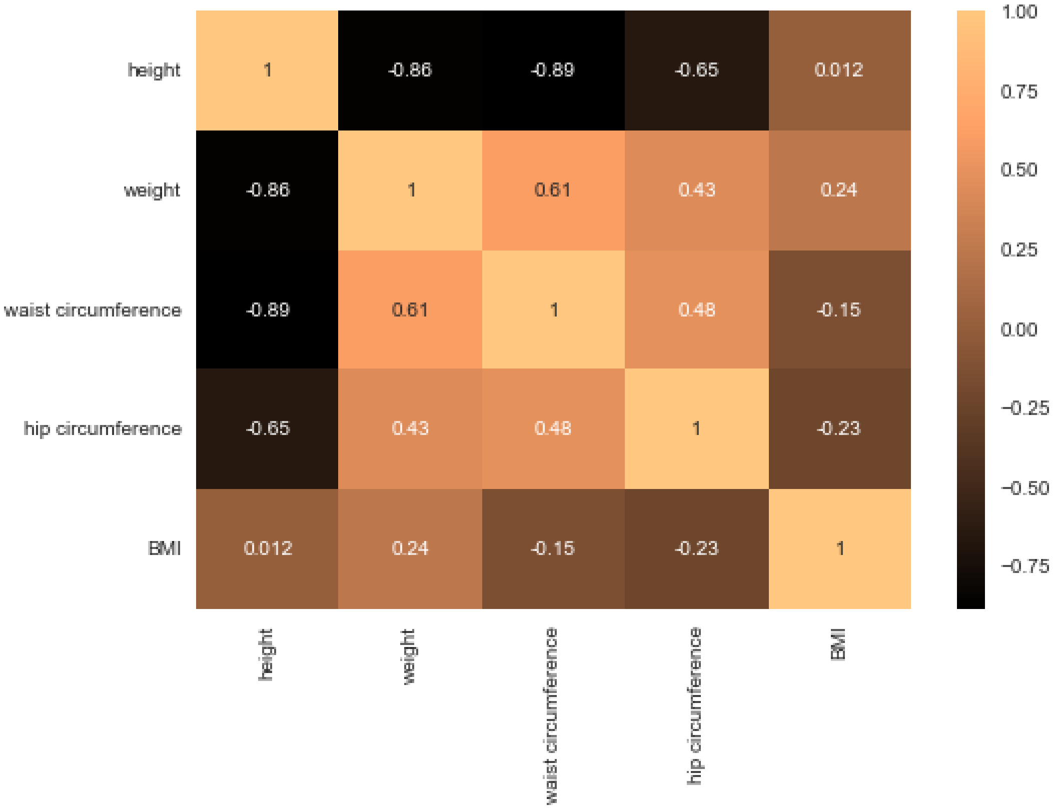
In [198…

```
plot_corr(C[:, 1], C[:, 2])
# Weight vs waist circumference
```



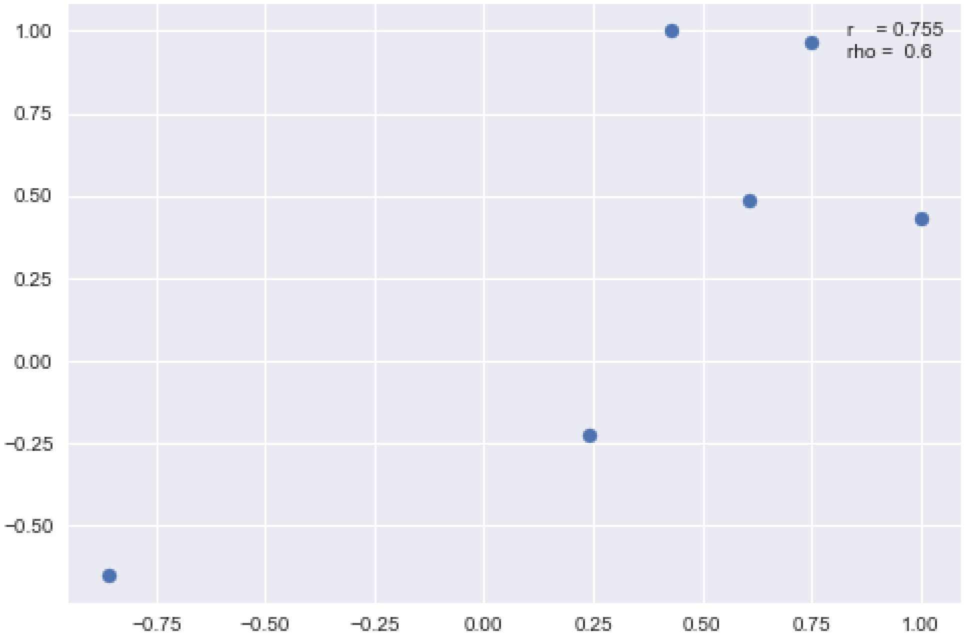
In [199…

```
C = np.corrcoef(stats.rankdata(spear, axis = 0).T)
C
clr = cm.get_cmap("copper")
cols = np.array([ "height", "weight", "waist circumference", "hip circumference", "BMI"
sns.heatmap(C, xticklabels=cols, yticklabels=cols, annot=True, cmap = clr)
plt.show()
```



In [200…

```
plot_corr(C[:, 1], C[:, 3])
```



Conclusion

From the results of pearson correlation and Spearmans rho we can conclude that there is a strong correlation between waist circumferance and weight which can be seen from the heatmaps of pearsons correlation = 0.896 and spearmans rho of 0.91. this proves to us that there is a correlation between the variables that are valid. The Pearson 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. while on the heatmap showing as 0.61, the correlation in rho and pearson still stays and 0.9 rounded