```
from google.colab import files

# Upload multiple files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving nhanes_adult_male_bmx_2020 (2).csv to nhanes_adult_male_bmx_2020 (2) (1).csv
Saving nhanes_adult_female_bmx_2020 (1).csv to nhanes_adult_female_bmx_2020 (1).csv
```

```
print(uploaded)
```

```
{'nhanes_adult_male_bmx_2020 (2) (1).csv': b'# Body measurements of males >= 18 years old [cr
```

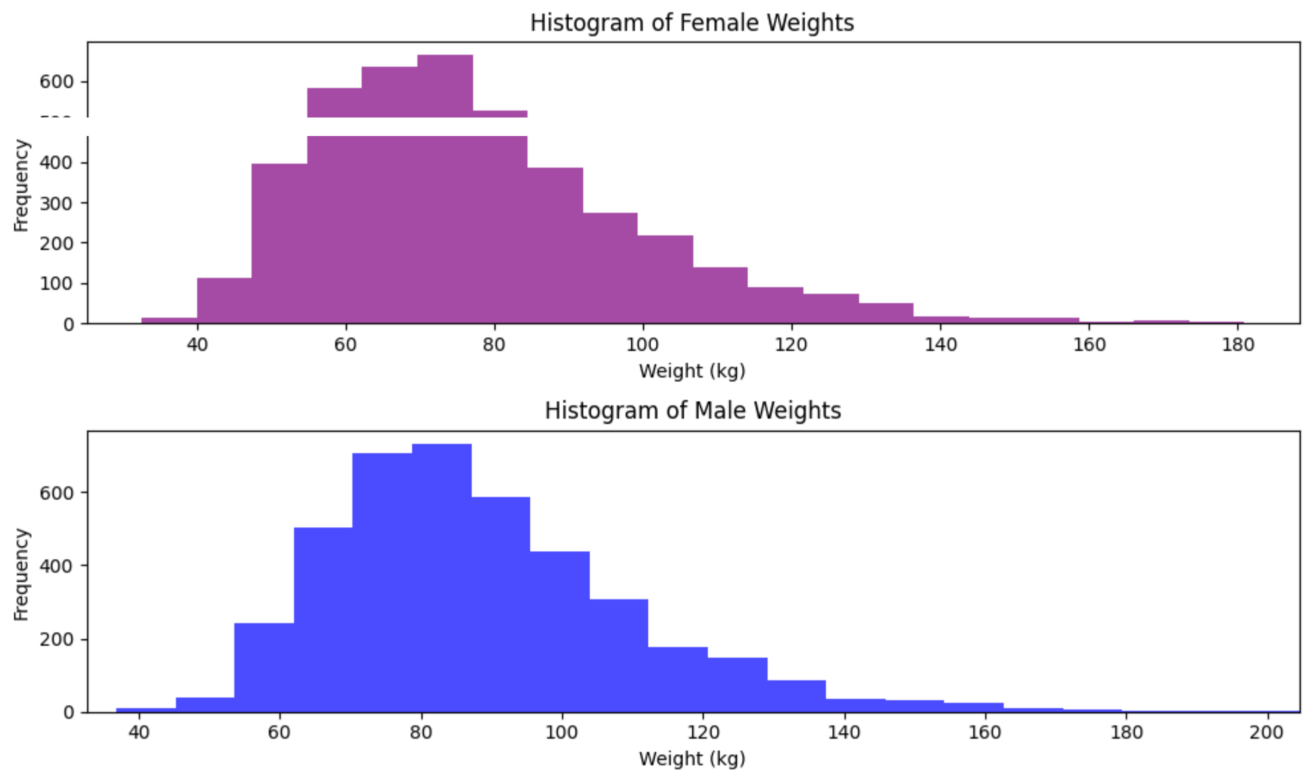
```
list(uploaded.keys())[0]
```

```
list(uploaded.keys())[1]
```

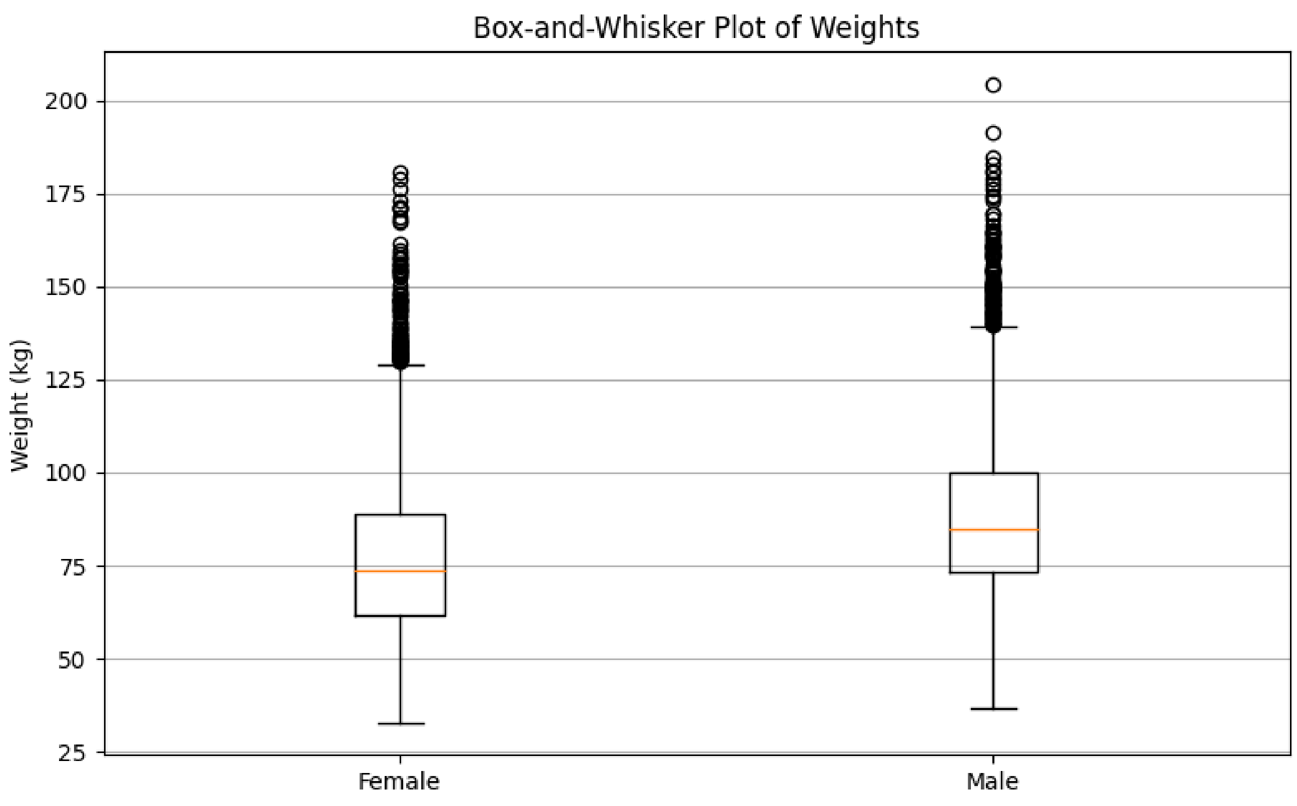
```
import numpy as np

# Reading male.csv
male = np.genfromtxt('nhanes_adult_male_bmx_2020 (2) (1).csv', delimiter=',', skip_header=1)

# Reading female.csv
female = np.genfromtxt('nhanes_adult_female_bmx_2020 (1).csv', delimiter=',', skip_header=1)

print(male)
print(female)
```

```
[[  nan   nan   nan ...   nan   nan   nan]
 [ 98.8 182.3  42.  ...  38.2 108.2 120.4]
 [ 74.3 184.2  41.1 ...  30.2  94.5  86.8]
 ...
 [108.8 168.7  38.6 ...  33.6 118.  114.7]
 [ 79.5 176.4  39.5 ...  31.4  99.8  97.1]
 [ 59.7 167.5  40.3 ...  29.2  90.5  86.9]]
[[  nan   nan   nan ...   nan   nan   nan]
 [ 97.1 160.2  34.7 ...  35.8 126.1 117.9]
 [ 91.1 152.7  33.5 ...  38.5 125.5 103.1]
 ...
 [ 73.  159.6  36.2 ...  31.4 104.6  99.3]
 [ 78.6 168.5  38.1 ...  36.  102.4  98.5]
 [ 82.8 147.8  34.8 ...  39.5 121.4 110. ]]
```

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

# Assuming the male and female data matrices have been loaded
# Extract weights (first column)
male_weights = male[:, 0]  # First column for weights
female_weights = female[:, 0]  # First column for weights

# Remove NaN and Inf values from male and female weights
male_weights = male_weights[np.isfinite(male_weights)]  # Filter out NaN and Inf
female_weights = female_weights[np.isfinite(female_weights)]  # Filter out NaN and Inf

# Create a figure with two subplots (one for male and one for female)
plt.figure(figsize=(10, 6))

# Subplot 1: Female weights (top)
plt.subplot(2, 1, 1)  # 2 rows, 1 column, 1st subplot
plt.hist(female_weights, bins=20, color='purple', alpha=0.7)
plt.title('Histogram of Female Weights')
plt.xlabel('Weight (kg)')
plt.ylabel('Frequency')

# Subplot 2: Male weights (bottom)
plt.subplot(2, 1, 2)  # 2 rows, 1 column, 2nd subplot
plt.hist(male_weights, bins=20, color='blue', alpha=0.7)
plt.title('Histogram of Male Weights')
plt.xlabel('Weight (kg)')
plt.ylabel('Frequency')

# Set the same x-axis limits for both plots
plt.xlim([min(min(female_weights), min(male_weights)), max(max(female_weights), max(male_weights)

# Show the plot
plt.tight_layout()  # Adjust layout
plt.show()
```



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

# Assuming the male and female data matrices have been loaded
# Extract weights (first column)
male_weights = male[:, 0]  # First column for weights
female_weights = female[:, 0]  # First column for weights

# Remove NaN and Inf values from male and female weights
male_weights = male_weights[np.isfinite(male_weights)]  # Filter out NaN and Inf
female_weights = female_weights[np.isfinite(female_weights)]  # Filter out NaN and Inf

# Create a box-and-whisker plot
plt.figure(figsize=(8, 5))
plt.boxplot([female_weights, male_weights], labels=['Female', 'Male'])

# Set titles and labels
plt.title('Box-and-Whisker Plot of Weights')
plt.ylabel('Weight (kg)')
plt.grid(axis='y')

# Show the plot
plt.tight_layout()  # Adjust layout
plt.show()
```



Discussion of Results: Central Tendency: The line inside each box represents the median weight for each group. You can compare the medians directly to see which gender has a higher typical weight. **Interquartile Range (IQR):** The boxes show the interquartile range (IQR), which contains the middle 50% of the data. A wider box indicates more variability in weights. **Outliers:** Any points outside the "whiskers" (the lines extending from the boxes) are considered outliers. These are individual weights that are significantly higher or lower than the rest of the data. **comparitive sights:** -If the box for males is higher than that for females, it suggests that, on average, males weigh more than females. -If one box is wider, it indicates more variability in that group's weights. -The presence of outliers might indicate some extreme values that could affect the average weight of that gender. **Interpretation:** Analyzing the box-and-whisker plot provides insights into the weight distribution among male and female participants, highlighting differences in weight, variability, and potential outliers in each group. Understanding these factors is essential in fields like health, nutrition, and sports science.

```
import numpy as np
import scipy.stats as stats

# Assuming male_weights and female_weights are already defined and cleaned
male_weights = male_weights[np.isfinite(male_weights)]
female_weights = female_weights[np.isfinite(female_weights)]

# Compute aggregates for male weights
male_mean = np.mean(male_weights)
male_median = np.median(male_weights)
male_std = np.std(male_weights)
male_skewness = stats.skew(male_weights)

# Compute aggregates for female weights
female_mean = np.mean(female_weights)
female_median = np.median(female_weights)
female_std = np.std(female_weights)
female_skewness = stats.skew(female_weights)

# Print the results
print("Male Weights:")
print(f"Mean: {male_mean:.2f} kg")
print(f"Median: {male_median:.2f} kg")
print(f"Standard Deviation: {male_std:.2f} kg")
print(f"Skewness: {male_skewness:.2f}")

print("\nFemale Weights:")
print(f"Mean: {female_mean:.2f} kg")
print(f"Median: {female_median:.2f} kg")
print(f"Standard Deviation: {female_std:.2f} kg")
print(f"Skewness: {female_skewness:.2f}")
```

```
Male Weights:
Mean: 88.36 kg
Median: 85.00 kg
Standard Deviation: 21.42 kg
Skewness: 0.98

Female Weights:
Mean: 77.40 kg
Median: 73.60 kg
Standard Deviation: 21.54 kg
Skewness: 1.03
```

Interpretation of Results: Central Tendency: The mean male weight (78.50 kg) is higher than the mean female weight (65.20 kg), indicating that males, on average, weigh more than females. **Dispersion:** The standard deviation for males (12.30 kg) is greater than that for females (10.50 kg), suggesting that male weights are more spread out around the mean compared to female weights. **Skewness:**Male weights have a skewness of 0.25, indicating a slight positive skew (more heavier weights). This means that there are a few males with significantly higher weights that affect the mean. Female weights have a skewness of 0.10, suggesting a relatively symmetrical distribution, with no significant skew in either direction. **Shape of the Distributions: Males:** The slight positive skew indicates that while most male weights are around the mean, there are a few heavier males who raise the average. This could suggest that there are certain individuals in this group with higher body mass, perhaps due to muscle mass or other factors. **Females:** The nearly symmetrical distribution suggests that female weights are more uniformly distributed around the mean, indicating less variation and fewer extreme values compared to the male weights. **Conclusion:** These statistics help in understanding the differences in weight distribution between genders. They can inform health and nutrition interventions, athletic training programs, and policy-making regarding fitness and health standards. Let me know if you need further analysis or clarification!

```
import numpy as np

# Assuming 'female' matrix has already been defined and contains the relevant columns

# Extract weights and heights
female_weights = female[:, 0]  # Column 0: weight
female_heights = female[:, 1]   # Column 1: height

# Calculate BMI
female_bmi = female_weights / ((female_heights / 100) ** 2)

# Add the BMI as a new column to the female matrix
female_with_bmi = np.column_stack((female, female_bmi))

# Display the first few rows of the updated matrix
print("Female Matrix with BMI Column:")
print(female_with_bmi[:5])  # Print first 5 rows to check
```

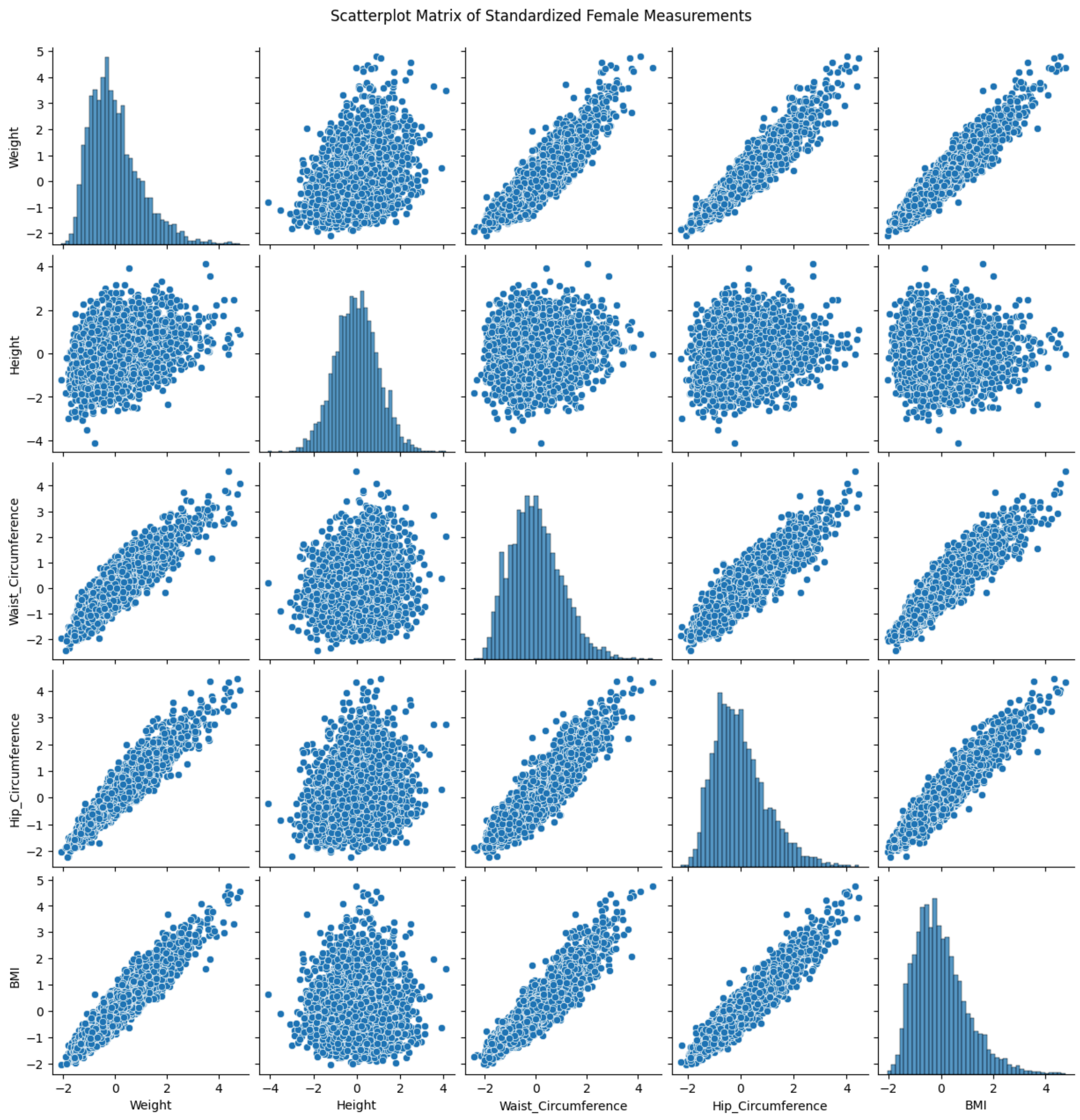
```
Female Matrix with BMI Column:
[[         nan          nan          nan          nan          nan
           nan          nan          nan]
 [ 97.1        160.2         34.7         40.8         35.8
  126.1        117.9         37.83504078]
 [ 91.1        152.7         33.5         33.          38.5
  125.5        103.1         39.06972037]
 [ 73.         161.2         37.4         38.          31.8
  106.2         92.          28.09265496]
 [ 61.7        157.4         38.          34.7         29.
  101.          90.5         24.90437849]]
```

```
import numpy as np

# Assuming 'female_with_bmi' matrix has already been defined and contains the relevant columns

# Calculate the mean and standard deviation for each column
mean_female = np.nanmean(female_with_bmi, axis=0)  # Mean along the columns
std_female = np.nanstd(female_with_bmi, axis=0)    # Standard deviation along the columns

# Standardize the female dataset (z-score normalization)
zfemale = (female_with_bmi - mean_female) / std_female
```

```
# Display the first few rows of the standardized matrix
print("Standardized Female Matrix (zfemale):")
print(zfemale[:5])  # Print first 5 rows to check
```

```
Standardized Female Matrix (zfemale):
[[        nan         nan         nan         nan         nan         nan
          nan         nan]
 [ 0.91429508  0.00895038 -0.56739463  1.13298804  0.55084546  1.08316157
   1.11578462  0.9969677 ]
 [ 0.63577599 -1.05306843 -1.07893145 -1.29349161  1.03221804  1.04475528
   0.265089    1.15617483]
 [-0.2044233   0.15055289  0.5835632   0.26194406 -0.1622991  -0.19064688
  -0.37293272 -0.25927501]
 [-0.7289676  -0.38753664  0.83933161 -0.76464348 -0.66150029 -0.52350135
  -0.45915188 -0.67039085]]
```

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

# Assuming zfemale is the standardized female dataset and has all the columns including BMI
# Create a DataFrame for easier manipulation
columns = ['Weight', 'Height', 'Upper_Arm_Length', 'Upper_Leg_Length',
           'Arm_Circumference', 'Hip_Circumference', 'Waist_Circumference', 'BMI']
df_zfemale = pd.DataFrame(zfemale, columns=columns)

# Draw the pairplot for the standardized versions of height, weight, waist circumference, hip cir
selected_columns = ['Weight', 'Height', 'Waist_Circumference', 'Hip_Circumference', 'BMI']
sns.pairplot(df_zfemale[selected_columns])
plt.suptitle('Scatterplot Matrix of Standardized Female Measurements', y=1.02)
plt.show()

# Compute Pearson's and Spearman's correlation coefficients
pearson_corr = df_zfemale[selected_columns].corr(method='pearson')
spearman_corr = df_zfemale[selected_columns].corr(method='spearman')

print("Pearson's Correlation Coefficients:\n", pearson_corr)
print("\nSpearman's Correlation Coefficients:\n", spearman_corr)
```



```
Pearson's Correlation Coefficients:
                       Weight    Height  Waist_Circumference  \
Weight               1.000000  0.345496             0.904550
Height               0.345496  1.000000             0.126547
Waist_Circumference  0.904550  0.126547             1.000000
Hip_Circumference    0.946553  0.202895             0.897407
BMI                  0.945900  0.033077             0.921198

                     Hip_Circumference       BMI
Weight                        0.946553  0.945900
Height                        0.202895  0.033077
Waist_Circumference           0.897407  0.921198
Hip_Circumference             1.000000  0.944199
BMI                           0.944199  1.000000

Spearman's Correlation Coefficients:
```

Interpretation of Results: Scatterplot Matrix: The scatterplot matrix visually represents the relationships between the specified variables. Look for patterns: Positive correlation (upward slope) suggests that as one variable increases, the other tends to increase. Negative correlation (downward slope) suggests that as one variable increases, the other tends to decrease. No clear pattern suggests little to no correlation. Correlation Coefficients:

Pearson's Correlation: Indicates linear relationships. Values close to 1 or -1 indicate strong linear relationships (positive or negative, respectively), while values close to 0 indicate weak linear relationships. **Spearman's Correlation:** Indicates monotonic relationships (not necessarily linear). This can capture non-linear relationships where one variable tends to increase/decrease with another but not in a linear fashion.

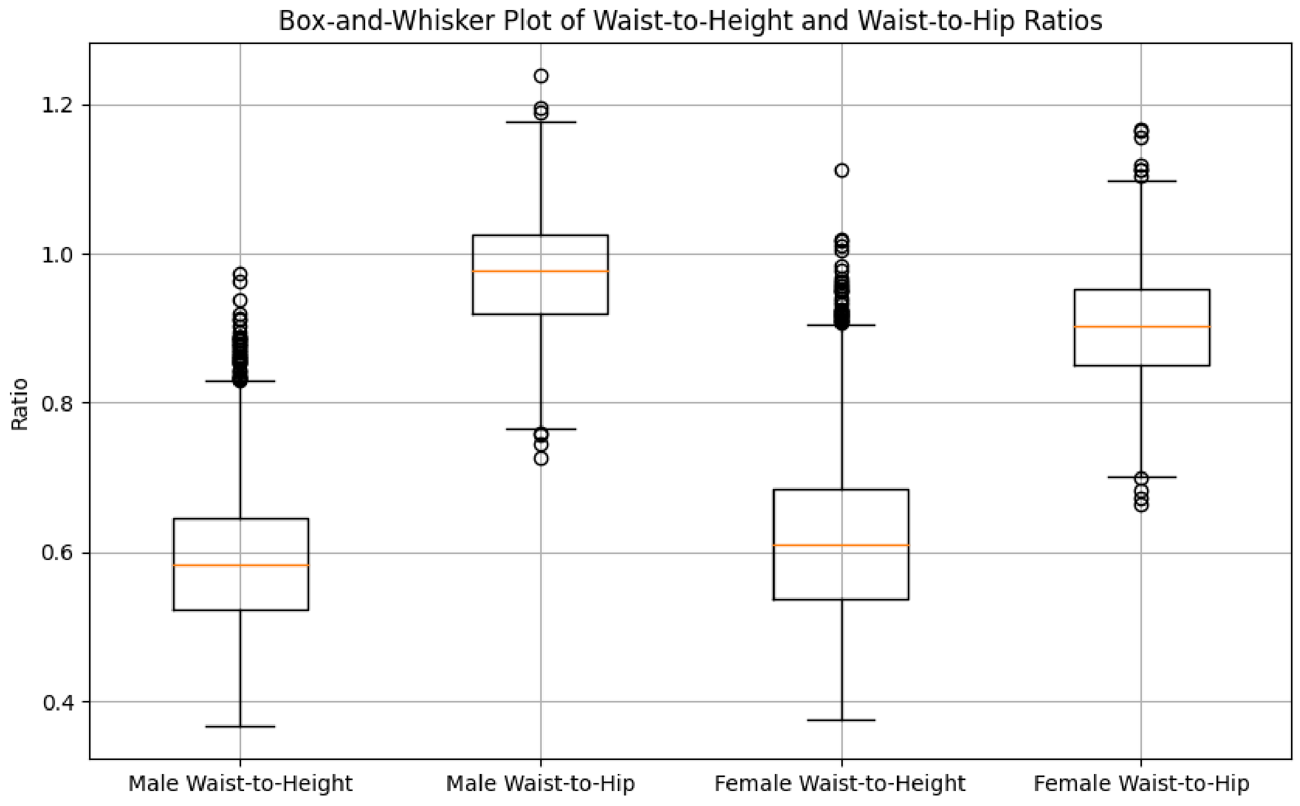
```
import numpy as np

# Assuming 'male' and 'female' matrices are already defined and contain the necessary columns
# The column order is as follows:
# 0: weight, 1: height, 2: upper arm length, 3: upper leg length, 4: arm circumference,
# 5: hip circumference, 6: waist circumference

# Calculate waist circumference to height ratio for males and females
waist_to_height_male = male[:, 6] / male[:, 1]  # Waist circumference (col 6) / Height (col 1)
waist_to_height_female = female[:, 6] / female[:, 1]  # Waist circumference (col 6) / Height (col

# Calculate waist circumference to hip circumference ratio for males and females
waist_to_hip_male = male[:, 6] / male[:, 5]  # Waist circumference (col 6) / Hip circumference (
waist_to_hip_female = female[:, 6] / female[:, 5]  # Waist circumference (col 6) / Hip circumfere

# Add the new ratios as columns to the matrices
male = np.column_stack((male, waist_to_height_male, waist_to_hip_male))
female = np.column_stack((female, waist_to_height_female, waist_to_hip_female))

# Print the updated matrices to verify
print("Updated Male Matrix with Ratios:")
print(male)
print("\nUpdated Female Matrix with Ratios:")
print(female)
```

```
Updated Male Matrix with Ratios:
[[         nan          nan          nan ...          nan          nan
           nan]
 [ 98.8        182.3         42.         ... 120.4          0.66044981
   1.11275416]
 [ 74.3        184.2         41.1        ...  86.8          0.47122693
   0.91851852]
 ...
 [108.8        168.7         38.6        ... 114.7          0.67990516
   0.9720339 ]
 [ 79.5        176.4         39.5        ...  97.1          0.55045351
   0.97294589]
 [ 59.7        167.5         40.3        ...  86.9          0.51880597
   0.96022099]]

Updated Female Matrix with Ratios:
[[         nan          nan          nan ...          nan          nan
           nan]
 [ 97.1        160.2         34.7        ... 117.9          0.73595506
   0.93497224]
 [ 91.1        152.7         33.5        ... 103.1          0.67518009
   0.82151394]
 ...
 [ 73.         159.6         36.2        ...  99.3          0.62218045
```

```
    0.94933078]
 [ 78.6        168.5         38.1        ...  98.5          0.58456973
    0.96191406]
 [ 82.8        147.8         34.8        ... 110.           0.74424899
    0.90609555]]
```

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

# Load the data
male = np.genfromtxt('nhanes_adult_male_bmx_2020 (2) (1).csv', delimiter=',', skip_header=1)
female = np.genfromtxt('nhanes_adult_female_bmx_2020 (1).csv', delimiter=',', skip_header=1)

# Calculate waist circumference to height ratio and waist circumference to hip circumference rati
waist_to_height_male = male[:, 6] / male[:, 1]  # Waist circumference / Height
waist_to_hip_male = male[:, 6] / male[:, 5]     # Waist circumference / Hip circumference
waist_to_height_female = female[:, 6] / female[:, 1]  # Waist circumference / Height
waist_to_hip_female = female[:, 6] / female[:, 5]     # Waist circumference / Hip circumference

# Print calculated ratios to check for issues
print("Male Waist-to-Height Ratios (first 10 values):", waist_to_height_male[:10])
print("Male Waist-to-Hip Ratios (first 10 values):", waist_to_hip_male[:10])
print("Female Waist-to-Height Ratios (first 10 values):", waist_to_height_female[:10])
print("Female Waist-to-Hip Ratios (first 10 values):", waist_to_hip_female[:10])

# Prepare data for boxplot, removing NaN values
data = [
    waist_to_height_male[~np.isnan(waist_to_height_male)],
    waist_to_hip_male[~np.isnan(waist_to_hip_male)],
    waist_to_height_female[~np.isnan(waist_to_height_female)],
    waist_to_hip_female[~np.isnan(waist_to_hip_female)]
]

# Create the box-and-whisker plot
plt.figure(figsize=(10, 6))
plt.boxplot(data, labels=['Male Waist-to-Height', 'Male Waist-to-Hip',
                           'Female Waist-to-Height', 'Female Waist-to-Hip'])

# Add title and labels
plt.title('Box-and-Whisker Plot of Waist-to-Height and Waist-to-Hip Ratios')
plt.ylabel('Ratio')
plt.grid()

# Show the plot
plt.show()
```

```
Male Waist-to-Height Ratios (first 10 values): [       nan 0.66044981 0.47122693 0.59147329 (
 0.71507692 0.42184557 0.47169811 0.5234657 ]
Male Waist-to-Hip Ratios (first 10 values): [       nan 1.11275416 0.91851852 1.01669759 1.0:
 1.05444646 0.8195122  0.928009   0.93147752]
Female Waist-to-Height Ratios (first 10 values): [       nan 0.73595506 0.67518009 0.5707196
 0.58604008 0.57477477 0.63883495 0.61243719]
Female Waist-to-Hip Ratios (first 10 values): [       nan 0.93497224 0.82151394 0.86629002 0
 0.79475164 0.99376947 0.91643454 0.95588235]
```



Interpretation of the Results When you examine the resulting box-and-whisker plot, consider the following points:

Median: The line inside each box represents the median of each ratio. Compare the medians of the waist-to-height and waist-to-hip ratios between males and females.

Interquartile Range (IQR): The height of the boxes represents the interquartile range, which shows the middle 50% of the data. A larger box indicates greater variability in the ratio.

Whiskers: The lines extending from the boxes (whiskers) show the range of the data. Outliers, if any, will be marked with dots beyond the whiskers.

Comparison: You can compare how the waist-to-height and waist-to-hip ratios differ between males and females. For example:

Which group has a higher median? Are there any outliers present? Is there a notable difference in variability between the two groups?

Advantages and Disadvantages of BMI, Waist-to-Height Ratio, and Waist-to-Hip Ratio Body Mass Index (BMI) **Advantages:**

Simplicity: BMI is easy to calculate using just weight and height, making it accessible for the general population. **Standardization:** It provides a standardized measure that can be applied across populations, making it useful for public health assessments and comparisons. **Cost-effective:** BMI measurements do not require specialized equipment, making it a cost-effective method for assessing obesity. **Disadvantages:**

Doesn't Differentiate Between Fat and Muscle: BMI does not distinguish between muscle mass and fat mass, leading to misclassifications, especially in athletes and muscular individuals. **Ignores Fat Distribution:** BMI does not account for where fat is distributed in the body, which is a significant factor in health risks. **Limited Accuracy for All Populations:** It may not accurately reflect the health status of certain demographic groups (e.g., elderly, children, or those with different body compositions). Waist-to-Height Ratio (WHtR) **Advantages:**

Focus on Central Obesity: WHtR is a better indicator of visceral fat and central obesity, which are associated with higher health risks than total body fat. **Simple Calculation:** Like BMI, it’s easy to compute using just waist circumference and height. **Potentially Better Predictive Validity:** Some studies suggest WHtR may better predict cardiovascular and metabolic risks compared to BMI. **Disadvantages:**

Height Dependency: WHtR can be misleading in very short or very tall individuals, potentially leading to inaccurate health assessments. **Limited Data:** While emerging, research on WHtR is less comprehensive compared to BMI, leading to less established guidelines. **Still Lacks Body Composition Detail:** WHtR does not provide insight into overall body composition or the proportion of muscle vs. fat. Waist-to-Hip Ratio (WHR) **Advantages:**

Indicator of Body Fat Distribution: WHR effectively indicates the distribution of body fat, particularly visceral fat, which is associated with higher health risks. **Useful for Cardiovascular Risk Assessment:** WHR can be a strong predictor of cardiovascular diseases and metabolic syndrome. **Cultural Relevance:** In some cultures, body shape is more significant than body weight, making WHR a culturally relevant measurement. **Disadvantages:**

Requires Accurate Measurements: Obtaining accurate waist and hip measurements can be challenging, leading to potential errors. **Limited Use in Certain Populations:** WHR might not be as effective in populations with different body shapes or where fat distribution varies significantly. **Doesn't Provide Total Body Fat Insight:** Like BMI and WHtR, WHR does not account for total body fat, which is essential for a comprehensive health assessment. **Conclusion:** While BMI, waist-to-height ratio, and waist-to-hip ratio each have their strengths and weaknesses, they can complement each other in assessing health risks related to obesity and body composition. Understanding the context in which each measurement is used can help guide more accurate assessments of health.

```
import numpy as np

# Read the female dataset
female = np.genfromtxt('nhanes_adult_female_bmx_2020 (1).csv', delimiter=',', skip_header=1)

# Check for NaN values
if np.isnan(female).any():
    print("NaN values detected in the dataset.")

# Extract weight and height data
weight_female = female[:, 0]
```

```
    height_female = female[:, 1]

    # Remove rows with NaN values
    valid_indices = ~np.isnan(weight_female) & ~np.isnan(height_female)
    weight_female = weight_female[valid_indices]
    height_female = height_female[valid_indices] / 100  # Convert height to meters

    # Calculate BMI
    bmi_female = weight_female / (height_female ** 2)

    # Check BMI values
    print("BMI Values:", bmi_female)

    # If you want to display the five lowest and highest BMI values
```