HW1 ASSIGNMENT

Q 2.1)

1. Write a code for the mean and find the following values:  
   μ(X) = ?

μ(Y ) = ?

μ(Z) = ?

* Code:-

μ(X) = 2.5 μ(Y ) = 2.1 ̄6 μ(Z) = 2.5

import statistics  
X = [3, 1, 2, 3, 1, 4, 3, 3]  
m = statistics.mean(X)  
print("The arithmetic mean is :", m)

A screenshot of a phone

AI-generated content may be incorrect.

def amean(X):  
 mX = 0  
 for ele in X:  
 mX += ele  
 return mX/len(X)

1. Write a code for the harmonic mean and find the following values:  
   μh(X) = ?

μh(Y ) = ?

μh(Z) = ?

* Code:-

from scipy.stats import hmean

harmonic\_mean\_X = hmean(X)

harmonic\_mean\_Y = hmean(Y)

harmonic\_mean\_Z = hmean(Z)

print("Harmonic Mean (X):", harmonic\_mean\_X)

print("Harmonic Mean (Y):", harmonic\_mean\_Y)

print("Harmonic Mean (Z):", harmonic\_mean\_Z)

μh(X) = 1.9591836735 μh(Y ) = 1.5319148936 μh(Z) = 1.92

def hmean(X):  
 mX = 0  
 for ele in X:  
 mX += 1/ele  
 mX = mX/len(X)  
 return 1/mX

1. Write a code for the geometric mean and find the following values:  
   μg(X) = ?

μg(Y ) = ?

μg(Z) = ?

* Code:-

from scipy.stats import gmean

geometric\_mean\_X = gmean(X)

geometric\_mean\_Y = gmean(Y)

geometric\_mean\_Z = gmean(Z)

print("Geometric Mean (X):", geometric\_mean\_X)

print("Geometric Mean (Y):", geometric\_mean\_Y)

print("Geometric Mean (Z):", geometric\_mean\_Z)

μg(X) = 2.2461919979 μg(Y ) = 1.8171205928 μg(Z) = 2.2133638394

def gmean(X):  
 mX = 1  
 for ele in X:  
 mX \*= ele  
 mX = mX\*\*(1/len(X))  
return mX

1. Write a code for the arithmetic-geometric mean and find the following values:  
   μag(X) = ?

μag(Y ) = ?

μag(Z) = ?

* Code:-

μag(X) = 2.37139789655 μag(Y ) = 1.9880506265 μag(Z) = 2.35450047778

Iterative version:-  
def myagmean(X):  
 am = amean(X)  
 gm = gmean(X)  
 while abs(am - gm) > 0.000001:  
 tam = (am + gm)/2  
 gm = (am \* gm)\*\*(1/2)  
 am = tam  
 return (am + gm)/2

Recursive version:-  
call agmean(amean(X), gmean(X)) initially

def agmean(am, gm):  
 if abs(am - gm) < 0.000001:  
 return (am + gm)/2  
 else:  
 tam = (am + gm)/2  
 tgm = (am \* gm)\*\*(1/2)  
 return agmean(tam, tgm)

1. Write a code for the arithmetic-geometric mean and find the following values:  
   μag(X) = ?

μag(Y ) = ?

μag(Z) = ?

* Code:-

μahg(X) = 2.224062384 μahg(Y ) = 1.8202847281 μahg(Z) = 2.1983271599

Iterative version:-

def myaghmean(X):  
 am = amean(X)  
 gm = gmean(X)  
 hm = hmean(X)  
 while max(abs(am - gm),abs(am - hm),abs(gm - hm)) > 0.00001:  
 tam = (am + gm + hm)/3  
 tgm = (am \* gm \* hm)\*\*(1/3)  
 hm = 3/(1/am + 1/gm + 1/hm)  
 am = tam  
 gm = tgm  
 return (am + gm + hm)/3

Recursive version  
call aghmean(amean(X), gmean(X)), hmean(X)) initially

def aghmean(am, gm, hm):  
 if max(abs(am - gm),abs(am - hm),abs(gm - hm)) < 0.00001:  
 return (am + gm + hm)/3  
 else:  
 tam = (am + gm + hm)/3  
 tgm = (am \* gm \* hm)\*\*(1/3)  
 thm = 3/(1/am + 1/gm + 1/hm)  
 return aghmean(tam, tgm, thm)

1. Write a code for the median and find the following values:  
   μi(X) = ?

μi(Y ) = ?

μi(Z) = ?

* Code:-

μi(X) = 3 μi(Y ) = 2 μi(Z) = 2.5

print(statistics.median(X))

def mymedian(X):  
 n = len(X)  
 X.sort()  
 if n % 2 == 0:  
 return (X[n//2-1] + X[n//2])/2  
 else:  
 return X[n//2]

1. Write a code for the mode and find the following values:  
   μo(X) = ?

μo(Y ) = ?

μo(Z) = ?

* Code:-

from scipy.stats import mode

μo(X) = 3 μo(Y ) = 1 μo(Z) = 1

statistics.mode(X)

from scipy import stats  
stats.mode(x)

hx=[2,1,4,1]  
np.argmax(hx)

1. Write a code for the midpoint and find the following values:  
   μm(X) = ?

μm(Y ) = ?

μm(Z) = ?

* Code:-

μm(X) = 2.5 μm(Y ) = 2.5 μm(Z) = 2.5

(max(X)+min(X))/2

1. Write a code for the p% trimmed midpoint and find the following values:  
   μm(X25∼75%) = ?

μm(Y25∼75%) = ?

μm(Z25∼75%) = ?

* Code:-

μm(X25∼75%) = 2.5 μm(Y25∼75%) = 2.0 μm(Z25∼75%) = 2.5

import numpy as np  
import math  
def trimmedmidpoint(X, p):  
 n = len(X)  
 X.sort()  
 s = math.floor(n\*p/100+1)  
 e = math.floor(n\*(100-p)/100)  
 S = np.array(X)  
 return S[(s-1):e]

tX = trimP(X,25)  
print((max(tX)+min(tX))/2)

import math  
def trimmedmidpoint(X, p):  
 n = len(X)  
 X.sort()  
 s = math.floor(n\*p/100+1)  
 e = math.floor(n\*(100-p)/100)  
 return (X[s-1]+X[e-1])/2  
print(trimmedmidpoint(X,25))

1. Write a code for the exclusive quartiles and find the following values:  
   xQ1(X) = ?

xQ1(Y ) = ?

xQ1(Z) = ?  
xQ3(X) = ?

xQ3(Y ) = ?

xQ3(Z) = ?

* Code:-

xQ1(X) = 1.5 xQ1(Y ) = 1.0 xQ1(Z) = 1.5  
xQ3(X) = 3.0 xQ3(Y ) = 3.0 xQ3(Z) = 3.5

def Quartile\_x(X):  
 n = len(X)  
 X.sort()  
 S = np.array(X)  
 e = math.floor(n/2)  
 s = math.floor((n+1)/2)  
 return(mymedian(S[:e]), mymedian(S[s:n]))

1. Write a code for the inclusive quartiles and find the following values:  
   iQ1(X) = ?

iQ1(Y ) = ?

iQ1(Z) = ?  
iQ3(X) = ?

iQ3(Y ) = ?

iQ3(Z) = ?

* Code:-

iQ1(X) = 1.5 iQ1(Y ) = 1.0 iQ1(Z) = 1.5  
iQ3(X) = 3.0 iQ3(Y ) = 3.0 iQ3(Z) = 3.5

def Quartile i(X):  
 n = len(X)  
 X.sort()  
 S = np.array(X)  
 e = math.ceil(n/2)  
 s = math.floor(n/2)  
 return(mymedian(S[:e]), mymedian(S[s:n]))

1. Write a code for the entity proportional quartiles and find the following values:  
   eQ1(X) = ?

eQ1(Y ) = ?

eQ1(Z) = ?  
eQ3(X) = ?

eQ3(Y ) = ?

eQ3(Z) = ?

* Code:-

eQ1(X) = 1.5 eQ1(Y ) = 1.0 eQ1(Z) = 1.5  
eQ3(X) = 3.0 eQ3(Y ) = 3.0 eQ3(Z) = 3.5

def Quartile ep(X):  
 n = len(X)  
 X.sort()  
 q1 = (4 - (n+2)%4)/4 \* X[math.floor((n+2)/4)-1]  
 + ((n+2)%4)/4 \* X[math.ceil((n+2)/4)-1]  
 q3 = ((n+2)%4)/4 \* X[math.floor((3\*n+2)/4)-1]  
 + (4-(n+2)%4)/4 \* X[math.ceil((3\*n+2)/4)-1]  
 return(q1, q3)

1. Write a code for the scale proportional quartiles and find the following values:  
   sQ1(X) = ?

sQ1(Y ) = ?

sQ1(Z) = ?  
sQ3(X) = ?

sQ3(Y ) = ?

sQ3(Z) = ?

* Code:-

sQ1(X) = 1.75 sQ1(Y ) = 1.0 sQ1(Z) = 1.75  
sQ3(X) = 3.0 sQ3(Y ) = 3.0 sQ3(Z) = 3.2

def Quartile ep(X):  
 n = len(X)  
 X.sort()  
 q1 = (4 - (n-1)%4)/4 \* X[math.floor((n-1)/4)]  
 + ((n-1)%4)/4 \* X[math.ceil((n-1)/4)]  
 q3 = ((n-1)%4)/4 \* X[math.floor((3\*n-3)/4)]  
 + (4-(n-1)%4)/4 \* X[math.ceil((3\*n-3)/4)]  
 return(q1, q3)

OR

import numpy as np  
print(np.quantile(X, .25), np.quantile(X, .75))

Q 2.2)

During the Peloponnesian war in 431 BC, attackers besieging Plataea wanted to find  
the height of the wall in order to build ladders needed. The height of the wall was estimated  
by counting the number of bricks. 40 soldiers reported their estimates of the number of  
bricks. Frequency of soldiers estimated each number of bricks is given as follows:

A table with numbers and a tower

AI-generated content may be incorrect.

**The Data**

The data consists of:

* X: Number of bricks
* fr(X): Frequency of estimates for each number of bricks

We first calculate the actual dataset by "expanding" the frequencies into a list of values.

import numpy as np

from scipy.stats import hmean, gmean, mode

# Frequency distribution

X = np.array([103, 104, 105, 106, 107, 108, 109, 110, 111]) # Brick counts

frequencies = np.array([5, 4, 6, 5, 6, 8, 3, 2, 1]) # Frequencies

# Expand the data

data = np.repeat(X, frequencies)

1. Write a code for the arithmetic mean and find the value, μ(B).

# Arithmetic mean

arithmetic\_mean = np.mean(data)

print("Arithmetic Mean (μ(B)):", arithmetic\_mean)

5 × 103 + 4 × 104 + 6 × 105 + 5 × 106 + 6 × 107 + 8 × 108 + 3 × 109 + 2 × 110 + 111  
40

μw(B, f r(B)) = 106.375

1. Write a code for the median and find the value, μi(B).

# Median

median = np.median(data)

print("Median (μ\_i(B)):", median)

1. Write a code for the mode and find the value, μo(B).

# Mode

mode\_value = mode(data).mode[0]

print("Mode (μ\_o(B)):", mode\_value)

1. Write a code for the harmonic mean and find the value, μh(B).

# Harmonic mean

harmonic\_mean = hmean(data)

print("Harmonic Mean (μ\_h(B)):", harmonic\_mean)

1. Write a code for the geometric mean and find the value, μg(B).

# Geometric mean

geometric\_mean = gmean(data)

print("Geometric Mean (μ\_g(B)):", geometric\_mean)

μag (B) = 106.3642304

1. Write a code for the arithmetic-geometric mean and find the value, μag(B).

A black background with white text

AI-generated content may be incorrect.

def arithmetic\_geometric\_mean(a, g, tol=1e-9):

while abs(a - g) > tol:

a, g = (a + g) / 2, np.sqrt(a \* g)

return a

# Initial values

a0 = arithmetic\_mean

g0 = geometric\_mean

# Compute the arithmetic-geometric mean

ag\_mean = arithmetic\_geometric\_mean(a0, g0)

print("Arithmetic-Geometric Mean (μ\_ag(B)):", ag\_mean)