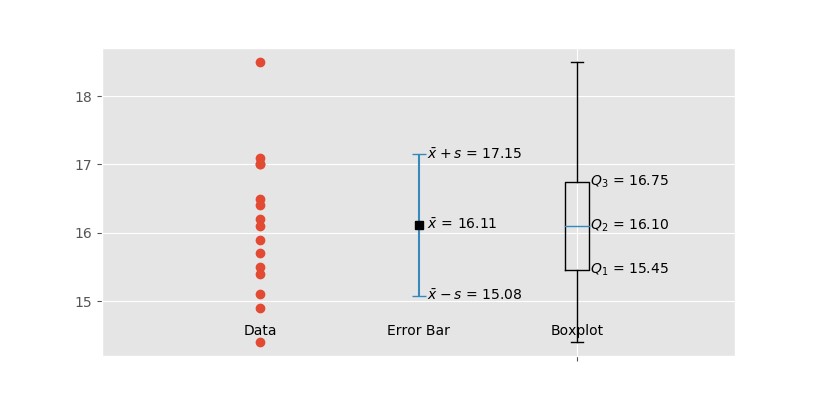
**Mid Exam Simulation #2**

1. A sample of the data of the amount of water in liter used by some washing machines are of the following.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5.5 | 1.1 | 6.5 | 4.9 | 6.4 |
| 7.0 | 1.5 | 5.7 | 5.9 | 5.4 |
| 6.1 | 1.2 | 7.3 | 6.1 | 4.4 |

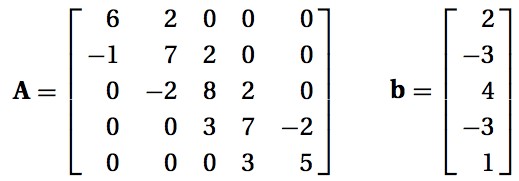
Write a python program to compute the statistic descriptive of the sample data.

* 1. Present the data graphically by using dot plot, error bar, and boxplot. Your results may look like the following.



**import** matplotlib.pyplot **as** plt  
**import** numpy **as** np  
  
*# https://matplotlib.org/api/\_as\_gen/matplotlib.axes.Axes.plot.html*plt.style.use(**'ggplot'**)  
  
*#  
# x= np.linspace(0, 4\*np.pi, 1000)  
# y = np.sinc(x)  
#  
# fig = plt.figure(1)  
# plt.clf()  
#  
# ax = fig.add\_subplot(1,1,1)  
# ax.plot(x,y)  
  
# x = np.array([3450, 4550, 4650, 3480, 3355, 3310, 3490, 3730, 3925, 3520, 3480])*x = np.array([  
 5.5, 1.1, 6.5, 4.9, 6.4,  
 7.0, 1.5, 5.7, 5.9, 5.4,  
 6.1, 1.2, 7.3, 6.1, 4.4,  
])  
mean = np.mean(x)  
std = np.std(x, ddof=1)  
n = len(x)  
Q = np.percentile(x, [25, 50, 75])  
  
fig = plt.figure(1)  
plt.clf()  
ax = fig.add\_subplot(1, 1, 1)  
*# ax.boxplot(x, vert=False, patch\_artist=True)  
# ax.plot(x, 0.75 \* np.ones(n) , 'or')  
# ax.plot(Q, 1.25 \* np.ones(3), 's')***for** i **in** x:  
 plt.plot(1, i, **'or'**)  
  
*# plt.errorbar(2, mean, std,marker='s',capsize=5,color='m')*plt.errorbar(2, mean, std, capsize=5, color=**'b'**)  
plt.plot(2, mean, **'s'**, color=**'k'**)  
  
up = mean + std  
down = mean - std  
  
ax.text(2.05, up, **'ẍ+s = '** + str(round(up, 2)), fontsize=8)  
ax.text(2.05, mean, **'ẍ = '** + str(round(mean, 2)), fontsize=8)  
ax.text(2.05, down, **'ẍ-s = '** + str(round(down, 2)), fontsize=8)  
  
plt.boxplot([[], [], x])  
  
ax.text(3.2, Q[0], **'Q1 = '** + str(Q[0]), fontsize=8)  
ax.text(3.2, Q[1], **'Q2 = '** + str(Q[1]), fontsize=8)  
ax.text(3.2, Q[2], **'Q3 = '** + str(Q[2]), fontsize=8)  
  
plt.xticks([1, 2, 3], [**"Data"**, **"Error Bar"**, **"Boxplot"**])  
  
plt.tight\_layout()  
plt.show()

1. We consider the linear algebra problem 𝐴𝑥 = 𝑏 where the system matrix and the right-hand-side vector are:

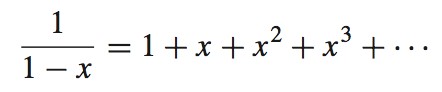


Use python, to compute the following. Copy your solutions to the answer sheet.

* 1. Transpose of matrix A and vector b
  2. The Euclidean norm of the vector 𝑥.

*# --------------------------------  
# 2*A = np.array([  
 [6, 2, 0, 0, 0],  
 [-1, 7, 2, 0, 0],  
 [0, -2, 8, 2, 0],  
 [0, 0, 3, 7, -2],  
 [0, 0, 0, 3, 5]  
])  
  
b = np.array([  
 [2],  
 [-3],  
 [4],  
 [-3],  
 [1]  
])  
  
*#2 a)*transpose\_A = np.transpose(A)  
transpose\_b = np.transpose(b)  
print(**"Transpose A"**, transpose\_A)  
print(**"Transpose b"**, transpose\_b)  
  
*#2 b)*x = np.linalg.solve(A, b)  
norm = np.linalg.norm(x)  
print(**"Euclidean norm of vector x "**, norm)

1. If |x| < 1, it is known that:



* 1. Write a python function that compute 1/(1-x) for x = 0.1. Your function should have the following interface that allows the user to adjust the number of the terms of the series 𝑛.

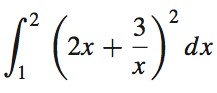
def taylor\_1(x, n):

# Your codes

* 1. Use your function to compute 1/(1-x) with x = 0.1 by using the first 5 and 10 terms of the series.
  2. Compute the relative errors of the results in (b).

*#---------------------------  
#3 a)***def** taylor\_1(x,n):  
 a = 0  
 **for** i **in** range(0, n):  
 a = a + x\*\*i  
 **return** a  
  
*#3 b)*tay\_5 = taylor\_1(0.1 , 5)  
tay\_10 = taylor\_1(0.1, 10)  
  
print(**"5 terms"**, tay\_5)  
print(**"10 terms"**, tay\_10)  
  
*#3 c)*original = 1/(1-0.1)  
error\_5 = original - tay\_5  
error\_10 = original - tay\_10  
print(**"Original"**, original)  
print(**"Error 5 "**, error\_5)  
print(**"Error 10"**, error\_10)

1. Compute the exact solution and the numerical solution for the problem:



As for the numerical solution using Gauss quadrature with two quadrature points of 𝜉’ = −0.577350 and 𝜉. = +0.577350, and the two points have the same weight of 𝐴0 = 1.0. Compute the relative error of the numerical solution compared to the exact integration result.

import numpy as np

from scipy.integrate import \*

def Gauss(f,a, b):

zeta = np.array([-0.577350,0.577350])

w = np.array([1.0, 1.0])

x = (b+a)/2 + (b-a)/2 \* zeta

area = (b-a)/2 \* np.sum(w \* f(x))

return area

f = lambda x : (2\*x + 3/x)\*\*2

a = 1

b = 2

area\_scipy = quad(f, a, b)[0]

area\_gauss = Gauss(f, a, b)

print("Area scipe " , area\_scipy)

print("Area gauss ", area\_gauss)