CENX570: Simulation and Modelling

HW #4

Submitted by:

Mohammed Shahzad

444105788@student.ksu.edu.sa

Gigen the traces of a generated long MPEG encoded video sequences:

1. Compute the important statistical parameters of each trace: mean, variance, autocorrelation, mode, mean, max.

2. Build a descriptive histogram for each trace;

3. Select the most appropriate PDF;

4. Draw the Q-Q diagram and make a decision;

5. Draw the autocorrelation function and choose a fitting curve.

6. Make a simulation of a similar trace, and compare the obtained statistical parameters.

7. Conclusions.

1. Compute the important statistical parameters of each trace: mean, variance, autocorrelation, mode, mean, max.

Answer:

<Refer attached - code1.py>

```
import statistics
import statsmodels

# race_I
with open('race_I.txt', 'r') as f:
    data = f.readlines()
    master_list = []
    lst = []
    for i in data:
        if '\n' in i:
            master_list.append(lst)
            lst = int(i)
        else:
            lst.append(i.replace('\n', ''))
master_list.append(61832)
master_list.remove([])
print("Trace - Race_I\n")

#Mean
average = statistics.mean(master_list)
print("Mean is",int(average))

#Variance
variancee = statistics.variance(master_list)
print("variance is",int(variancee))

#autocorrelation
#generate delayed by time 1
```

```
master_list_lagby1 = []
for i in range(0,len(master_list)):
    master_list_lagby1.append(int(master_list[i])+1)

autocorr = statistics.correlation(master_list,master_list_lagby1)
# correlation(x, y, /, *, method='linear')
print("Autocorrelation is", autocorr)

#mode
modee = statistics.mode(master_list)
print("Mode is",int(modee))

#median
mediann = statistics.median(master_list)
print("median is",int(mediann))

#Max
maxx = max(master_list)
print("The maximum value is",maxx)

print("\n----\n")

#   race_B
with open('race_B.txt', 'r') as f:
    data = f.readlines()
    master_list2 = []
    lst = []
    for i in data:
        if '\n' in i:
            master_list2.append(lst)
            lst = int(i)
        else:
            lst.append(i.replace('\n', ''))
master_list2.append(13144)
master_list2.remove([])
print("Trace - Race_B\n")

#Mean
average = statistics.mean(master_list2)
print("Mean is",int(average))

#Variance
varianceе = statistics.variance(master_list2)
print("variance is",int(variancee))

#autocorrelation
#generate delayed by time 1
master_list_lagby1 = []
for i in range(0,len(master_list2)):
    master_list_lagby1.append(int(master_list2[i])+1)
```

```
autocorr = statistics.correlation(master_list2,master_list_lagby1)
# correlation(x, y, /, *, method='linear')
print("Autocorrelation is", autocorr)

#mode
modee = statistics.mode(master_list2)
print("Mode is",int(modee))

#median
mediann = statistics.median(master_list2)
print("median is",int(mediann))

#Max
maxx = max(master_list2)
print("The maximum value is",maxx)

print("\n----\n")

#   race_IPB
with open('race_IPB.txt', 'r') as f:
    data = f.readlines()
    master_list3 = []
    lst = []
    for i in data:
        if '\n' in i:
            master_list3.append(lst)
            lst = int(i)
        else:
            lst.append(i.replace('\n', ''))
master_list3.append(13144)
master_list3.remove([])
print("Trace - Race_IPB\n")

#Mean
average = statistics.mean(master_list3)
print("Mean is",int(average))

#Variance
variancee = statistics.variance(master_list3)
print("variance is",int(variancee))

#autocorrelation
#generate delayed by time 1
master_list_lagby1 = []
for i in range(0,len(master_list3)):
    master_list_lagby1.append(int(master_list3[i])+1)

autocorr = statistics.correlation(master_list3,master_list_lagby1)
# correlation(x, y, /, *, method='linear')
```

```
print("Autocorrelation is", autocorr)

#mode
modee = statistics.mode(master_list3)
print("Mode is",int(modee))

#median
mediann = statistics.median(master_list3)
print("median is",int(mediann))

#Max
maxx = max(master_list3)
print("The maximum value is",maxx)

print("\n----\n")
```

Results:

Trace - Race_I

Mean is 79241
variance is 433748571
Autocorrelation is 1.0
Mode is 84480
median is 75808
The maximum value is 186048

Trace - Race_B

Mean is 21891
variance is 99934158
Autocorrelation is 1.0
Mode is 12920
median is 20040
The maximum value is 165448

Trace - Race_IPB

Mean is 30749
variance is 448078471
Autocorrelation is 1.0
Mode is 17976
median is 23808
The maximum value is 202416

----

2. Build a descriptive histogram for each trace

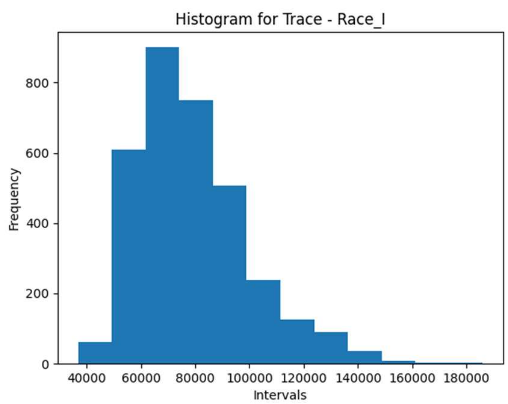
Answer:

<Refer attached - code2.py>

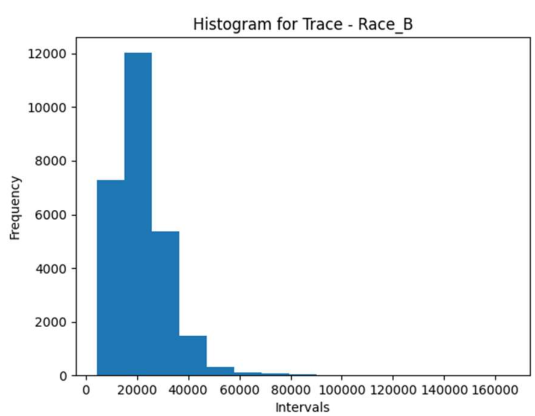
```
import matplotlib.pyplot as plt

# race_I
with open('race_I.txt', 'r') as f:
    data = f.readlines()
    master_list = []
    lst = []
    for i in data:
        if '\n' in i:
            master_list.append(lst)
            lst = int(i)
        else:
            lst.append(i.replace('\n', ''))
master_list.append(61832)
master_list.remove([])

# histogram of Race_I
print("Histogram for Trace - Race_I\n")
plt.title("Histogram for Trace - Race_I")
plt.xlabel("Intervals")
plt.ylabel("Frequency")
plt.hist(master_list, bins=12)
plt.show()

#   race_B
with open('race_B.txt', 'r') as f:
    data = f.readlines()
    master_list2 = []
    lst = []
    for i in data:
        if '\n' in i:
            master_list2.append(lst)
            lst = int(i)
        else:
            lst.append(i.replace('\n', ''))
master_list2.append(13144)
master_list2.remove([])
print("Histogram for Trace - Race_B\n")
plt.title("Histogram for Trace - Race_B")
plt.xlabel("Intervals")
plt.ylabel("Frequency")
plt.hist(master_list2, bins=15)
plt.show()

#   race_IPB
```

```
with open('race_IPB.txt', 'r') as f:
    data = f.readlines()
    master_list3 = []
    lst = []
    for i in data:
        if '\n' in i:
            master_list3.append(lst)
            lst = int(i)
        else:
            lst.append(i.replace('\n', ''))
master_list3.append(13144)
master_list3.remove([])
print("Histogram for Trace - Race_IPB\n")
plt.title("Histogram for Trace - Race_IPB")
plt.xlabel("Intervals")
plt.ylabel("Frequency")
plt.hist(master_list3, bins=10)
plt.show()
```

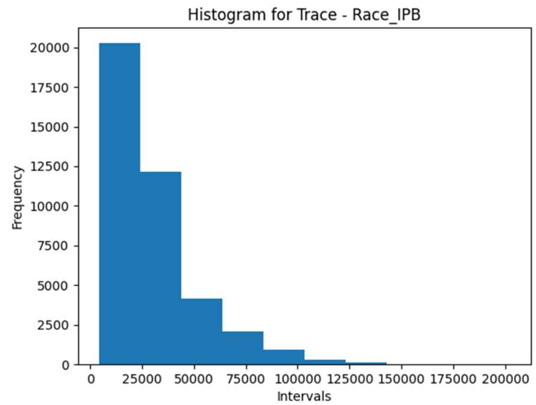
Results:



Histogram for Trace - Race_I



Histogram for Trace - Race_B



Histogram for Trace - Race_IPB

3. Select the most appropriate PDF

Answer:

Trace_I - Slight Left skewed, then uniform – Poisson distribution

Trace_P - Left skew – Chi square distribution

Trace_IPB - Starts from left and gradually decreases – Geometric distribution

4. Draw the Q-Q diagram and make a decision

Answer:

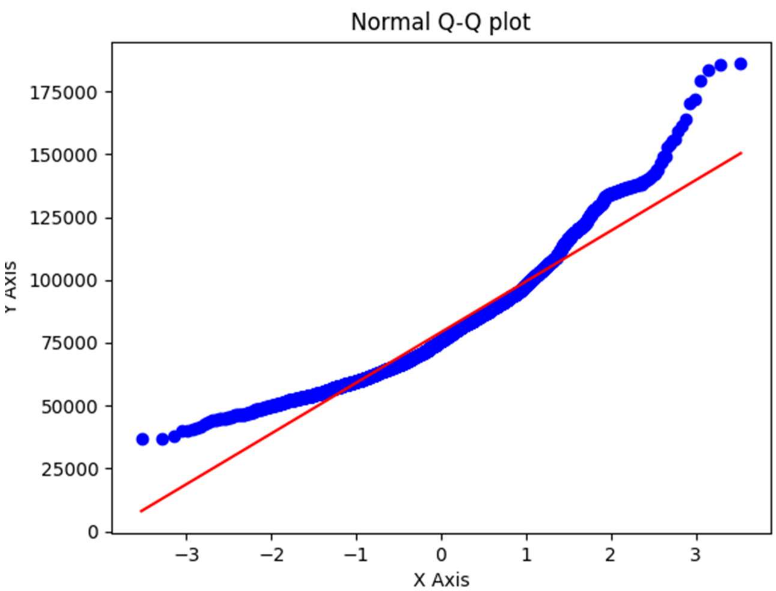
<Please refer attached code4.py>

```
# code
import scipy.stats as stats
import matplotlib.pyplot as plt
import statsmodels.api as sm

#Trace_I
with open('race_I.txt', 'r') as f:
    data = f.readlines()
    master_list = []
    lst = []
    for i in data:
        if '\n' in i:
            master_list.append(lst)
            lst = int(i)
        else:
            lst.append(i.replace('\n', ''))
master_list.append(61832)
master_list.remove([])
master_list.sort()
```

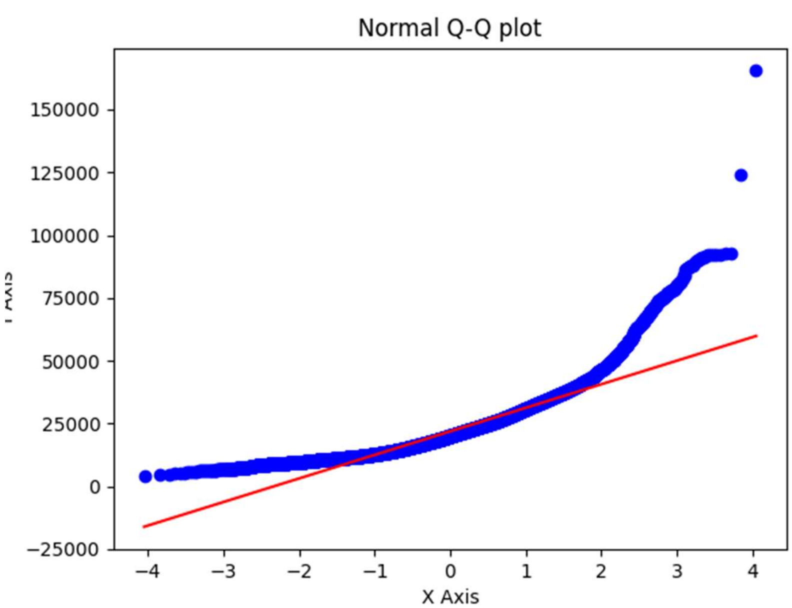
```
print("\n---Trace_I-----\n")

#q-q plot
stats.probplot(master_list, dist="norm", plot=plt)
plt.title("Normal Q-Q plot")
plt.xlabel("X Axis")
plt.ylabel("Y Axis")
plt.show()

#Trace_B
with open('race_B.txt', 'r') as f:
    data = f.readlines()
    master_list2 = []
    lst = []
    for i in data:
        if '\n' in i:
            master_list2.append(lst)
            lst = int(i)
        else:
            lst.append(i.replace('\n', ''))
master_list2.append(61832)
master_list2.remove([])
master_list2.sort()

print("\n---Trace_B-----\n")

#q-q plot
stats.probplot(master_list2, dist="norm", plot=plt)
plt.title("Normal Q-Q plot")
plt.xlabel("X Axis")
plt.ylabel("Y Axis")
plt.show()

#Trace_IPB
with open('race_IPB.txt', 'r') as f:
    data = f.readlines()
    master_list3 = []
    lst = []
    for i in data:
        if '\n' in i:
            master_list3.append(lst)
            lst = int(i)
        else:
            lst.append(i.replace('\n', ''))
master_list3.append(61832)
master_list3.remove([])
master_list3.sort()

print("\n---Trace_IPB-----\n")
```

```
#q-q plot
stats.probplot(master_list3, dist="norm", plot=plt)
plt.title("Normal Q-Q plot")
plt.xlabel("X Axis")
plt.ylabel("Y Axis")
plt.show()
```

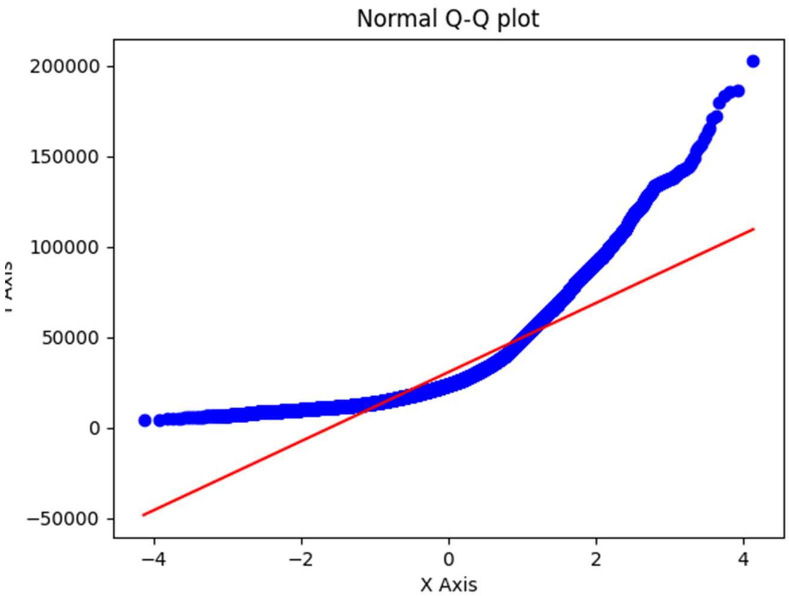
Results:



Trace I Q-Q Plot



Trace B Q-Q plot



Trace IPB Q-Q plot

Trace I and Trace B fit. Trace IPB does not pass the Q-Q test.

5. Draw the autocorrelation function and choose a fitting curve.

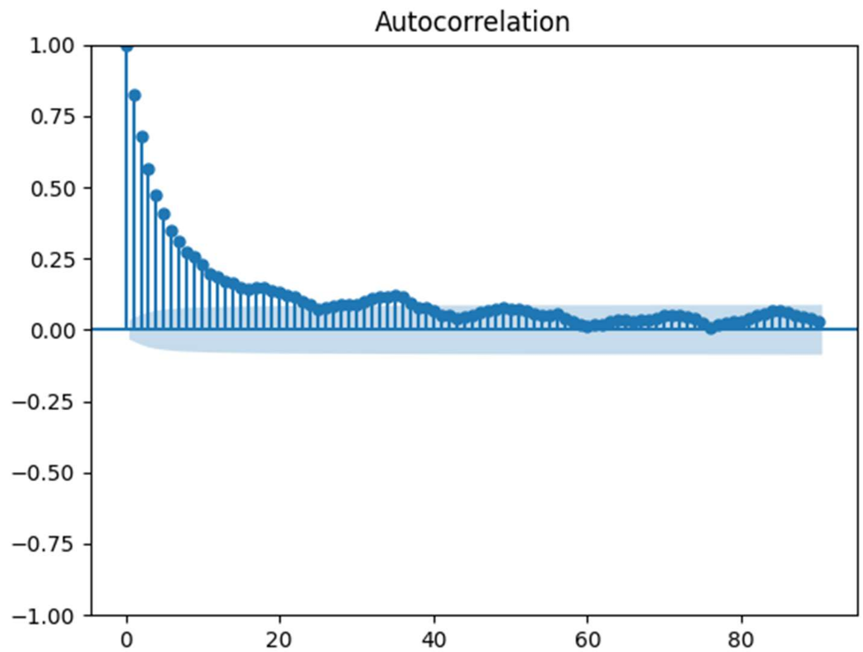
Answer:

<Please see attached code5.py>

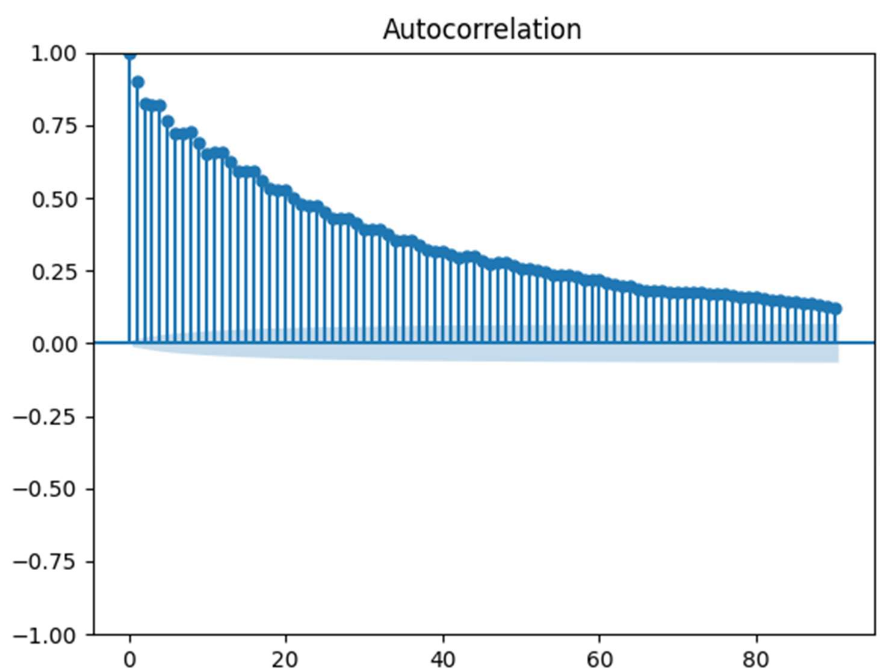
```
import pandas as pd
from statsmodels.graphics.tsaplots import plot_acf
import numpy as np
import matplotlib.pyplot as plt

# race_I
with open('race_I.txt', 'r') as f:
    data = f.readlines()
    master_list = []
    lst = []
    for i in data:
        if '\n' in i:
            master_list.append(lst)
            lst = int(i)
        else:
            lst.append(i.replace('\n', ''))
master_list.append(61832)
master_list.remove([])
print("\nThe autocorrelation function for the given trace - Trace I\n")
```

```
master_list = np.array(master_list)
plt.figure(figsize=(10,5))
plt.plot(master_list)
plot_acf(master_list,lags = 90)
plt.show()

#   race_B
with open('race_B.txt', 'r') as f:
    data = f.readlines()
    master_list2 = []
    lst = []
    for i in data:
        if '\n' in i:
            master_list2.append(lst)
            lst = int(i)
        else:
            lst.append(i.replace('\n', ''))
master_list2.append(13144)
master_list2.remove([])
print("\nThe autocorrelation function for the given trace - Trace B\n")
master_list2 = np.array(master_list2)
plt.figure(figsize=(10,5))
plt.plot(master_list2)
plot_acf(master_list2,lags = 90)
plt.show()

#   race_IPB
with open('race_IPB.txt', 'r') as f:
    data = f.readlines()
    master_list3 = []
    lst = []
    for i in data:
        if '\n' in i:
            master_list3.append(lst)
            lst = int(i)
        else:
            lst.append(i.replace('\n', ''))
master_list3.append(13144)
master_list3.remove([])
print("\nThe autocorrelation function for the given trace - Trace IPB\n")
master_list3 = np.array(master_list3)
plt.figure(figsize=(10,5))
plt.plot(master_list3)
plot_acf(master_list3,lags = 90)
plt.show()
```

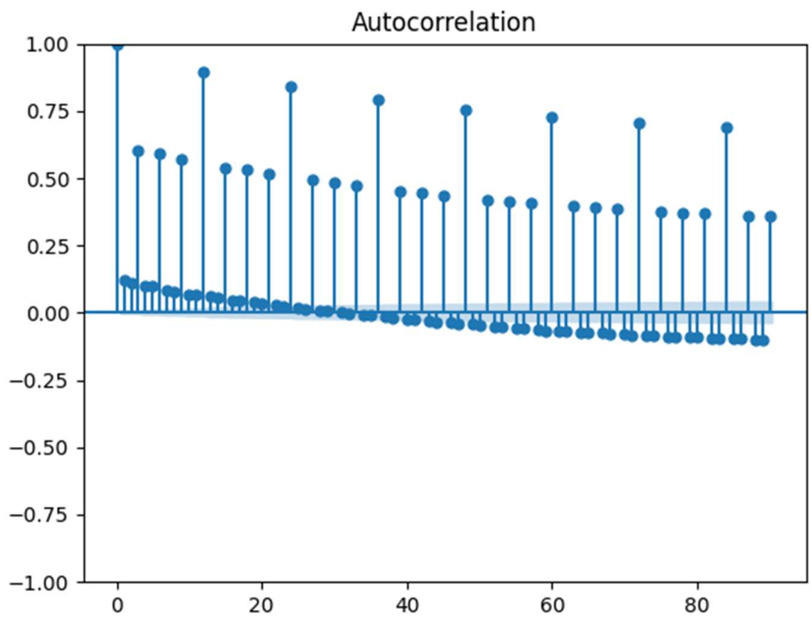
Results: (With 95% confidence band)



The autocorrelation function for the given trace - Trace I



The autocorrelation function for the given trace - Trace B



The autocorrelation function for the given trace - Trace IPB

Result inference:

Autocorrelation of trace I shows that there is no autocorrelation.

Autocorrelation of trace B shows that there is large positive correlation.

Autocorrelation of trace IPB, indicates presence of negative autocorrelation.

6. Make a simulation of a similar trace, and compare the obtained statistical parameters.

<u>**Answer:**</u>

<Please see code6.py>

```
import statistics
import statsmodels

# generate random integer values
from random import seed
from random import randint

# race_I
with open('race_I.txt', 'r') as f:
    data = f.readlines()
    master_list = []
```

```
    lst = []
    for i in data:
        if '\n' in i:
            master_list.append(lst)
            lst = int(i)
        else:
            lst.append(i.replace('\n', ''))

# seed random number generator
n = len(master_list)
seed(311)
valueList = []
value = []
# generate some integers
for _ in range(n):
    value = randint(0, 1000)
    valueList.append(value)
#print(master_list)

master_list.append(61832)
master_list.remove([])
for i in range(n):
    master_list[i] = int(master_list[i])+valueList[i]

#new similar trace generated
print("New Trace \n")

#Mean
average = statistics.mean(master_list)
print("Mean is",int(average))

#Variance
variancee = statistics.variance(master_list)
print("variance is",int(variancee))

#autocorrelation
#generate delayed by time 1
master_list_lagby1 = []
for i in range(0,len(master_list)):
    master_list_lagby1.append(int(master_list[i])+1)

autocorr = statistics.correlation(master_list,master_list_lagby1)
# correlation(x, y, /, *, method='linear')
print("Autocorrelation is", autocorr)

#mode
modee = statistics.mode(master_list)
print("Mode is",int(modee))

#median
```

```
mediann = statistics.median(master_list)
print("median is",int(mediann))

#Max
maxx = max(master_list)
print("The maximum value is",maxx)

print("\n----\n")
```

Result:

New Trace

Mean is 79732

variance is 433782637

Autocorrelation is 1.0

Mode is 55369

median is 76244

The maximum value is 186308

----

7. Conclusion

Answer:

In this work, we studied three traces, trace I, B and IBP by calculating their measures of central tendencies, maximum value and autocorrelation using python programming. The results discussed above for each trace highlight some characteristics of the traces. We also generated a similar trace and calculated its mean, median, mode, variance and autocorrelation.

Also, after performing the Quantile-Quantile test we were able to identify positive and negative correlation between the elements of the trace.

***