Report for the 4th Assignment of ELEC-E7130

[redacted]

[redacted]

Report, task 1

1. Box plot

A box plot is a type of chart which can show the distribution of numerical data and skewness through displaying the data quartiles and averages.
Basically, the x-axis stands for a specific category, the y-axis represents the minimum, first quartile, median, third quartile, and maximum value in the set of numbers of the specific category.

2. Parallel coordinates plot

A parallel coordinates plot is usually used to study the features of samples for several quantitative variables.
Typically, the x-axis stands for the quantitative variables of the samples, each variable has a vertical line showing the corresponding value, and then each variable will be normalized and sharing the same unit, which can be seen in the y-axis. This is to say, y-axis will show the values of each variable.

3. Lag plot

A lag plot is a special type of scatter plot, which can be used to check for model suitability, outliers, randomness, and serial correlation as well as seasonality.
As for the x-axis and y-axis, given a data set Y1, Y2 ..., Yn, usually a plot of lag 1 is a plot of the values of Yi versus Yi-1, this is to say, y-axis represents Yi for all i and x-axis represents Yi-1 for all i.

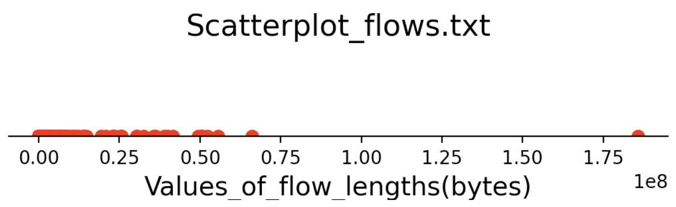
4. Autocorrelation plot

Autocorrelation plot graphically summarizes the strength of a relationship with an observation in a time series with observations at prior time steps, which are heavily used in time series analysis and forecasting.
Typically, y-axis represents the autocorrelation coefficient, which is calculated by the result of an autocovariance function dividing the result of a variance function, the x-axis shows the time lag.

Report, task 2

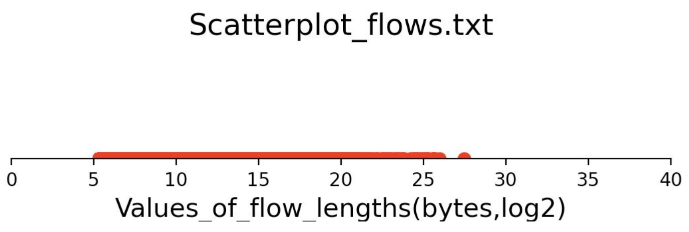
1. Code and result of scatter plot for flows.txt

```
import pandas as pd
import matplotlib.pyplot as plt
data =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-4/r-data/flows.txt',names=["A"]
, header=None)
x=[]
y=[]
for index, row in data.iterrows():
   x.append(row["A"])
   y.append(0)
fig = plt.figure()
ax = fig.add_subplot(111)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)
plt.title("Scatterplot_flows.txt",fontsize=16)
plt.xlabel("Values_of_flow_lengths(bytes)",fontsize=14)
plt.yticks([])
plt.ylim((0, 0.1))
plt.scatter(x, y, s=40, c='r')
plt.show()
```



2. Code and result of scatter plot using logarithmic values

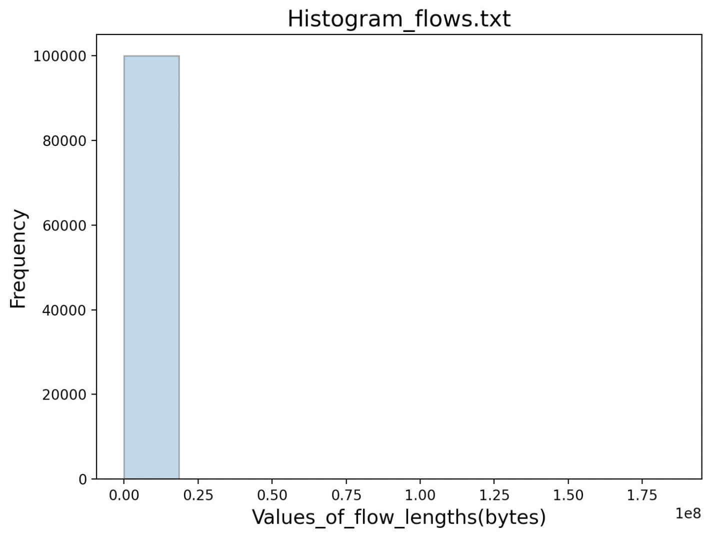
```
import math
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
data =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-4/r-data/flows.txt',names=["A"]
, header=None)
x=[]
y=[]
for index, row in data.iterrows():
   x.append(math.log2(int(row["A"])))
   y.append(0)
fig = plt.figure()
ax = fig.add_subplot(111)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)
ax.xaxis.set_major_locator(ticker.MultipleLocator(5))
plt.title("Scatterplot_flows.txt",fontsize=16)
plt.xlabel("Values_of_flow_lengths(bytes,log2)",fontsize=14)
plt.yticks([])
plt.ylim((0, 0.1))
plt.xlim((0, 40))
plt.scatter(x, y, s=40, c='r')
plt.show()
```



3. Code and result of histogram plot for flows.txt

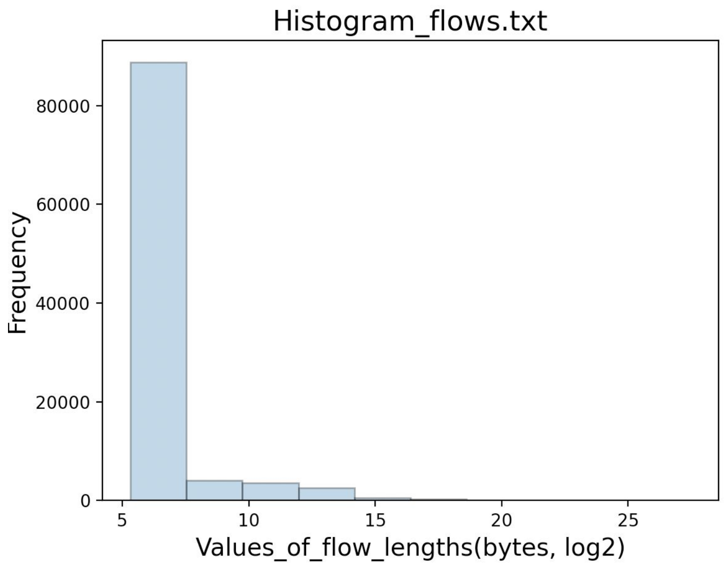
```
import pandas as pd
import matplotlib.pyplot as plt
```

```
data =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-4/r-data/flows.txt',names=["A"]
, header=None)
x=[]
for index, row in data.iterrows():
   x.append(row["A"])
plt.title("Histogram_flows.txt",fontsize=16)
plt.xlabel("Values_of_flow_lengths(bytes)",fontsize=14)
plt.ylabel("Frequency",fontsize=14)
plt.hist(x, edgecolor='k', alpha=0.35)
plt.show()
```



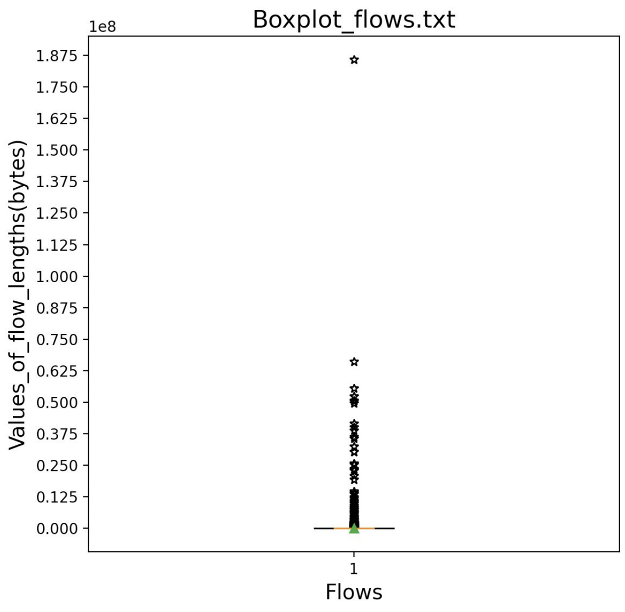
4. Code and result of histogram plot using logarithmic values

```
import math
import pandas as pd
import matplotlib.pyplot as plt
data =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-4/r-data/flows.txt',names=["A"]
, header=None)
x=[]
for index, row in data.iterrows():
   x.append(math.log2(int(row["A"])))
plt.title("Histogram_flows.txt",fontsize=16)
plt.xlabel("Values_of_flow_lengths(bytes, log2)",fontsize=14)
plt.ylabel("Frequency",fontsize=14)
plt.hist(x, edgecolor='k', alpha=0.35)
plt.show()
```



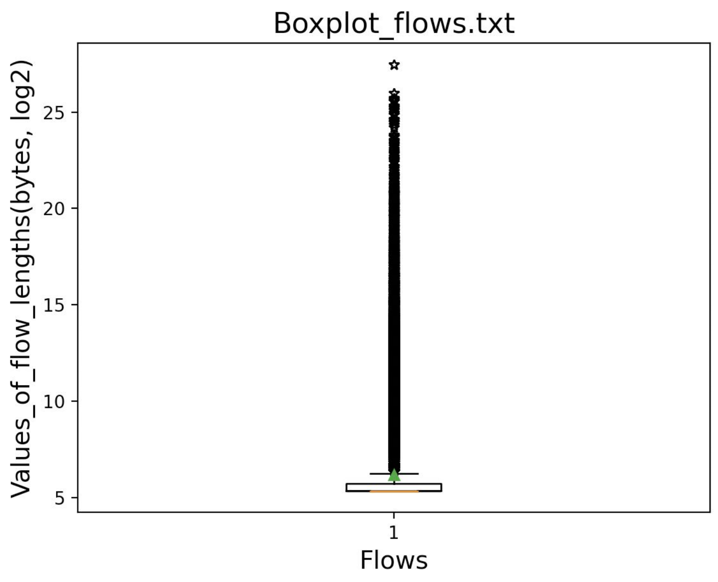
5. Code and result of box plot for flows.txt

```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
data =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-4/r-data/flows.txt',names=["A"]
, header=None)
x=[]
for index, row in data.iterrows():
   x.append(row["A"])
plt.figure()
plt.figure().add_subplot(111).yaxis.set_major_locator(ticker.MultipleLocator(125000
00))
plt.title("Boxplot_flows.txt",fontsize=16)
plt.xlabel("Flows",fontsize=14)
plt.ylabel("Values_of_flow_lengths(bytes)",fontsize=14)
plt.boxplot(x, showmeans=True, notch = True, sym = '*')
plt.show()
```



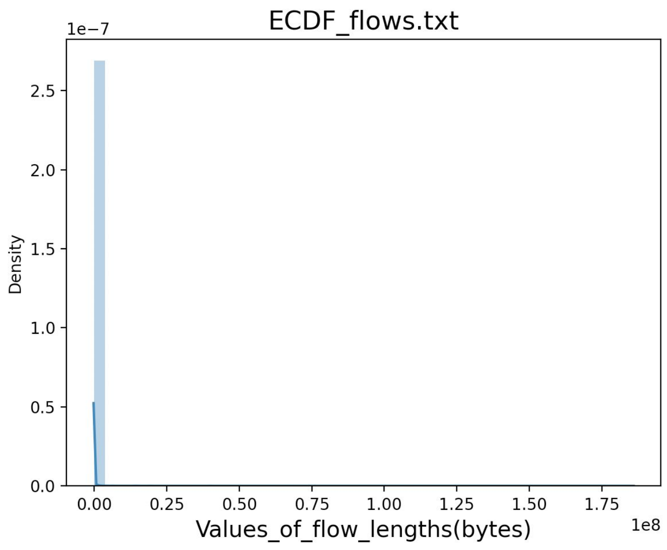
6. Code and result of box plot for flows.txt using logarithmic values

```
import math
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
data =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-4/r-data/flows.txt',names=["A"]
, header=None)
x=[]
for index, row in data.iterrows():
  x.append(math.log2(int(row["A"])))
plt.figure()
plt.figure().add_subplot(111).yaxis.set_major_locator(ticker.MultipleLocator(5))
plt.title("Boxplot_flows.txt",fontsize=16)
plt.xlabel("Flows",fontsize=14)
plt.ylabel("Values_of_flow_lengths(bytes, log2)",fontsize=14)
plt.boxplot(x, showmeans=True, notch = True, sym = '*')
plt.show()
```



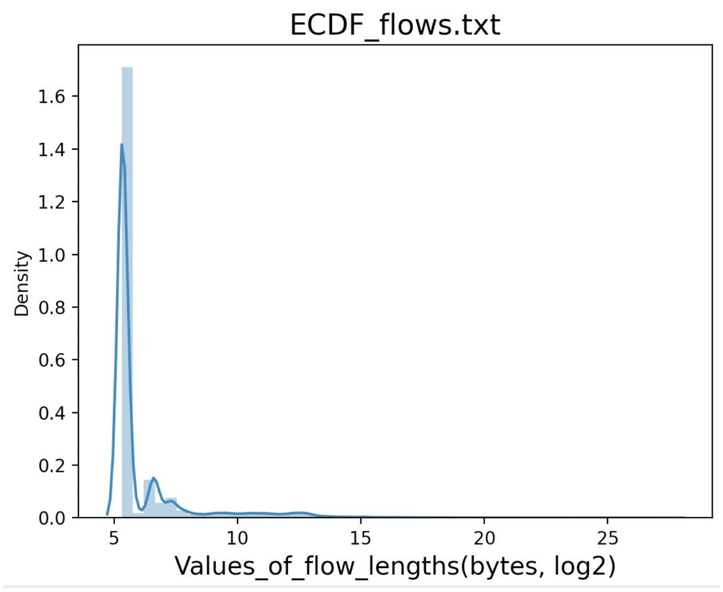
7. Code and result of Empirical CDF plot for flows.txt

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
data =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-4/r-data/flows.txt',names=["A"]
, header=None)
x=[]
for index, row in data.iterrows():
   x.append(row["A"])
plt.title("ECDF_flows.txt",fontsize=16)
plt.xlabel("Values_of_flow_lengths(bytes)",fontsize=14)
sns.distplot(x)
plt.show()
```



8. Code and result of Empirical CDF plot for flows.txt using logarithmic values

```
import math
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
data =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-4/r-data/flows.txt',names=["A"]
, header=None)
x=[]
for index, row in data.iterrows():
   x.append(math.log2(int(row["A"])))
plt.title("ECDF_flows.txt",fontsize=16)
plt.xlabel("Values_of_flow_lengths(bytes, log2)",fontsize=14)
sns.distplot(x)
plt.show()
```



9. Conclusions based on the flow information, what are the best methods to describe the data?

After comparing the 8 plots above generated by different plotting methods and variables, I think empirical CDF plot using logarithmic values is the best to describe the data in flows.txt. First of all, all methods including scatter plot, histogram plot, box plot and empirical CDF without using logarithmic values are bad ways to describe the data in flows.txt, cause the data is too scattered ranging from 40 to 185758396, which means, using logarithmic values will reduce the scope of the data.

```
yanjing@yanjingdeMacBook-Pro r-data % sort -n flows.txt | head -n1
40
yanjing@yanjingdeMacBook-Pro r-data % sort -n flows.txt | tail -n1
185758396
```

Secondly, comparing the four methods including scatter plot, histogram plot, box plot and empirical CDF using logarithmic values, we can get the following conclusions, for the scatter plot using logarithmic values, all the scattering points are in x-axis, we can not even see the density; for the box plot using logarithmic values, the minimum, first quartile, median, third quartile, and maximum value makes no sense here cause outliers are too many; for the histogram plot using logarithmic values, I think this method is the second best way to describe the data, but still, the frequencies for the data differs a lot, which causes the plot hard to know the details; for the empirical CDF using logarithmic values, I do think it is the best way, the density showed in y-axis is the values after some process of the frequency showed in y-axis in the histogram plot using logarithmic values in some way, which makes it easier to see the detailed differences.

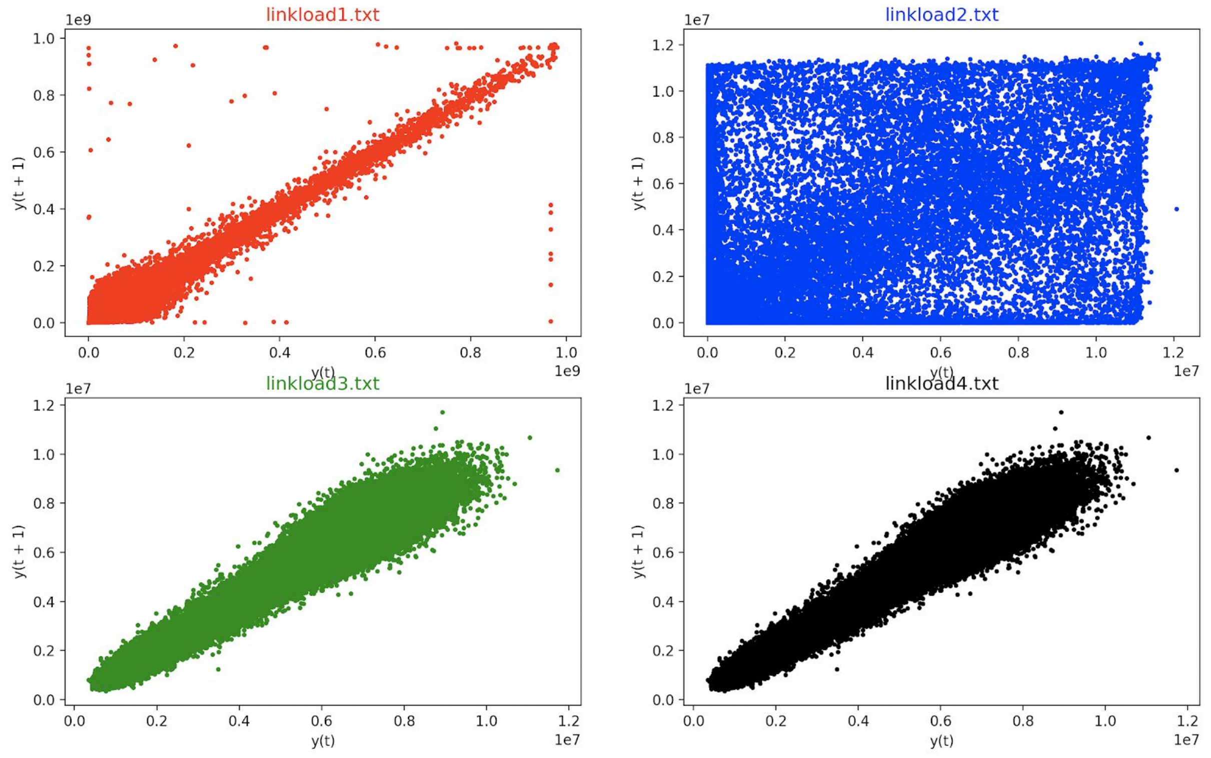
Report, task 3

1. Plots according to instructions

1-1. Code and result of lag plot for each link

Note: before the code, I replaced the “blank space” in each txt file with “,” and modified some of them to csv files (cause there is no column title and there are more than one columns).

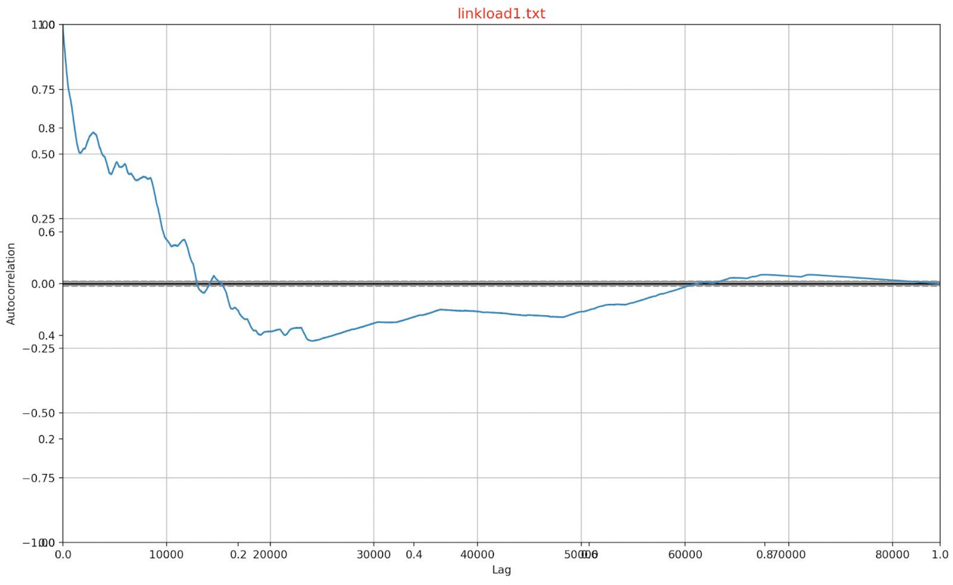
```
import pandas as pd
import matplotlib.pyplot as plt
from pandas.plotting import lag_plot
linkload1 =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-4/r-data/linkload-1.csv',
names=['A','B'], header=None)
linkload2 =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-4/r-data/linkload-2.txt',
names=['A'], header=None)
linkload3 =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-4/r-data/linkload-3.csv',
names=['A','B'], header=None)
linkload4 =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-4/r-data/linkload-4.csv',
names=['A','B'], header=None)
plt.figure(1)
plt.subplot(221)
plt.title("linkload1.txt",fontsize=13,c="red")
lag_plot(linkload1.B,lag=1,s=5,c='red')
plt.subplot(222)
plt.title("linkload2.txt",fontsize=13,c="blue")
lag_plot(linkload2.A,lag=1,s=5,c='blue')
plt.subplot(223)
plt.title("linkload3.txt",fontsize=13,c="green")
lag_plot(linkload3.B,lag=1,s=5,c='green')
plt.subplot(224)
plt.title("linkload4.txt",fontsize=13,c="black")
lag_plot(linkload3.B,lag=1,s=5,c='black')
plt.show()
```



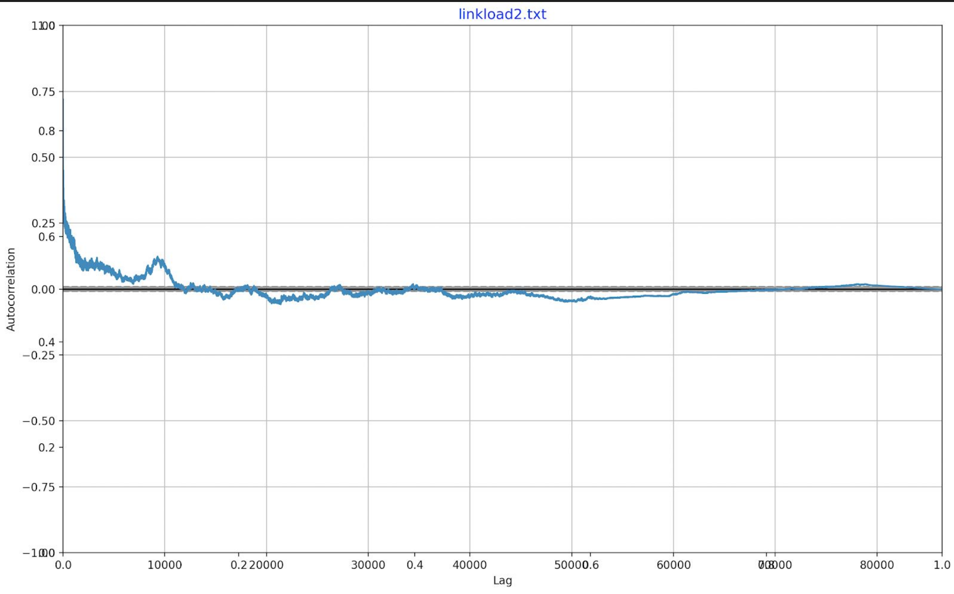
1-2. Code and result of autocorrelation plot for each link

```
import pandas as pd
import matplotlib.pyplot as plt
from pandas.plotting import autocorrelation_plot
linkload1 =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-4/r-data/linkload-1.csv',
names=['A','B'], header=None)
plt.title("linkload1.txt",fontsize=13,c="red")
```

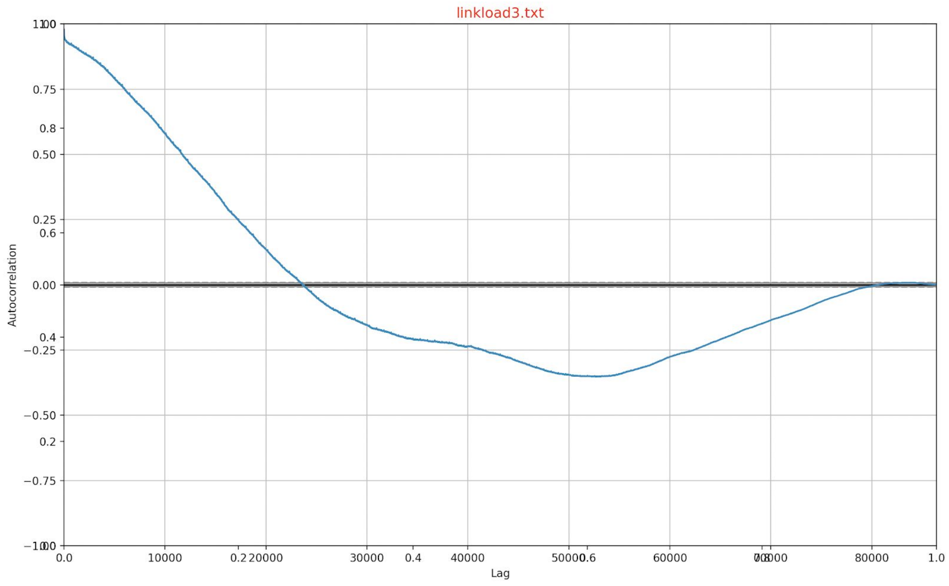
```
autocorrelation_plot(linkload1.B)
plt.show()
```



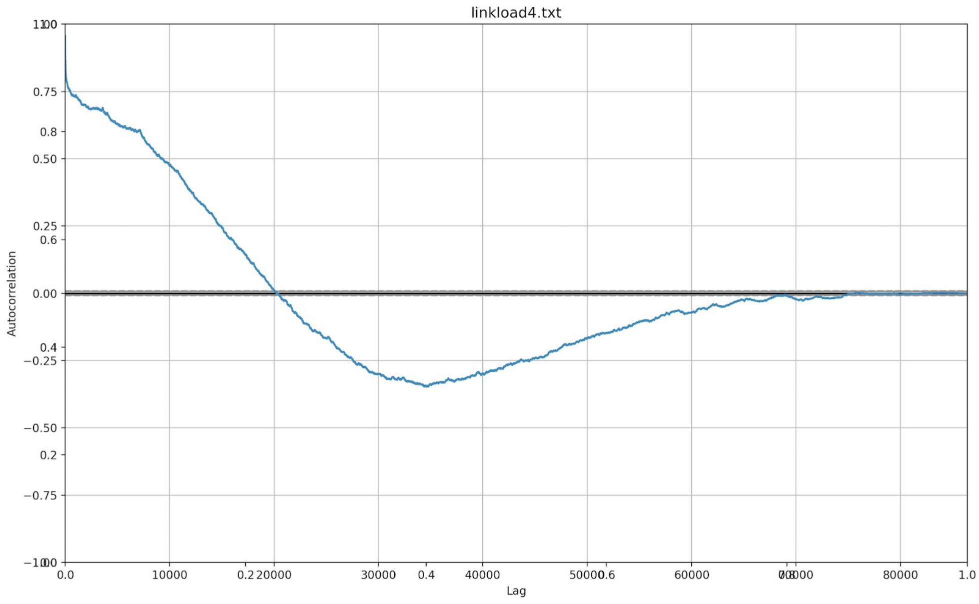
```
import pandas as pd
import matplotlib.pyplot as plt
from pandas.plotting import autocorrelation_plot
linkload2 =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-4/r-data/linkload-2.txt',
names=['A'], header=None)
plt.title("linkload2.txt",fontsize=13,c="blue")
autocorrelation_plot(linkload2.A)
plt.show()
```



```
import pandas as pd
import matplotlib.pyplot as plt
from pandas.plotting import autocorrelation_plot
linkload3 =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-4/r-data/linkload-3.csv',
names=['A','B'], header=None)
plt.title("linkload3.txt",fontsize=13,c="red")
autocorrelation_plot(linkload3.B)
plt.show()
```



```
import pandas as pd
import matplotlib.pyplot as plt
from pandas.plotting import autocorrelation_plot
linkload4 =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-4/r-data/linkload-4.csv',
names=['A','B'], header=None)
plt.title("linkload4.txt",fontsize=13,c="black")
autocorrelation_plot(linkload4.B)
plt.show()
```



2. Data inspection results and conclusions of each data set

From the five pictures above, we can get identical conclusions from both lag plot and autocorrelation plot.
For lag plot, more points tighter into the diagonal line (the function y=x) suggests a stronger relationship and more spread from the line suggests a weaker relationship. So we can see the picture shown in 1-1. Code and result of lag plot for each link and get the following info:

- There are some outliers spread from the diagonal line in the picture generated based on “linkload-1.txt” and lag plot, but basically, most of the data suggests a strong relationship.
- For the picture generated based on “linkload-2.txt” and lag plot, we can not see the feature that most points are tighter into the diagonal line, just see all points spread across the whole picture, this is to say, a very weak relationship among them.
- As we said above, the pictures generated based on “linkload-3.txt” and “linkload-4.txt” with lag plot show a much stronger relationship among the data inside as almost all points are tighter into the diagonal line.

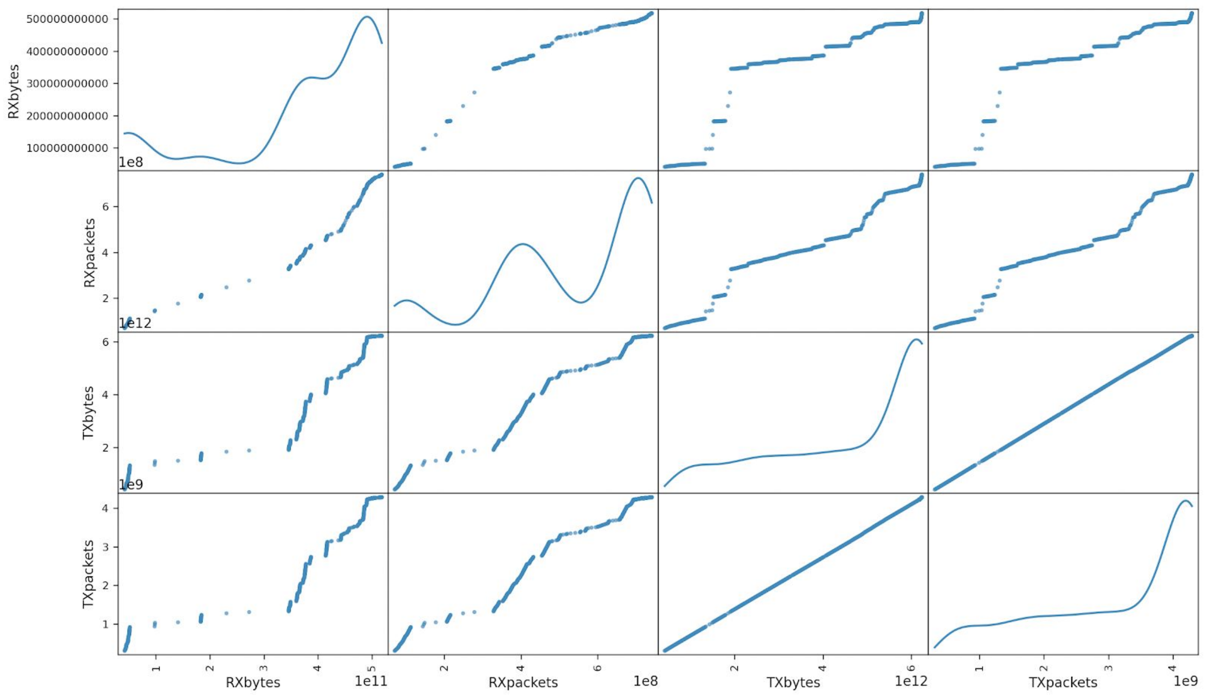
For autocorrelation plot, a correlation value calculated between the observations and their lag1 values will result in a number between -1 and 1. The sign of this number indicates a negative or positive correlation respectively. A value close to zero suggests a weak correlation, whereas a value closer to -1 or 1 indicates a strong correlation. So we can see the pictures shown in 1-2. Code and result of autocorrelation plot for each link and get the following info:

- So for the picture generated by “linkload-1.txt” and autocorrelation plot, almost half values of the curve are very close to 0 and other half values are close to 1, which shows positive correlation among some data, and also negative correlation among the rest data (This is corresponding to the outliers showed in lag plot).
- For the picture generated by “linkload-2.txt” and autocorrelation plot, almost all the values are very close to 0, this is to say, negative correlation among all data, which is also the same with the result shown in lag plot.
- For the pictures generated by “linkload-3.txt” and “linkload-4.txt” with autocorrelation plot, almost all values are closer to 1 rather than 0, which means the data in “linkload-3.txt” and “linkload-4.txt” showing positive correlation. (This is also the same conclusion with the lag plot.)

Report, task 4

1. Code and result of pairs plot (scatter matrix) for bytes.csv

```
import pandas as pd
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
data = pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-4/r-data/bytes.csv')
scatter_matrix(data, alpha=0.7, figsize=(14,8), diagonal='kde')
plt.show()
```



2. Analysis for the result of pairs plot (scatter matrix) based on bytes.csv

From the picture generated based on bytes.csv and scatter matrix method, we can get the following info:

- The variable “TXpackets” correlates most to the variable “TXbytes”
- For the other variables, they all correlate to each other more or less, for example, when “RXbytes” increases, the “TXpackets”, “TXbytes” and “RXpackets” all increase more or less. The conclusion I can see from the picture really matches a user’s Internet habit, most packets generated while a user is surfing the Internet are TCP, which is sending-receiving mode. But basically, the packets received may be possibly more than the packets transmitted.
- “TXbytes” and “RXbytes” are the least informative columns, cause the info in “TXbytes” can be totally seen through “TXpackets” and the info in “RXbytes” can be seen through “RXpackets”.

Report, task 5

1. Translate flowdata_unclean.txt to flowdata_unclean.csv

Replace all “\t” to “,” and we can see the flowdata_unclean.csv as follows:
Note: just display some of the flowdata_unclean.csv, more details can be seen through AS4.zip file in the final submission

flowdata_unclean

src	dst	proto	ok	sport	dport	pkts	bytes	flows	first	latest
203.246.146.19	106.22.48.22	1.0	1.0	8.0	0.0	1.0	32.0	1.0	1491969542.5880100	1491969542.5880100
203.246.146.19	218.201.19.92	1.0	1.0	8.0	0.0	1.0		1.0		
176.52.159.166	203.246.146.19	1.0	1.0	0.0	0.0	1.0	32.0	1.0	1491969545.8431500	1491969545.8431500
203.246.146.19	5.57.5.49	1.0	1.0	8.0	0.0	1.0	32.0	1.0	1491969545.8646800	1491969545.8646800
203.246.146.19	217.226.102.29	1.0	1.0	8.0	0.0	1.0	32.0	1.0	1491969549.900830	
101.156.145.60	133.60.159.199	6.0	1.0	1214.0	445.0	2.0	96.0	1.0	1491969557.269190	1491969560.227220
203.246.146.19	71.32.111.81	1.0	1.0	8.0	0.0	1.0	32.0	1.0	1491969562.070120	1491969562.070120
203.246.146.19	77.205.87.213	1.0	1.0	8.0	0.0	1.0	32.0	1.0	1491969563.7858200	1491969563.7858200

2. Delete rows in flowdata_unclean.csv with NaN label

Code:

```
import pandas as pd
df = pd.read_csv(r"/Users/yanjing/Desktop/ITMA/Assignment-4/flowdata_unclean.csv",
na_values='NULL', encoding='utf-8')
df.dropna().to_csv('/Users/yanjing/Desktop/ITMA/Assignment-4/flowdata_clean.csv',
encoding='utf-8')
```

Result:

Note: just display some of the flowdata_clean.csv, more details can be seen through AS4.zip file in the final submission

flowdata_clean

	src	dst	proto	ok	sport	dport	pkts	bytes	flows	first	latest
0	203.246.146.19	106.22.48.22	1.0	1.0	8.0	0.0	1.0	32.0	1.0	1491969542.5880100	1491969542.5880100
2	176.52.159.166	203.246.146.19	1.0	1.0	0.0	0.0	1.0	32.0	1.0	1491969545.8431500	1491969545.8431500
3	203.246.146.19	5.57.5.49	1.0	1.0	8.0	0.0	1.0	32.0	1.0	1491969545.864680	1491969545.864680
5	101.156.145.60	133.60.159.199	6.0	1.0	1214.0	445.0	2.0	96.0	1.0	1491969557.269190	1491969560.227220
6	203.246.146.19	71.32.111.81	1.0	1.0	8.0	0.0	1.0	32.0	1.0	1491969562.070120	1491969562.070120
7	203.246.146.19	77.205.87.213	1.0	1.0	8.0	0.0	1.0	32.0	1.0	1491969563.7858200	1491969563.7858200
8	203.246.146.19	73.11.78.246	1.0	1.0	8.0	0.0	1.0	32.0	1.0	1491969564.7227800	1491969564.7227800
9	203.246.146.19	62.202.235.183	1.0	1.0	8.0	0.0	1.0	32.0	1.0	1491969566.5572500	1491969566.5572500

3. Add an additional column according to the values of bytes column in flowdata_clean.csv

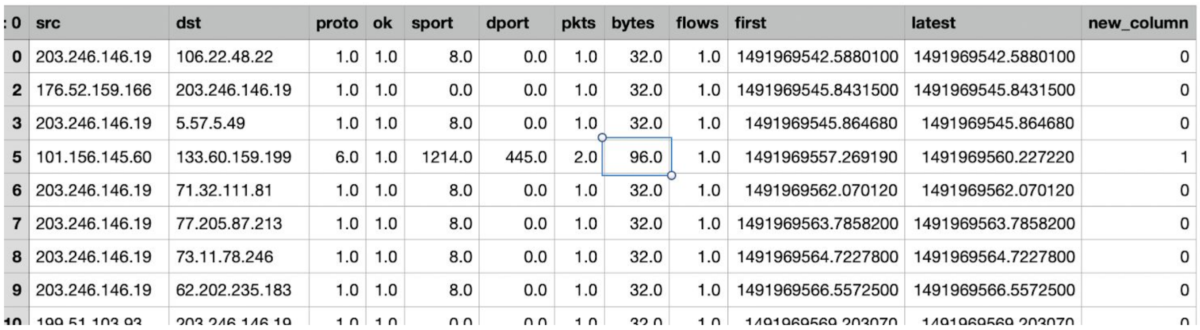
Code:

```
import pandas as pd
df1 = pd.read_csv(r"/Users/yanjing/Desktop/ITMA/Assignment-4/flowdata_clean.csv")
x = []
for i in df1["bytes"]:
   if i>=50:
       x.append(1)
   else:
       x.append(0)
data = pd.DataFrame(x)
df1['new_column'] = data
df1.to_csv(r"flowdata_clean_new.csv",mode = 'a',index =False)
```

Result:

Note: just display some of the flowdata_clean_new.csv, more details can be seen through AS4.zip file in the final submission

flowdata_clean_new



: 0	src	dst	proto	ok	sport	dport	pkts	bytes	flows	first	latest	new_column
0	203.246.146.19	106.22.48.22	1.0	1.0	8.0	0.0	1.0	32.0	1.0	1491969542.5880100	1491969542.5880100	0
2	176.52.159.166	203.246.146.19	1.0	1.0	0.0	0.0	1.0	32.0	1.0	1491969545.8431500	1491969545.8431500	0
3	203.246.146.19	5.57.5.49	1.0	1.0	8.0	0.0	1.0	32.0	1.0	1491969545.864680	1491969545.864680	0
5	101.156.145.60	133.60.159.199	6.0	1.0	1214.0	445.0	2.0	96.0	1.0	1491969557.269190	1491969560.227220	1
6	203.246.146.19	71.32.111.81	1.0	1.0	8.0	0.0	1.0	32.0	1.0	1491969562.070120	1491969562.070120	0
7	203.246.146.19	77.205.87.213	1.0	1.0	8.0	0.0	1.0	32.0	1.0	1491969563.7858200	1491969563.7858200	0
8	203.246.146.19	73.11.78.246	1.0	1.0	8.0	0.0	1.0	32.0	1.0	1491969564.7227800	1491969564.7227800	0
9	203.246.146.19	62.202.235.183	1.0	1.0	8.0	0.0	1.0	32.0	1.0	1491969566.5572500	1491969566.5572500	0
10	199.51.103.93	203.246.146.19	1.0	1.0	0.0	0.0	1.0	32.0	1.0	1491969569.203070	1491969569.203070	0

Report, task 6

1. Is there any trend or seasonality?

There is no a trend or seasonality (I mean RTT does not increase or decrease when the time is increasing), most of the time, the RTT fluctuates around 200ms; occasionally, it soars a lot or falls some, which is really matching the real internet environment, most of the time, network is stable, occasionally, network may be unstable, resulting in large RTT or packet loss, as well as faster network speed.

2. Is the time series stationary?

The time series is stationary, as I replied in the last question, there is no direct relationship between time and RTT. We can calculate the statistics like mean or the variance of the RTT, basically, they are consistent over time as the network is stable most of the time.