ELEC-E7130 - Internet Traffic Measurements and Analysis

Assignment 4 Analyzing data using R or python

Haibi Peng 875552

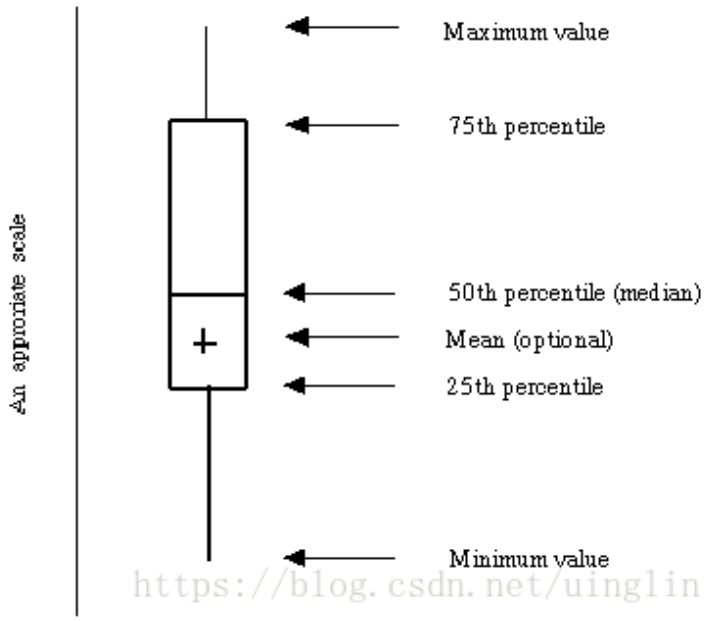
Task 1: Understanding different plots

Solution:

1. Discussion on different plots.

I. Boxplot

A box plot shows the distribution of quantitative data in a way that facilitates comparisons between variables or across levels of a categorical variable. The box shows the quartiles of the dataset while the whiskers extend to show the rest of the distribution, except for points that are determined to be "outliers" using a method that is a function of the inter-quartile range. A boxplot is a standardized way of displaying the dataset based on a five-number summary: the minimum, the maximum, the sample median, and the first and third quartiles.

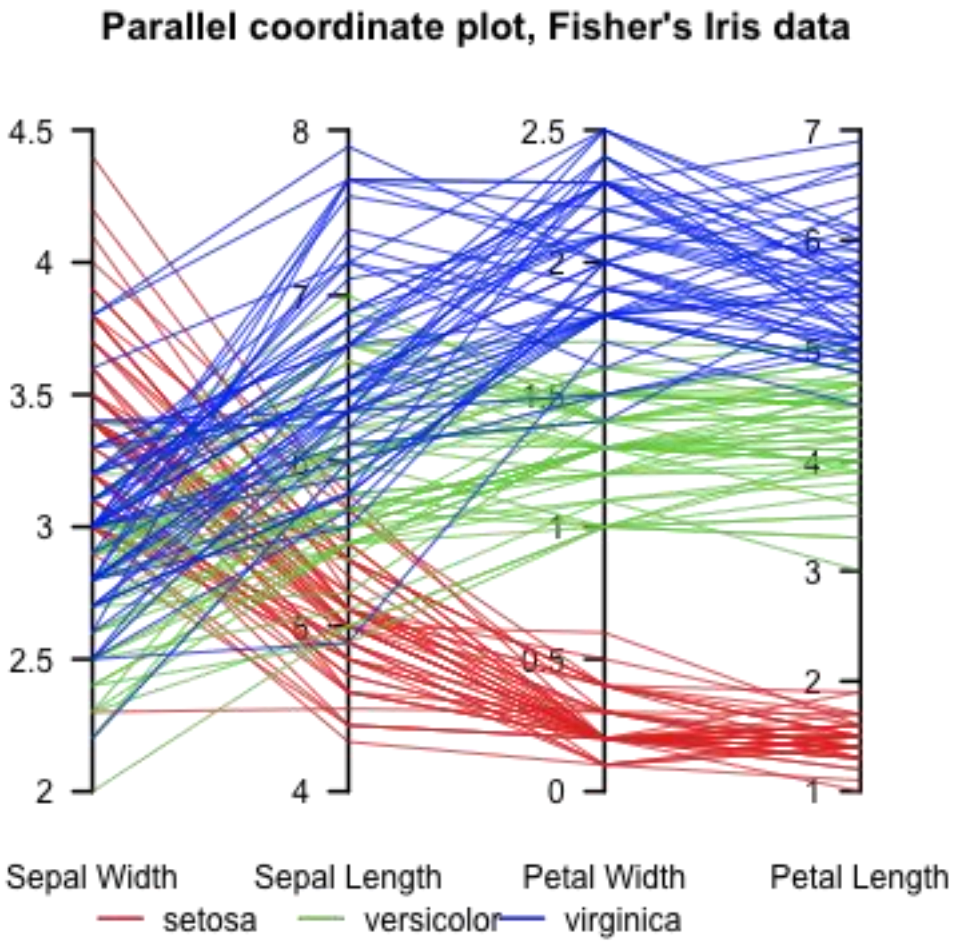


Source: https://blog.csdn.net/uinglin/article/details/79895993

II. Parallel plot

Parallel coordinates are a common way of visualizing and analyzing high-dimensional datasets. Parallel plot or parallel coordinates plot allows to compare the feature of several individual observations (series) on a set of numeric variables. Each vertical bar represents a variable and often has its own scale. (The units can even be different). Values are then plotted as series of lines connected across each axis. A parallel plot allows to study the features of samples for several

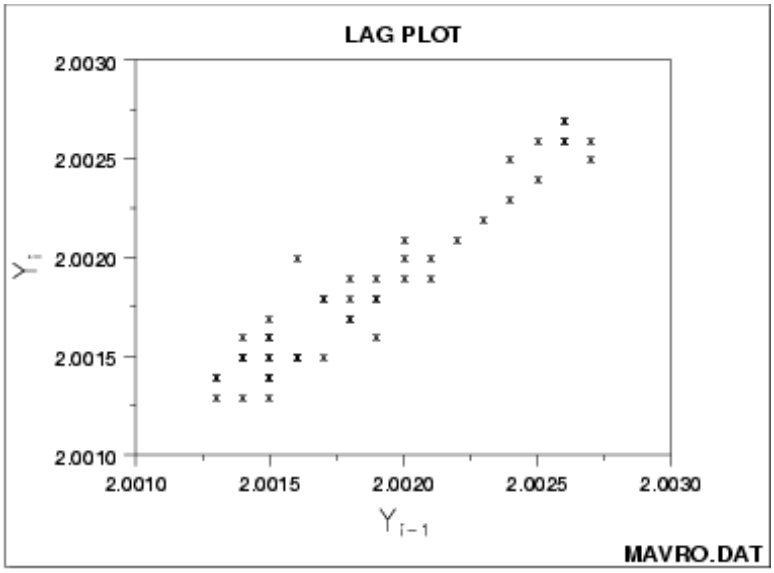
quantitative variables. Its strength is that the variables can even be completely different: different ranges and even different units.



Source: https://en.wikipedia.org/wiki/Parallel_coordinates

III. Lag plot

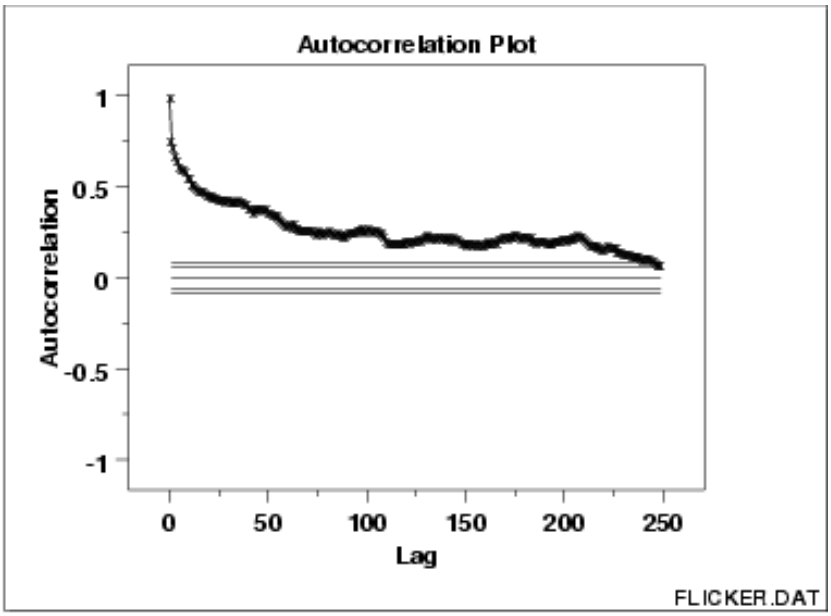
A lag plot is a simplistic and non-statistical approach for analyzing the relationship between a series and its lags, with the use of two-dimensional scatter plots for visualizing the series (typically on the y-axis) against the k lag of the series. Hence, each pair of points represents a combination of the series observations and their corresponding lagged values. A lag plot checks whether a data set or time series is random or not. As more points on the lag plot are closer to the 45° line, the higher the correlation will be between the series and the corresponding lag. Its y-axis/x-axis represent Yi and Yi-1



Source: https://www.itl.nist.gov/div898/handbook/eda/section3/lagplot.htm

IV. Autocorrelation plot

Autocorrelation plot is a commonly-used tool for checking randomness in a data set. This randomness is ascertained by computing autocorrelations for data values at varying time lags. If random, such autocorrelations should be near zero for any and all time-lag separations. If non-random, then one or more of the autocorrelations will be significantly non-zero. Its y-axis/x-axis represent autocorrelation coefficient and time lag.



Source: https://www.itl.nist.gov/div898/handbook/eda/section3/autocopl.htm

Task 2: Plot data

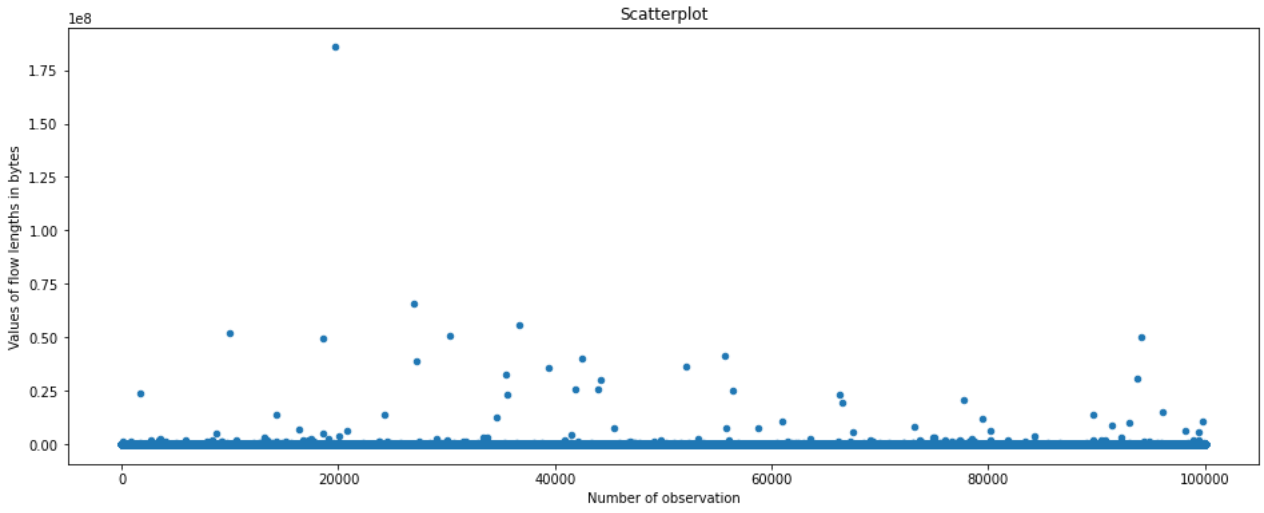
Solution:

1. Different plots of flows.txt

```
1.  #get data
2.  flowsdata=[]
3.  with open("/Users/jdszsl/Desktop/Aalto/Aalto-Communications Engineering/ELEC-
    E7130 - Internet Traffic Measurements and Analysis/Assignments/Assignment4/data/flows.txt",
     "r") as f:  #open file
4.      for line in f.readlines():
5.          line = line.strip('\n') #get rid of '\n'
6.          flowsdata.append(int(line)) #str to int
```

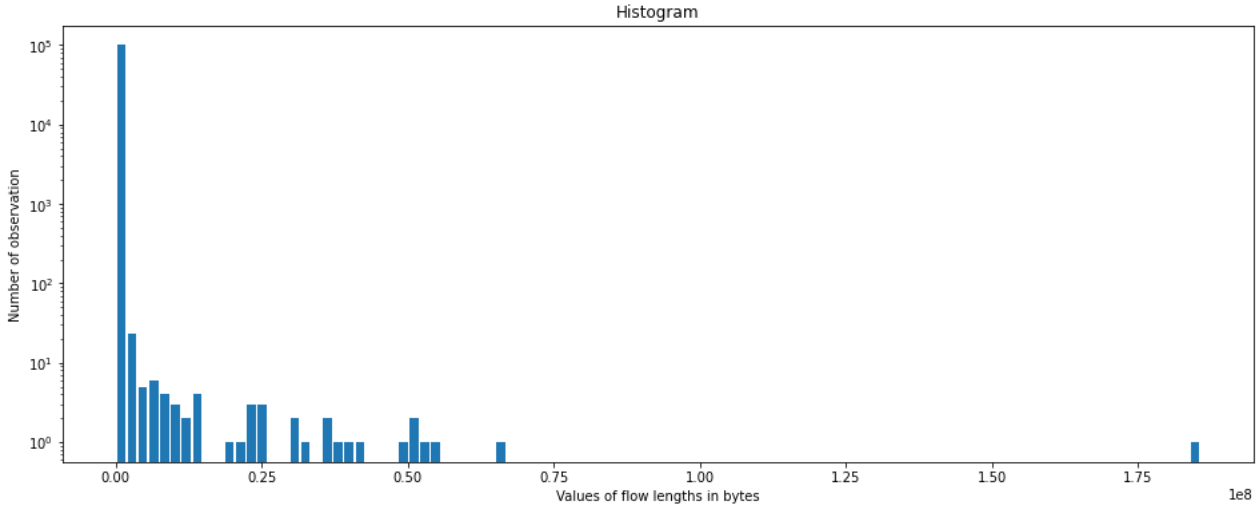
I. Scatterplot (Number of observation will reside on X axis)

```
1.  #Scatterplot
2.  xValue = list(range(1, len(flowsdata)+1))
3.  yValue = [x for x in flowsdata]
4.
5.  plt.figure(figsize=(16,6))
6.  plt.title('Scatterplot')
7.  plt.xlabel('Number of observation')
8.  plt.ylabel('Values of flow lengths in bytes')
9.
10. plt.scatter(xValue, yValue, s=20, marker='o')
11. plt.show()
```



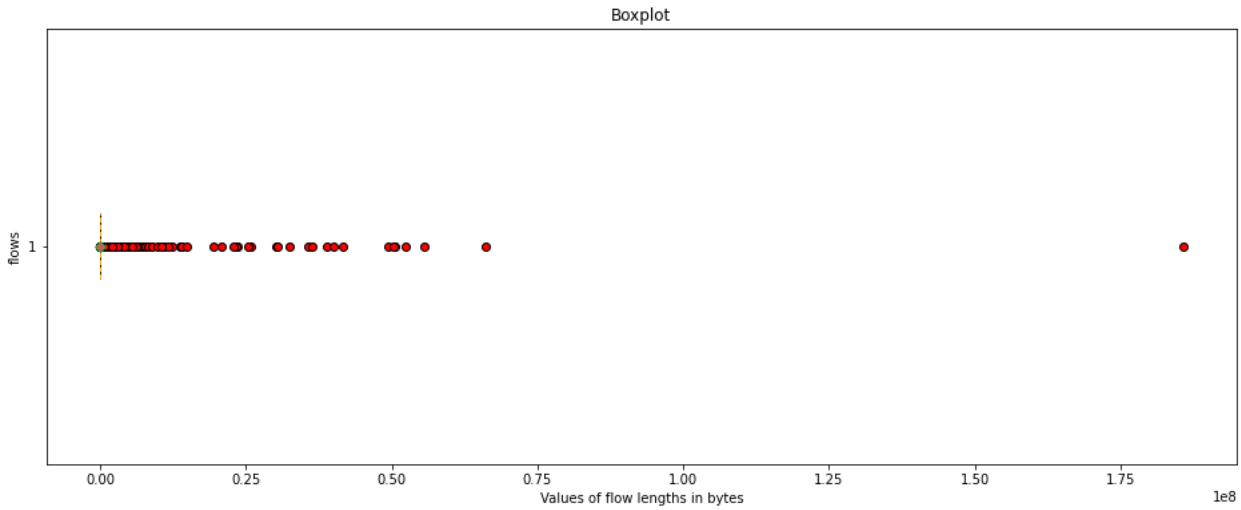
II. Histogram (Using suitable number of bins)

```
1.  #Histogram
2.  plt.figure(figsize=(16,6))
3.  plt.title('Histogram')
4.  plt.xlabel('Values of flow lengths in bytes')
5.  plt.ylabel('Number of observation')
6.  plt.hist([x for x in flowsdata], bins=100, log=1, histtype='bar', rwidth=0.8)
```



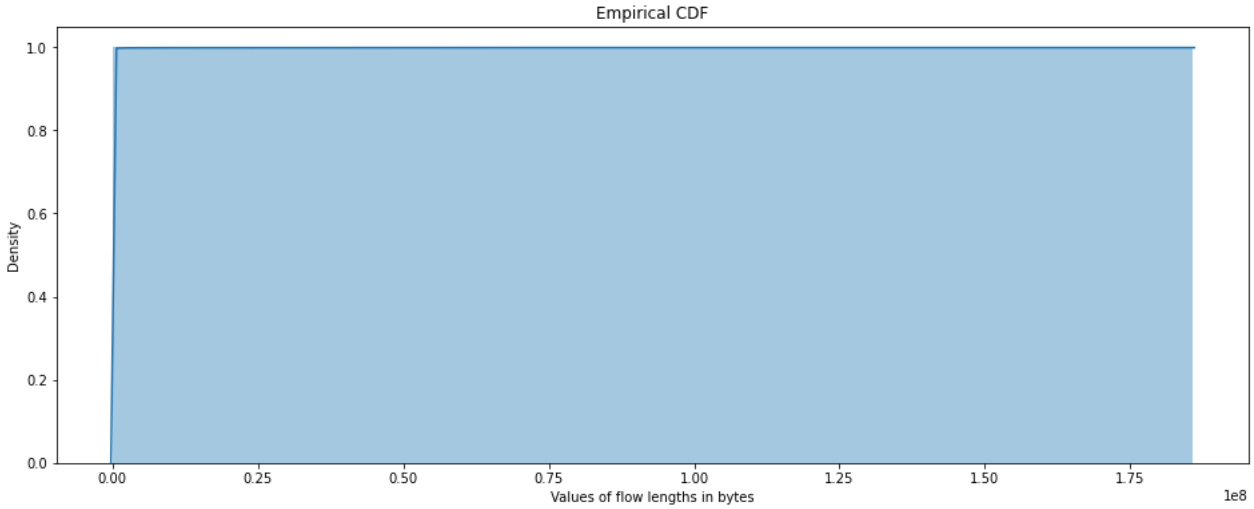
III. Boxplot

```
1.  #Boxplot
2.  plt.figure(figsize=(16,6))
3.  plt.title('Boxplot')
4.  plt.xlabel('Values of flow lengths in bytes')
5.  plt.ylabel('flows')
6.
7.  plt.boxplot(x = [x for x in flowsdata],
8.              patch_artist=True,
9.              showmeans=True,
10.             showfliers=True,
11.             vert=False,
12.             boxprops = {'color':'black','facecolor':'#9999ff'},
13.             flierprops = {'marker':'o','markerfacecolor':'red','color':'black'},
14.             meanprops = {'marker':'D','markerfacecolor':'indianred'},
15.             medianprops = {'linestyle':'--','color':'orange'})
16. plt.show()
```



IV. Empirical CDF of the variable

```
#Empirical CDF
plt.figure(figsize=(16,6))
x = [x for x in flowsdata]
kwargs = {'cumulative': True, 'density': True}
sns.distplot(x, hist_kws=kwargs, kde_kws={'cumulative': True})
plt.title('Empirical CDF')
plt.xlabel('Values of flow lengths in bytes')
```



2. Summarize distribution using different number of variables

```
# calculate quartiles
quartiles = percentile(flowsdata, [25, 50, 75])
# calculate min/max/mean
data_min, data_max, data_mean = min(flowsdata), max(flowsdata), np.mean(flowsdata)
# print 5-number summary
print('Min: %.3f' % data_min)
print('Q1: %.3f' % quartiles[0])
print('Median: %.3f' % quartiles[1])
print('Q3: %.3f' % quartiles[2])
print('Max: %.3f' % data_max)
print('Mean: %.3f' % data_mean)
```

The results are as follows:

```
Min: 40.000
Q1: 40.000
Median: 40.000
Q3: 52.000
Max: 185758396.000
Mean: 14545.657
```

I. One number

Mean: 14546

II. Two numbers

Q1: 40

Q3: 52

III. Three numbers

Min: 40

Median: 40

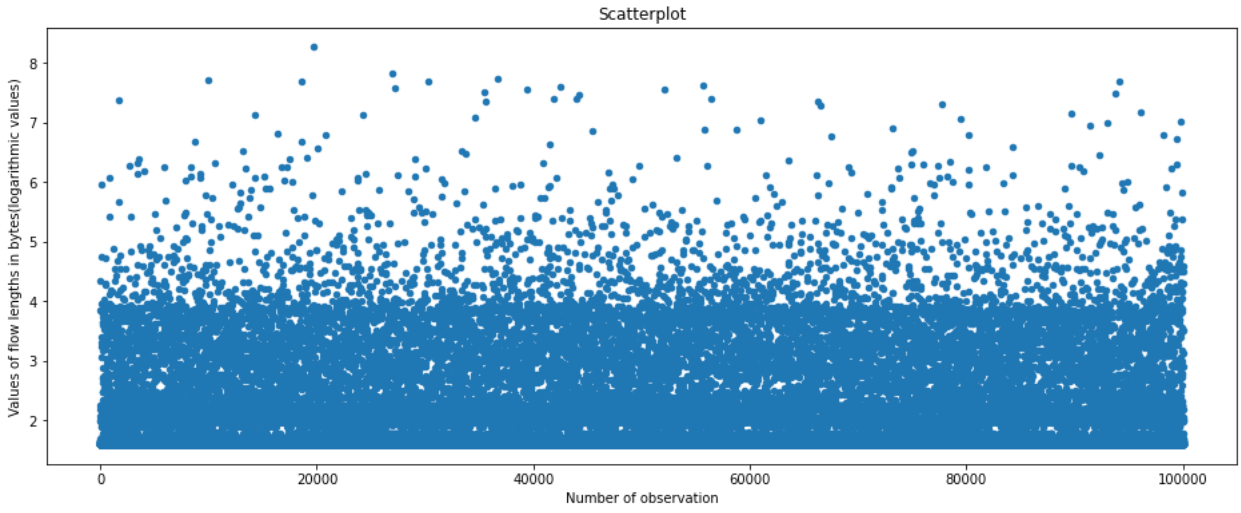
Max: 185758396

3. Different plots of flows.txt using logarithmic values

I choose to use log 10 to calculate the logarithmic values.

I. Scatterplot (Number of observation will reside on X axis)

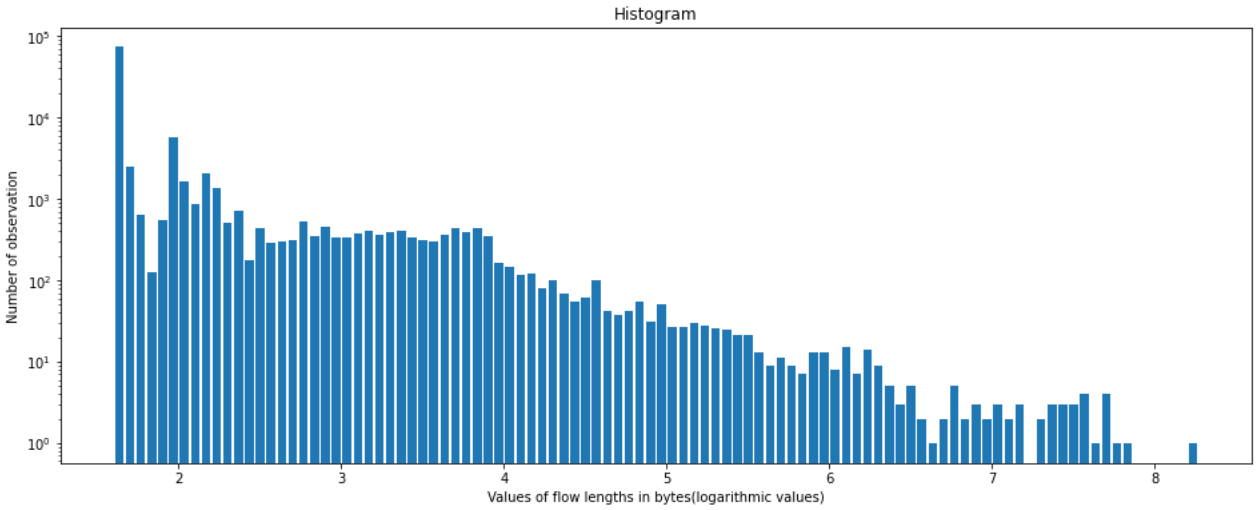
```
#Scatterplot
xValue = list(range(1, len(flowsdata)+1))
yValue = [math.log(x,10) for x in flowsdata]

plt.figure(figsize=(16,6))
plt.title('Scatterplot')
plt.xlabel('Number of observation')
plt.ylabel('Values of flow lengths in bytes(logarithmic values)')

plt.scatter(xValue, yValue, s=20, marker='o')
plt.show()
```



II. Histogram (Using suitable number of bins)

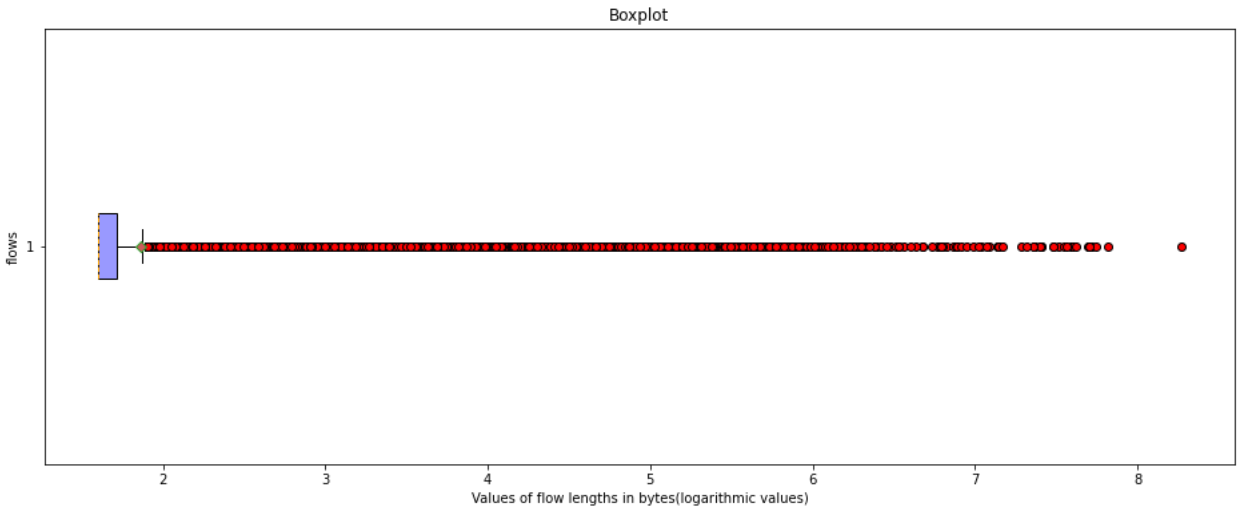
```
#Histogram
plt.figure(figsize=(16,6))
plt.title('Histogram')
```

```
4.  plt.xlabel('Values of flow lengths in bytes(logarithmic values)')
5.  plt.ylabel('Number of observation')
6.  plt.hist([math.log(x,10) for x in flowsdata], bins=100, log=1, histtype='bar', rwidth=0.8)
```



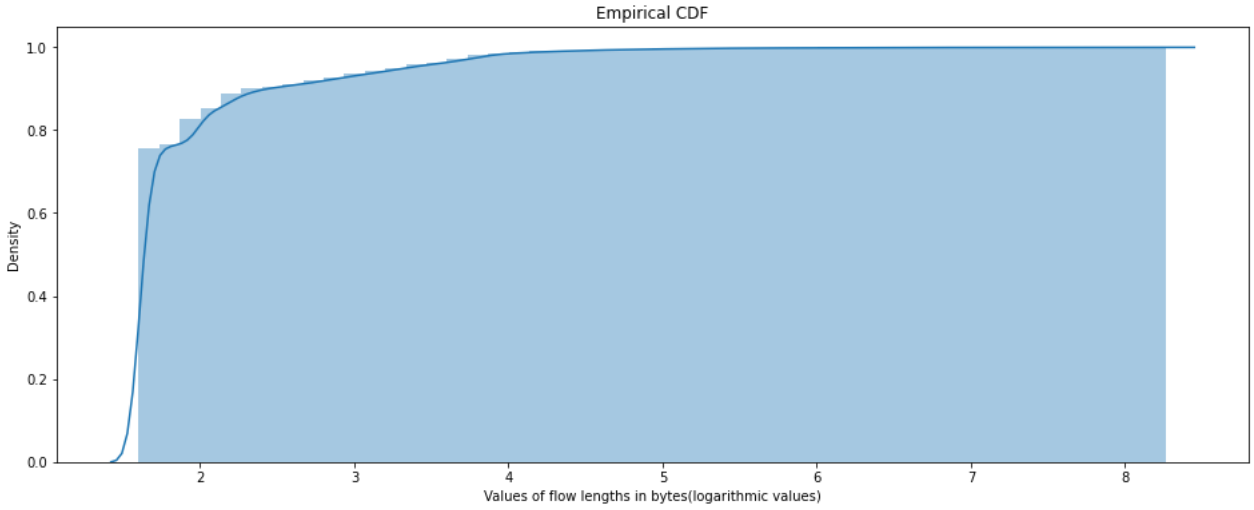
III. Boxplot

```
1.  #Boxplot
2.  plt.figure(figsize=(16,6))
3.  plt.title('Boxplot')
4.  plt.xlabel('Values of flow lengths in bytes(logarithmic values)')
5.  plt.ylabel('flows')
6.  plt.boxplot(x = [math.log(x,10) for x in flowsdata],
7.              patch_artist=True,
8.              showmeans=True,
9.              showfliers=True,
10.             vert=False,
11.             boxprops = {'color':'black','facecolor':'#9999ff'},
12.             flierprops = {'marker':'o','markerfacecolor':'red','color':'black'},
13.             meanprops = {'marker':'D','markerfacecolor':'indianred'},
14.             medianprops = {'linestyle':'--','color':'orange'})
15. plt.show()
```



IV. Empirical CDF of the variable

```
#Empirical CDF
plt.figure(figsize=(16,6))
x = [math.log(x,10) for x in flowsdata]
kwargs = {'cumulative': True, 'density': True}
sns.distplot(x, hist_kws=kwargs, kde_kws={'cumulative': True})
plt.title('Empirical CDF')
plt.xlabel('Values of flow lengths in bytes(logarithmic values)')
```



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

Based on the plotted flow information, nearly 80% of the flows have less than 100 bytes and nearly 10% of the flows have 100-1000 bytes, only around 10% of the flows have more than 1000 bytes. Basically, using logarithmic values are more suitable for describing the data since in that case the data plots are more apparent and it is easier to find the distribution and patterns of the data.

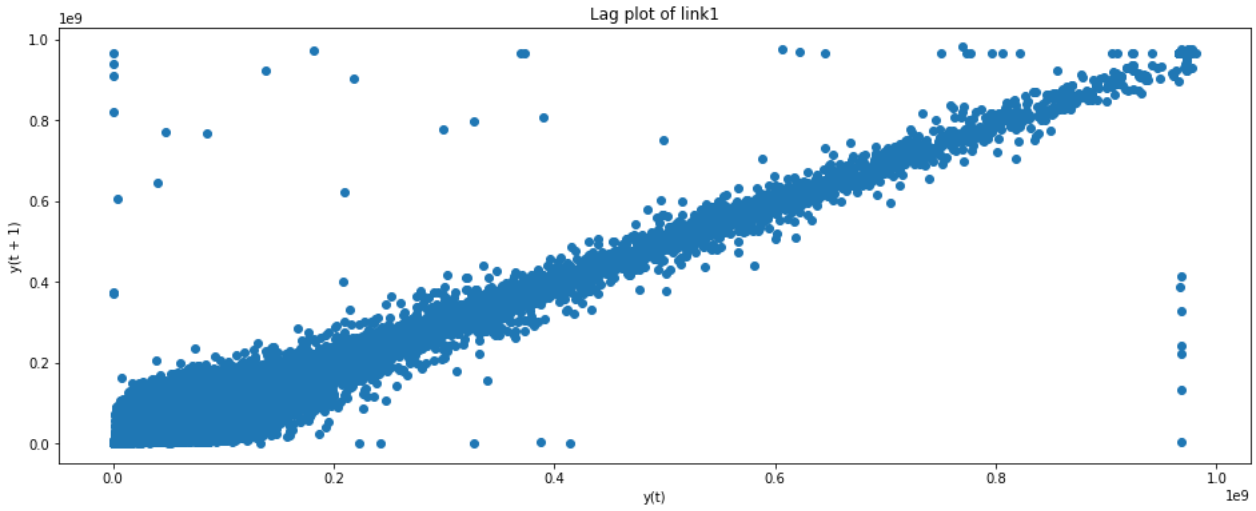
Task 3: Link loads

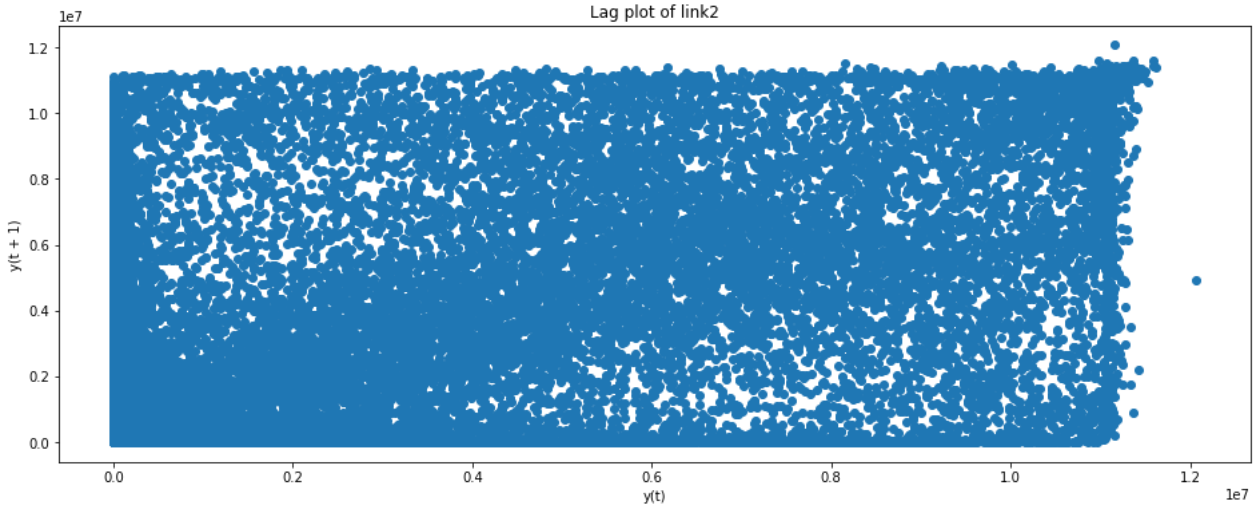
Solution:

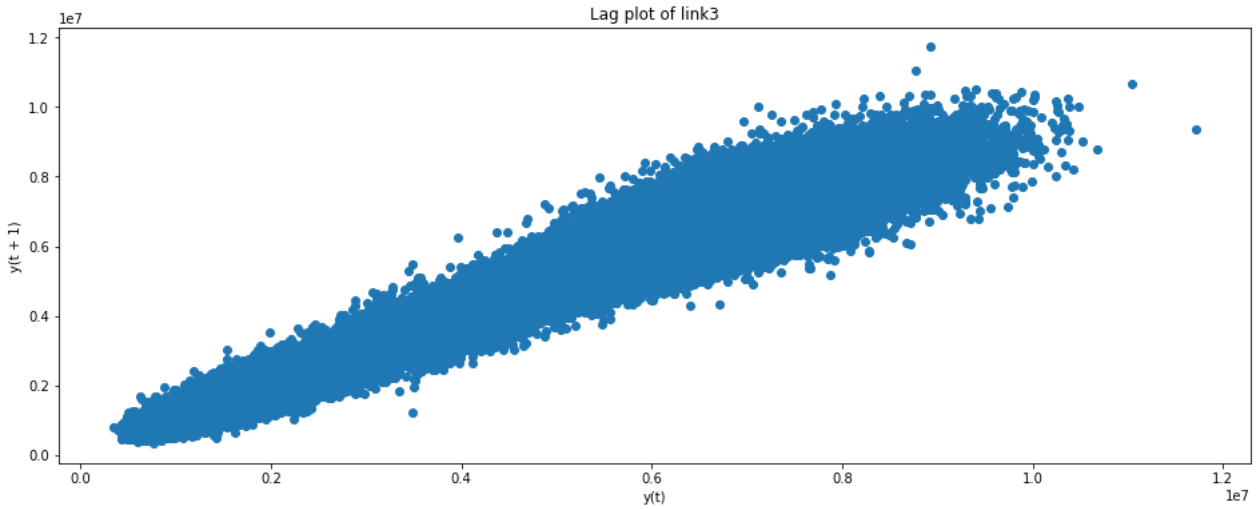
1. Plots according to instructions

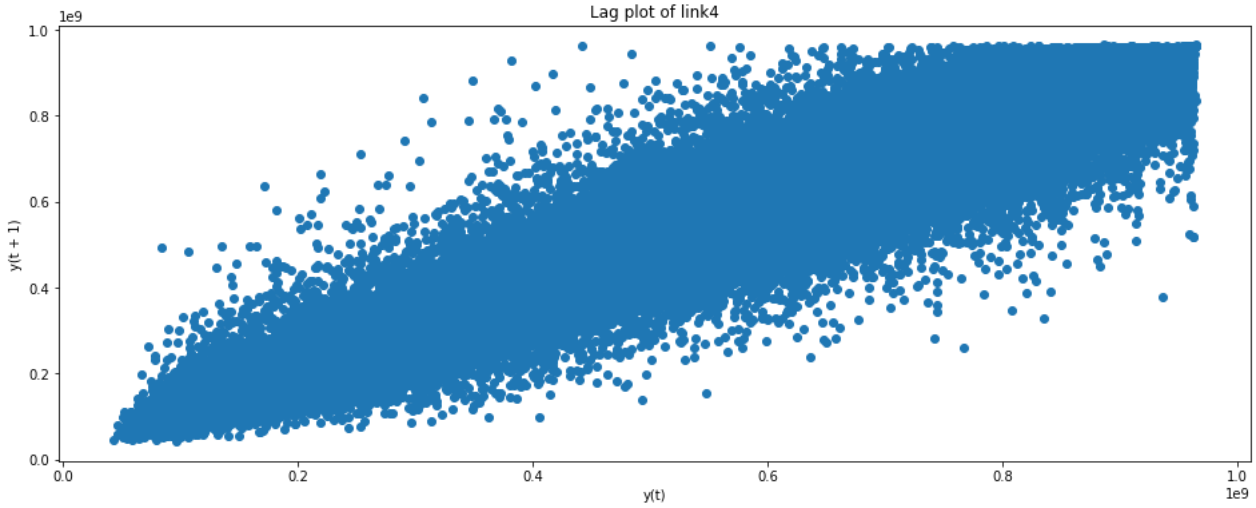
```
#get data
linkdata=[]
for i in range(1,5):
    data=[]
    with open("/Users/jdszsl/Desktop/Aalto/Aalto-Communications Engineering/ELEC-E7130 - Internet Traffic Measurements and Analysis/Assignments/Assignment4/data/linkload-"+str(i)+".txt", "r") as f:  # 打开文件
        for line in f.readlines():
            line = line.strip('\n') #get rid of '\n'
            if i == 2:
                data.append(float(line.split(' ')[0])) #str to float
            else:
                data.append(float(line.split(' ')[1])) #str to float
        linkdata.append(data)
        data=[]
```

```
#lag plot
for i in range(0,4):
    s = pd.Series(linkdata[i])
    plt.figure(figsize=(16,6))
    lag_plot(s, lag=1)
    plt.title('Lag plot of link'+str(i+1))
    plt.show()
```



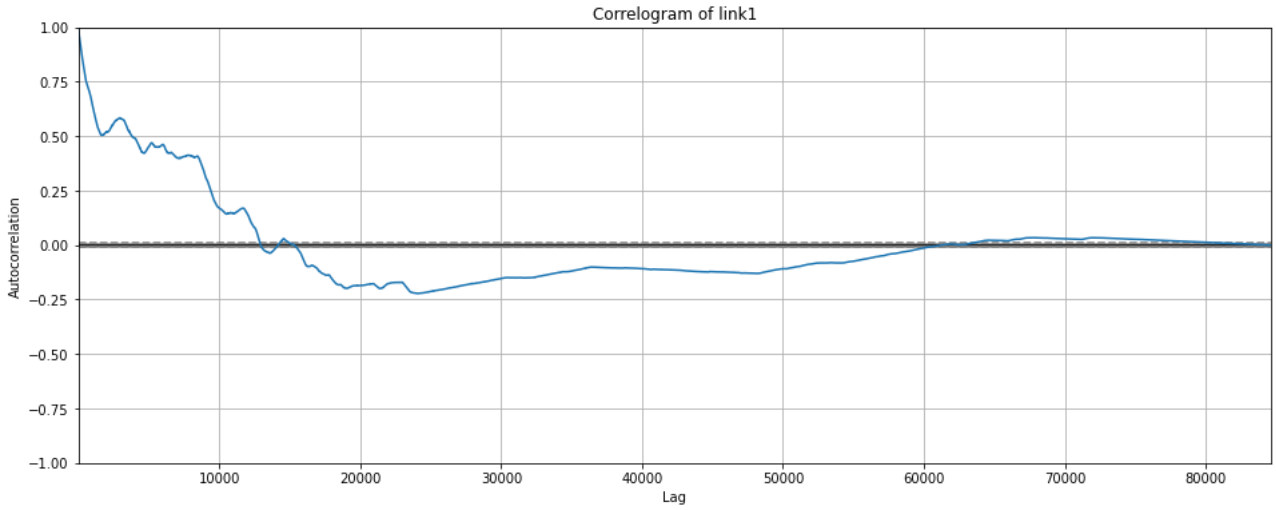


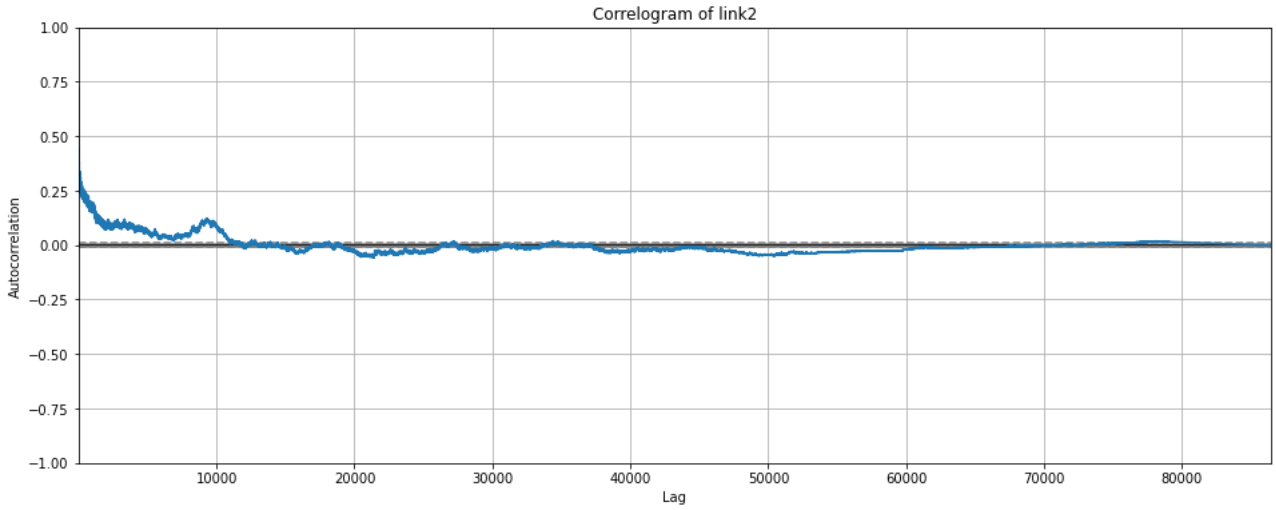


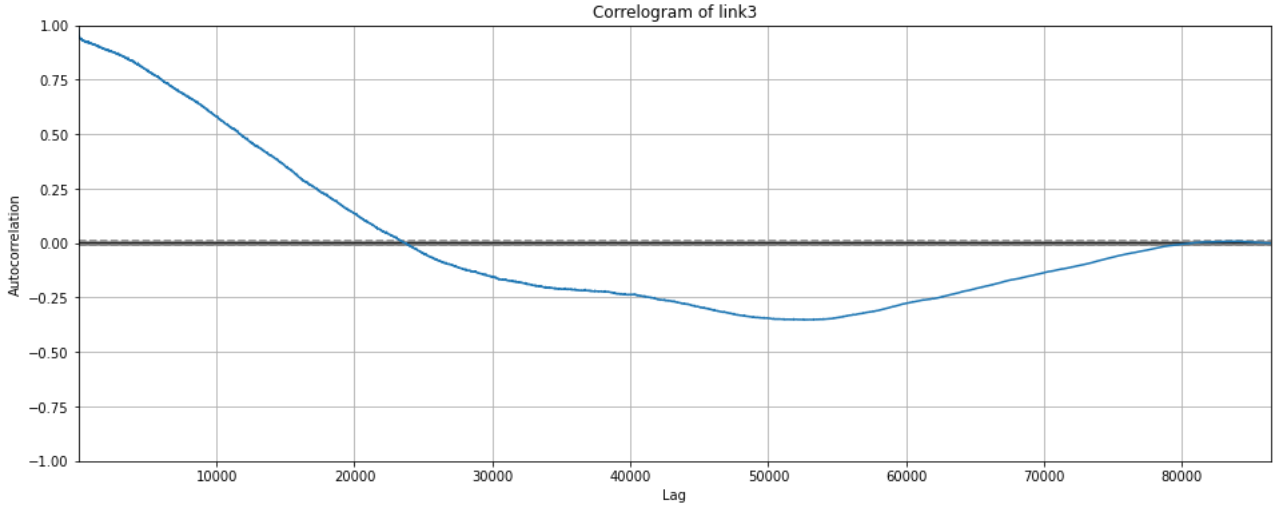


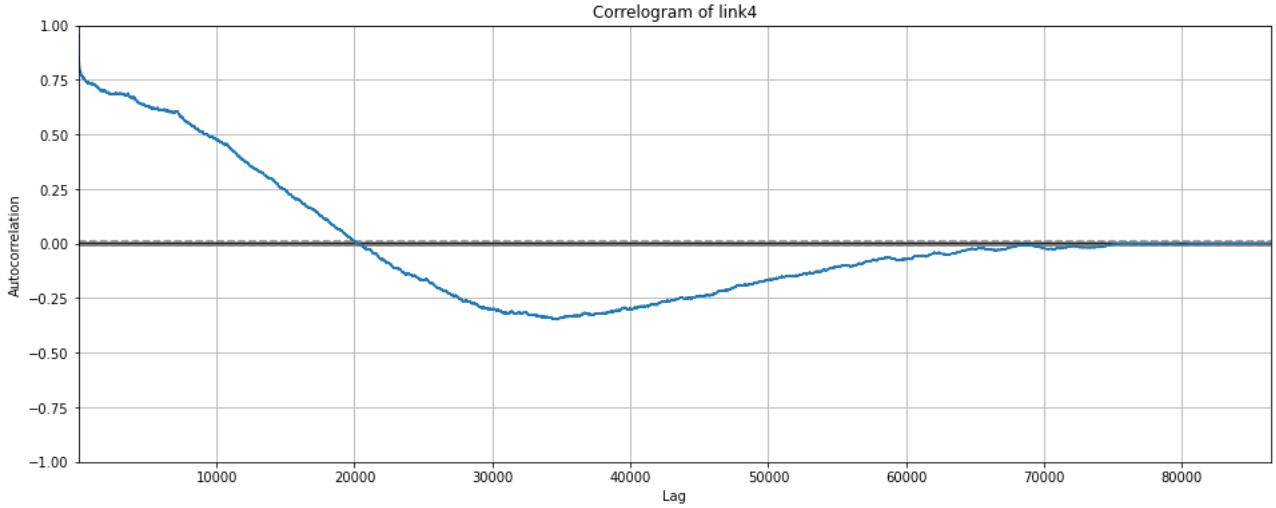

```
#correlogram
for i in range(0,4):
    s = pd.Series(linkdata[i])
    plt.figure(figsize=(10,4))
```

```
5.      autocorrelation_plot(s)
6.      plt.title('Correlogram of link'+str(i+1))
7.      plt.show()
```









2. Data inspection results

For link load 1, 3 and 4, basically their points on the lag plot are close to the 45° line, so there are correlations between the series and the corresponding lag, which means their previous values contribute to the present value. Among them, link load 3 has least outliers (most stable) and according to the autocorrelation plot more recent values (smaller lags, less than 23000) are stronger correlated with the current value than older values (larger lags, more than 23000), and at extremely large lags (80000) the correlation decays to 0; link load 1 has more outliers and according to the autocorrelation plot it has fluctuations in autocorrelation and its autocorrelation decays faster and becomes less correlated earlier than link load 3 at 13000 lags, and also the point where correlation dacays to 0 is earlier (60000 lags) than link load 3; as for link load 4, its points are less concentrated on the 45° line so it has less autocorrelation than link load 3, and it also has more outliers than link load 3 so becomes less correlated a bit earlier than link load 3 at 20000 lags, and also the point where correlation dacays to 0 is earlier (70000 lags)than link load 3 but later than link load 1. However, the scatter points of link load 2 show no sign of close to the 45° line and are randomly plotted (least stable) so it has the least autocorrelation and keep uncorrelated for most of the lags.

3. Conclusions of each data set.

In conclusion, link load 3 has the longest range memory and highest stability and link load 2 has the shortest range memory, and least stability, with link load 1 and 4 placed between them.

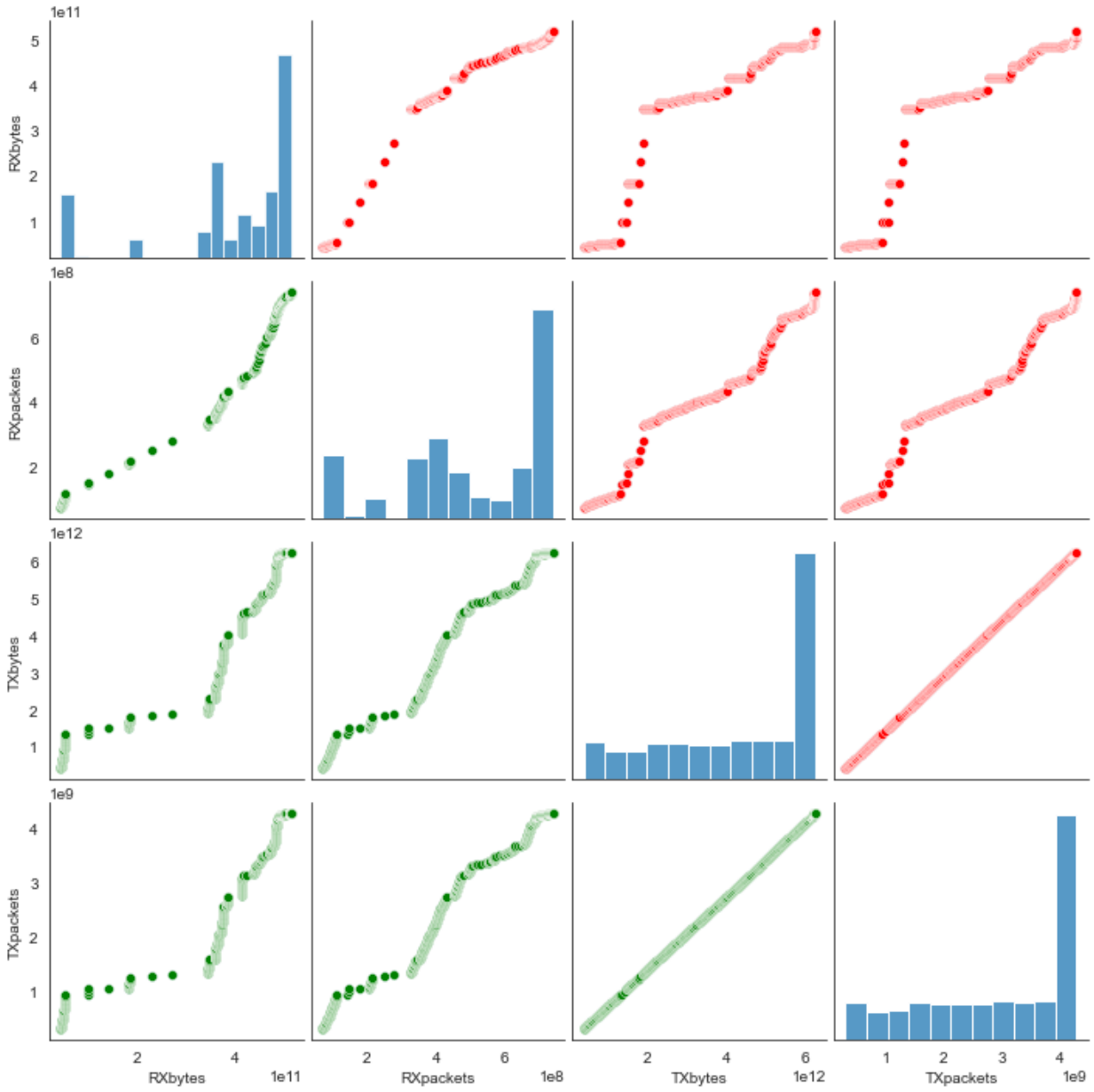
Task 4: Pairs plot

Solution:

1. Pair plots and analysis.

```
#get data as dataframe
csv_data = pd.read_csv('/Users/jdszsl/Desktop/Aalto/Aalto-Communications Engineering/ELEC-E7130 - Internet Traffic Measurements and Analysis/Assignments/Assignment4/data/bytes.csv')
print(csv_data)
```

```
#pairs plot
sns.set_style('white')
p = sns.pairplot(csv_data, kind = 'scatter', diag_kind = 'hist', palette = 'husl')
p.map_upper(sns.scatterplot,color='red')
p.map_lower(sns.scatterplot, color='green')
p.map_diag(plt.hist)
```



According to the pair plot, TXbytes and TXpackets are most correlated to each other since the scatter points are almost on the 45° line, which shows that they have stable linearithmic relations between each other.

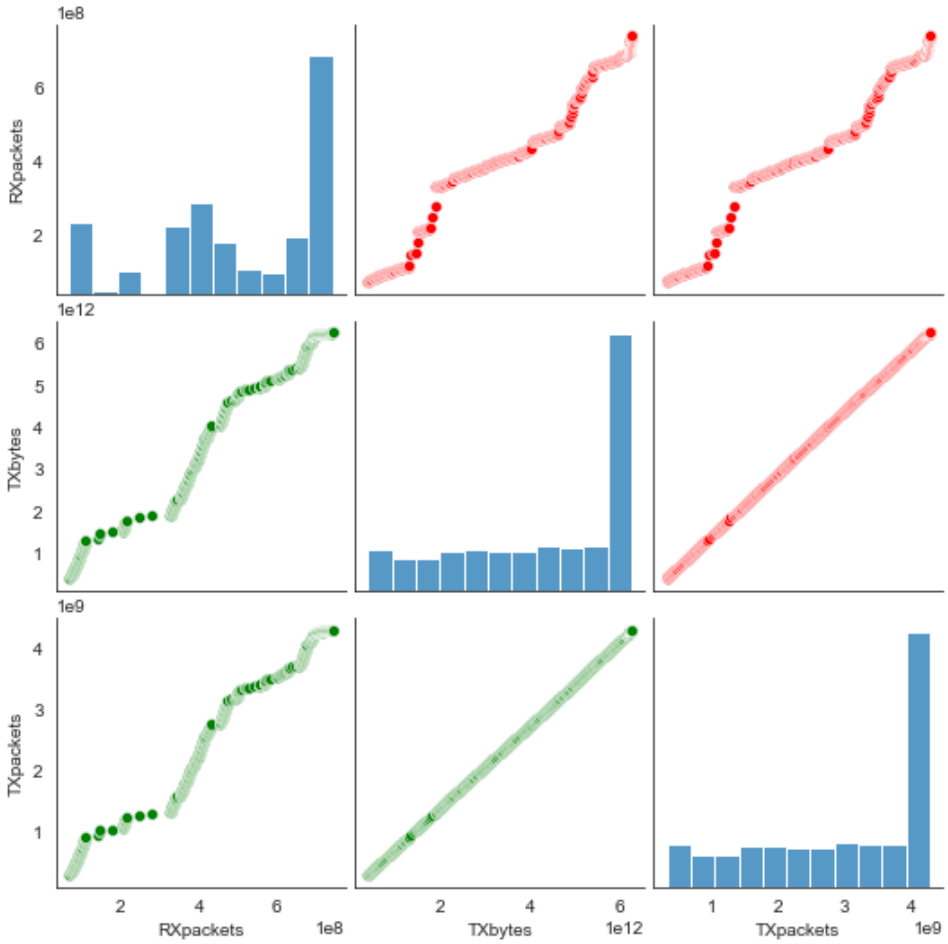
2. What is the least informative column and why?

If we remove the RXbytes column then do pair plot again and see the process time:

```
csv_data1 = csv_data.drop(['RXbytes'], axis=1)
T1 = time.time()
sns.set_style('white')
p = sns.pairplot(csv_data1, kind = 'scatter', diag_kind = 'hist', palette = 'husl')
p.map_upper(sns.scatterplot,color='red')
p.map_lower(sns.scatterplot, color='green')
p.map_diag(plt.hist)
T2 = time.time()
print('data handling takes:%ss' % (T2 - T1))
```

The results are as follows:

```
data handling takes:5.738476991653442s
```

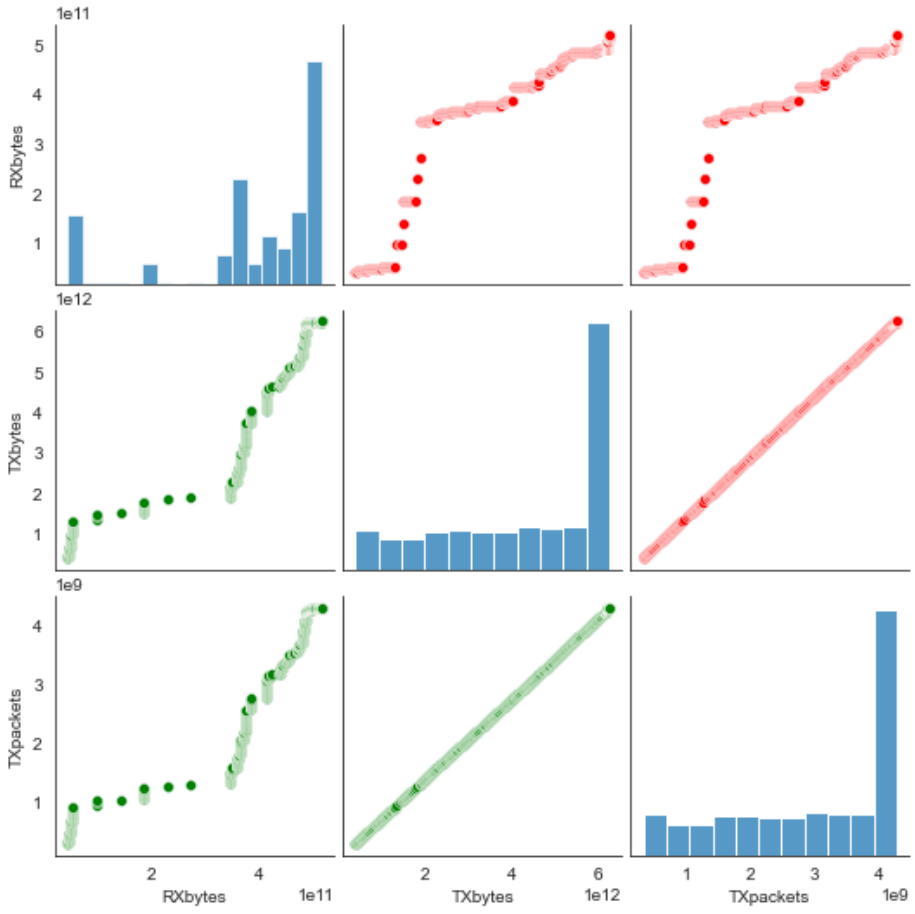


And if we remove the RXpackets column then do pair plot again and see the process time:

```
csv_data2 = csv_data.drop(['RXpackets'], axis=1)
T1 = time.time()
sns.set_style('white')
p = sns.pairplot(csv_data2, kind = 'scatter', diag_kind = 'hist', palette = 'husl')
p.map_upper(sns.scatterplot,color='red')
p.map_lower(sns.scatterplot, color='green')
p.map_diag(plt.hist)
T2 = time.time()
print('data handling takes:%ss' % (T2 - T1))
```

The results are as follows:

```
data handling takes:3.325695037841797s
```



The processing time of removing RXpackets is less than that of removing RXbytes. So the RXbytes is the least informative column, since the points on the plot are more deviated from the 45° line so it is less correlated with other three variables.

Task 5: Practising with dataframes

Solution:

1. Describe method you used to clean the data and added the additional column.

Firstly, I used *pd.read_table()* method from **pandas** library to get the data in the *flowdata_unclean.txt* into a dataframe. Then I used *dropna()* method to get rid of the *NaN* data. Next, I use *to_numpy()* method to convert bytes column into array and determined corresponding number to each flow data according to the instructions. At last, I used *assign()* method to assign the label column to the dataframe.

2. Provide code and sample of data

The codes and sample of data are as follows:

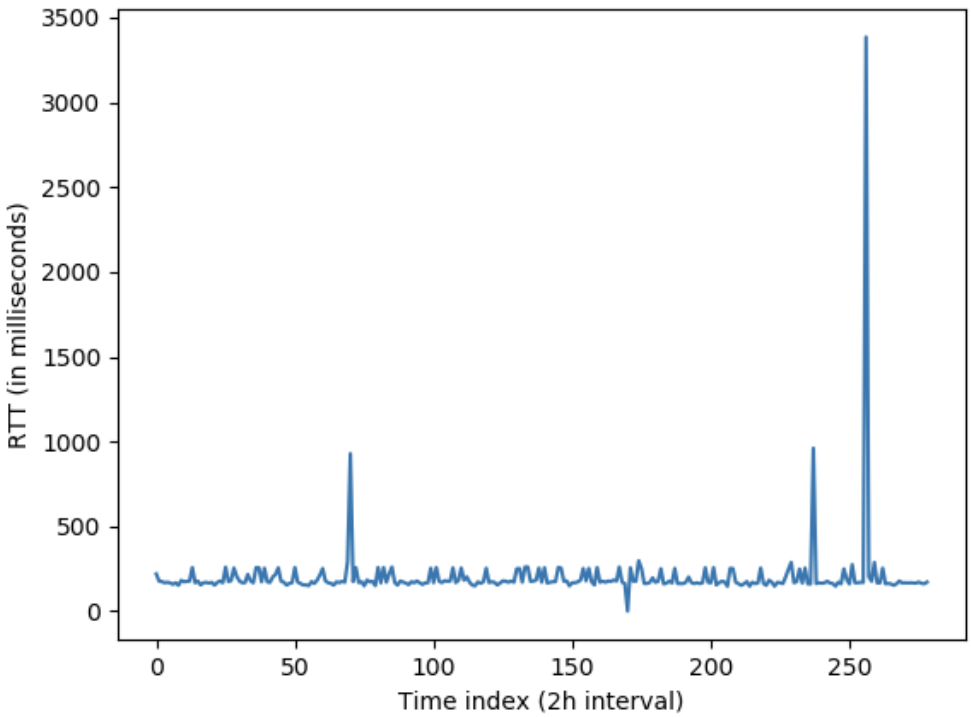
```
#get data into dataframe
txt_data_unclean = pd.read_table('/Users/jdszsl/Desktop/Aalto/Aalto-Communications Engineering/ELEC-E7130 - Internet Traffic Measurements and Analysis/Assignments/Assignment4/data/flowdata_unclean.txt')
print(txt_data_unclean)
txt_data_clean = txt_data_unclean.dropna() #clean the data
print(txt_data_clean)
txt_bytes = txt_data_clean.iloc[:,7].to_numpy() #convert the bytes column into array
print(txt_bytes)
#determine corresponding values
label = []
for i in txt_bytes:
    if i >=50:
        label.append(1)
    else:
        label.append(0)
txt_data_clean = txt_data_clean.assign(Y=label) #assgin the label column to the dataframe
#write the cleaned data into csv
txt_data_clean.to_csv('/Users/jdszsl/Desktop/Aalto/Aalto-Communications Engineering/ELEC-E7130 - Internet Traffic Measurements and Analysis/Assignments/Assignment4/data/flowdata_clean.csv', sep=',', header=True, index=True)
```

	src	dst	proto	ok	sport	dport	pkts	bytes	flows	first	latest	Y
0	203.246.146.19	106.22.48.22	1	1	8	0	1	32	1	1491969543	1491969543	0
2	176.52.159.166	203.246.146.19	1	1	0	0	1	32	1	1491969546	1491969546	0
3	203.246.146.19	5.57.5.49	1	1	8	0	1	32	1	1491969546	1491969546	0
5	101.156.145.60	133.60.159.199	6	1	1214	445	2	96	1	1491969557	1491969560	1
6	203.246.146.19	71.32.111.81	1	1	8	0	1	32	1	1491969562	1491969562	0
7	203.246.146.19	77.205.87.213	1	1	8	0	1	32	1	1491969564	1491969564	0
8	203.246.146.19	73.11.78.246	1	1	8	0	1	32	1	1491969565	1491969565	0

9	203.246.146.19	62.202.235.183	1	1	8	0	1	32	1	1491969567	1491969567	0
10	199.51.103.93	203.246.146.19	1	1	0	0	1	32	1	1491969569	1491969569	0
…	…	…	…	…	…	…	…	…	…	…	…	…

Task 6: Understanding time series concepts

Solution:



1. Answer to questions with reasoning

I. Is there any trend or seasonality?

According to the RTT plot, there is no apparent trend or seasonality, since there are no periodic fluctuations in the whole time period, only with random fluctuations and sometimes extremely big RTT, which are considered outliers.

II. Is the time series stationary?

The time series is stationary since it keep fluctuating around 200 ms for most of the time and only with four outliers.