Report for the Final Assignment of ELEC-E7130

Report, task1

1.0 Data pre-processing

As the last digit of my student number is 6, so I need to deal with the data based on “133.60.172.0/24” subnetwork, the following are the scripts to get the needed data.
For the files in “flow-continue/”, after some filtering work, I got “flow_continue.t2” and showed the first ten lines here *(more detailed info about “flow_continue.t2” can be seen in the AS_Final.zip)*.

```
# /work/courses/unix/T/ELEC/E7130/general/trace/flow-continue
# ls
15-2-1800.t2*  15-2-2100.t2*  15-3-0000.t2*  15-3-0300.t2*  15-3-0600.t2*
15-3-0900.t2*  15-3-1200.t2*  15-3-1500.t2*  15-3-1800.t2*
15-2-1900.t2*  15-2-2200.t2*  15-3-0100.t2*  15-3-0400.t2*  15-3-0700.t2*
15-3-1000.t2*  15-3-1300.t2*  15-3-1600.t2*  15-3-1800.t2.25*
15-2-2000.t2*  15-2-2300.t2*  15-3-0200.t2*  15-3-0500.t2*  15-3-0800.t2*
15-3-1100.t2*  15-3-1400.t2*  15-3-1700.t2*

# cat /work/courses/unix/T/ELEC/E7130/general/trace/flow-continue/*.t2 | gawk
'$1~/^133\.60\.172\./||$2~/^133\.60\.172\./' >> ./flow_continue.t2
# cat flow_continue.t2 | wc -l
1099743

# head -n5 flow_continue.t2
190.247.151.140 133.60.172.0    6   1   4873    445 2   96  1
1491923010.262914000    1491923013.272384000
85.135.83.137   133.60.172.138  6   1   3111    445 2   96  1
1491925517.250238000    1491925520.009444000
77.5.153.148    133.60.172.94   6   1   1528    445 2   96  1
1491925751.527469000    1491925754.574771000
190.183.127.119 133.60.172.15   6   1   1289    445 2   96  1
1491923077.276329000    1491923080.256957000
191.37.134.178  133.60.172.19   6   1   18248   23  1   44  1
1491925700.491105000    1491925700.491105000
```

After the preprocessing above, replacing all “\t” with “,” and adding header for the “flow_continue.t2” file, then renaming the file as “flow_continue.csv”, and I copied this file to my local computer

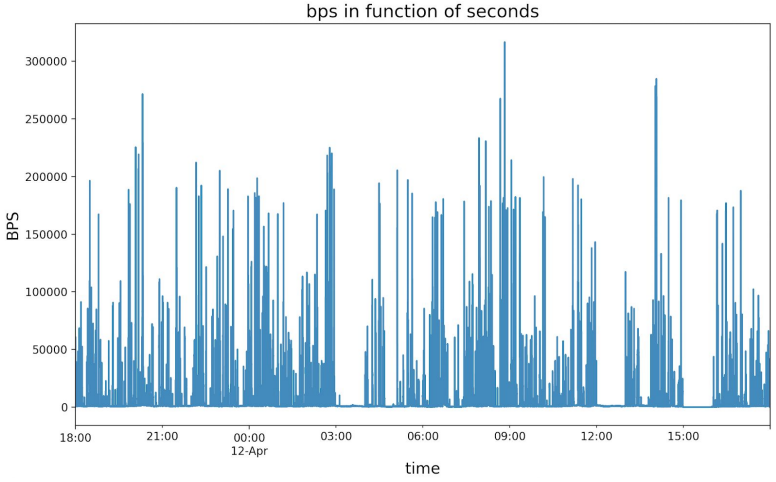
```
# head -n5 flow_continue.csv
src,dst,proto,valid,sport,dport,pkt,bytes,flows,start,end
190.247.151.140,133.60.172.0,6,1,4873,445,2,96,1,1491923010.262914000,1491923013.27
2384000
85.135.83.137,133.60.172.138,6,1,3111,445,2,96,1,1491925517.250238000,1491925520.00
9444000
77.5.153.148,133.60.172.94,6,1,1528,445,2,96,1,1491925751.527469000,1491925754.5747
71000
190.183.127.119,133.60.172.15,6,1,1289,445,2,96,1,1491923077.276329000,1491923080.2
56957000
```

1.1 Plot traffic volume

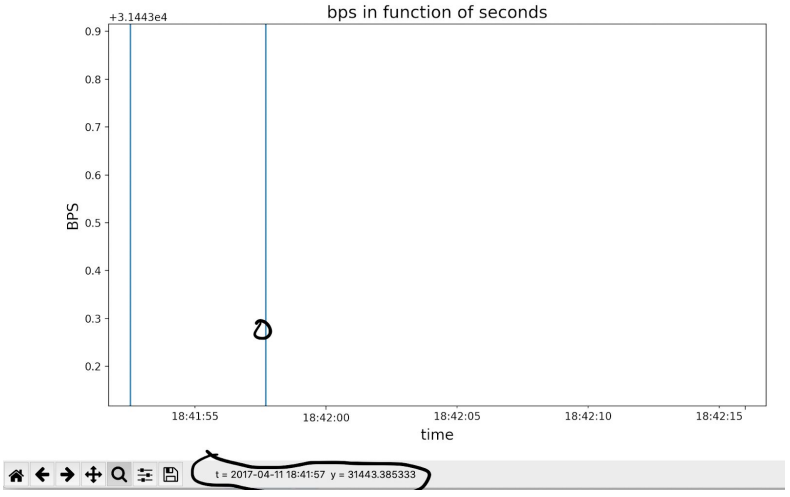
1.1.1 Code and plot of bits per second in function of seconds

```
import time
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from dateutil.parser import parse
df = pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/flow_continue.csv')
beginTime=1E30
endTime=0
start_x = []
for i in df["start"]:
    if beginTime > i:
        beginTime =i
    if endTime<i:
        endTime=i
    start_x.append(int(i))
end_x = []
for i in df["end"]:
    if beginTime > i:
        beginTime =i
    if endTime<i:
        endTime=i
    end_x.append(int(i))
time_x = []
for i in range(len(start_x)):
    tmp = end_x[i]-start_x[i]
    if int(tmp) == 0:
        time_x.append(1)
    else:
        time_x.append(int(tmp))
bytes_x = []
for i in df["bytes"]:
    bytes_x.append(i)
bps_x = []
for i in range(len(start_x)):
    bps_x.append(round(bytes_x[i]*1.0/time_x[i], 3))
beginTime=int(beginTime)
endTime=int(endTime)
timeStr=[]
for i in range(beginTime,endTime+1):
    timeStr.append(time.strftime("%Y-%m-%d %H:%M:%S", time.localtime(int(i))))
########
timeLine = [0]*(endTime-beginTime+1)
for i in range(len(start_x)):
    start=start_x[i]
    end=end_x[i]
    bps=bps_x[i]
    for idx in range(start,end):
```

```
        abs_idx=idx-beginTime
        timeLine[abs_idx]=timeLine[abs_idx]+bps
data={
   "time":timeStr,
   "bps":timeLine
}
df = pd.DataFrame(data)
df['time'] = pd.to_datetime(df['time'])
plt.title("bps in function of seconds",fontsize=16)
plt.xlabel("TIME",fontsize=14)
plt.ylabel("BPS",fontsize=14)
df=df.set_index('time')
df['bps'].plot()
plt.show()
```



The plot above is shown in this way because of too many values for the x-axis, but when I click some point in the plot and click “Zoom In”, it will show the detailed info for ***some second*** as follows:

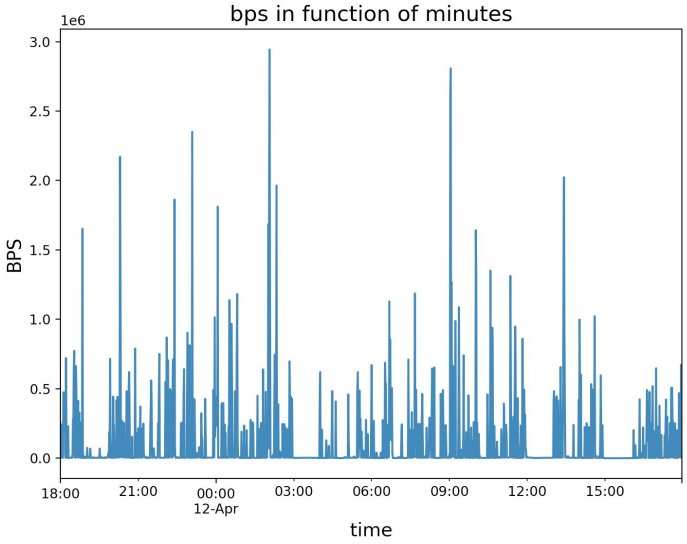


1.1.2 Code and plot of bits per second in function of minutes

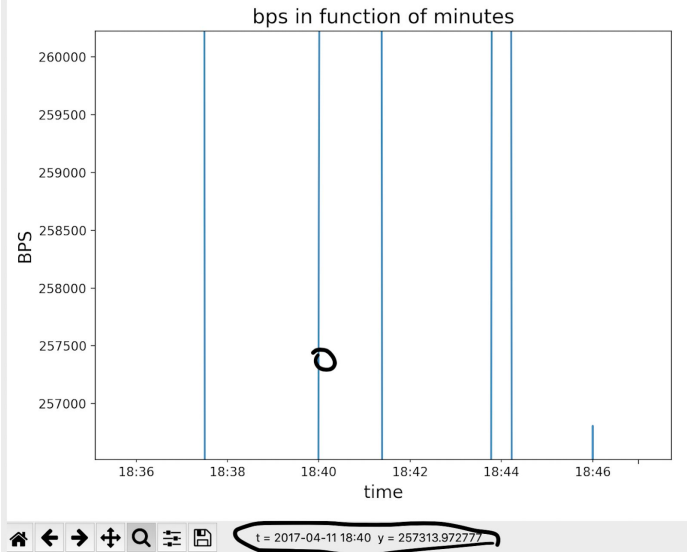
Modify some lines as follows based on the previous code (from “########”)

```
timeLine = [0]*(len(timeStr))
for i in range(len(start_x)):
   start=int(start_x[i]/60)*60
   end=int(end_x[i]/60)*60
   bps=bps_x[i]
   for idx in range(start,end,60):
       abs_idx=int(int(idx-beginTime)/60)
       timeLine[abs_idx]=timeLine[abs_idx]+bps
```

```
data={
    "time":timeStr,
    "bps":timeLine
}
df = pd.DataFrame(data)
df['time'] = pd.to_datetime(df['time'])
plt.title("bps in function of minutes",fontsize=16)
plt.xlabel("TIME",fontsize=14)
plt.ylabel("BPS",fontsize=14)
df=df.set_index('time')
df['bps'].plot()
plt.show()
```



The plot above is shown in this way because of too many values for the x-axis, but when I click some point in the plot and click “Zoom In”, it will show the detailed info for ***some minute*** as follows:

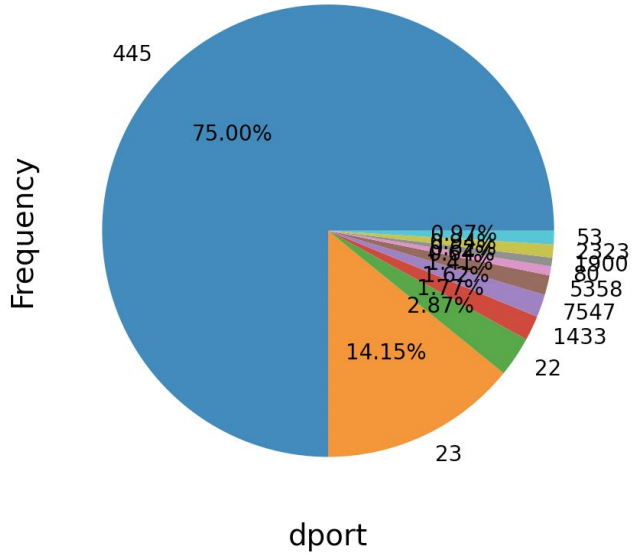


1.2 Code and plot for visualizing flows by port numbers

In this part, as we all know, sport numbers are random based on the resources of the client in sessions, while dport numbers are decided by the protocol (the dport numbers are discrete values), and after some checking, I also found that the value of the “flows” in each flow is 1. Based on the analysis above, displaying the code and plot as follows (I filtered

some ports through “dportMap[i] > 5000, so in the following plot, only displayed most of the data, not all of them”):

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/flow_continue.csv')
dportMap = {}
for index,row in df.iterrows():
      sport = row["sport"]
      dport = row["dport"]
      if sport >= dport:
            tmp = dport
      else:
            tmp = sport
      if tmp in dportMap:
            dportMap[tmp] = dportMap[tmp] + 1
      else:
            dportMap[tmp] = 1
x=[]
y=[]
for i in dportMap:
  if dportMap[i]>5000:
      x.append(i)
      y.append(dportMap[i])
plt.plot(y,x)
plt.xlabel("dport",fontsize=14)
plt.ylabel("Frequency",fontsize=14)
plt.pie(y,labels=x,autopct='%3.2f%%')
plt.show()
```

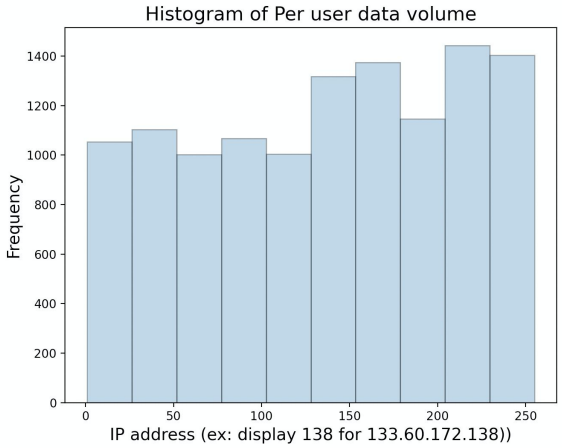


1.3 OD-pairs

1.4 Code and plot for visualising distribution of user aggregated data

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/flow_continue.csv')
subnetwork = []
ip2int = lambda x:sum([256**j*int(i) for j,i in enumerate(x.split('.')[::-1])])
base = ip2int("133.60.172.0")
for index, row in df.iterrows():
 if row["src"].startswith("133.60.172"):
        tmp = ip2int(row["src"])-base
```

```
        if tmp > 0 and tmp < 256:
                subnetwork.append(ip2int(row["src"])-base)
 else:
        tmp = ip2int(row["src"])-base
        if tmp > 0 and tmp < 256:
                subnetwork.append(ip2int(row["dst"])-base)
plt.title("Histogram of Per user data volume",fontsize=16)
plt.xlabel("IP address (ex: display 138 for 133.60.172.138))",fontsize=14)
plt.ylabel("Frequency",fontsize=14)
plt.hist(subnetwork, edgecolor='k', alpha=0.35)
plt.show()
```



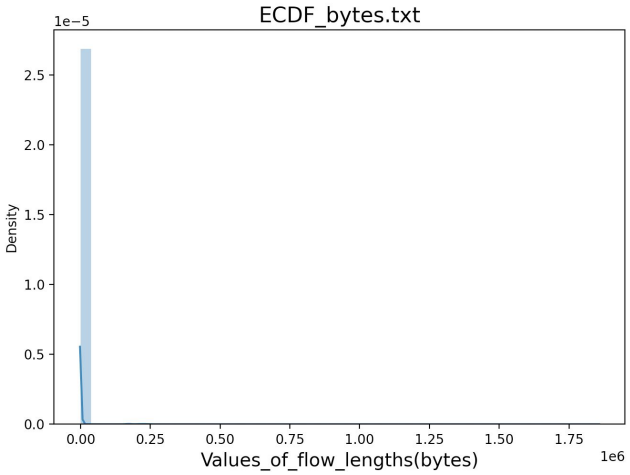
1.5 Flow length distribution

1.5.1 Key summary statistics of flow length

```
> bytes_set = flow_continue[, "bytes"]
> bytes_set <- as.numeric(bytes_set)
> summary(bytes_set)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    26      44      96     439      96 1856463
```

1.5.2 Code and ECDF plot for the flow length distribution

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
data =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/flow_continue.csv')
x=[]
for index, row in data.iterrows():
  x.append(row["bytes"])
plt.title("ECDF_bytes.txt",fontsize=16)
plt.xlabel("Values_of_flow_lengths(bytes)",fontsize=14)
sns.distplot(x)
plt.show()
```



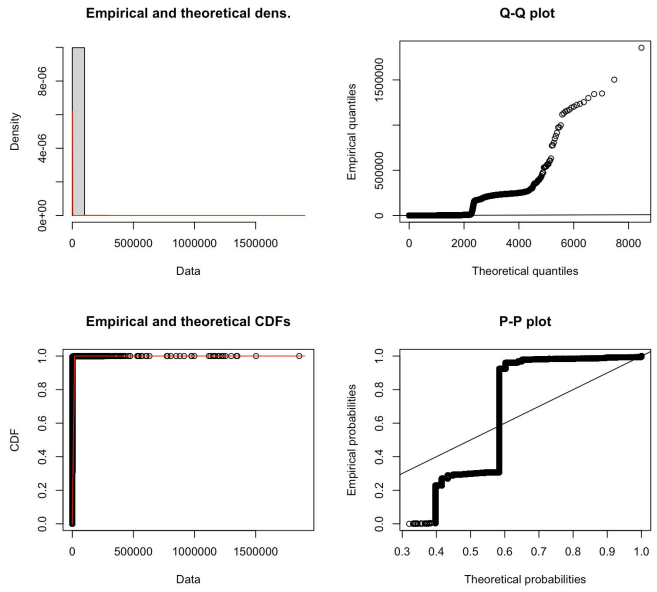
1.5.3 Distribution judgement for flow length

The result shows that the "flow length(bytes)" of this data set does not follow any kind of distribution, which is caused the data are highly discrete.

I tried to fit the followings kinds of distributions through "fitdist" function for flow length, including "norm", "lnorm", "exp", "weibull" and "unif", after checking the "Parameters" calculated by "fitdist" function with different kinds of distributions and validating the distributions through plotting, no one fits.

- Code and plot for "weibull" distribution

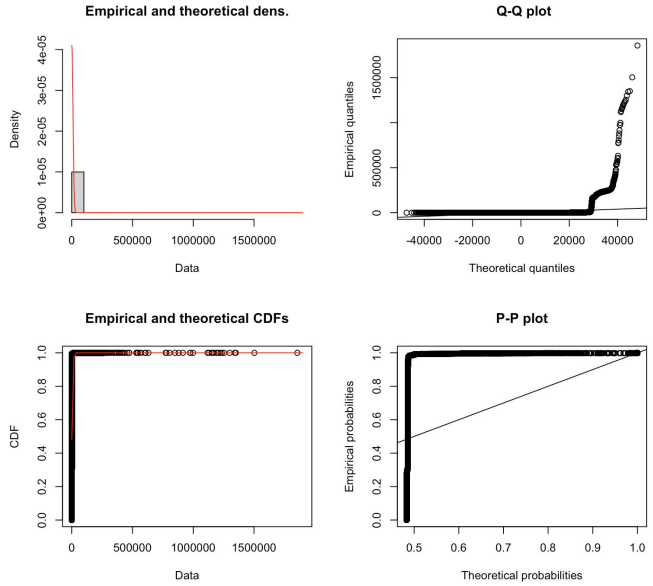
```
> bytes_set = flow_continue[, "bytes"]
> bytes_set <- as.numeric(bytes_set)
> fit_para <- fitdist(bytes_set, "weibull")
> print(fit_para)
Fitting of the distribution ' weibull ' by maximum likelihood
Parameters:
        estimate  Std. Error
shape   0.6278835 0.000280037
scale 118.3671783 0.190577727
> plot(fit_para)
```



- Code and plot for "norm" distribution

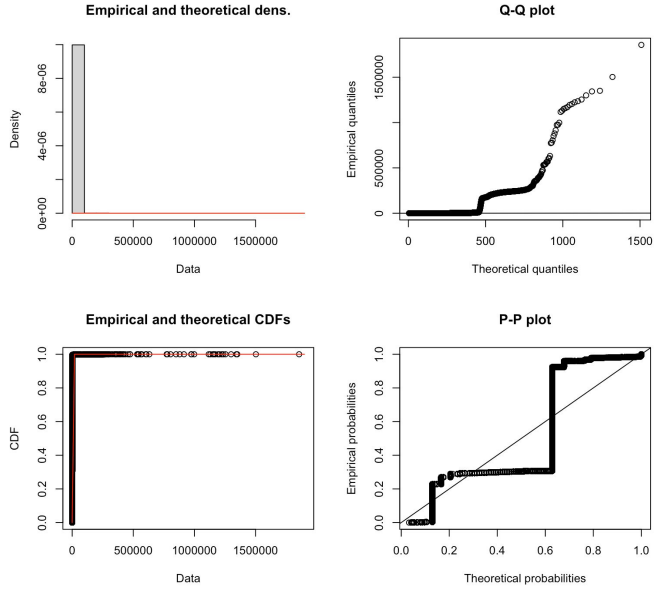
```
> bytes_set = flow_continue[, "bytes"]
> bytes_set <- as.numeric(bytes_set)
> fit_para <- fitdist(bytes_set, "norm")
```

```
> print(fit_para)
Fitting of the distribution ' norm ' by maximum likelihood
Parameters:
     estimate Std. Error
mean  439.0202   4.807517
sd   9724.4859   5.648131
> plot(fit_para)
```



- Code and plot for “lnorm” distribution

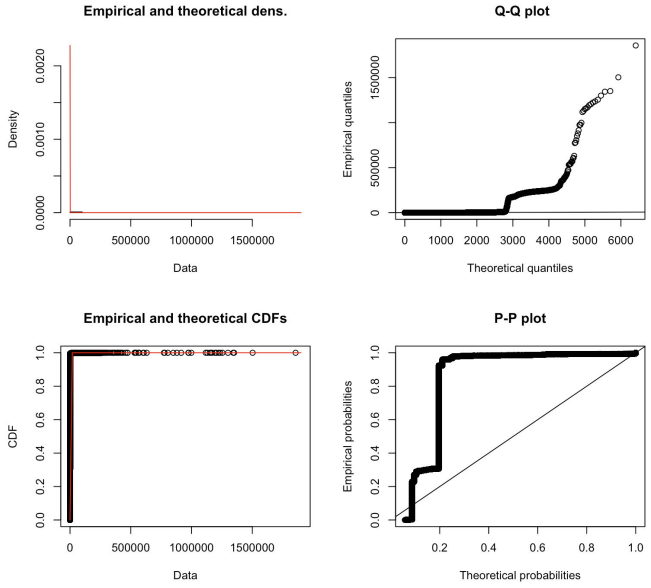
```
> bytes_set = flow_continue[, "bytes"]
> bytes_set <- as.numeric(bytes_set)
> fit_para <- fitdist(bytes_set, "lnorm")
> print(fit_para)
Fitting of the distribution ' lnorm ' by maximum likelihood
Parameters:
        estimate   Std. Error
meanlog 4.3666270 0.0005733147
sdlog   0.6012273 0.0004053897
> plot(fit_para)
```



- Code and plot for “exp” distribution

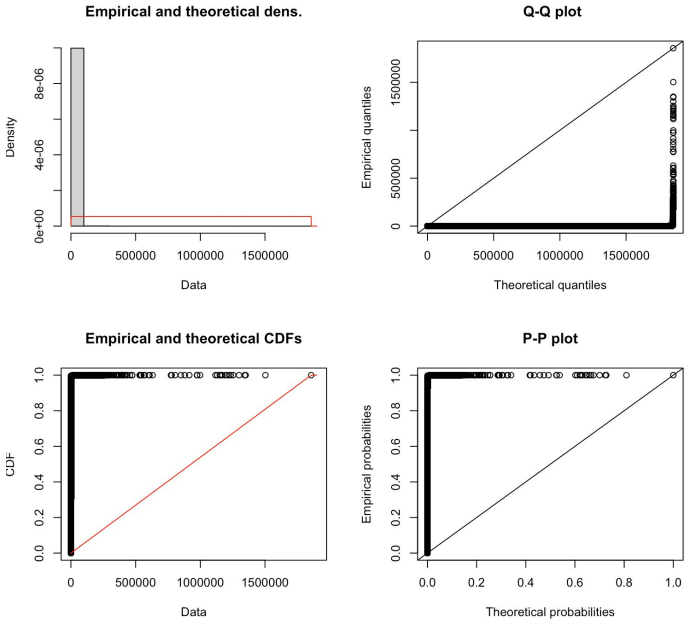
```
> bytes_set = flow_continue[, "bytes"]
> bytes_set <- as.numeric(bytes_set)
> fit_para <- fitdist(bytes_set, "exp")
```

```
> print(fit_para)
Fitting of the distribution ' exp ' by maximum likelihood
Parameters:
       estimate  Std. Error
rate 0.002277799 1.57094e-06
> plot(fit_para)
```



- Code and plot for “unif” distribution

```
> bytes_set = flow_continue[, "bytes"]
> bytes_set <- as.numeric(bytes_set)
> fit_para <- fitdist(bytes_set, "unif")
> print(fit_para)
Fitting of the distribution ' unif ' by maximum likelihood
Parameters:
    estimate Std. Error
min       26         NA
max  1856463         NA
> plot(fit_para)
```



1.6 Conclusions

1.6.1 Recognisable patterns for traffic volume at different time scales

Even though the two plots are in different time scales, essentially the plots from the same data set, so they follow the same trend. And there are other findings from these two plots, 3:00-4:00, 12:00-13:00, 15:00-16:00 are the bottoms of the traffic volume, and 2:00-3:00, 9:00-10:00 are the peaks of the traffic volume.

1.6.2 The 5 most common applications based on the study of the port numbers

TCP 445 = Port for SMB Protocol (for example: used for common file sharing)
TCP 23= Port for Telnet protocol
TCP 22= Port for SSH protocol
TCP 1433 = Default port for SQL Server
TCP 7547 = Port for CWMP protocol on H3C system

1.6.3 Speculate on what kind of network this network could be based on traffic volumes and user profiles

Based on the plots for “flows by port numbers” and “per user data volume”, and the ranking info in the previous question, this network is a small office network or a small school network. During which, people will share files among each other, remotely control other computers through telnet/ssh, access sql servers and configure routers/switches.

Report, task2

2.0 Description about packet capture

I captured the packets through wireshark software, during the capture, I normally used my computer to search materials on the Internet, listen to music, use instant talk software, upload/download some files and so on. The basic info about the packets is as follows:

```
Name: /Users/yanjing/Desktop/test.pcap
Length: 318 MB
Hash (SHA256): bb91993710ff74379435cd48821093faa7e2b89b1026490cb27efb5a748a6c73
Hash (RIPEMD160): 3e32331917b1d9edaac0ba459c4351ad7fa3f240
Hash (SHA1): 9084c5e25033a014010d6d47e4d307960666e535
Format: Wireshark/... - pcap
Encapsulation: Ethernet

Time
First packet: 2020-11-22 09:53:10
Last packet: 2020-11-22 12:11:52
Elapsed: 02:18:41
```

Statistics

Measurement	Captured	Displayed	Marked
Packets	404029	404029 (100.0%)	—
Time span, s	8321.699	8321.699	—
Average pps	48.6	48.6	—
Average packet size, B	754	754	—
Bytes	304630701	304630701 (100.0%)	0
Average bytes/s	36 k	36 k	—
Average bits/s	292 k	292 k	—

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.100	142.250.74.46	TCP	78	63206 → 443 [SYN] Seq=0 Win=65535 Len=0 M
2	0.010839	142.250.74.46	192.168.1.100	QUIC	1392	Initial, SCID=9ed5010886b70837
3	0.011097	142.250.74.46	192.168.1.100	QUIC	265	Handshake, SCID=9ed5010886b70837
4	0.011100	142.250.74.46	192.168.1.100	QUIC	106	Protected Payload (KP0)
5	0.012046	192.168.1.100	142.250.74.46	QUIC	1392	Initial, DCID=9ed5010886b70837
6	0.012216	192.168.1.100	142.250.74.46	QUIC	83	Handshake, DCID=9ed5010886b70837
7	0.015967	192.168.1.100	142.250.74.46	QUIC	115	Handshake, DCID=9ed5010886b70837
8	0.016020	192.168.1.100	142.250.74.46	QUIC	112	Protected Payload (KP0), DCID=9ed5010886b
9	0.016356	192.168.1.100	142.250.74.46	QUIC	1291	Protected Payload (KP0), DCID=9ed5010886b
10	0.022827	142.250.74.46	192.168.1.100	TCP	74	443 → 63206 [SYN, ACK] Seq=0 Ack=1 Win=65
11	0.022837	159.89.89.188	192.168.1.100	TCP	1514	[TCP segment of a reassembled PDU]
12	0.022899	192.168.1.100	142.250.74.46	TCP	66	63206 → 443 [ACK] Seq=1 Ack=1 Win=131840
13	0.023112	192.168.1.100	142.250.74.46	TLSv1.3	652	Client Hello

2.1 Data pre-processing

2.1.1 Cleaning the data packets (PS1)

In this part, I dealt with the PS1 dataset using the same method with PS2, which can be seen in 2.1.2, so all the tasks which should use PS1 dataset will use PS2.csv file.

2.1.2 Converting packet trace to flow data (PS2)

```
% crl_flow -Ci=3600 -cl -Tf60 -O %i.t2 -Cai=1 test.pcap
% ls -l
total 312648
-rw-r--r-- 1 yanj3 domain users     34474 Nov 23 11:17 0.t2
-rw-r--r-- 1 yanj3 domain users    294427 Nov 23 11:17 1.t2
-rw-r--r-- 1 yanj3 domain users    344028 Nov 23 11:17 2.t2
-rw-r--r-- 1 yanj3 domain users     72941 Nov 23 11:17 3.t2
-rw-r--r-- 1 yanj3 domain users      6994 Nov 23 11:17 4.t2
-rw------- 1 yanj3 domain users 318116368 Nov 23 11:16 test.pcap
% for i in {0..6}
for> do
for> grep '^[^#]' $i.t2 >> flow.t2
for> done
% vim flow.t2 (:%s/\t/\,/g :wq)
% sed -i '1i\src,dst,pro,ok,sport,dport,pkts,bytes,flows,first,latest' flow.t2
% mv flow.csv PS2.csv
yanjing@yanjingdeMacBook-Pro DataSet2 % head -n10 PS2.csv
src,dst,pro,ok,sport,dport,pkts,bytes,flows,first,latest
192.168.1.100,82.130.63.1,17,1,55883,53,1,75,1,1606031670.000235000,1606031670.0002
35000
192.168.1.100,142.250.74.14,17,1,52886,443,95,11496,1,1606031701.534319000,16060317
24.083220000
82.130.63.1,192.168.1.100,17,1,53,38008,1,347,1,1606031663.830470000,1606031663.830
470000
192.168.1.100,142.250.74.35,6,1,63246,443,9,1115,1,1606031800.890871000,1606031878.
749332000
82.130.63.1,192.168.1.100,17,1,53,26227,1,353,1,1606031791.768314000,1606031791.768
314000
82.130.0.1,192.168.1.100,17,1,53,51651,1,347,1,1606031894.334940000,1606031894.3349
40000
172.217.21.129,192.168.1.100,17,1,443,55054,11,6432,1,1606031663.998775000,16060316
64.125169000
172.217.21.138,192.168.1.100,17,1,443,61332,11,4923,1,1606031705.225699000,16060317
16.277498000
192.168.1.100,216.58.211.142,17,1,50814,443,14,4026,1,1606031663.973862000,16060316
64.125200000
```

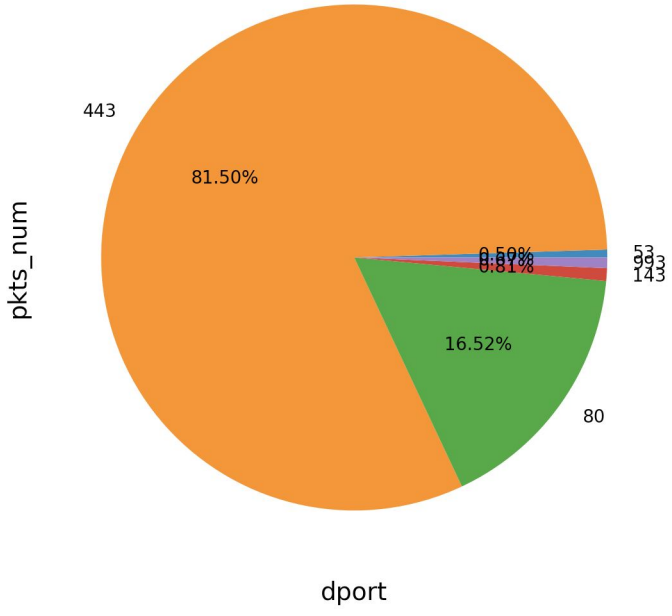
2.1.3 TCP connection statistics (PS3)

```
% tcptrace -l -r -n --csv /Users/yanjing/Desktop/test/test.pcap > PS3.csv
# Do not display the detailed content of PS3.csv here cause too much content in
each line
```

2.2 Packet data PS1

2.2.1 Code and plot for visualizing packet distribution by port numbers

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet2/PS2.csv')
dportMappacket = {}
for index,row in df.iterrows():
      sport = row["sport"]
      dport = row["dport"]
      pkts = row["pkts"]
      if sport >= dport:
            tmp = dport
      else:
            tmp = sport
      if tmp in dportMappacket:
            dportMappacket[tmp] = dportMappacket[tmp] + pkts
      else:
            dportMappacket[tmp] = pkts
x=[]
y=[]
for i in dportMappacket:
      if dportMappacket[i] > 1000:
            x.append(i)
            y.append(dportMappacket[i])
plt.plot(y,x)
plt.xlabel("dport",fontsize=14)
plt.ylabel("pkts_num",fontsize=14)
plt.pie(y,labels=x,autopct='%3.2f%%')
plt.show()
```

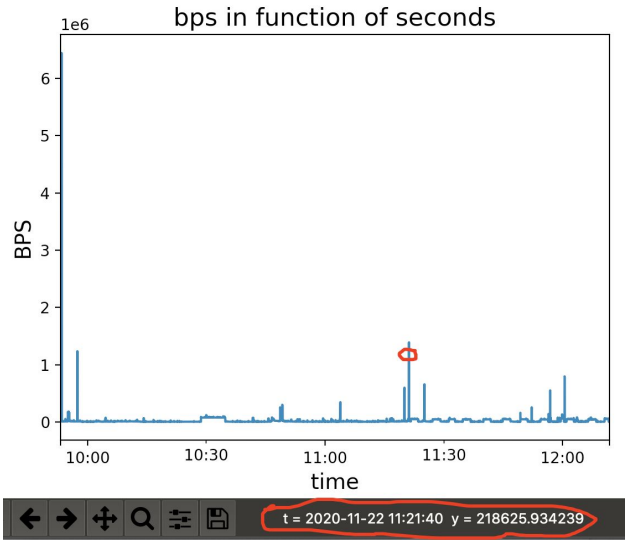


2.2.2 Plot traffic volume

2.2.2.1 Code and plot of bits per second in function of seconds

```
import time
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from dateutil.parser import parse
df = pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet2/PS2.csv')
```

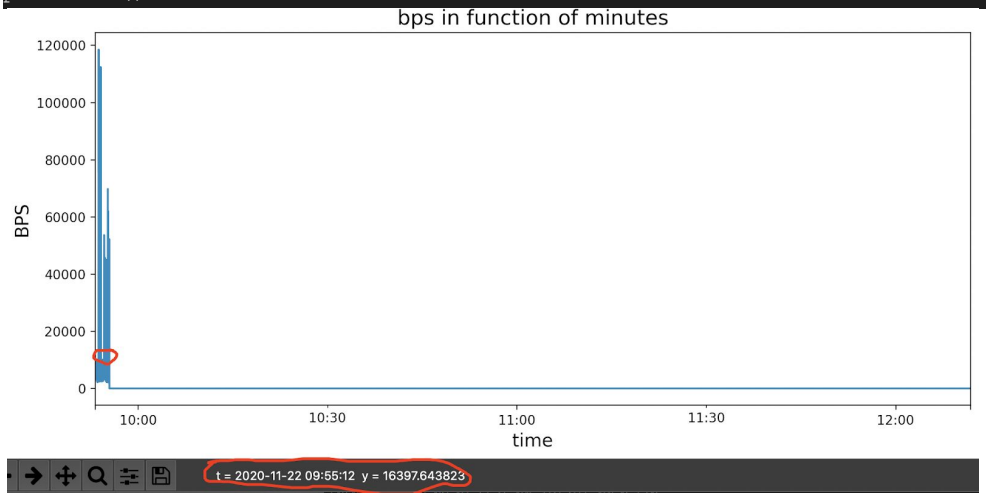
```
beginTime=1E30
endTime=0
start_x = []
for i in df["first"]:
  if beginTime > i:
      beginTime =i
  if endTime<i:
      endTime=i
  start_x.append(int(i))
end_x = []
for i in df["latest"]:
  if beginTime > i:
      beginTime =i
  if endTime<i:
      endTime=i
  end_x.append(int(i))
time_x = []
for i in range(len(start_x)):
  tmp = end_x[i]-start_x[i]
  if int(tmp) == 0:
      time_x.append(1)
  else:
      time_x.append(int(tmp))
bytes_x = []
for i in df["bytes"]:
  bytes_x.append(i)
bps_x = []
for i in range(len(start_x)):
  bps_x.append(round(bytes_x[i]*1.0/time_x[i], 3))
beginTime=int(beginTime)
endTime=int(endTime)
timeStr=[]
for i in range(beginTime,endTime+1):
  timeStr.append(time.strftime("%Y-%m-%d %H:%M:%S", time.localtime(int(i))))
########
timeLine = [0]*(endTime-beginTime+1)
for i in range(len(start_x)):
  start=start_x[i]
  end=end_x[i]
  bps=bps_x[i]
  for idx in range(start,end):
      abs_idx=idx-beginTime
      timeLine[abs_idx]=timeLine[abs_idx]+bps
data={
  "time":timeStr,
  "bps":timeLine
}
df = pd.DataFrame(data)
df['time'] = pd.to_datetime(df['time'])
plt.title("bps in function of seconds",fontsize=16)
plt.xlabel("TIME",fontsize=14)
plt.ylabel("BPS",fontsize=14)
df=df.set_index('time')
df['bps'].plot()
plt.show()
```



2.2.2.2 Code and plot of bits per second in function of minutes

Modify some lines as follows based on the previous code (from "########")

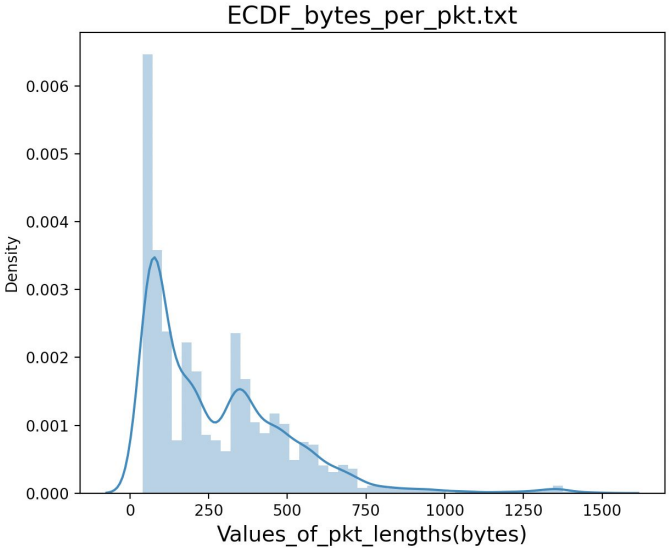
```
timeLine = [0]*(len(timeStr))
for i in range(len(start_x)):
  start=int(start_x[i]/60)*60
  end=int(end_x[i]/60)*60
  bps=bps_x[i]
  for idx in range(start,end,60):
      abs_idx=int(int(idx-beginTime)/60)
      timeLine[abs_idx]=timeLine[abs_idx]+bps
data={
  "time":timeStr,
  "bps":timeLine
}
df = pd.DataFrame(data)
df['time'] = pd.to_datetime(df['time'])
plt.title("bps in function of minutes",fontsize=16)
plt.xlabel("TIME",fontsize=14)
plt.ylabel("BPS",fontsize=14)
df=df.set_index('time')
df['bps'].plot()
plt.show()
```



2.2.3 Packet length distribution

2.2.3.1 Code and ECDF plot for the packet length distribution

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet2/PS2.csv')
bpp_list = []
for index,row in df.iterrows():
     pkts_x = row["pkts"]
     bytes_x = row["bytes"]
     bpp_list.append(float('%.3f'%(bytes_x/pkts_x)))
plt.title("ECDF_bytes_per_pkt.txt",fontsize=16)
plt.xlabel("Values_of_pkt_lengths(bytes)",fontsize=14)
sns.distplot(bpp_list)
plt.show()
```



2.2.3.2 Key summary statistics of the packet length

```
import numpy as np
import pandas as pd
df = pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet2/PS2.csv')
bpp_list = []
for index,row in df.iterrows():
     pkts_x = row["pkts"]
     bytes_x = row["bytes"]
     bpp_list.append(float('%.3f'%(bytes_x/pkts_x)))
print("The min value is: ", float('%.3f'%(np.min(bpp_list))))
print("The 1st Qu. is: ", float('%.3f'%(np.percentile(bpp_list, (25)))))
print("The median value is: ", float('%.3f'%(np.median(bpp_list))))
print("The mean value is: ", float('%.3f'%(np.mean(bpp_list))))
print("The 3st Qu. is: ", float('%.3f'%(np.percentile(bpp_list, (75)))))
print("The max value is: ", float('%.3f'%(np.max(bpp_list))))
```

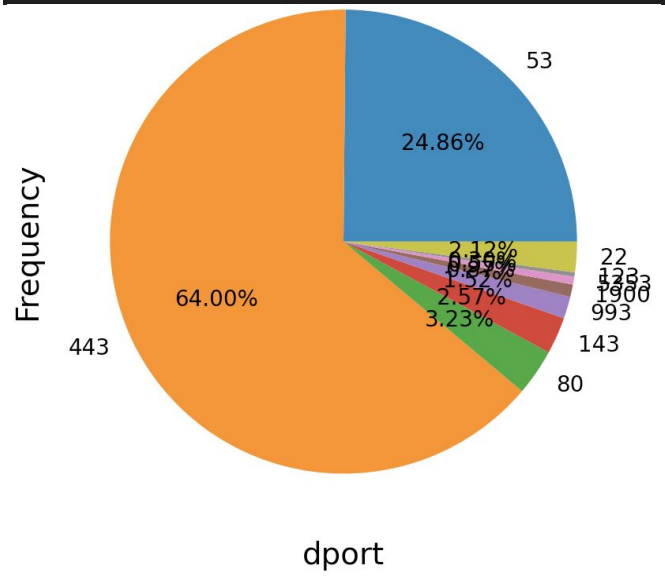
```
The min value is:  40.0
The 1st Qu. is:  82.119
The median value is:  204.982
The mean value is:  276.344
The 3st Qu. is:  398.091
The max value is:  1499.942
```

2.3 Flow data PS2

2.3.1 Code and plot for visualising flow distribution by port

This task is similar with “1.2 Flows by port numbers”, so I will not describe it again, just display the code and plot as follows:

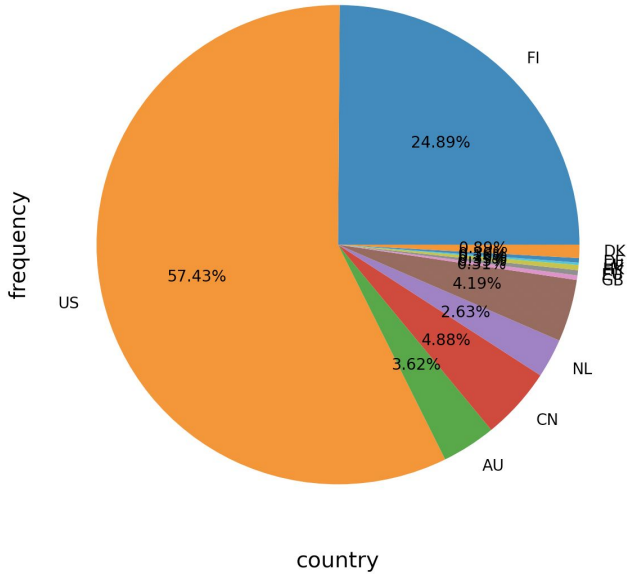
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet2/PS2.csv')
dportMap = {}
for index,row in df.iterrows():
     sport = row["sport"]
     dport = row["dport"]
     if sport >= dport:
            tmp = dport
     else:
            tmp = sport
     if tmp in dportMap:
            dportMap[tmp] = dportMap[tmp] + 1
     else:
            dportMap[tmp] = 1
x=[]
y=[]
for i in dportMap:
 if dportMap[i]>10:
     x.append(i)
     y.append(dportMap[i])
plt.plot(y,x)
plt.xlabel("dport",fontsize=14)
plt.ylabel("Frequency",fontsize=14)
plt.pie(y,labels=x,autopct='%3.2f%%')
plt.show()
```



2.3.2 Code and plot for visualising flow distribution by country

```
import subprocess
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet2/PS2.csv")
ip_list = []
for index, row in df.iterrows():
   if row["src"].startswith("192.168.1.100"):
       tmp = row["dst"]
```

```
    else:
        tmp = row["src"]
    ip_list.append(tmp)
ip_dir = {}
for i in ip_list:
    if i in ip_dir:
        ip_dir[i] = ip_dir[i] + 1
    else:
        ip_dir[i] = 1
res = {}
for i in ip_dir:
    tmp = subprocess.getoutput("whois %s |grep country -i -m 1 |cut -d ':' -f 2
|xargs"%i)
    if tmp in res:
        res[tmp] = res[tmp] + ip_dir[i]
    else:
        res[tmp] = ip_dir[i]
x = []
y = []
for i in res:
    x.append(i)
    y.append(res[i])
plt.plot(y,x)
plt.xlabel("country",fontsize=14)
plt.ylabel("frequency",fontsize=14)
plt.pie(y,labels=x,autopct='%3.2f%%')
plt.show()
```



2.3.3 Plot origin-destination pairs by both by data volume and by flows (Zipf type plot)

2.3.4 Flow length distribution

2.3.4.1 Key summary statistics of flow length

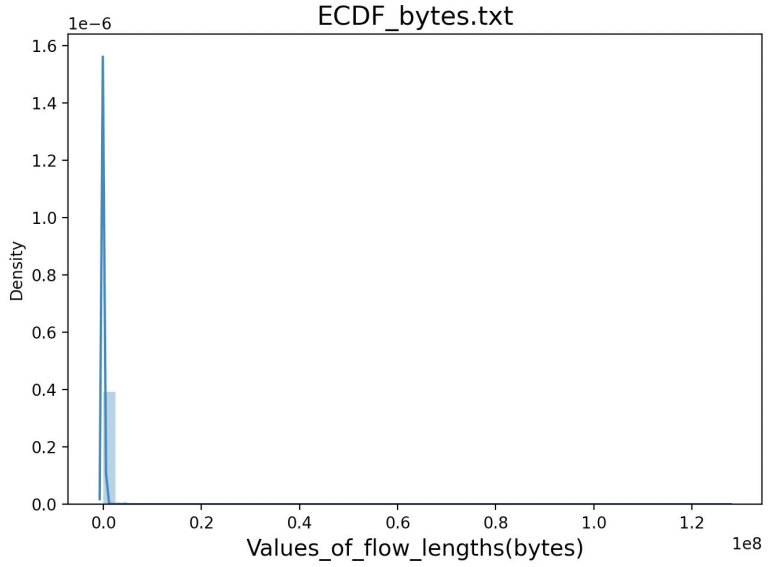
```
import numpy as np
import pandas as pd
data = pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet2/PS2.csv')
x=[]
for index, row in data.iterrows():
 x.append(row["bytes"])
print("The min value is: ", float('%.3f'%(np.min(x))))
print("The 1st Qu. is: ", float('%.3f'%(np.percentile(x, (25)))))
print("The median value is: ", float('%.3f'%(np.median(x))))
```

```
print("The mean value is: ", float('%.3f'%(np.mean(x))))
print("The 3st Qu. is: ", float('%.3f'%(np.percentile(x, (75)))))
print("The max value is: ", float('%.3f'%(np.max(x))))
```

```
The min value is:  40.0
The 1st Qu. is:  264.75
The median value is:  1153.0
The mean value is:  37087.722
The 3st Qu. is:  4778.25
The max value is:  127156080.0
```

2.3.4.2 Code and ECDF plot for the flow length distribution

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
data = pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet2/PS2.csv')
x=[]
for index, row in data.iterrows():
 x.append(row["bytes"])
plt.title("ECDF_bytes.txt",fontsize=16)
plt.xlabel("Values_of_flow_lengths(bytes)",fontsize=14)
sns.distplot(x)
plt.show()
```



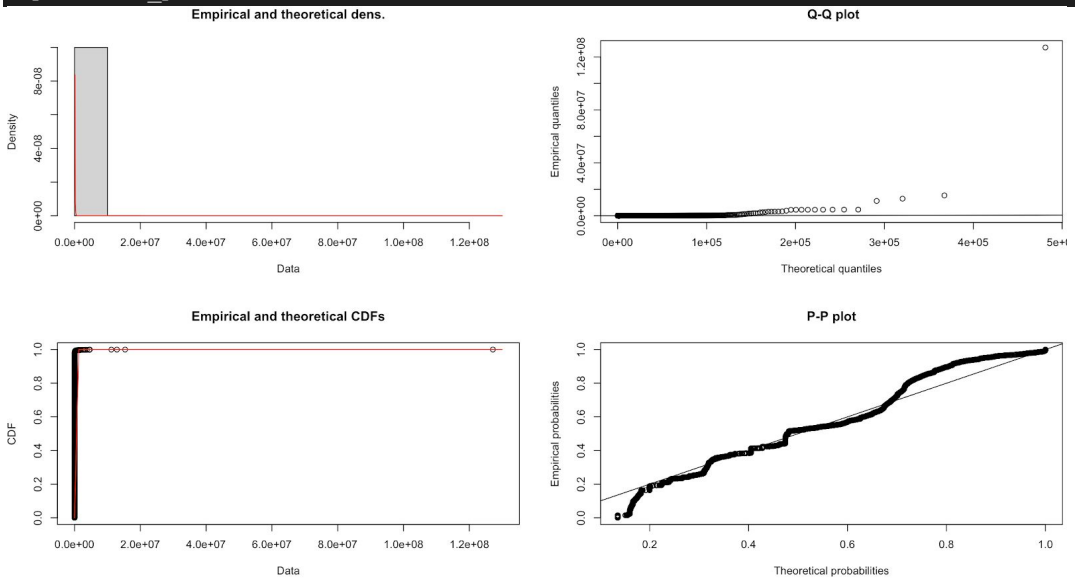
2.3.4.3 Distribution judgement for flow length

After several times trials of the fitdist function, I found that the flow length of my network volume follows “weibull” and “lnorm” distribution, which can be also explained by the pie plot for visualising flow distribution by port above, the data of my network volume highly continuous (the ports of most flows are 443, 53, 80), cause most of them are related to browsing websites.

- Code and plot for “weibull” distribution

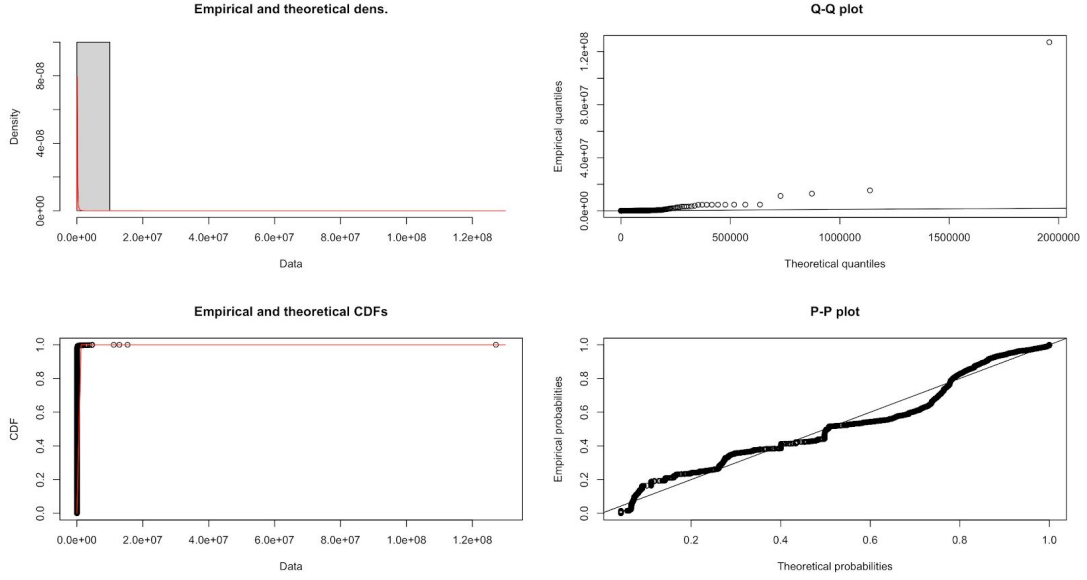
```
> PS2 <- read_csv("Desktop/ITMA/Assignment-Final/DataSet2/PS2.csv")
> bytes <- as.numeric(PS2$bytes)
> fit_para <- fitdist(bytes, 'weibull')
> print(fit_para)
Fitting of the distribution ' weibull ' by maximum likelihood
Parameters:
         estimate   Std. Error
shape    0.4468875  0.003094861
scale 2987.9281829 78.502154163
```

```
> plot(fit_para)
```



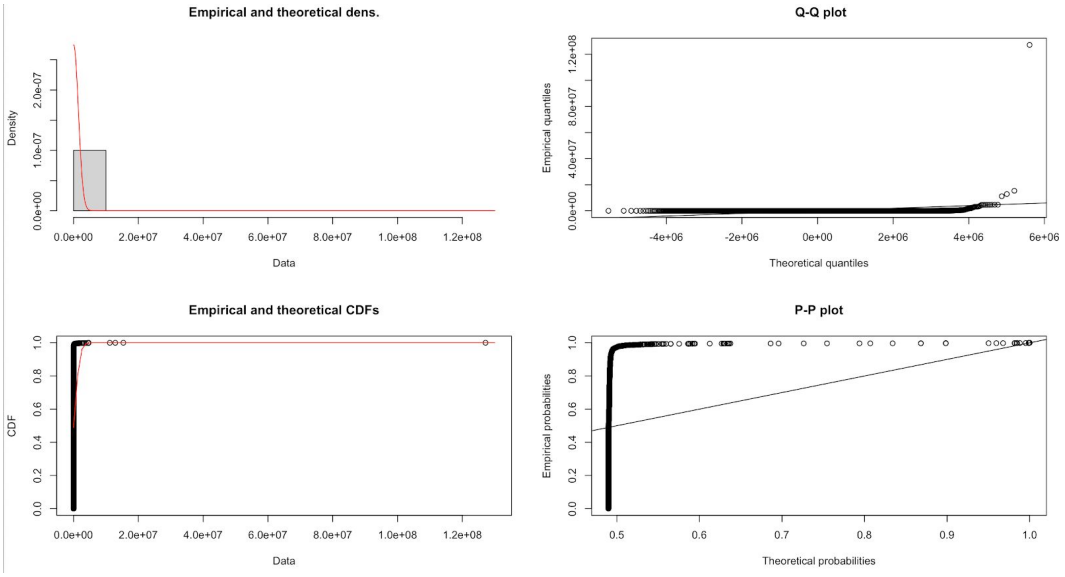
- Code and plot for “lnorm” distribution

```
> PS2 <- read_csv("Desktop/ITMA/Assignment-Final/DataSet2/PS2.csv")
> fit_para <- fitdist(bytes, 'lnorm')
> print(fit_para)
Fitting of the distribution ' lnorm ' by maximum likelihood
Parameters:
         estimate Std. Error
meanlog 7.022946 0.02166272
sdlog   1.944825 0.01531784
> plot(fit_para)
```



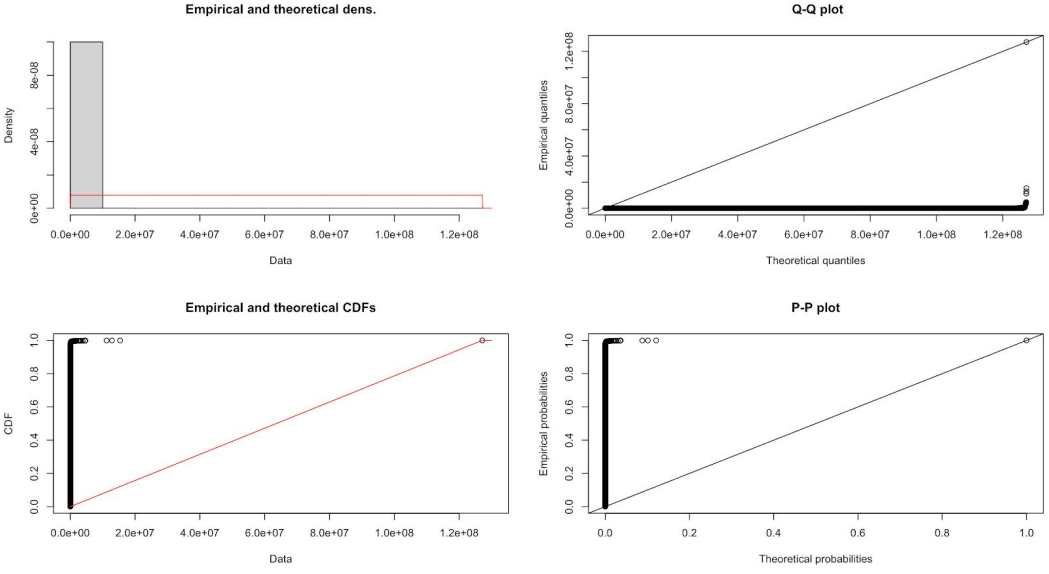
- Code and plot for “norm” distribution

```
> PS2 <- read_csv("Desktop/ITMA/Assignment-Final/DataSet2/PS2.csv")
> fit_para <- fitdist(bytes, 'norm')
> print(fit_para)
Fitting of the distribution ' norm ' by maximum likelihood
Parameters:
      estimate Std. Error
mean   37087.72         NA
sd   1451713.90         NA
> plot(fit_para)
```



- Code and plot for “unif” distribution

```
> PS2 <- read_csv("Desktop/ITMA/Assignment-Final/DataSet2/PS2.csv")
> fit_para <- fitdist(bytes, 'unif')
> print(fit_para)
Fitting of the distribution ' unif ' by maximum likelihood
Parameters:
    estimate Std. Error
min       40         NA
max 127156080         NA
> plot(fit_para)
```



2.4 TCP connection data PS3

2.4.1 Round-trip times and their variance

Got the following rtt.csv from PS3.csv

```
% head -n5 rtt.csv
RTT_samples_a2b,RTT_samples_b2a,RTT_min_a2b,RTT_min_b2a,RTT_max_a2b,RTT_max_b2a,RTT
_avg_a2b,RTT_avg_b2a,RTT_stdev_a2b,RTT_stdev_b2a
4,4,21.6,0.1,26.6,0.2,23.3,0.1,2.2,0.1
18942,1,0.0,110.6,1.7,110.6,0.2,110.6,0.1,0.0
2,0,0.2,0.0,0.2,0.0,0.2,0.0,0.0,0.0
```

```
2,0,0.1,0.0,0.1,0.0,0.1,0.0,0.0,0.0
% tail -n5 rtt.csv
3,3,24.4,0.1,54.8,0.2,43.9,0.2,16.9,0.0
3,3,24.2,0.2,52.8,0.3,39.4,0.2,14.4,0.1
8,9,336.2,0.1,417.6,0.4,391.7,0.2,28.4,0.1
1,0,364.7,0.0,364.7,0.0,364.7,0.0,0.0,0.0
1,0,340.9,0.0,340.9,0.0,340.9,0.0,0.0,0.0
```

```
> library(readr)
> rtt <- read_csv("Desktop/ITMA/Assignment-Final/DataSet2/rtt.csv")
> summary(rtt$RTT_samples_a2b)
   Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
   0.00     2.00     4.00    20.86     7.00 18942.00
> var(rtt$RTT_samples_a2b)
[1] 216749
> summary(rtt$RTT_samples_b2a)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.000   2.000   4.000   8.315   9.000 433.000
> var(rtt$RTT_samples_b2a)
[1] 468.3278
> summary(rtt$RTT_min_a2b)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.0    21.5    31.3   114.3   259.7  1067.3
> var(rtt$RTT_min_a2b)
[1] 19065.8
> summary(rtt$RTT_min_b2a)
   Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
 0.0000   0.0000   0.1000   0.1296   0.1000 110.6000
> var(rtt$RTT_min_b2a)
[1] 7.341965
> summary(rtt$RTT_max_a2b)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.0    29.0    63.4   154.3   301.6  1351.4
> var(rtt$RTT_max_a2b)
[1] 30701.14
> summary(rtt$RTT_max_b2a)
   Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
 0.0000   0.1000   0.2000   0.6793   0.3000 110.6000
> var(rtt$RTT_max_b2a)
[1] 49.83785
> summary(rtt$RTT_avg_a2b)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.0    23.5    43.9   129.4   285.9  1067.2
> var(rtt$RTT_avg_a2b)
[1] 22230.27
> summary(rtt$RTT_avg_b2a)
   Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
 0.0000   0.1000   0.1000   0.2005   0.2000 110.6000
> var(rtt$RTT_avg_b2a)
[1] 7.486021
> summary(rtt$RTT_stdev_a2b)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.00    0.00    7.20   15.24   19.00  466.60
> var(rtt$RTT_stdev_a2b)
[1] 792.1902
> summary(rtt$RTT_stdev_b2a)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.0000  0.0000  0.0000  0.1338  0.1000 31.8000
> var(rtt$RTT_stdev_b2a)
[1] 2.105314
```

2.4.2 Total traffic volume during the connection

```
import pandas as pd
from datetime import datetime
df = pd.read_csv("/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet2/PS2.csv")
bytes_all = 0
for index, row in df.iterrows():
```

```
   bytes_all = bytes_all + row["bytes"]
beginTime = 1E30
endTime = 0
for i in df["first"]:
  if beginTime > i:
      beginTime = i
for i in df["latest"]:
  if endTime < i:
      endTime = i
time = (datetime.utcfromtimestamp(endTime) -
datetime.utcfromtimestamp(beginTime)).seconds
print("The bps for the whole connection is: ",float('%.3f'%(bytes_all/time)))
```

```
The bps for the whole connection is:  35924.413
```

2.5 Conclusions

2.5.1 Traffic volume at different time scales

Even though the two plots are in different time scales, essentially the plots from the same data set, so they follow the same trend. And another finding is that around 9:00, there is a peak, cause at that time, I was having a video call with my family.

2.5.2 Characteristics of top 5 most common applications used

TCP 443 = Port for HTTPS
TCP 53= Port for DNS
TCP 80= Port for HTTP
TCP 143 = Port for IMAP
TCP 22 = Port for SSH protocol

2.5.3 Comparison of above results with result from data set FS2

There is a very obvious difference between FS2 and the result above shown in “Distribution judgement for flow length”, which is caused by the packets generated by different user behaviors. All the flows info in FS2 are highly discrete, but mine is continuous. Another difference caused by the same reason is also displayed through pie plots in the “visualising flow distribution by port” parts.

2.5.4 Differences of flow and packet measurements in the example case

In this case, overall, I can see that the flow measurements are more comprehensive and persuasive than packet measurements, for each flow, there are many info, like bytes, packets, ports, from all the info, we can calculate the bps, traffic volume distribution and density and so on. For the packet measurements, we can also get some conclusions, however, without the length of each packet, then the conclusion will not be persuasive.

2.5.5 Your findings on retransmissions

As we all know that the TCP retransmission mechanism ensures that data is reliably sent from end to end. If retransmissions are detected in a TCP connection, it is logical to assume that packet loss has occurred on the network somewhere between client and server. The basic theory also corresponds to the PS3.csv, as I can find the value of “**_retrans_a2b(b2a)” will not be 0 if there are packet loss.

Report, task3

3.0 Description about where the measurements are from and data pre-processing

3.0.1 Description about where the measurements are from

I collected all the data during the “Assignment 2: Basic measurements” with the following script on “force.aalto.fi” server, and from the following code, I tested 6 name servers for DNS and ICMP, 2 research servers (based on the last digit of my student number, calculated repeated result, that is why I only test 2 research servers), 2 iperf servers.

```
yanj3@force ~ % crontab -l
6 * * * * /u/94/yanj3/unix/task1/task1-1.sh >> /u/94/yanj3/unix/task1/task1-1 2>&1
6,16,26,36,46,56 * * * * /u/94/yanj3/unix/task1/task1-2.sh >>
/u/94/yanj3/unix/task1/task1-2 2>&1
6,16,26,36,46,56 * * * * /u/94/yanj3/unix/task1/task1-3.sh >>
/u/94/yanj3/unix/task1/task1-3 2>&1
6 * * * * /u/94/yanj3/unix/task2/task2.sh >> /u/94/yanj3/unix/task2/task2 2>&1
yanj3@force ~ % cat task1/task1-1.sh
for name_server in d.dns.br e.dns.br f.dns.br a.dns.br c.dns.br b.dns.br
do
date
dig @8.8.8.8 $name_server
ping $name_server -D -O -c1
done
yanj3@force ~ % cat task1/task1-2.sh
for research_server in pna-es.ark.caida.org hlz-nz.ark.caida.org
do
date
ping $research_server -c5
done
yanj3@force ~ % cat task1/task1-3.sh
for iperf_server in ok1.iperf.comnet-student.eu blr1.iperf.comnet-student.eu
do
date
ping $iperf_server -c5
echo "curl iperf_server is $iperf_server"
curl -o /dev/null http://$iperf_server/1K.bin -w "%{time_connect}\n"
Done
yanj3@force ~ % cat task2/task2.sh
date
echo "HTTP download tool"
curl -o /dev/null http://ok1.iperf.comnet-student.eu/500M.bin
echo "iperf3 tool - up"
iperf3 -c ok1.iperf.comnet-student.eu
echo "iperf3 tool - down"
iperf3 -c ok1.iperf.comnet-student.eu -R
echo "speedtest tool"
python3 /u/94/yanj3/unix/task2/speedtest.py
```

3.0.2 Data Pre-processing

Basically, I have shown all the processing methods (including shell cmds, like awk, sed, grep and so on, and also some manually operations and filtering) in the “Assignment 2: Basic measurements”, so I will not provide the tedious processing steps and only show the final dataset here.

During the process, I also manually deal with some data, like modify the msec as 2000 ms if the packet loss is 100% and no rtt value returned from the ping cmd.

3.0.2.1 AS1_NameServer_DNS.csv

```
yanjing@yanjingdeMacBook-Pro DataSet3 % head -n5 AS1_NameServer_DNS.csv
Date,a.dns.br_query_time(msec),b.dns.br_query_time(msec),c.dns.br_query_time(msec)
2020-10-28T00:06,24,24,28
2020-10-28T01:06,24,24,68
2020-10-28T02:06,24,24,268
2020-10-28T03:06,284,284,92
yanjing@yanjingdeMacBook-Pro DataSet3 % tail -n5 AS1_NameServer_DNS.csv
2020-11-13T11:06,24,28,28
2020-11-13T12:06,28,28,28
2020-11-13T13:06,24,28,104
2020-11-13T14:06,28,24,28
2020-11-13T15:06,348,28,268
```

3.0.2.2 AS1_NameServer_ICMP.csv

```
yanjing@yanjingdeMacBook-Pro DataSet3 % head -n5 AS1_NameServer_ICMP.csv
date,a.dns.br_pkt_loss,a.dns.br_rtt_max(msec),b.dns.br_pkt_loss,b.dns.br_rtt_max(ms
ec),c.dns.br_pkt_loss,c.dns.br_rtt_max(msec),
2020-10-28T00:06,0%,215.479,0%,173.102,0%,234.096
2020-10-28T01:06,0%,215.708,0%,173.206,0%,234.160
2020-10-28T02:06,0%,215.495,0%,173.272,0%,234.054
2020-10-28T03:06,0%,215.584,0%,173.129,0%,261.637
yanjing@yanjingdeMacBook-Pro DataSet3 % tail -n5 AS1_NameServer_ICMP.csv
2020-11-13T11:06,0%,212.997,0%,189.338,0%,245.654
2020-11-13T12:06,0%,212.414,0%,188.814,0%,245.438
2020-11-13T13:06,0%,211.025,0%,189.749,0%,245.868
2020-11-13T14:06,0%,210.163,0%,189.066,0%,245.651
2020-11-13T15:06,0%,228.938,0%,189.006,0%,271.615
```

3.0.2.3 AS1_ResearchServer_ICMP.csv

```
yanjing@yanjingdeMacBook-Pro DataSet3 % head -n5 AS1_ResearchServer_ICMP.csv
date,hlz_pkt_loss,hlz_rtt(msec),pna_pkt_loss,pna_rtt(msec)
2020-09-27T00:06:01 EEST,0%,302.253,20%,82.847
2020-09-27T00:06:06 EEST,0%,302.309,40%,82.871
2020-09-27T00:16:01 EEST,0%,302.288,40%,82.820
2020-09-27T00:16:06 EEST,0%,302.324,60%,82.875
yanjing@yanjingdeMacBook-Pro DataSet3 % tail -n5 AS1_ResearchServer_ICMP.csv
2020-10-05T07:36:06 EEST,0%,312.023,20%,86.499
2020-10-05T07:46:01 EEST,0%,312.280,20%,86.482
2020-10-05T07:46:07 EEST,0%,312.045,40%,86.507
2020-10-05T07:56:01 EEST,0%,312.320,0%,86.344
2020-10-05T07:56:06 EEST,0%,312.043,20%,86.326
```

3.0.2.4 AS1_IperfServer_ICMP.csv

```
yanjing@yanjingdeMacBook-Pro DataSet3 % head -n5 AS1_IperfServer_ICMP.csv
date,blz1_iperf_pkt_loss,blz1_iperf_rtt(msec),iperf_netlab_pkt_loss,iperf_netlab_rt
t(msec)
2020-09-27T00:06:01 EEST,0%,316.612,0%,0.710
2020-09-27T00:06:06 EEST,0%,316.505,0%,0.717
2020-09-27T00:16:01 EEST,0%,316.756,0%,0.673
2020-09-27T00:16:06 EEST,0%,316.653,0%,0.664
yanjing@yanjingdeMacBook-Pro DataSet3 % tail -n5 AS1_IperfServer_ICMP.csv
2020-10-05T07:26:01 EEST,0%,337.044,0%,0.731
2020-10-05T07:26:05 EEST,0%,335.800,0%,0.716
2020-10-05T07:36:01 EEST,0%,336.806,0%,2.072
2020-10-05T07:36:06 EEST,0%,335.702,0%,0.716
2020-10-05T07:46:01 EEST,0%,335.662,0%,0.788
```

3.0.2.5 AS2_iperf.csv

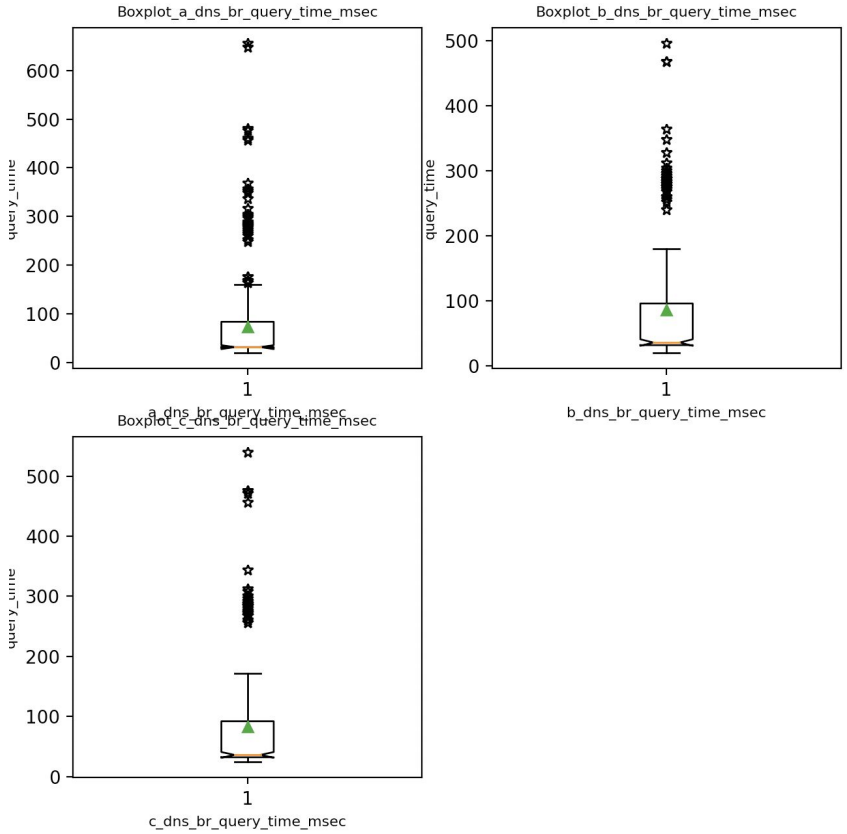
```
yanjing@yanjingdeMacBook-Pro DataSet3 % head -n5 AS2_iperf.csv
date,curl_http_throughput(M),iperf_throughput_up(Gbits/sec),iperf_throughput_down(G
bits/sec)
2020-11-01T00:06:01 EET,412,6.25,3.77
```

```
2020-11-01T01:06:01 EET,424,5.54,3.78
2020-11-01T02:06:01 EET,360,5.34,3.93
2020-11-01T03:06:01 EET,360,6.47,3.74
yanjing@yanjingdeMacBook-Pro DataSet3 % tail -n5 AS2_iperf.csv
2020-11-21T18:06:01 EET,750,8.43,5.19
2020-11-21T19:06:01 EET,582,8.57,9.21
2020-11-21T20:06:01 EET,544,8.51,9.20
2020-11-21T21:06:01 EET,494,8.27,9.20
2020-11-21T22:06:01 EET,637,7.72,9.06
```

3.1 Latency data plots (AS1.x)

3.1.1 Code and Box plot for AS1_NameServer_DNS.csv (No packet loss)

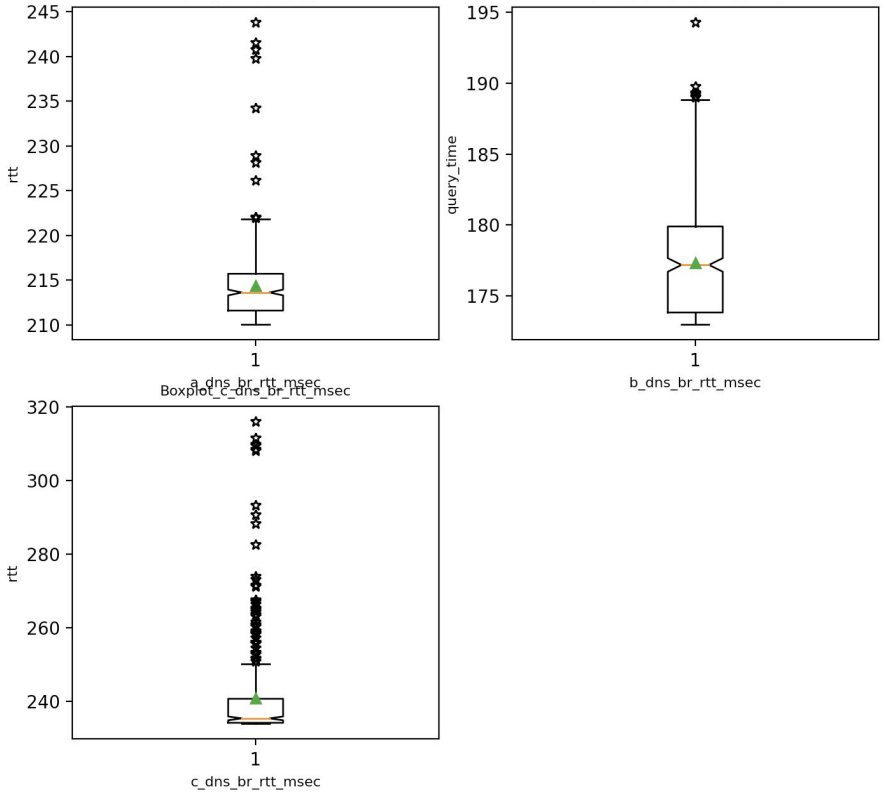
```
import pandas as pd
import matplotlib.pyplot as plt
data =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet3/AS1_NameServer_D
NS.csv')
a_dns_br_query_time_msec = []
for index, row in data.iterrows():
 a_dns_br_query_time_msec.append(row["a.dns.br_query_time(msec)"])
b_dns_br_query_time_msec = []
for index, row in data.iterrows():
 b_dns_br_query_time_msec.append(row["b.dns.br_query_time(msec)"])
c_dns_br_query_time_msec = []
for index, row in data.iterrows():
 c_dns_br_query_time_msec.append(row["c.dns.br_query_time(msec)"])
plt.figure(1)
plt.subplot(221)
plt.title("Boxplot_a_dns_br_query_time_msec",fontsize=8)
plt.xlabel("a_dns_br_query_time_msec",fontsize=8)
plt.ylabel("query_time",fontsize=8)
plt.boxplot(a_dns_br_query_time_msec, showmeans=True, notch = True, sym = '*')
plt.subplot(222)
plt.title("Boxplot_b_dns_br_query_time_msec",fontsize=8)
plt.xlabel("b_dns_br_query_time_msec",fontsize=8)
plt.ylabel("query_time",fontsize=8)
plt.boxplot(b_dns_br_query_time_msec, showmeans=True, notch = True, sym = '*')
plt.subplot(223)
plt.title("Boxplot_c_dns_br_query_time_msec",fontsize=8)
plt.xlabel("c_dns_br_query_time_msec",fontsize=8)
plt.ylabel("query_time",fontsize=8)
plt.boxplot(c_dns_br_query_time_msec, showmeans=True, notch = True, sym = '*')
plt.show()
```



3.1.2 Code and Box plot for AS1_NameServer_ICMP.csv without packet loss

```
import pandas as pd
import matplotlib.pyplot as plt
data =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet3/AS1_NameServer_I
CMP.csv')
a_dns_br_rtt_msec = []
b_dns_br_rtt_msec = []
c_dns_br_rtt_msec = []
########
for index, row in data.iterrows():
      if row["a.dns.br_rtt_max(msec)"] < 2000:
            a_dns_br_rtt_msec.append(row["a.dns.br_rtt_max(msec)"])
for index, row in data.iterrows():
      if row["b.dns.br_rtt_max(msec)"] < 2000:
            b_dns_br_rtt_msec.append(row["b.dns.br_rtt_max(msec)"])
for index, row in data.iterrows():
      if row["c.dns.br_rtt_max(msec)"] < 2000:
            c_dns_br_rtt_msec.append(row["c.dns.br_rtt_max(msec)"])
########
plt.figure(1)
plt.subplot(221)
plt.title("Boxplot_a_dns_br_rtt_msec",fontsize=8)
plt.xlabel("a_dns_br_rtt_msec",fontsize=8)
plt.ylabel("rtt",fontsize=8)
plt.boxplot(a_dns_br_rtt_msec, showmeans=True, notch = True, sym = '*')
plt.subplot(222)
plt.title("Boxplot_b_dns_br_query_time_msec",fontsize=8)
plt.xlabel("b_dns_br_rtt_msec",fontsize=8)
plt.ylabel("query_time",fontsize=8)
plt.boxplot(b_dns_br_rtt_msec, showmeans=True, notch = True, sym = '*')
plt.subplot(223)
plt.title("Boxplot_c_dns_br_rtt_msec",fontsize=8)
plt.xlabel("c_dns_br_rtt_msec",fontsize=8)
plt.ylabel("rtt",fontsize=8)
```

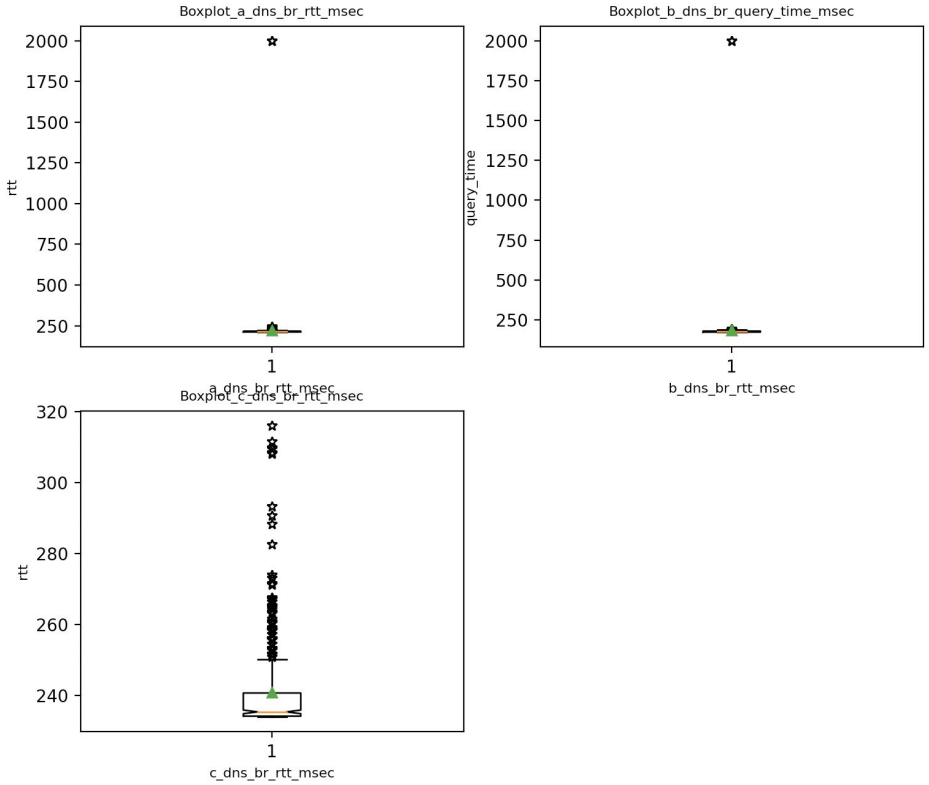
```
plt.boxplot(c_dns_br_rtt_msec, showmeans=True, notch = True, sym = '*')
plt.show()
```



Boxplot_c_dns_br_rtt_msec

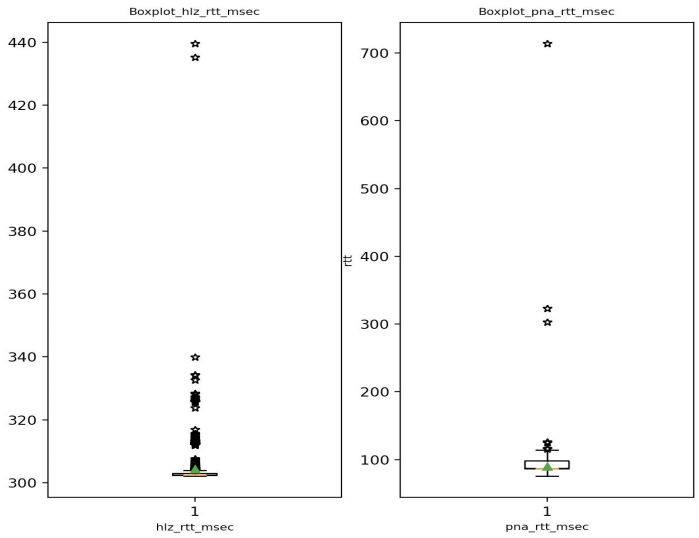
3.1.3 Code and Box plot for AS1_NameServer_ICMP.csv with packet loss

```
# Just modify the code among "########" in 3.1.2 as follows
for index, row in data.iterrows():
      a_dns_br_rtt_msec.append(row["a.dns.br_rtt_max(msec)"])
for index, row in data.iterrows():
      b_dns_br_rtt_msec.append(row["b.dns.br_rtt_max(msec)"])
for index, row in data.iterrows():
      c_dns_br_rtt_msec.append(row["c.dns.br_rtt_max(msec)"])
```



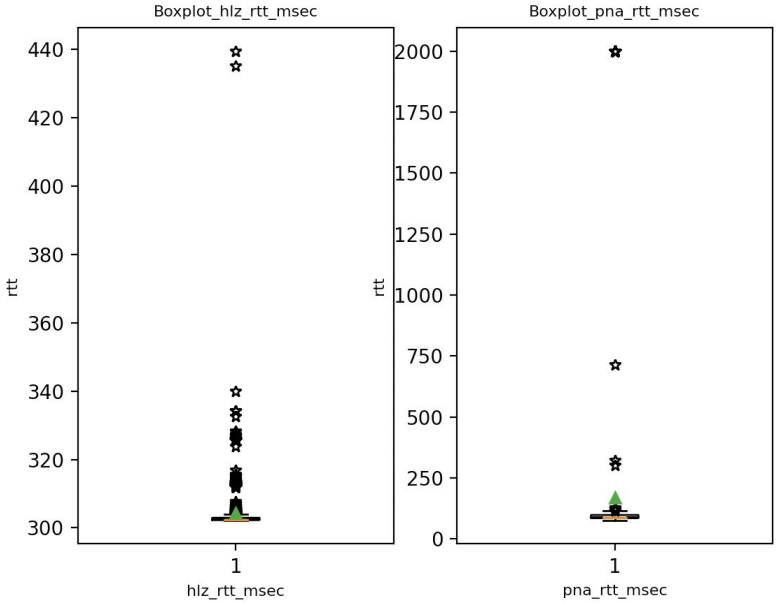
3.1.4 Code and Box plot for AS1_ResearchServer_ICMP.csv without packet loss

```
import pandas as pd
import matplotlib.pyplot as plt
data =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet3/AS1_ResearchServ
er_ICMP.csv')
hlz_rtt_msec = []
pna_rtt_msec = []
########
for index, row in data.iterrows():
      if row["hlz_rtt(msec)"] < 2000:
            hlz_rtt_msec.append(row["hlz_rtt(msec)"])
for index, row in data.iterrows():
      if row["pna_rtt(msec)"] < 2000:
            pna_rtt_msec.append(row["pna_rtt(msec)"])
########
plt.figure(1)
plt.subplot(121)
plt.title("Boxplot_hlz_rtt_msec",fontsize=8)
plt.xlabel("hlz_rtt_msec",fontsize=8)
plt.ylabel("rtt",fontsize=8)
plt.boxplot(hlz_rtt_msec, showmeans=True, notch = True, sym = '*')
plt.subplot(122)
plt.title("Boxplot_pna_rtt_msec",fontsize=8)
plt.xlabel("pna_rtt_msec",fontsize=8)
plt.ylabel("rtt",fontsize=8)
plt.boxplot(pna_rtt_msec, showmeans=True, notch = True, sym = '*')
plt.show()
```



3.1.5 Code and Box plot for AS1_ResearchServer_ICMP.csv with packet loss

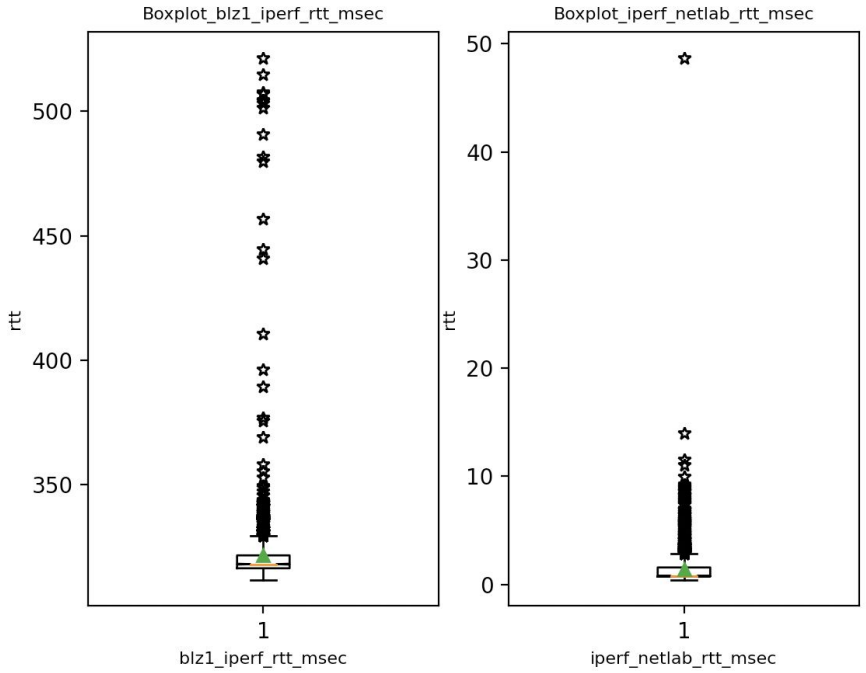
```
# Just modify the code among "########" in 3.1.4 as follows
for index, row in data.iterrows():
      hlz_rtt_msec.append(row["hlz_rtt(msec)"])
for index, row in data.iterrows():
      pna_rtt_msec.append(row["pna_rtt(msec)"])
```



3.1.6 Code and Box plot for AS1_IperfServer_ICMP.csv without packet loss

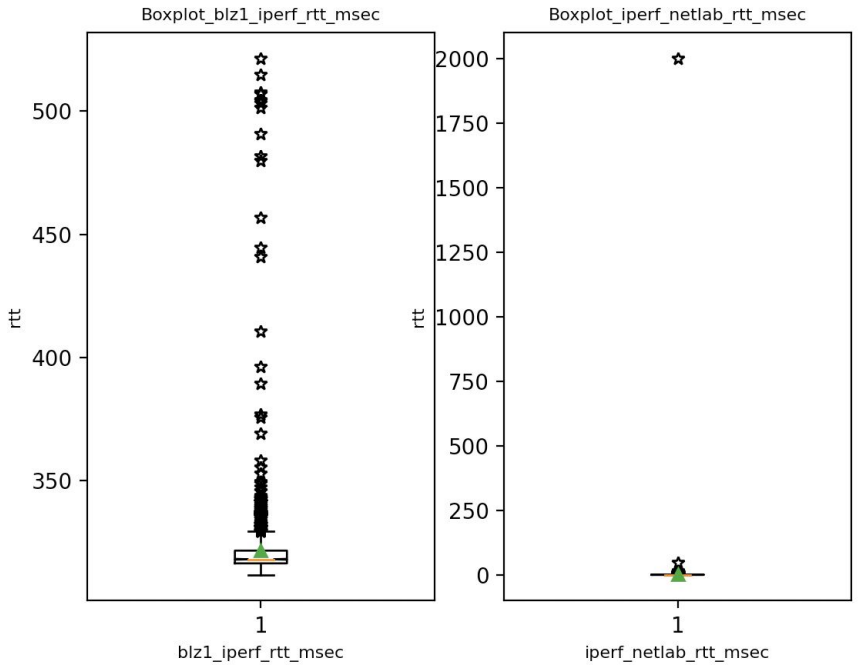
```
import pandas as pd
import matplotlib.pyplot as plt
data =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet3/AS1_IperfServer_
ICMP.csv')
blz1_iperf_rtt_msec = []
iperf_netlab_rtt_msec = []
########
for index, row in data.iterrows():
     if row["blz1_iperf_rtt(msec)"] < 2000:
           blz1_iperf_rtt_msec.append(row["blz1_iperf_rtt(msec)"])
for index, row in data.iterrows():
```

```
    if row["iperf_netlab_rtt(msec)"] < 2000:
            iperf_netlab_rtt_msec.append(row["iperf_netlab_rtt(msec)"])
########
plt.figure(1)
plt.subplot(121)
plt.title("Boxplot_blz1_iperf_rtt_msec",fontsize=8)
plt.xlabel("blz1_iperf_rtt_msec",fontsize=8)
plt.ylabel("rtt",fontsize=8)
plt.boxplot(blz1_iperf_rtt_msec, showmeans=True, notch = True, sym = '*')
plt.subplot(122)
plt.title("Boxplot_iperf_netlab_rtt_msec",fontsize=8)
plt.xlabel("iperf_netlab_rtt_msec",fontsize=8)
plt.ylabel("rtt",fontsize=8)
plt.boxplot(iperf_netlab_rtt_msec, showmeans=True, notch = True, sym = '*')
plt.show()
```



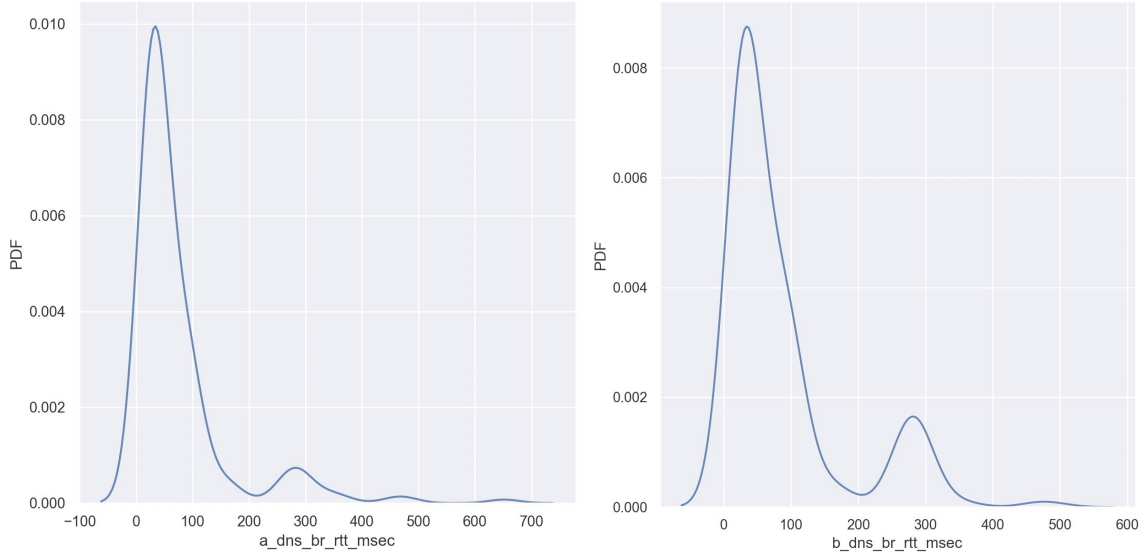
3.1.7 Code and Box plot for AS1_IperfServer_ICMP.csv with packet loss

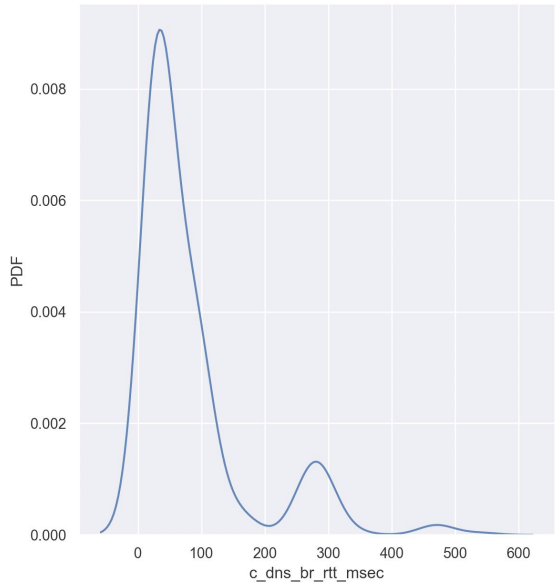
```
# Just modify the code among "########" in 3.1.6 as follows
for index, row in data.iterrows():
      blz1_iperf_rtt_msec.append(row["blz1_iperf_rtt(msec)"])
for index, row in data.iterrows():
      iperf_netlab_rtt_msec.append(row["iperf_netlab_rtt(msec)"])
```



3.1.8 Code and PDF plot for AS1_NameServer_DNS.csv

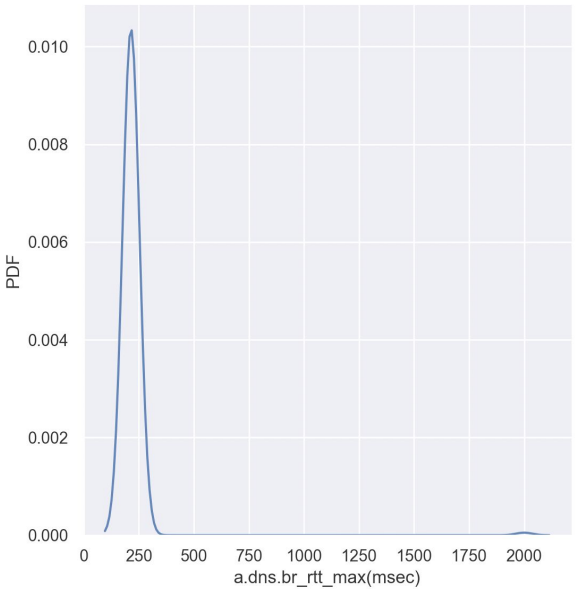
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(color_codes=True)
sns.set(rc={'figure.figsize':(5,5)})
data =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet3/AS1_NameServer_D
NS.csv')
a = []
for index, row in data.iterrows():
   a.append(row["a.dns.br_query_time(msec)"])
ax = sns.displot(a, kind="kde")
ax.set(xlabel='a_dns_br_rtt_msec', ylabel='PDF')
plt.show()
# Modify the code above for b.dns.br and c.dns.br
```

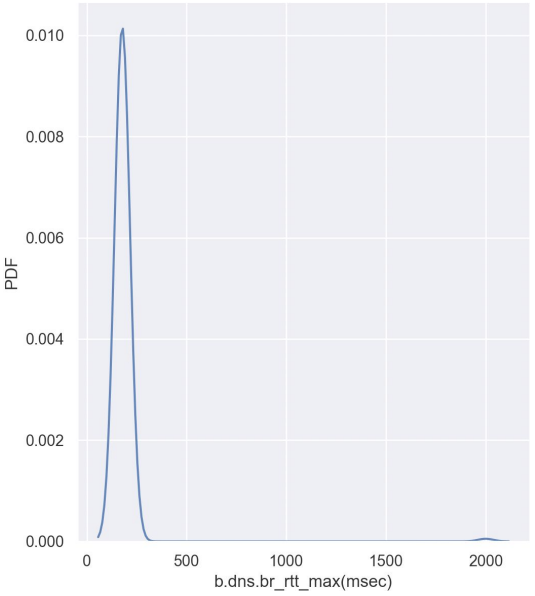


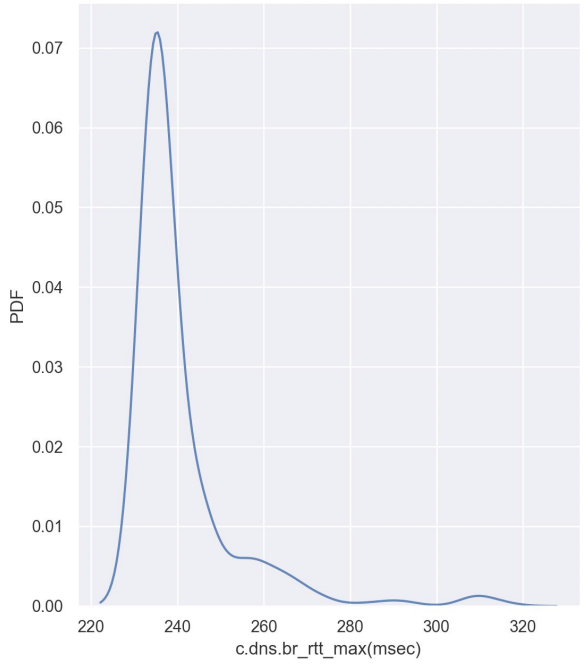


3.1.9 Code and PDF plot for AS1_NameServer_ICMP.csv

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(color_codes=True)
sns.set(rc={'figure.figsize':(5,5)})
data =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet3/AS1_NameServer_I
CMP.csv')
x = []
for index, row in data.iterrows():
   x.append(row["a.dns.br_rtt_max(msec)"])
ax = sns.displot(x, kind="kde")
ax.set(xlabel='a.dns.br_rtt_max(msec)', ylabel='PDF')
plt.show()
# Modify the code above for b.dns.br and c.dns.br
```

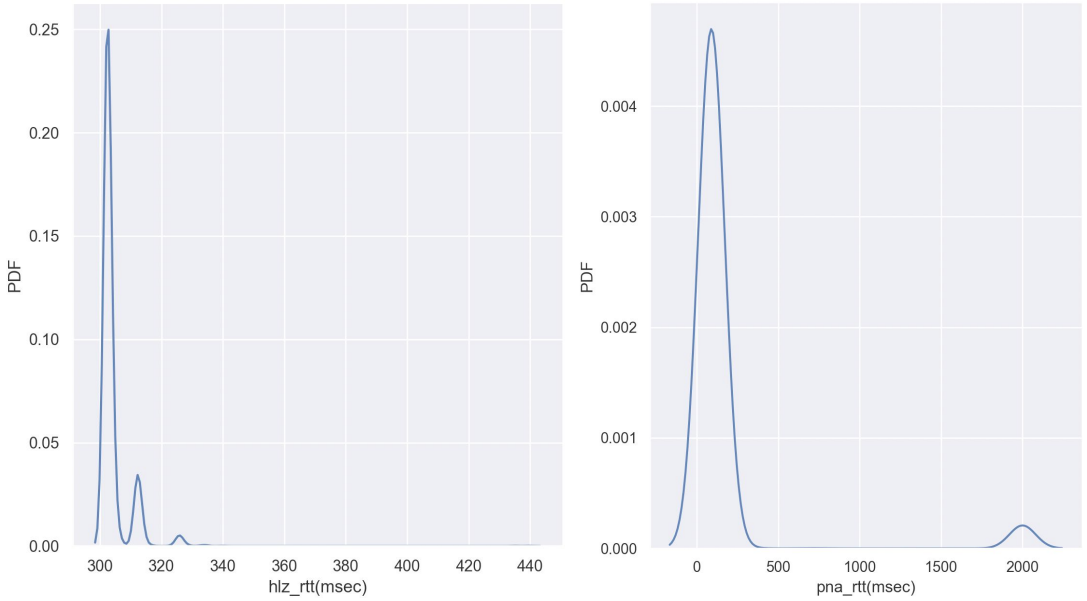






3.1.10 Code and PDF plot for AS1_ResearchServer_ICMP.csv

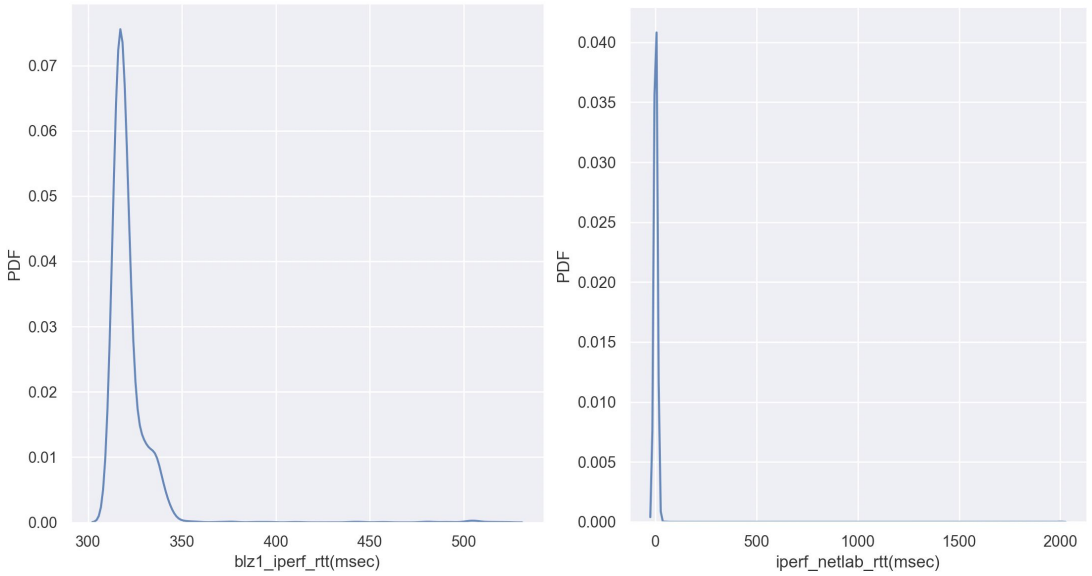
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(color_codes=True)
sns.set(rc={'figure.figsize':(5,5)})
data =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet3/AS1_ResearchServ
er_ICMP.csv')
x = []
for index, row in data.iterrows():
   x.append(row["hlz_rtt(msec)"])
ax = sns.displot(x, kind="kde")
ax.set(xlabel='hlz_rtt(msec)', ylabel='PDF')
plt.show()
# Modify the code above for pna
```



3.1.11 Code and PDF plot for AS1_IperfServer_ICMP.csv

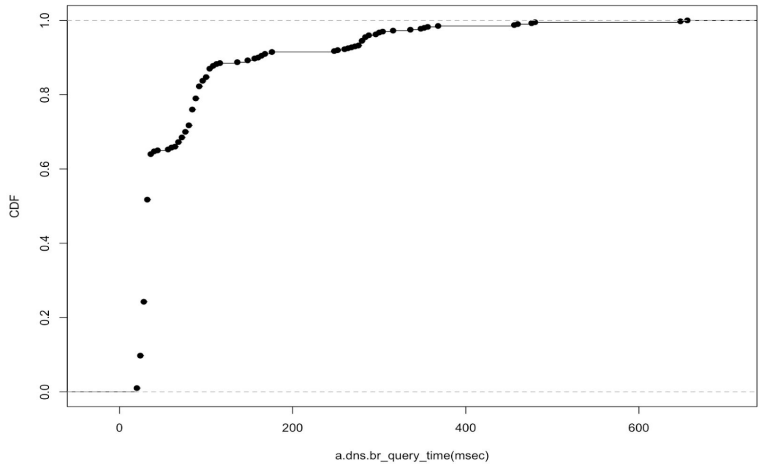
```
import pandas as pd
```

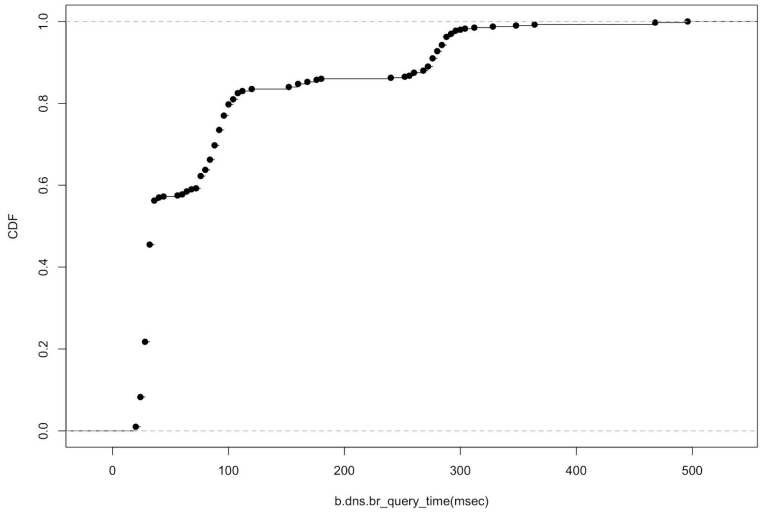
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(color_codes=True)
sns.set(rc={'figure.figsize':(5,5)})
data =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet3/AS1_IperfServer_
ICMP.csv')
x = []
for index, row in data.iterrows():
   x.append(row["blz1_iperf_rtt(msec)"])
ax = sns.displot(x, kind="kde")
ax.set(xlabel='blz1_iperf_rtt(msec)', ylabel='PDF')
plt.show()
# Modify the code above for iperf
```

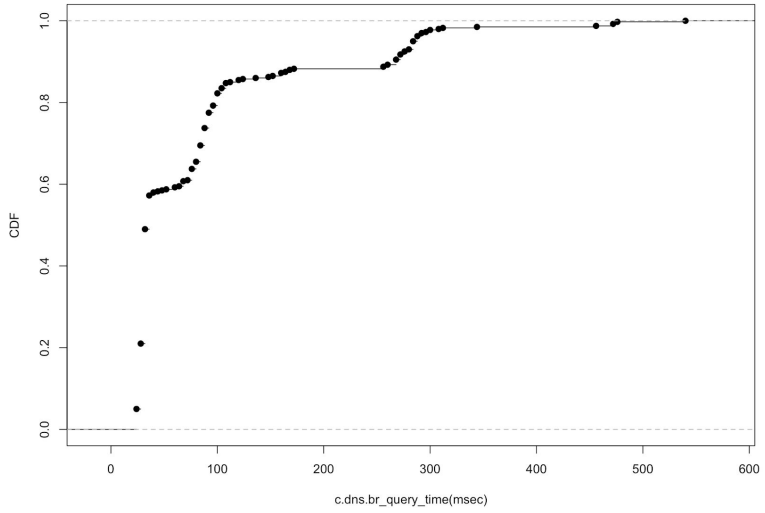


3.1.12 Code and CDF plot for AS1_NameServer_DNS.csv

```
> plot(ecdf(AS1_NameServer_DNS$`a.dns.br_query_time(msec)`), main = "CDF",  ann =
F)
> title(xlab = "a.dns.br_query_time(msec)", ylab = "CDF")
# Modify the code above for b.dns.br and c.dns.br
```

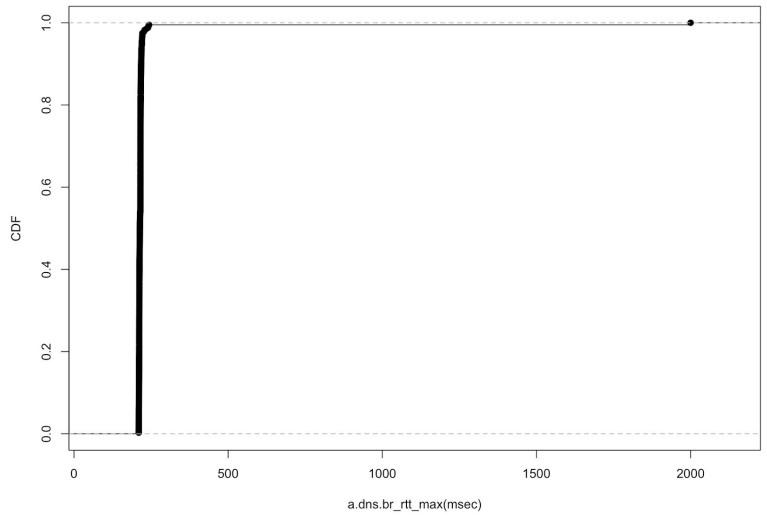


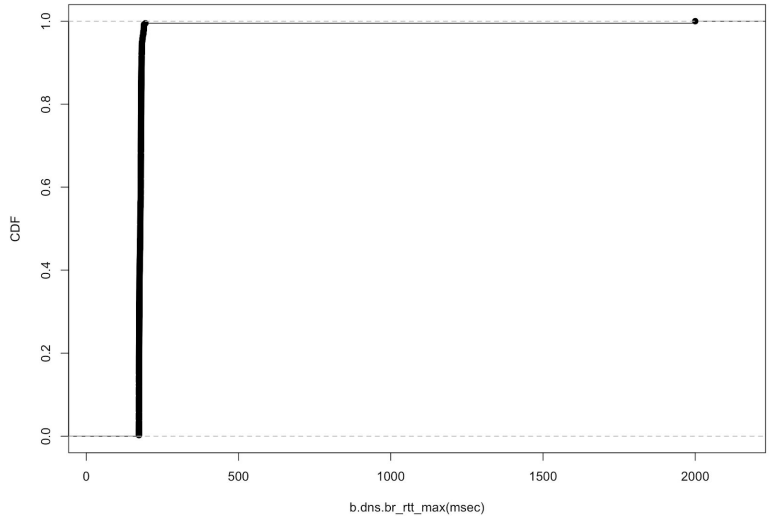


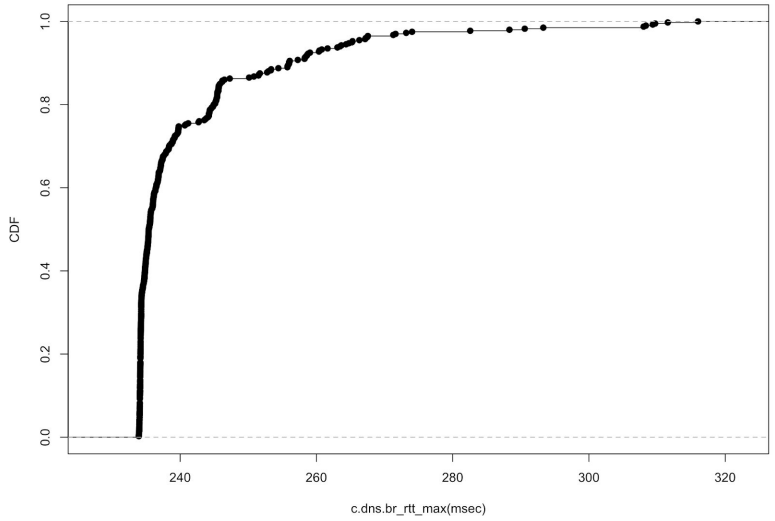


3.1.13 Code and CDF plot for AS1_NameServer_ICMP.csv

```
> plot(ecdf(AS1_NameServer_ICMP$`a.dns.br_rtt_max(msec)`), main = "CDF",  ann = F)
> title(xlab = "a.dns.br_rtt_max(msec)", ylab = "CDF")
# Modify the code above for b.dns.br and c.dns.br
```

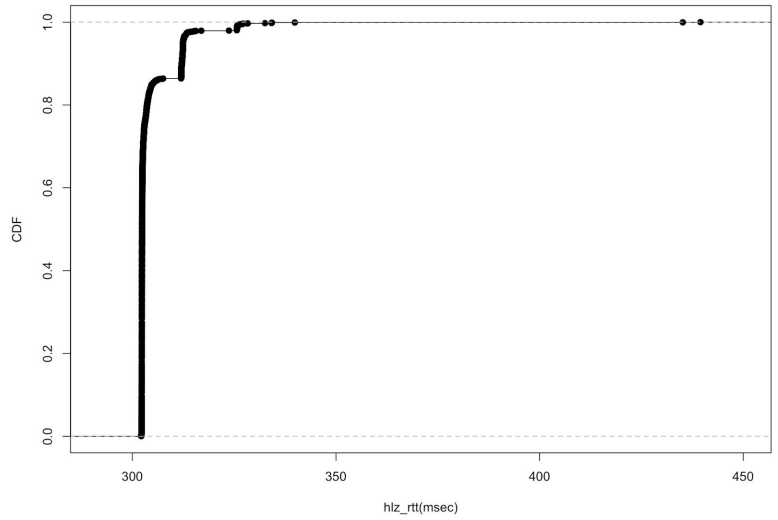


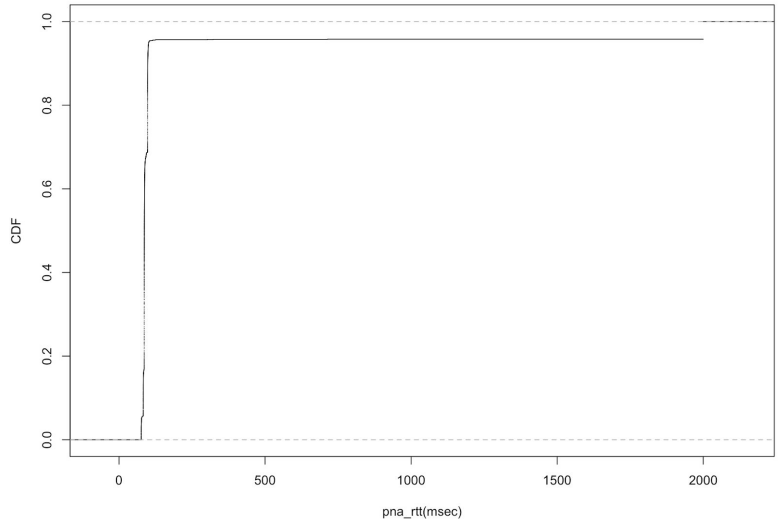




3.1.14 Code and CDF plot for AS1_ResearchServer_ICMP.csv

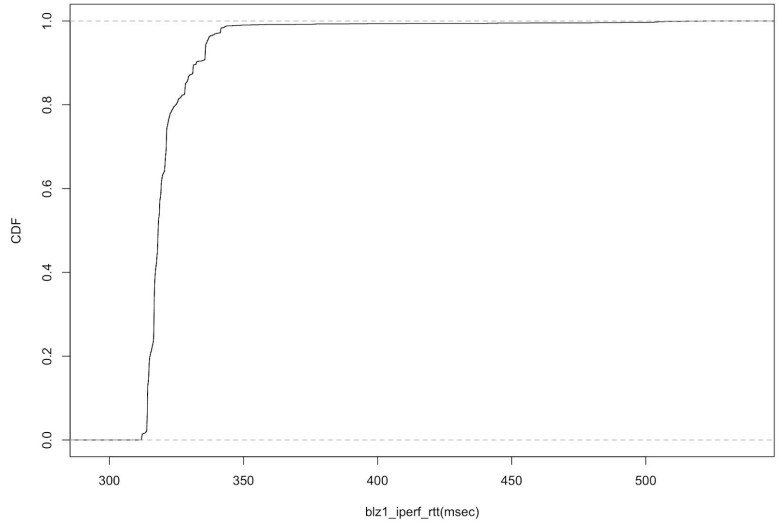
```
> plot(ecdf(AS1_ResearchServer_ICMP$`hlz_rtt(msec)`), main = "CDF",  ann = F)
> title(xlab = "hlz_rtt(msec)", ylab = "CDF")
# Modify the code above for pna
```

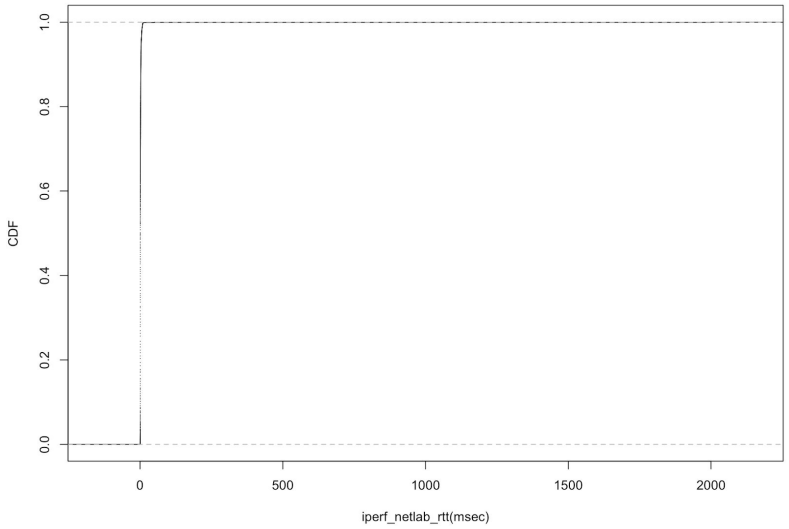




3.1.15 Code and CDF plot for AS1_IperfServer_ICMP.csv

```
> plot(ecdf(AS1_IperfServer_ICMP$`blz1_iperf_rtt(msec)`), main = "CDF",  ann = F)
> title(xlab = "blz1_iperf_rtt(msec)", ylab = "CDF")
# Modify the code above for iperf
```



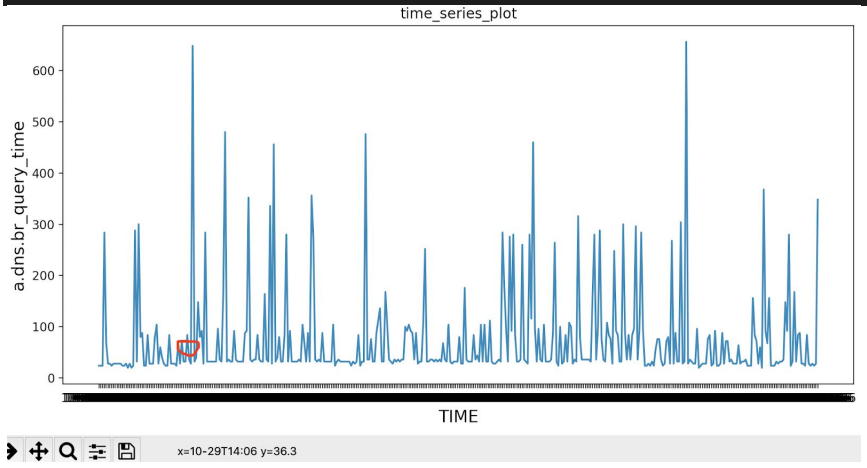


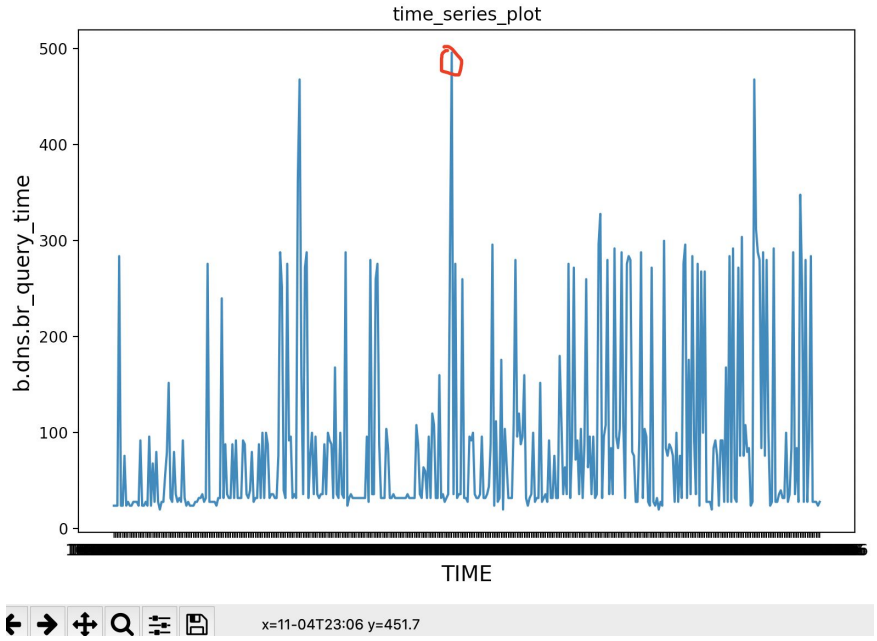
3.2 Latency data time series

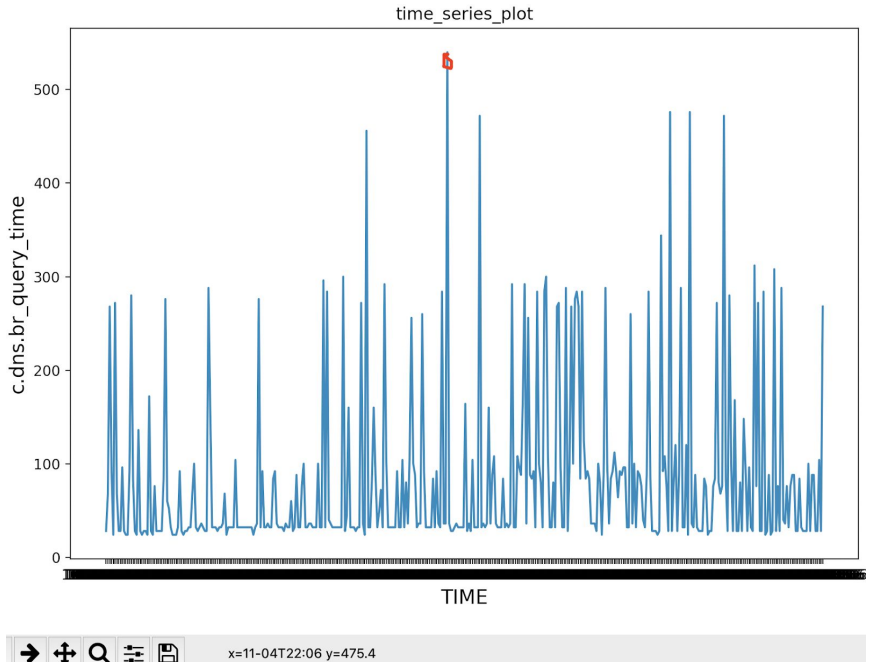
3.2.1 Code and plot time series of each data set AS1.x

3.2.1.1 AS1_NameServer_DNS.csv

```
import pandas as pd
import matplotlib.pyplot as plt
df =
pd.read_csv(r'/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet3/AS1_NameServer_
DNS.csv')
plt.plot(df['Date'],df['c.dns.br_query_time(msec)'])
plt.xlabel("TIME",fontsize=14)
plt.ylabel("c.dns.br_query_time",fontsize=14)
plt.title("time_series_plot")
plt.show()
# Modify code for a.dns.br and b.dns.br
```

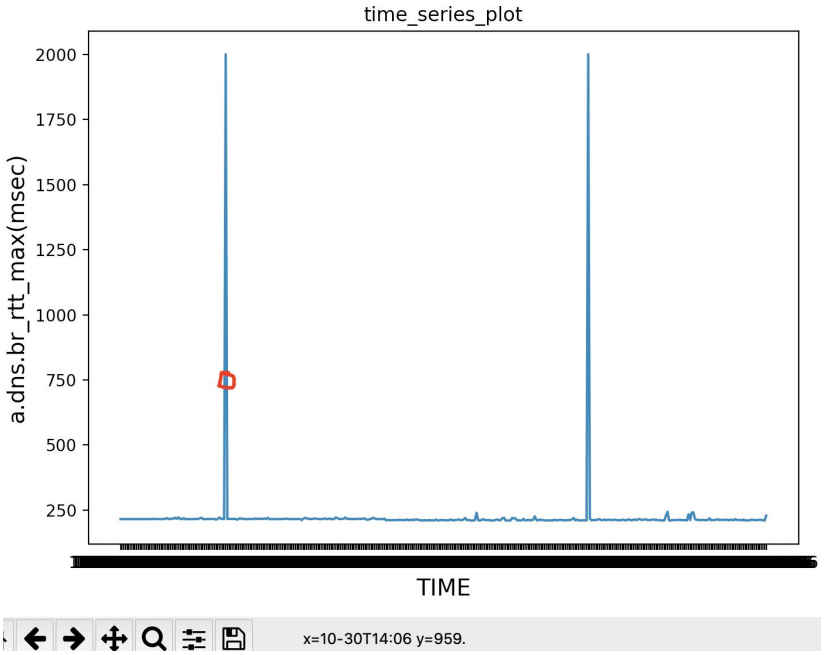


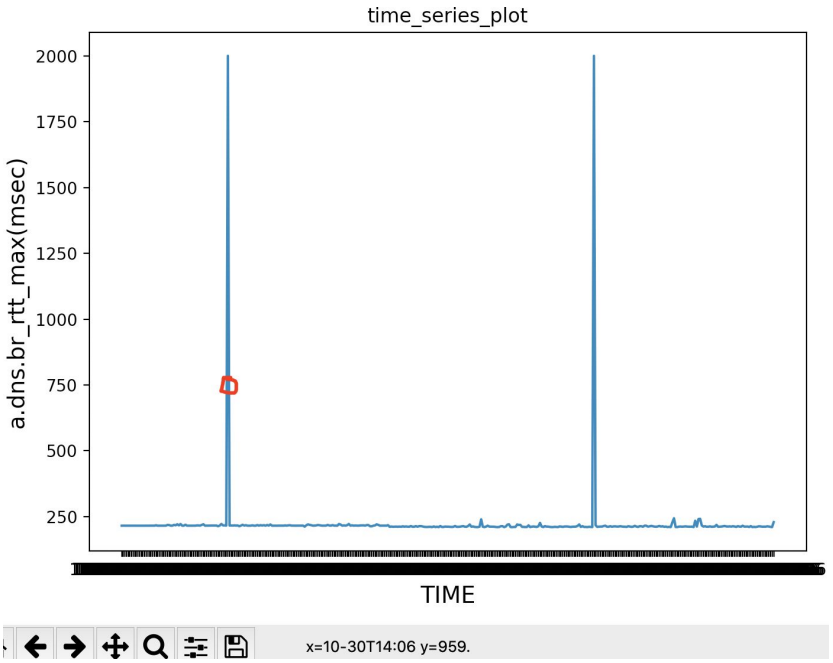


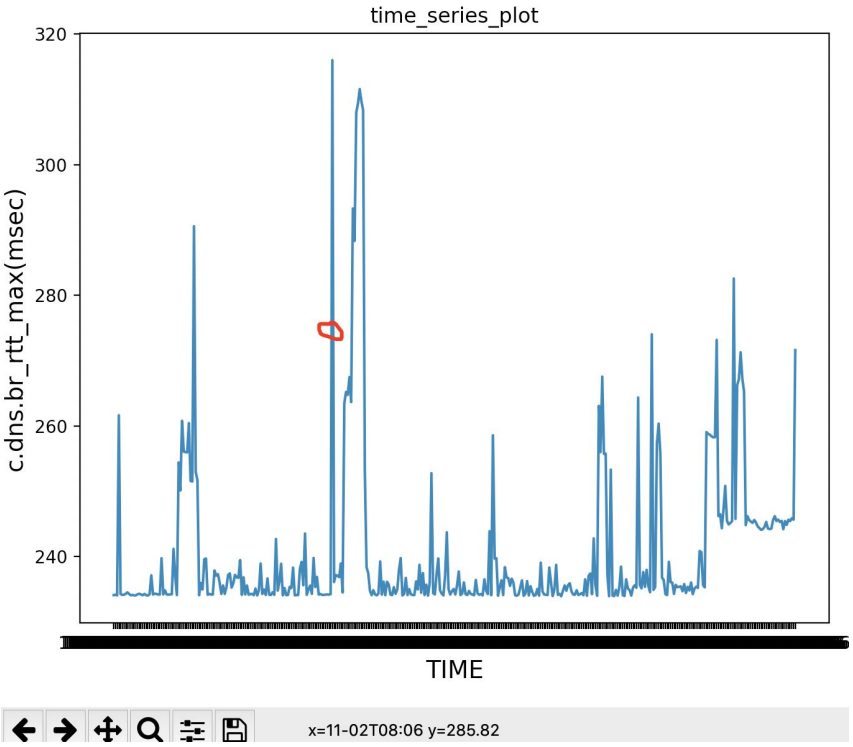


3.2.1.2 AS1_NameServer_ICMP.csv

```
import pandas as pd
import matplotlib.pyplot as plt
df =
pd.read_csv(r'/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet3/AS1_NameServer_
ICMP.csv')
plt.plot(df['date'],df['c.dns.br_rtt_max(msec)'])
plt.xlabel("TIME",fontsize=14)
plt.ylabel("c.dns.br_rtt_max(msec)",fontsize=14)
plt.title("time_series_plot")
plt.show()
```

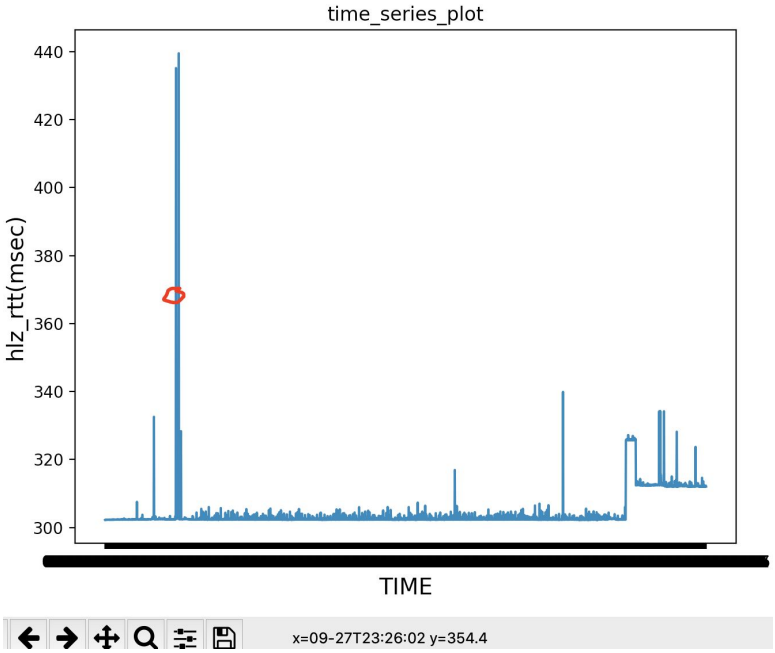

time_series_plot
a.dns.br_rtt_max(msec)
2000
1750
1500
1250
1000
750
500
250
TIME
x=10-30T14:06 y=959.

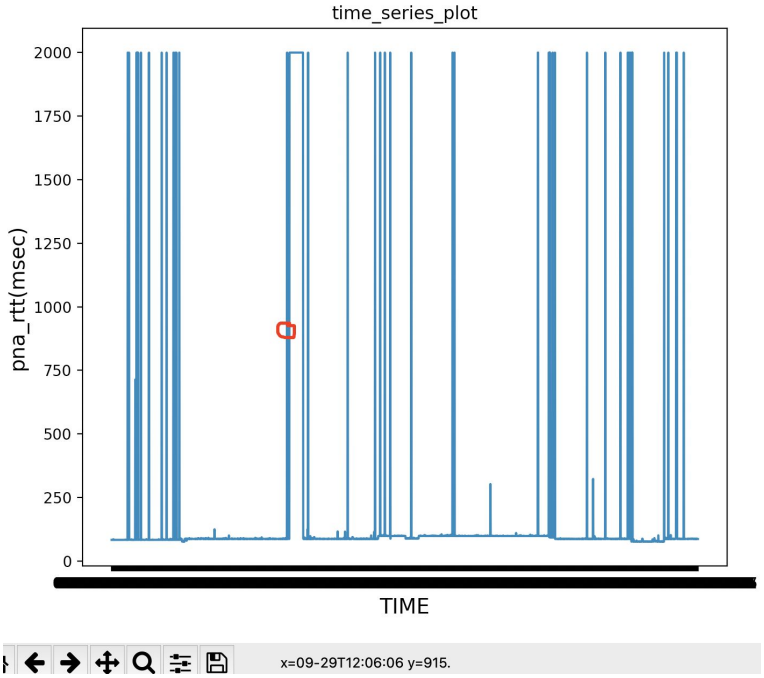

time_series_plot
a.dns.br_rtt_max(msec)
2000
1750
1500
1250
1000
750
500
250
TIME
x=10-30T14:06 y=959.



3.2.1.3 AS1_ResearchServer_ICMP.csv

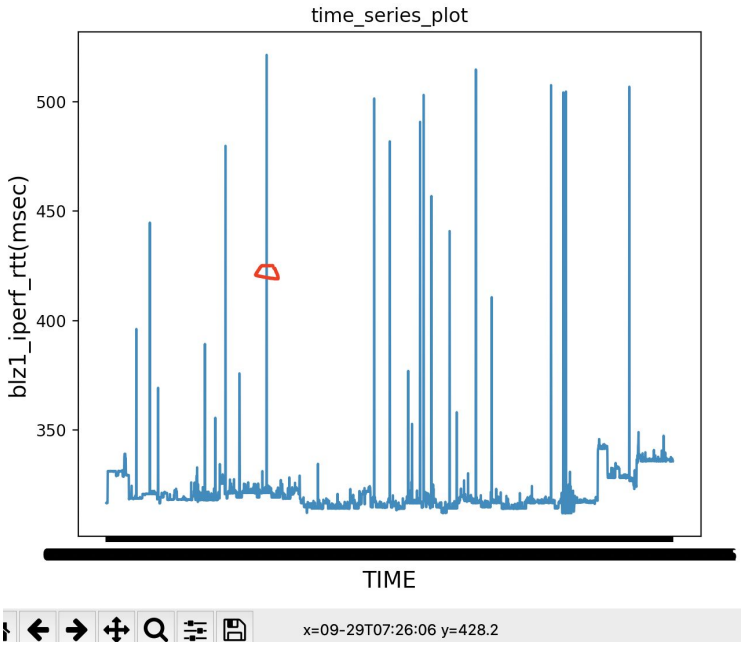
```
import pandas as pd
import matplotlib.pyplot as plt
df =
pd.read_csv(r'/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet3/AS1_ResearchSer
ver_ICMP.csv')
plt.plot(df['date'],df['pna_rtt(msec)'])
plt.xlabel("TIME",fontsize=14)
plt.ylabel("pna_rtt(msec)",fontsize=14)
plt.title("time_series_plot")
plt.show()
# Modify the code for hlz server
```

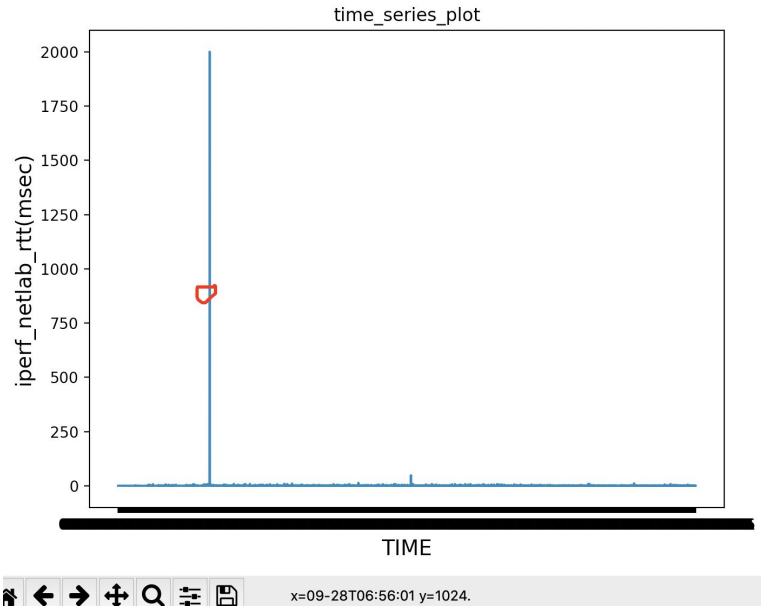




3.2.1.4 AS1_IperfServer_ICMP.csv

```
import pandas as pd
import matplotlib.pyplot as plt
df =
pd.read_csv(r'/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet3/AS1_IperfServer
_ICMP.csv')
plt.plot(df['date'],df['iperf_netlab_rtt(msec)'])
plt.xlabel("TIME",fontsize=14)
plt.ylabel("iperf_netlab_rtt(msec)",fontsize=14)
plt.title("time_series_plot")
plt.show()
# Modify the code for blz1 server
```

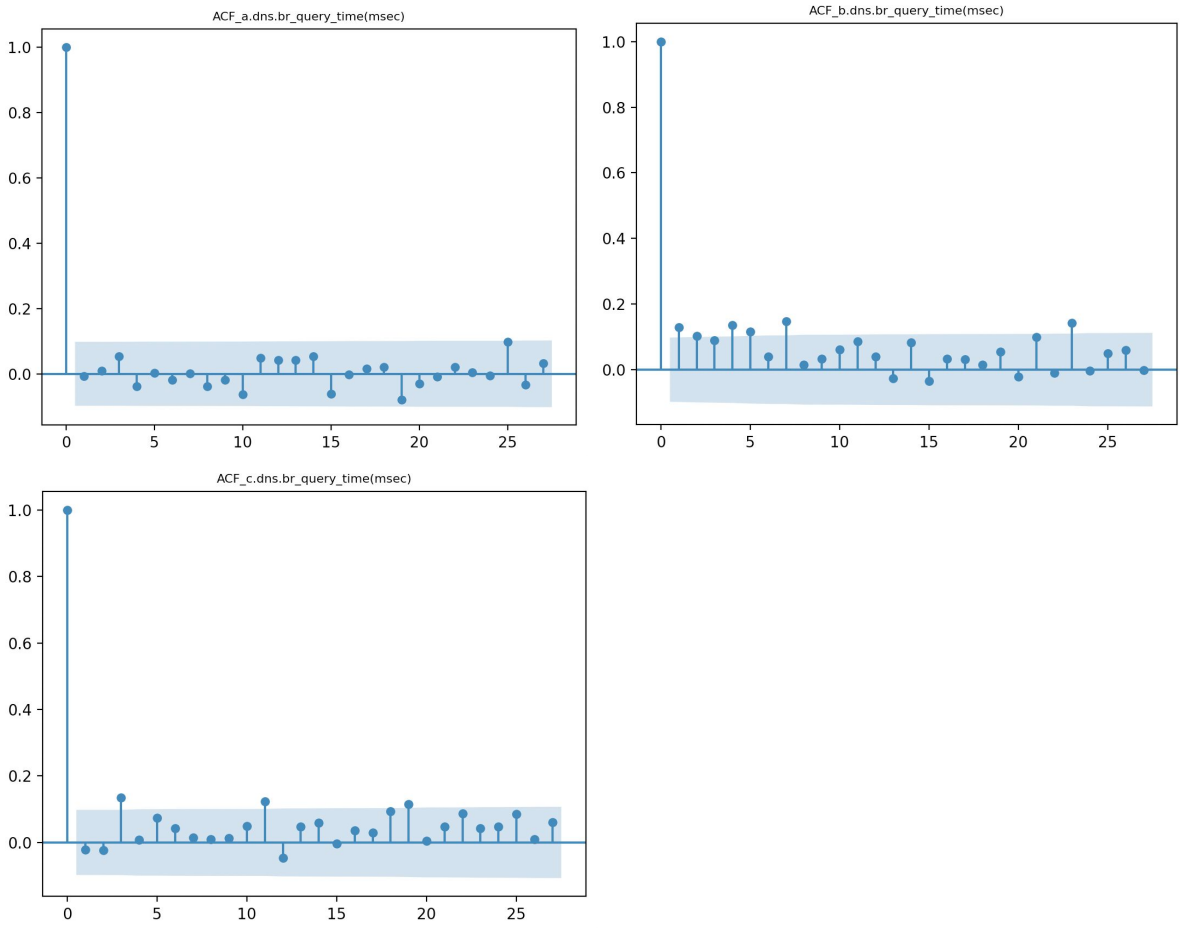




3.2.2 Code and autocorrelation plot on AS1.x data sets

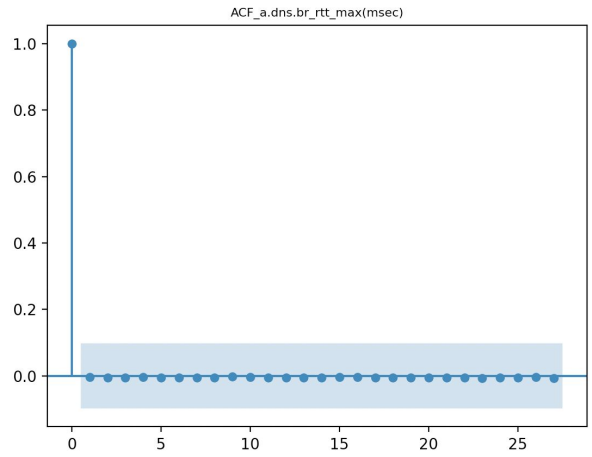
3.2.2.1 AS1_NameServer_DNS.csv

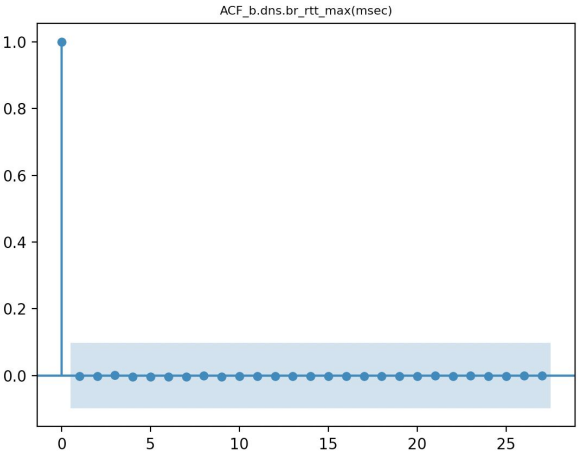
```
from pandas import read_csv
from matplotlib import pyplot
from statsmodels.graphics.tsaplots import plot_acf
series =
read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet3/AS1_NameServer_DNS.
csv')
plot_acf(series["c.dns.br_query_time(msec)"])
pyplot.title("ACF_c.dns.br_query_time(msec)",fontsize=8)
pyplot.show()
# Modify the code for a.dns.br and b.dns.br
```

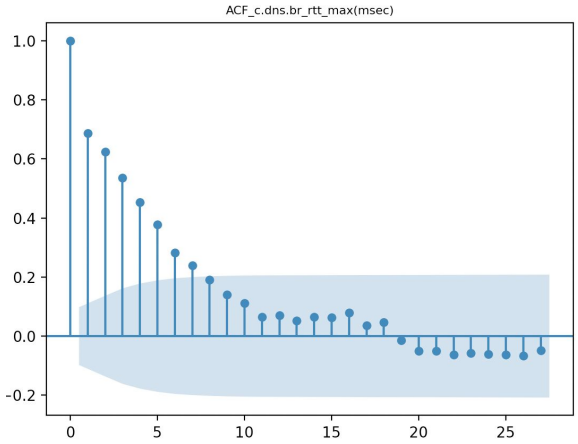


3.2.2.2 AS1_NameServer_ICMP.csv

```
from pandas import read_csv
from matplotlib import pyplot
from statsmodels.graphics.tsaplots import plot_acf
series =
read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet3/AS1_NameServer_ICMP
.csv')
plot_acf(series["c.dns.br_rtt_max(msec)"])
pyplot.title("ACF_c.dns.br_rtt_max(msec)",fontsize=8)
pyplot.show()
# Modify the code for a.dns.br and b.dns.br
```

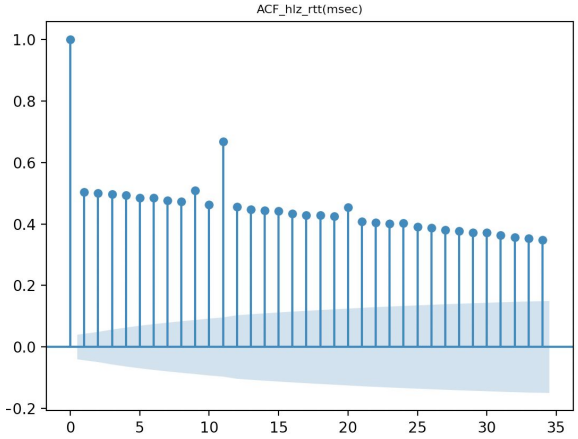


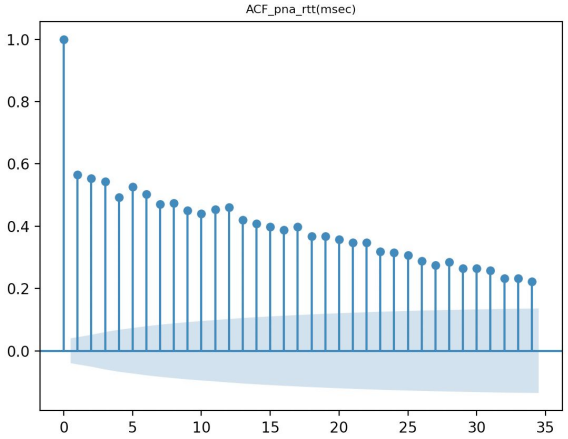




3.2.2.3 AS1_ResearchServer_ICMP.csv

```
from pandas import read_csv
from matplotlib import pyplot
from statsmodels.graphics.tsaplots import plot_acf
series =
read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet3/AS1_ResearchServer_
ICMP.csv')
plot_acf(series["pna_rtt(msec)"])
pyplot.title("ACF_pna_rtt(msec)",fontsize=8)
pyplot.show()
# Modify the code for hlz server
```

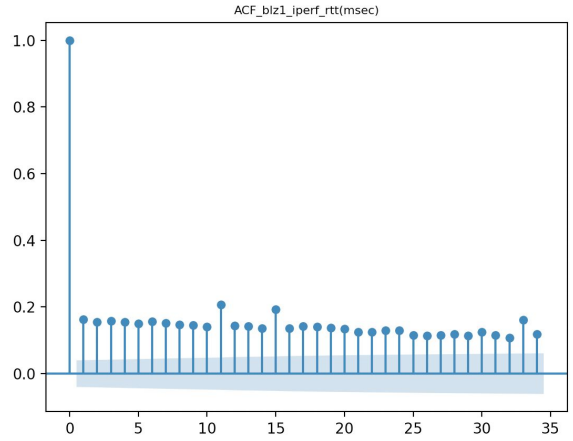


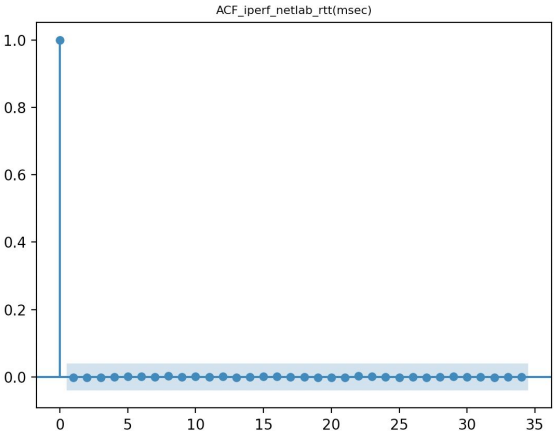


3.2.2.4 AS1_IperfServer_ICMP.csv

```
from pandas import read_csv
```

```
from matplotlib import pyplot
from statsmodels.graphics.tsaplots import plot_acf
series =
read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet3/AS1_IperfServer_ICM
P.csv')
plot_acf(series["iperf_netlab_rtt(msec)"])
pyplot.title("ACF_iperf_netlab_rtt(msec)",fontsize=8)
pyplot.show()
# Modify the code for blz server
```

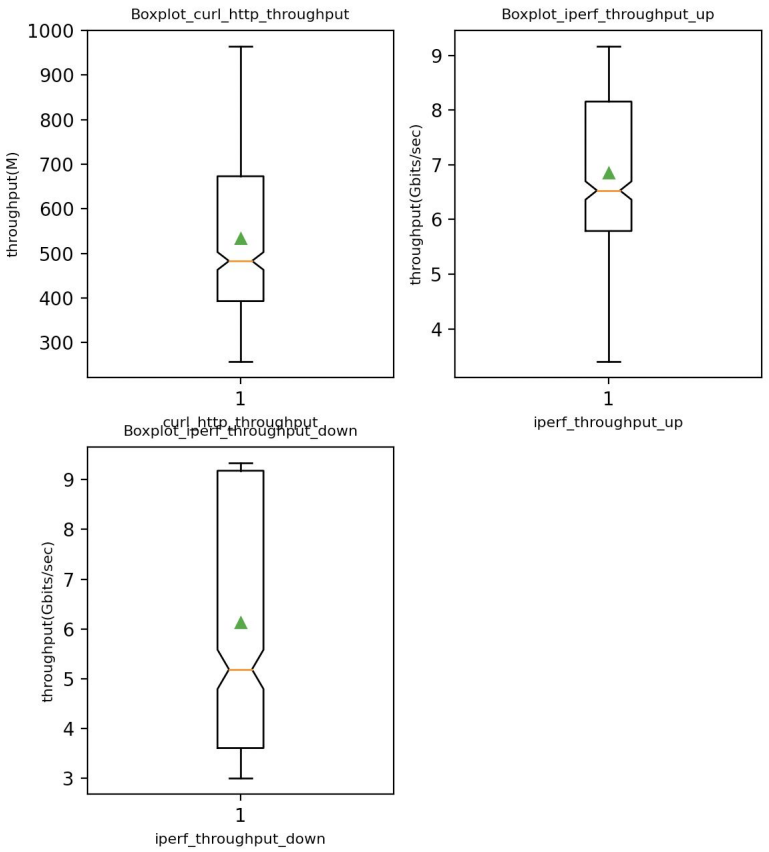




3.3 Throughput

3.3.1 Code and Box plot for AS2_iperf.csv

```
import pandas as pd
import matplotlib.pyplot as plt
data =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet3/AS2_iperf.csv')
curl_http_throughput = []
iperf_throughput_up = []
iperf_throughput_down = []
for index, row in data.iterrows():
      curl_http_throughput.append(row["curl_http_throughput(M)"])
for index, row in data.iterrows():
      iperf_throughput_up.append(row["iperf_throughput_up(Gbits/sec)"])
for index, row in data.iterrows():
      iperf_throughput_down.append(row["iperf_throughput_down(Gbits/sec)"])
plt.figure(1)
plt.subplot(221)
plt.title("Boxplot_curl_http_throughput",fontsize=8)
plt.xlabel("curl_http_throughput",fontsize=8)
plt.ylabel("throughput(M)",fontsize=8)
plt.boxplot(curl_http_throughput, showmeans=True, notch = True, sym = '*')
plt.subplot(222)
plt.title("Boxplot_iperf_throughput_up",fontsize=8)
plt.xlabel("iperf_throughput_up",fontsize=8)
plt.ylabel("throughput(Gbits/sec)",fontsize=8)
plt.boxplot(iperf_throughput_up, showmeans=True, notch = True, sym = '*')
plt.subplot(223)
plt.title("Boxplot_iperf_throughput_down",fontsize=8)
plt.xlabel("iperf_throughput_down",fontsize=8)
plt.ylabel("throughput(Gbits/sec)",fontsize=8)
plt.boxplot(iperf_throughput_down, showmeans=True, notch = True, sym = '*')
plt.show()
```



3.3.2 Compute and tabulate representative values AS2_iperf.csv

```
import pandas as pd
import numpy as np
from scipy.stats import hmean
from scipy.stats.mstats import gmean
data =
pd.read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet3/AS2_iperf.csv')
curl_http_throughput = []
curl_http_throughput_list = []
iperf_throughput_up = []
iperf_throughput_up_list = []
iperf_throughput_down = []
iperf_throughput_down_list = []
for index, row in data.iterrows():
    curl_http_throughput.append(row["curl_http_throughput(M)"])
for index, row in data.iterrows():
    iperf_throughput_up.append(row["iperf_throughput_up(Gbits/sec)"])
for index, row in data.iterrows():
    iperf_throughput_down.append(row["iperf_throughput_down(Gbits/sec)"])
curl_http_throughput_list.append(float('%.3f'%np.mean(curl_http_throughput)))
curl_http_throughput_list.append(float('%.3f'%hmean(curl_http_throughput)))
curl_http_throughput_list.append(float('%.3f'%gmean(curl_http_throughput)))
curl_http_throughput_list.append(float('%.3f'%np.median(curl_http_throughput)))
iperf_throughput_up_list.append(float('%.3f'%np.mean(iperf_throughput_up)))
iperf_throughput_up_list.append(float('%.3f'%hmean(iperf_throughput_up)))
iperf_throughput_up_list.append(float('%.3f'%gmean(iperf_throughput_up)))
iperf_throughput_up_list.append(float('%.3f'%np.median(iperf_throughput_up)))
iperf_throughput_down_list.append(float('%.3f'%np.mean(iperf_throughput_down)))
iperf_throughput_down_list.append(float('%.3f'%hmean(iperf_throughput_down)))
iperf_throughput_down_list.append(float('%.3f'%gmean(iperf_throughput_down)))
iperf_throughput_down_list.append(float('%.3f'%np.median(iperf_throughput_down)))
print("mean","\t","harmonic mean","\t","geometric mean","\t","median")
for i in range(4):
  print(curl_http_throughput_list[i], end='\t')
```

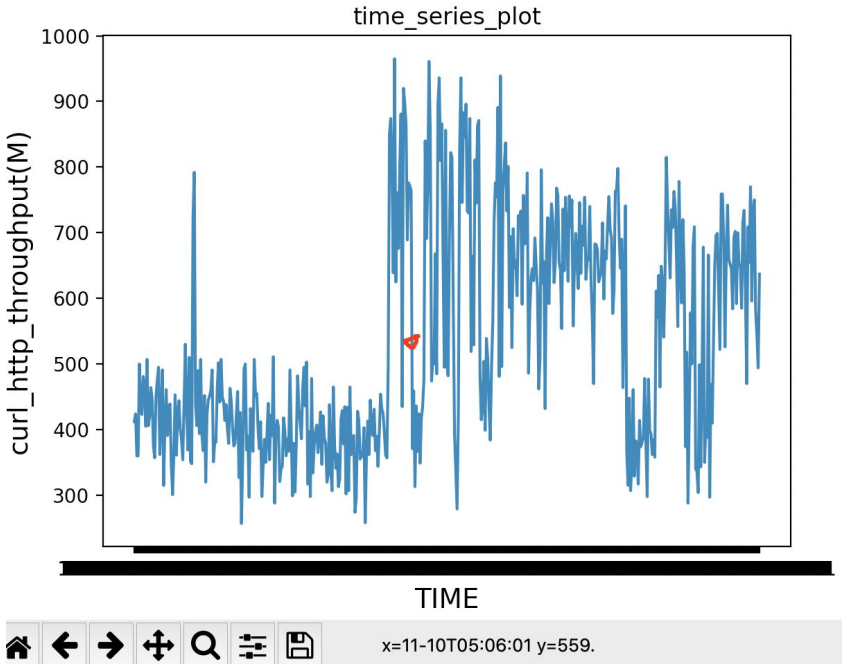
```
print("")
for i in range(4):
      print(iperf_throughput_up[i], end='\t')
print("")
for i in range(4):
      print(iperf_throughput_down[i], end='\t')
```

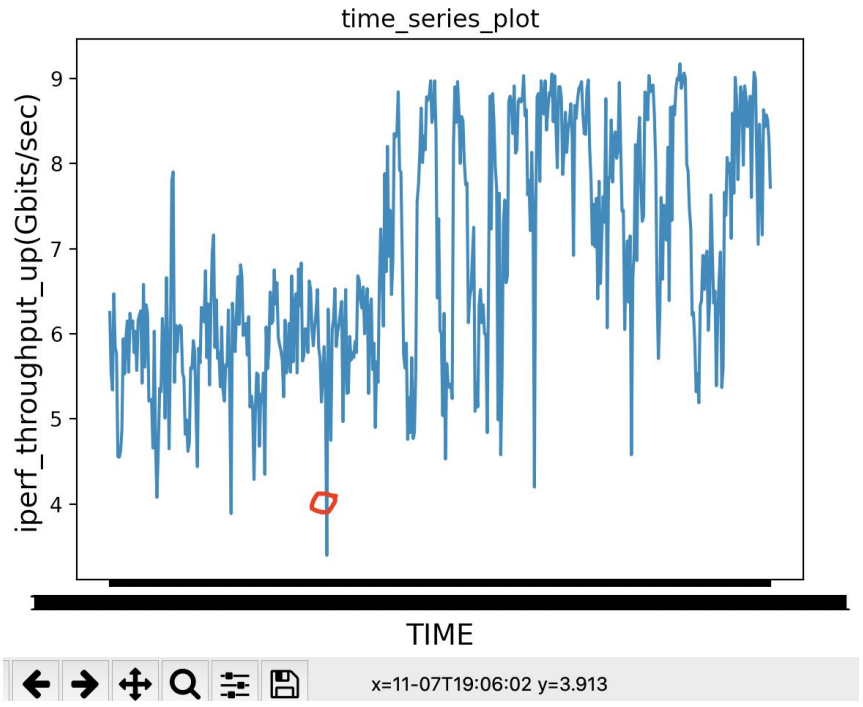
```
mean      harmonic mean    geometric mean          median
533.538 485.234 508.56   483.0
6.25     5.54     5.34     6.47
3.77     3.78     3.93     3.74
```

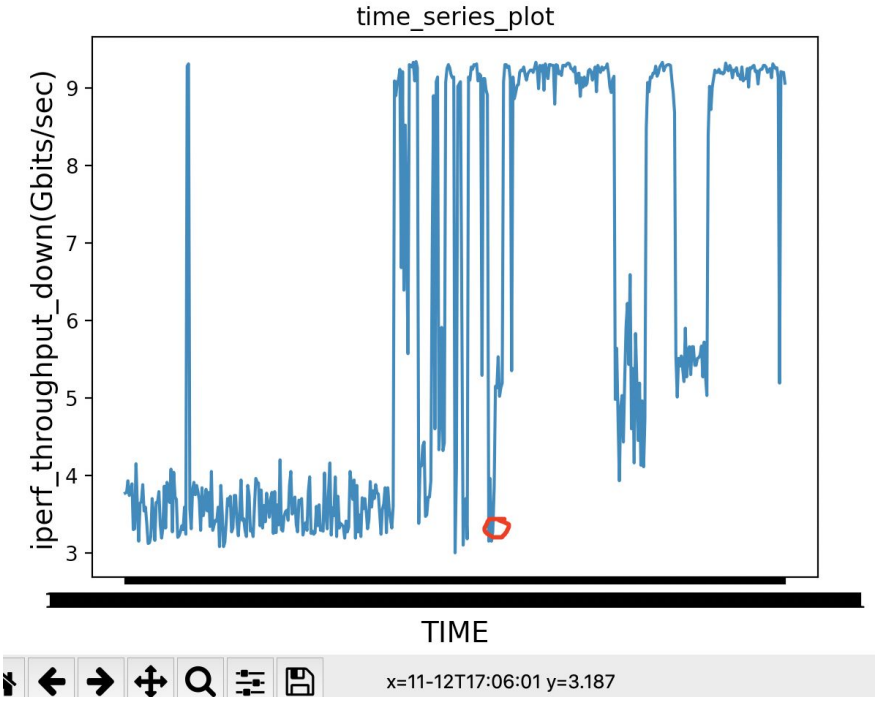
3.4 Throughput time series

3.4.1 Code and plot time series of each data set AS2.x

```
import pandas as pd
import matplotlib.pyplot as plt
df =
pd.read_csv(r'/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet3/AS2_iperf.csv')
plt.plot(df['date'],df['iperf_throughput_down(Gbits/sec)'])
plt.xlabel("TIME",fontsize=14)
plt.ylabel("iperf_throughput_down(Gbits/sec)",fontsize=14)
plt.title("time_series_plot")
plt.show()
# Modify the code for curl and iperf_up
```

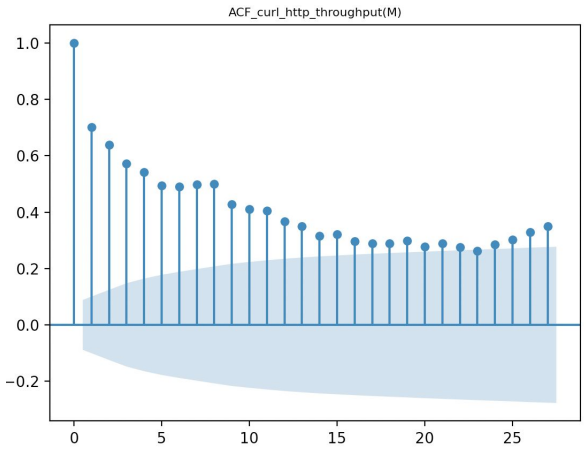


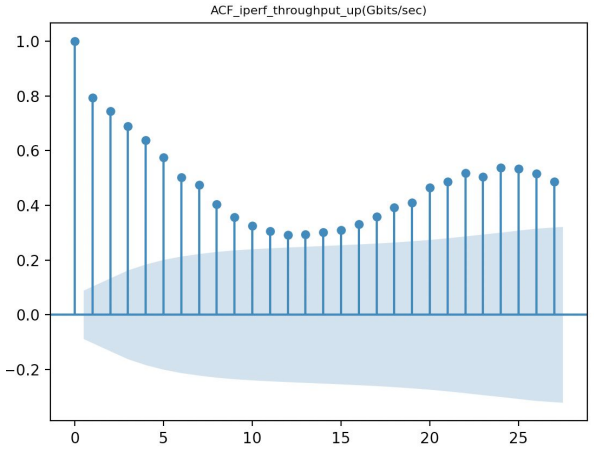


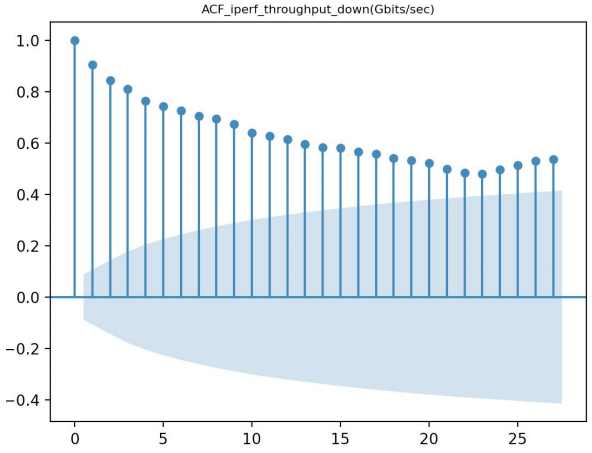


3.4.2 Code and autocorrelation plot on AS2.x data sets

```
from pandas import read_csv
from matplotlib import pyplot
from statsmodels.graphics.tsaplots import plot_acf
series =
read_csv('/Users/yanjing/Desktop/ITMA/Assignment-Final/DataSet3/AS2_iperf.csv')
plot_acf(series["iperf_throughput_down(Gbits/sec)"])
pyplot.title("ACF_iperf_throughput_down(Gbits/sec)",fontsize=8)
pyplot.show()
# Modify the code for curl and iperf_down
```







3.5 Conclusion

1> Based on AS1_NameServer_DNS.csv, AS1_NameServer_ICMP.csv

1.1> For the box plots

Among these 3 servers, a.dns.br's box plot is showing that the dns query time is concentrated around 40-80ms, which is the best performance.

Among these 3 servers, b.dns.br's box plot is showing that the ping rtt time is concentrated around 175-180ms without considering pkt loss, which is the best performance. When taking pkt loss into consideration, then c.dns.br's ping rtt time is the best.

So the conclusion is that, from my workstation to a.dns.br, there are least dns servers compared to other 2 servers which makes the dns query time is lowest, and from my workstation to b.dns.br, there are least hops compared to other 2 servers, but the network cables to c.dns.br is the most stable one.

1.2> For the PDF plot and CDF plots

a.dns.br, b.dns.br and c.dns.br behave similarly in PDF plot and CDF plot in dns query time.

a.dns.br and b.dns.br behave similarly in PDF plot and CDF plot in ping rtt time, while c.dns.br differs a little bit.

1.3> For the time series plots

a.dns.br, b.dns.br and c.dns.br behave similarly and even have basically identical fluctuation trends in dns query time, and can not see any recognisable patterns from the time series plots.

a.dns.br and b.dns.br behave similarly and even have basically identical fluctuation trends in ping rtt time, at the most of time, a,dns.br and b.dns.br are stable with time changes, there only two peaks, while c.dns.br differs a lot, it fluctuates a lot with time changes.

1.4> For the autocorrelation plots

The conclusion generated from autocorrelation plots for a.dns.br, b.dns.br and c.dns.br are basically similar with the time series plots in dns query time, as the dns query time for these 3 servers shows slightly positive correlation with time changes.

The conclusion generated from autocorrelation plots for a.dns.br, b.dns.br and c.dns.br are basically similar with the time series plots in ping rtt time, a.dns.br and b.dns.br's ping rtt time shows totally negative correlation with time changes, while c.dns.br shows slightly positive correlation with time changes.

2> Based on AS1_ResearchServer_ICMP.csv

2.1> From the box plots, no matter if I take pkt loss into consideration or not, the network cable is more stable and fast between my workstation and pna server compared with hlz server.

2.2> For the PDF plot and CDF plots

Pna server and hlz server behave very similarly.

2.3> For the time series plots

Both pna server and hlz server's ping rtt time shows changes with time changes, but the amplitude of pna server is more violent while the hlz server's is more light.

2.4> For the autocorrelation plots

The conclusion generated from autocorrelation plots for pna server and hlz server are basically similar with the time series plots in ping rtt time, both of them are showing positive correlation with time changes to some extent.

3> Based on AS1_IperfServer_ICMP.csv

3.1> From the box plots, no matter if I take pkt loss into consideration or not, the network cable is more stable and fast between my workstation and iperf_netlab server compared with blz1_iperf server.

3.2> For the PDF plot and CDF plots

iperf_netlab server and blz1_iperf server behave very similarly.

3.3> For the time series plots

Iperf_netlab's ping rtt time shows no change with time changes while the blz1_iperf fluctuates with time regularly.

3.4> For the autocorrelation plots

The conclusion generated from autocorrelation plots for Iperf_netlab server and blz1_iperf server are basically similar with the time series plots in ping rtt time, blz1_iperf is showing positive correlation with time changes to some extent, while Iperf_netlab shows no correlation with time changes.

4> Did there exist some correlation between path length (number of routers, it can be checked with traceroute and/or with TTL value of ICMP Echo Responses) and measurement stability? If you happened to also record TTL value, did it change over time?

Yes, there is correlation between patch length and stability.

For each router a packet passes, then the TTL will subtract 1, Which means the final value of TTL is lower, then the path is longer for the packet transferring. Longer paths will cause high possibilities to lose packet, timeout and so on.

5> Did throughput and latency have any correlation?
Yes, there is some kind of correlation between throughput and latency.
When the throughput is small, then it is highly possible that the latency is relatively low due to the resource usage of the physical limitations of network latency and disk IO latency is not that low, too. But with the increase of throughput, the system throughput can be effectively improved, at the same time, the delay will also increase. Furthermore, when the pressure exceeds a certain critical value of the system, the throughput does not rise but drops, and the delay will rise sharply.

Final Conclusion

1> How was your own traffic (Task 2) different from the data provided (Task 1)? What kind of differences can you identify? What could be a reason for that?
In the question above, I have stated the reason. The network traffic (I gave the conclusion about the category of the traffic network is from a small school network or small IT company network) in Task1 is highly discrete, which can be seen from the pie plot and distribution judgment. While the network traffic in Task2 is generated from my behavior when using the internet, all the data is highly continuous, cause I only browsed websites (HTTPS and HTTP), and checked email and used ssh protocol, all the rest actions are finished locally.

2> Comparing RTT latency about TCP connections (2.10), were active latency measurements around the same magnitude or was another much larger than the other?
The latency of TCP connections is a little higher than the RTT latency, which is also corresponding to the TCP/IP model. When building a TCP connection, the session will build a reliable connection between the client and the server through three-time handshake, while RTT is about ICMP packet.

3> How do you rate complexity of different tasks? Was some tasks more difficult or laborious than others? Did data volume cause any issues with your analysis?
This final assignment is a lot more complex than the previous weekly assignment, in my opinion, it demands more about the ability to understand the questions, because it took me lots of time to understand the questions, and then started to code, answer.