Report for the 5th Assignment of ELEC-E7130

[redacted]

[redacted]

Report, task 1

0. Pre-processing

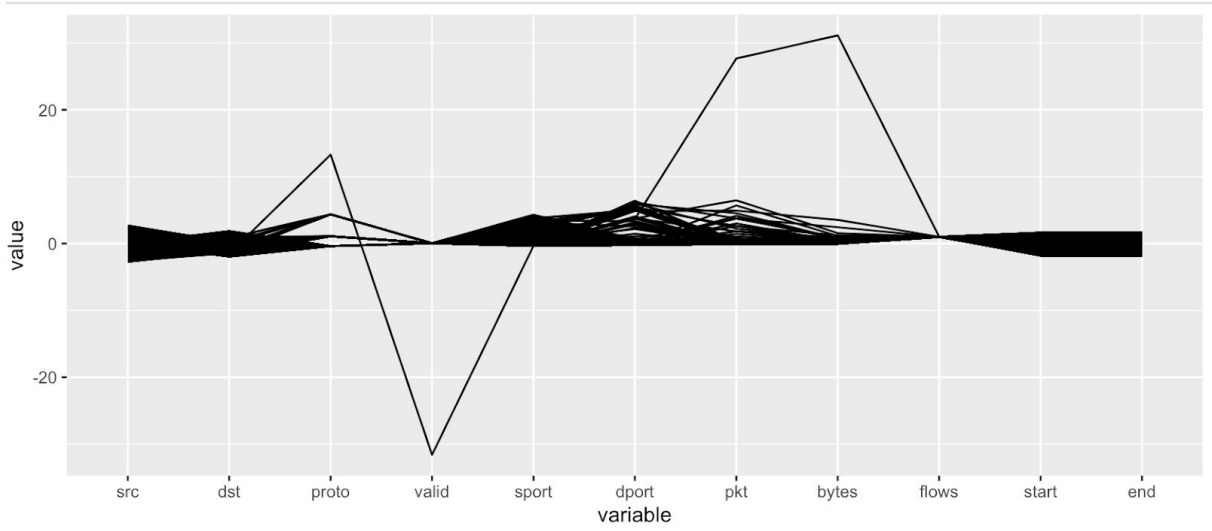
Modify the “flowdata.txt” as follows by replacing “\t” with “,” and adding csv header

```
yanjing@yanjingdeMacBook-Pro Assignment-5 % head -n5 flowdata.csv
src,dst,proto,valid,sport,dport,pkt,bytes,flows,start,end
216.239.82.13,163.35.104.114,17,1,53,52339,1,93,1,1491969541.801740000,1491969541.8
01740000
203.246.146.19,106.22.48.22,1,1,8,0,1,32,1,1491969542.588011000,1491969542.58801100
0
203.246.146.19,218.201.19.92,1,1,8,0,1,32,1,1491969545.793305000,1491969545.7933050
00
176.52.159.166,203.246.146.19,1,1,0,0,1,32,1,1491969545.843147000,1491969545.843147
000
```

1. Parallel plot based on 1000 random sample data from flowdata.txt

1.1 Code and result

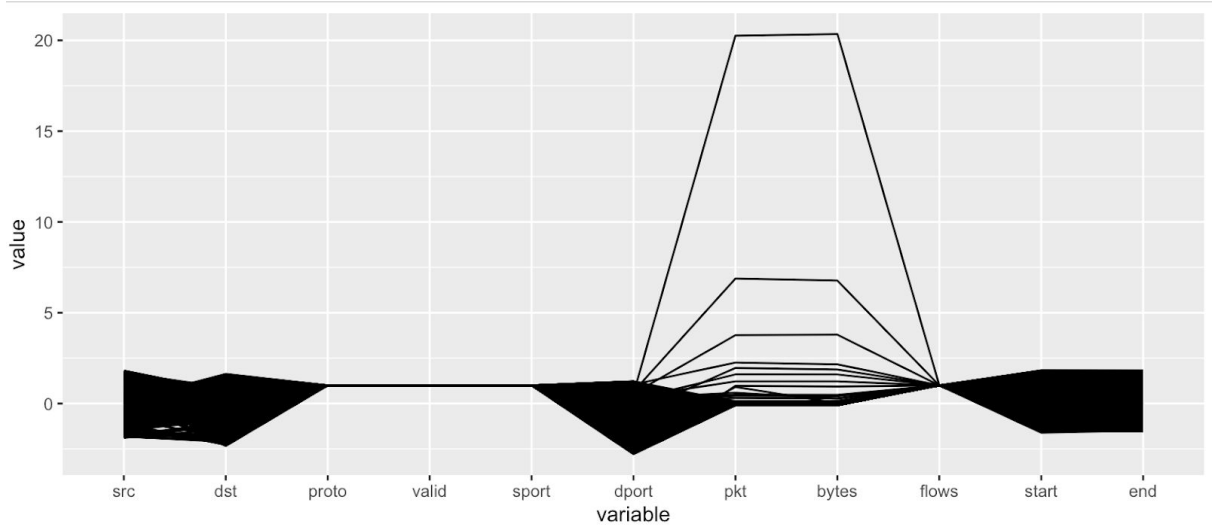
```
> library(readr)
> flowdata <- read_csv("Assignment-5/flowdata.csv")
> View(flowdata)
> colnames(flowdata)
[1] "src"   "dst"   "proto" "valid" "sport" "dport" "pkt"   "bytes" "flows"
[10] "start" "end"
> index <- sample(1:dim(flowdata)[1],1000)
> install.packages("GGally")
> library(GGally)
> ggparcoord(flowdata[index,])
```



2. Parallel plot based on flows with source port 80 (WWW) from flowdata.txt

2.1 Code and result

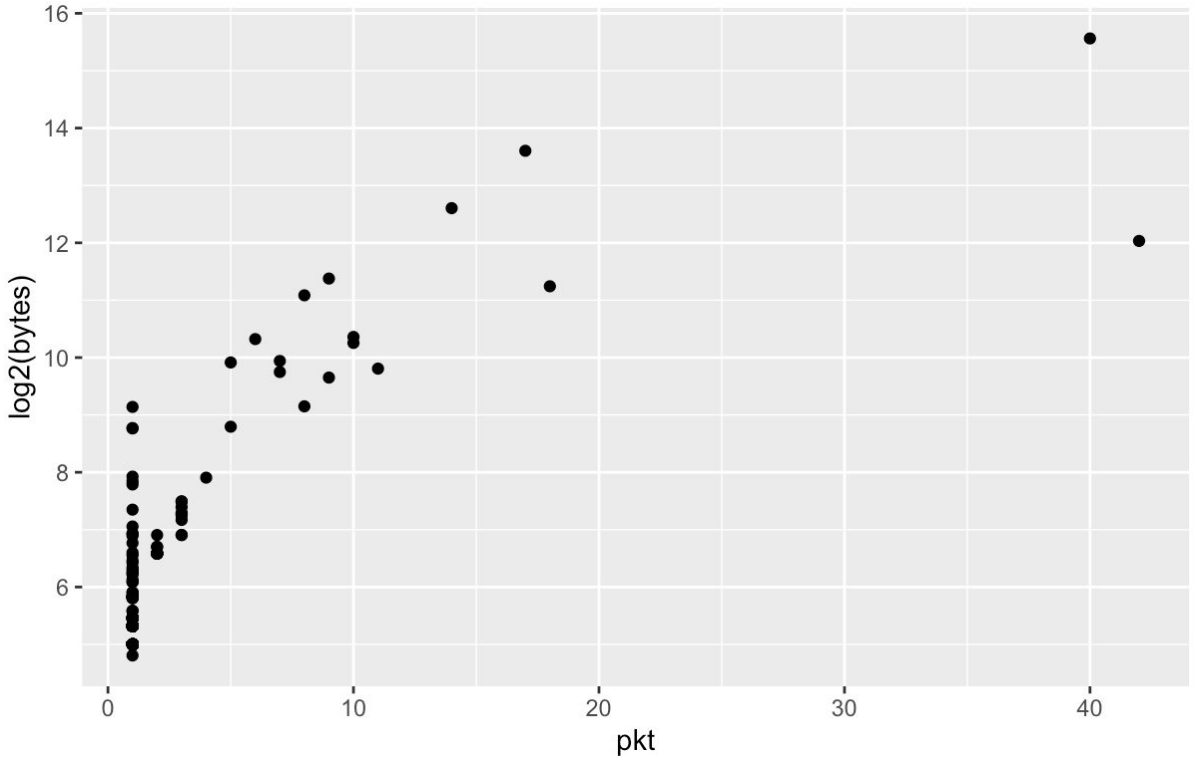
```
> www <- flowdata[flowdata$sport == 80,]
> ggparcoord(www)
```



3. Scatterplot based on number of bytes against packets from the 1000 sample data generated from flowdata.txt

3.1 Code and result

```
> install.packages("ggplot2")
> library(ggplot2)
> ggplot(data=flowdata[index,]) + geom_point(mapping = aes(x=pkt,y=log2(bytes)))
```



3.2 Analysis

Basically, the ‘pkt’ and the ‘bytes’ are related, as we can see the picture above, as ‘pkt’ increases, the ‘bytes’ also increases.
As for the “average packet size”, it is “125.9” bytes.

```
> summary(flowdata[index,])
```

```
      pkt              bytes
Min.   : 1.000   Min.   :   28.0
1st Qu.: 1.000   1st Qu.:   32.0
Median : 1.000   Median :   32.0
Mean   : 1.242   Mean   :  125.9
3rd Qu.: 1.000   3rd Qu.:   32.0
Max.   :42.000   Max.   :48399.0
```

4. Average throughput of the connections

4.1 Code and result

```
import csv
import pandas as pd
import numpy as np
filepath = open('/Users/yanjing/Desktop/ITMA/Assignment-5/flowdata.csv','r')
csv = pd.read_csv(filepath)
throughput = {}
for index, row in csv.iterrows():
   throughput[index] = []
   transfer_time = row["end"] - row["start"]
   throughput[index].append(transfer_time)
   throughput[index].append(row["bytes"])
```

```
throughput_result = []
for index in throughput:
   if (throughput[index][0] != 0):
       throughput_result.append(throughput[index][1]/throughput[index][0])
print(np.mean(throughput_result))
```

```
yanjing@yanjingdeMBP Assignment-5 % /usr/bin/python3
/Users/yanjing/Desktop/ITMA/Assignment-5/test.py
1118377.9541919837
```

4.2 Analysis

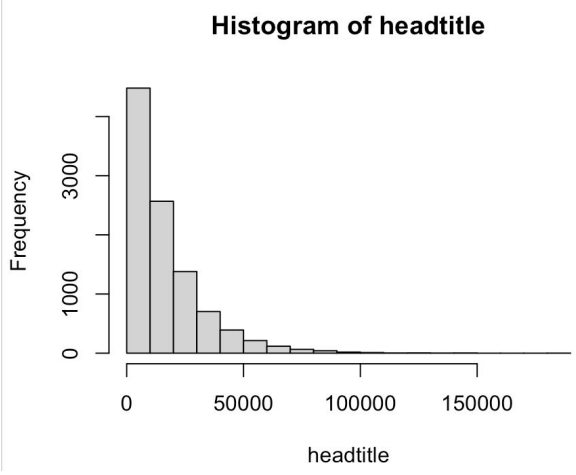
In the code above, I can see lots of records that transfer some bytes within 0 second through the variable “throughput”. When I calculated the average throughput of the connections, I dropped these kinds of records. As a result, I get the result “1118377.9541919837 (bytes/ms)” as the average throughput of the connections.
But I do not think this is a valid result, when we calculate throughput, we should focus on the bytes transferred in 1 time unit (like 1 second, minutes or hours) among the same src host and dst host pair, rather than pay attention to the flows, which is not accurate.

Report, task 2

1. Plot the histogram of the original data and compute the mean

1.1 Code and result for histogram plot

```
> sampling <- read.csv("Assignment-5/sampling/sampling.txt", header = F)
> headtitle = sampling[,1]
> headtitle <- as.numeric(headtitle)
> hist(headtitle)
```



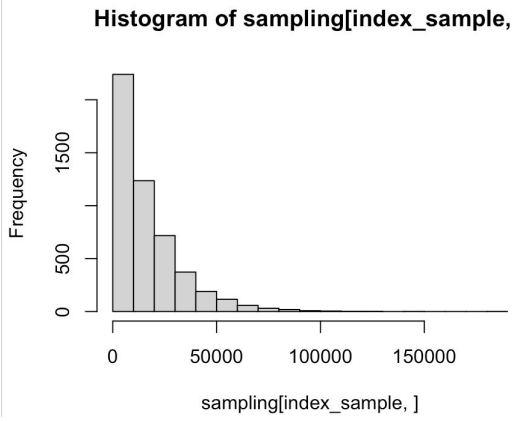
1.2 Code and result for the mean

```
> summary(sampling)
      V1
Min.   :     1.387
1st Qu.:  4731.549
Median : 11503.404
Mean   : 16432.251
3rd Qu.: 22410.163
Max.   :181717.713
```

2. Plot the histogram based on the 5000 sample from the original data and compute the mean

2.1 Code and result for histogram plot

```
> index_sample <- sample(1:dim(sampling)[1],5000)
> index_sample = as.numeric(index_sample)
> hist(sampling[index_sample,])
```



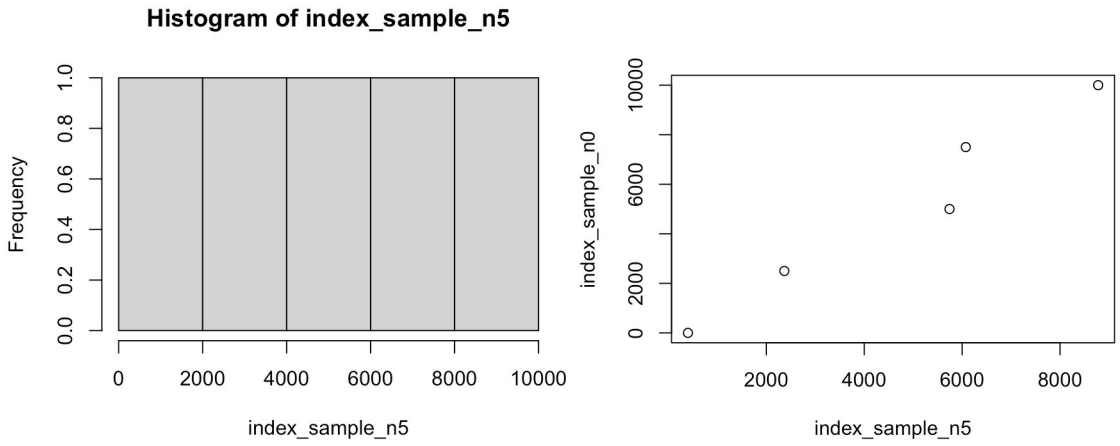
2.2 Code and result for the mean

```
> summary(sampling[index_sample,])
     Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
    1.387   4705.773  11371.564  16449.192  22183.910 181717.713
```

3 Histogram plot, Q-Q plot based on the 10000 sample when size=5 from the original data, and compute the mean and standard deviation

3.1 Code and result for histogram plot and Q-Q plot

```
> index_sample_n0 <- sample(1:dim(sampling)[1], 10000)
> index_sample_n0 <- as.numeric(index_sample_n0)
> index_sample_n5 <- sample(1:dim(sampling)[1], size = 5, 10000)
> index_sample_n5 <- as.numeric(index_sample_n5)
> hist(index_sample_n5)
> qqplot(index_sample_n5, index_sample_n0)
```



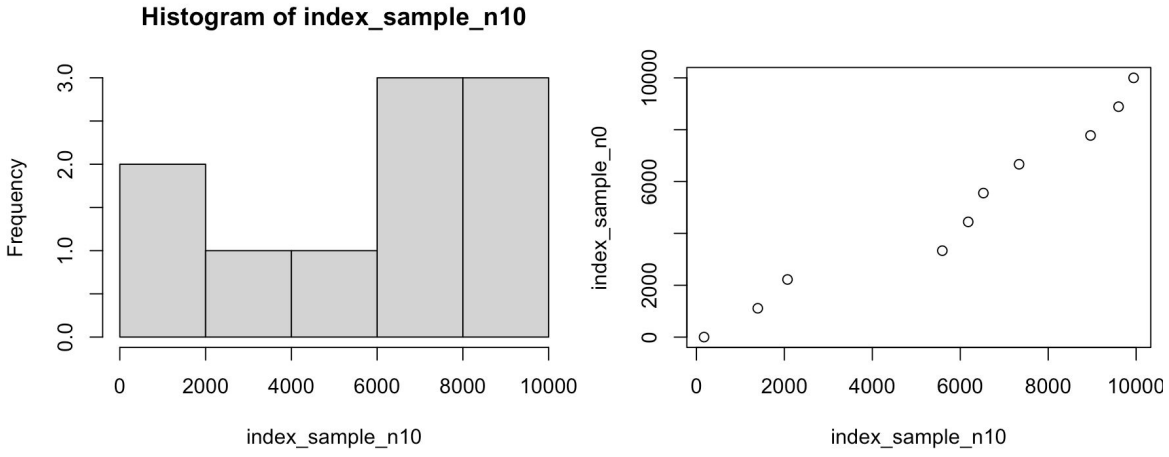
3.2 Code and result for the mean and the standard deviation

```
> mean(index_sample_n5)
[1] 4672.6
> sd(index_sample_n5)
[1] 3299.5089
```

4 Histogram plot, Q-Q plot based on the 10000 sample when size=10 from the original data, and compute the mean and standard deviation

4.1 Code and result for histogram plot and Q-Q plot

```
> index_sample_n0 <- sample(1:dim(sampling)[1], 10000)
> index_sample_n0 <- as.numeric(index_sample_n0)
> index_sample_n10 <- sample(1:dim(sampling)[1], size = 10, 10000)
> index_sample_n10 <- as.numeric(index_sample_n10)
> hist(index_sample_n10)
> qqplot(index_sample_n10, index_sample_n0)
```



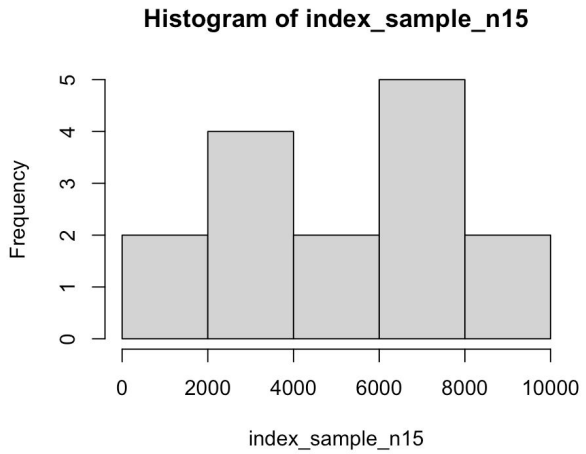
4.2 Code and result for the mean and the standard deviation

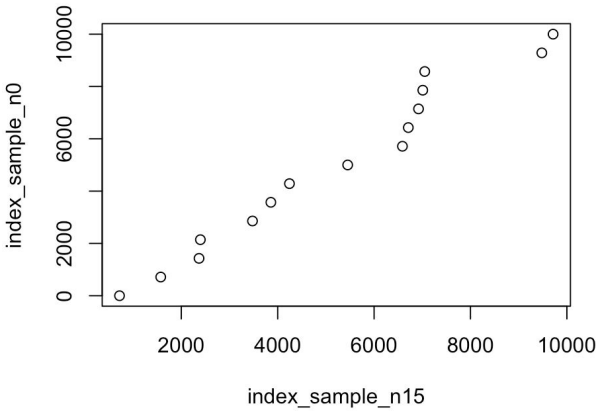
```
> mean(index_sample_n10)
[1] 5778.1
> sd(index_sample_n10)
[1] 3490.4549
```

5 Histogram plot, Q-Q plot based on the 10000 sample when size=10 from the original data, and compute the mean and standard deviation

5.1 Code and result for histogram plot and Q-Q plot

```
> index_sample_n0 <- sample(1:dim(sampling)[1], 10000)
> index_sample_n0 <- as.numeric(index_sample_n0)
> index_sample_n15 <- sample(1:dim(sampling)[1], size = 15,10000)
> index_sample_n15 <- as.numeric(index_sample_n15)
> hist(index_sample_n15)
> qqplot(index_sample_n15, index_sample_n0)
```





5.2 Code and result for the mean and the standard deviation

```
> mean(index_sample_n15)
[1] 5170.7333
> sd(index_sample_n15)
[1] 2770.1109
```

6 Discussion about the effects of sample size to the sampling distribution

When the sample size is larger, the distribution is more of a perfect normal distribution. And the center point(mean) is more accurate. Also most of the sample mean does not diverge from the mean.

Report, task 3

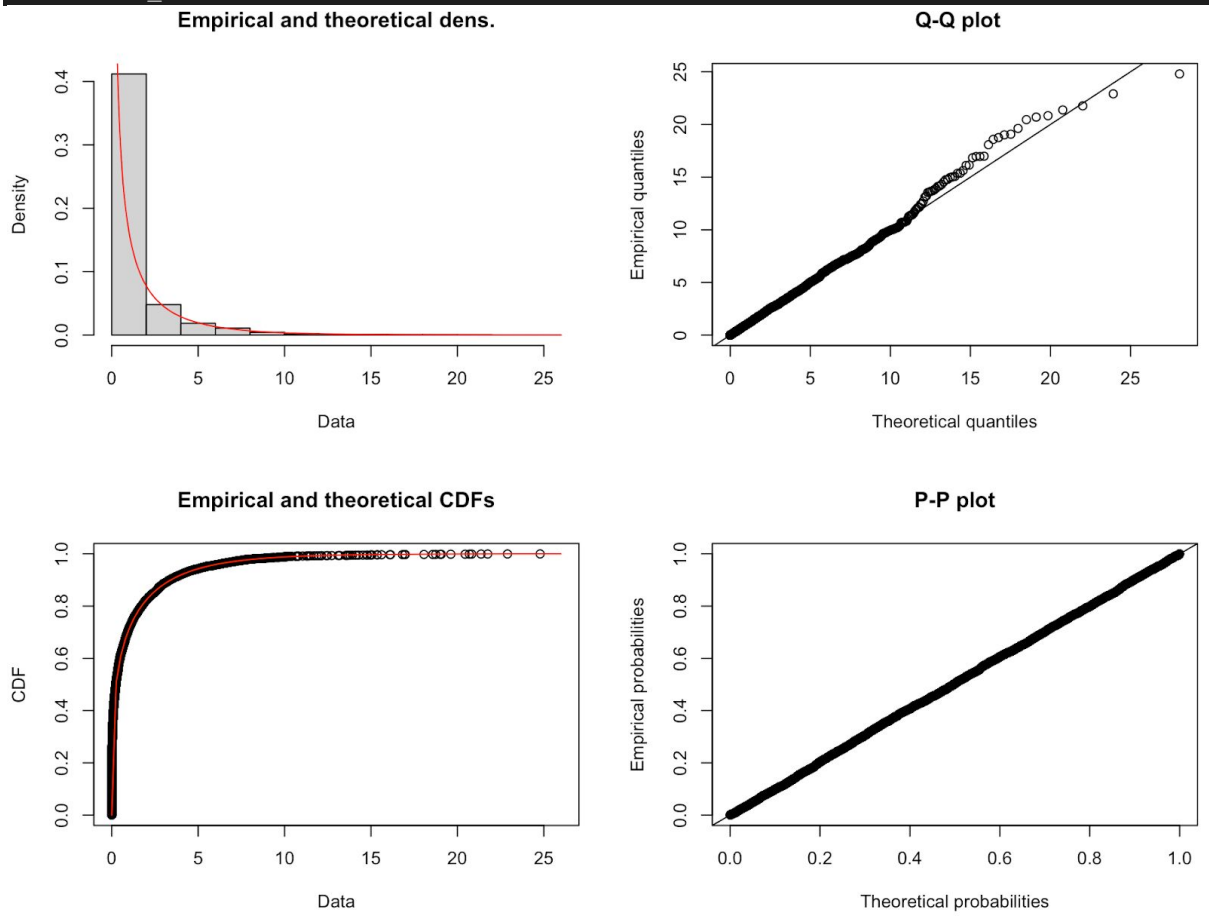
3.1 Code and result of how to judge the distribution of distr_a.txt

```
> install.packages("fitdistrplus")
> library(fitdistrplus)
> distr_a <- read.csv("Assignment-5/sampling/distr_a.txt", header = F)


> fit_para <- fitdist(distr_a[,1], "norm")
> print(fit_para)
Fitting of the distribution ' norm ' by maximum likelihood
Parameters:
     estimate  Std. Error
mean 1.1466768 0.028418919
sd   2.2203161 0.020095192

> fit_para <- fitdist(distr_a[,1], "exp")
> print(fit_para)
Fitting of the distribution ' exp ' by maximum likelihood
Parameters:
      estimate  Std. Error
rate 0.87208533 0.011162235

> fit_para <- fitdist(distr_a[,1], "gamma")
> print(fit_para)
Fitting of the distribution ' gamma ' by maximum likelihood
Parameters:
       estimate   Std. Error
shape 0.27590237 0.0039172059
```

```
rate  0.24060219 0.0067853718
```

I checked the result of the code above, especially the value of “Std. Error(rate)” and found the value of “gamma”’s “Std. Error(rate)” is the smallest, and then I plotted the picture when the “distr” = “gamma”, which can be seen as follows. The plot also validates that the dataset of “distr_a.txt” is distributed according to the “gamma” distribution.

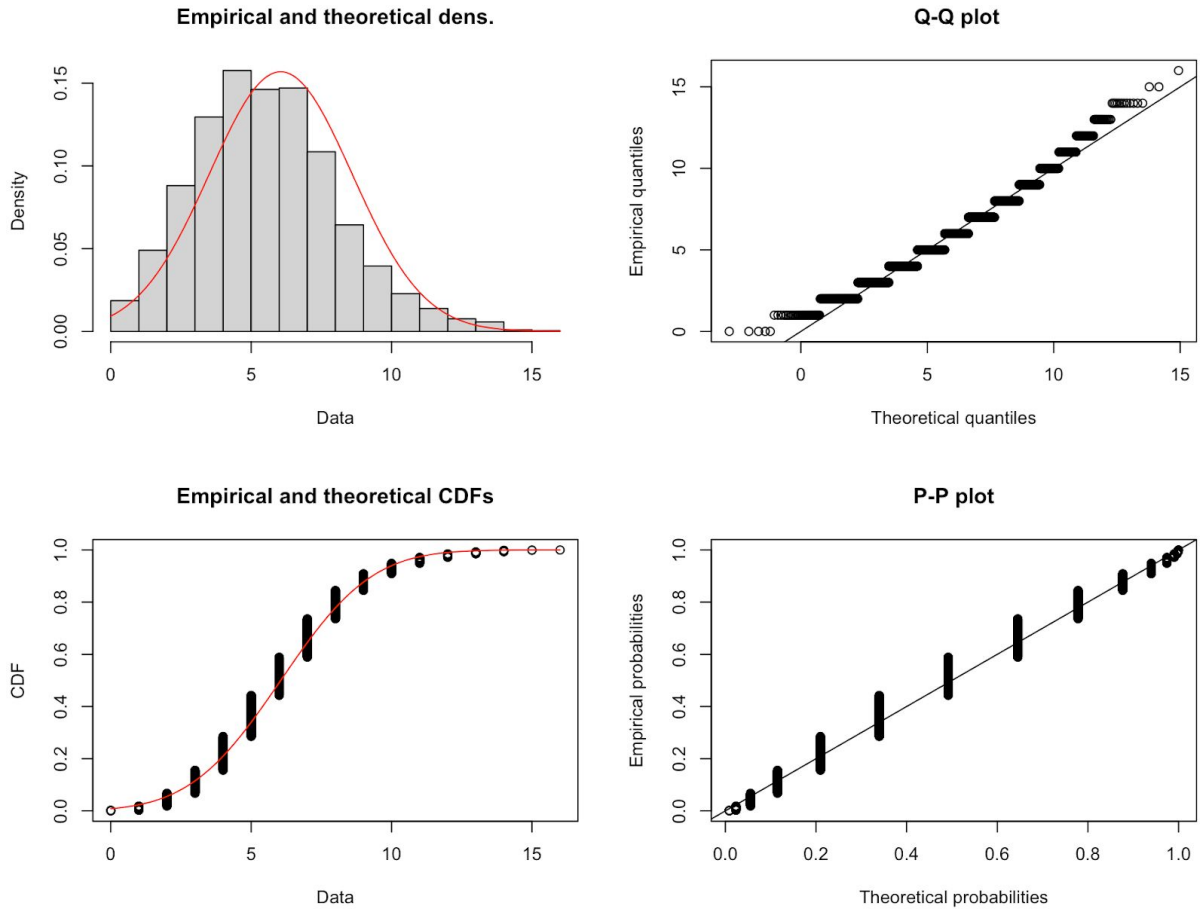
```
> fit_para <- fitdist(distr_a[,1], "gamma")
> plot(fit_para)
```



3.2 Code and result of how to judge the distribution of distr_b.txt

Same process with 3.1, checked “Std. Error(sd)” the plot, which also showed that the dataset of “distr_b.txt” is distributed according to the “norm” distribution.

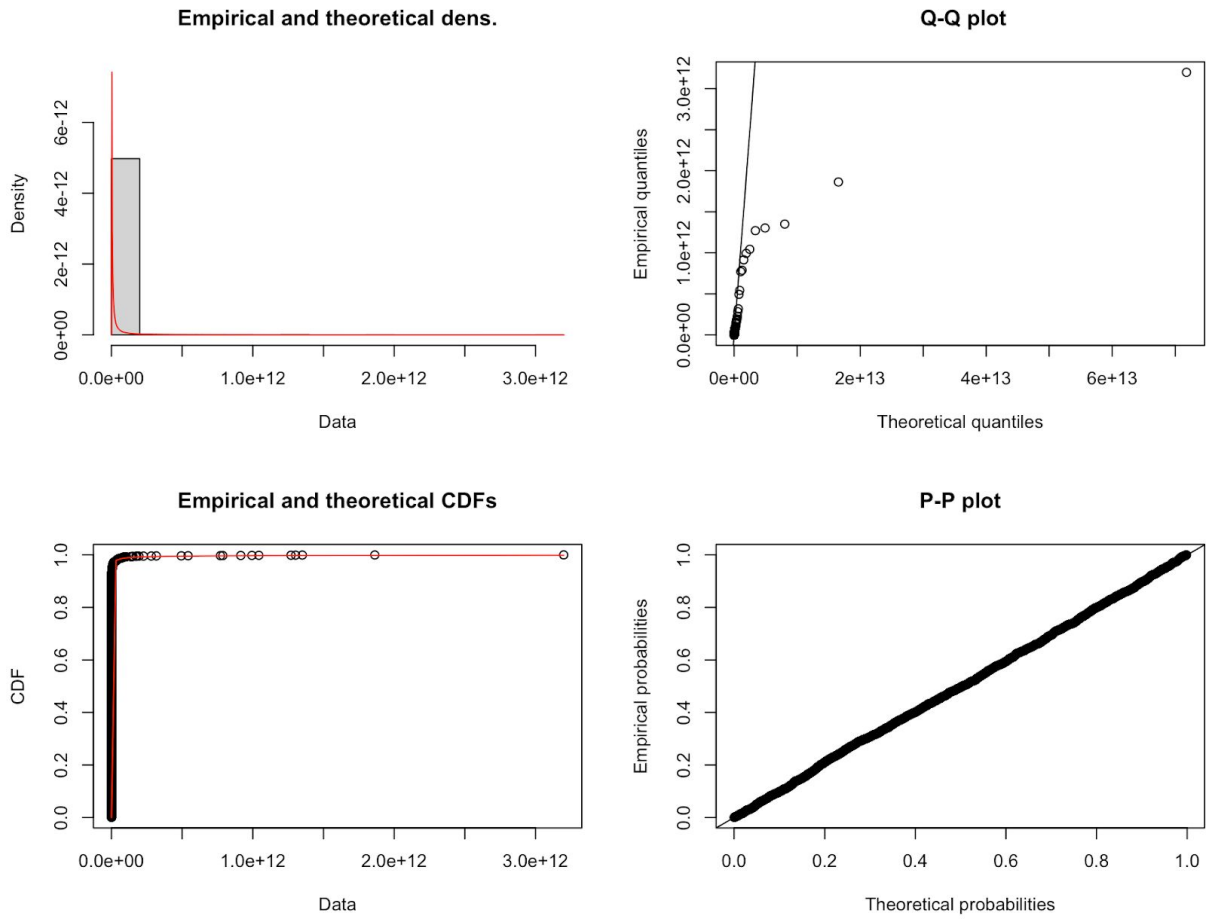
```
> distr_b <- read.csv("Assignment-5/sampling/distr_b.txt", header = F)
> fit_para <- fitdist(distr_b[,1], "norm")
> print(fit_para)
Fitting of the distribution ' norm ' by maximum likelihood
Parameters:
     estimate  Std. Error
mean 6.0523810 0.055488427
sd   2.5427992 0.039236216
> plot(fit_para)
```



3.3 Code and result of how to judge the distribution of distr_c.txt

Same process with 3.1, checked “Std. Error(sd)” the plot, which also showed that the dataset of “distr_c.txt” is basically distributed according to the “lnorm” distribution.

```
> distr_c <- read.csv("Assignment-5/sampling/distr_c.txt", header = F)
> fit_para <- fitdist(distr_c[,1], "lnorm")
> print(fit_para)
Fitting of the distribution ' lnorm ' by maximum likelihood
Parameters:
         estimate  Std. Error
meanlog 14.1580395 0.089454946
sdlog    4.9403109 0.063254188
> plot(fit_para)
```



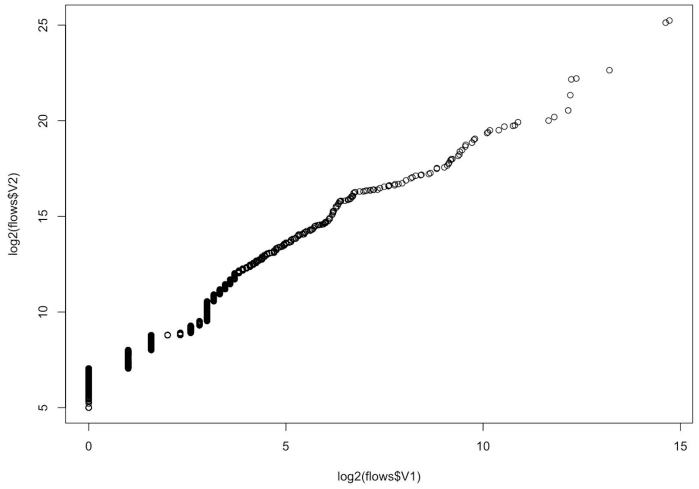
Report, task 4

Modify all the “\t” with “,” and we can see the flows.txt as follows:

```
1,44
1,40
1,40
...
```

4.1 Code and result of plotting the data and computing its mean and median for both packets and bytes

```
> flows <- read.csv("Assignment-5/sampling/flows.txt", header = F, sep = ",")
> View(flows)
> qqplot(log2(flows$V1),log2(flows$V2))
```




```
# packets (V1), bytes (V2)
> summary(flows)
       V1                    V2
Min.   :    1.0000   Min.   :       32
1st Qu.:    1.0000   1st Qu.:       40
Median :    1.0000   Median :       40
Mean   :    7.7614   Mean   :     6016
3rd Qu.:    1.0000   3rd Qu.:       88
Max.   :27037.0000   Max.   :39563775
```

4.2 Code and result of the expression for running mean

```
import pandas as pd
import numpy as np
df = pd.read_csv("/Users/yanjing/Desktop/ITMA/Assignment-5/sampling/flows.txt",
names=["packets", "bytes"], header=None)
x1_bytes = []
x2_bytes_mean = []
for i in df["bytes"]:
   x1_bytes.append(i)
   x2_bytes_mean.append(np.mean(x1_bytes))
x2_bytes_mean_df = pd.DataFrame(x2_bytes_mean)
df["bytes_mean"] = x2_bytes_mean_df
df.to_csv(r"flows_mean.csv",mode = 'a',index =False)
```

Just display some of the “flows_mean.csv” file as follows, the detailed info can be seen in AS5.zip

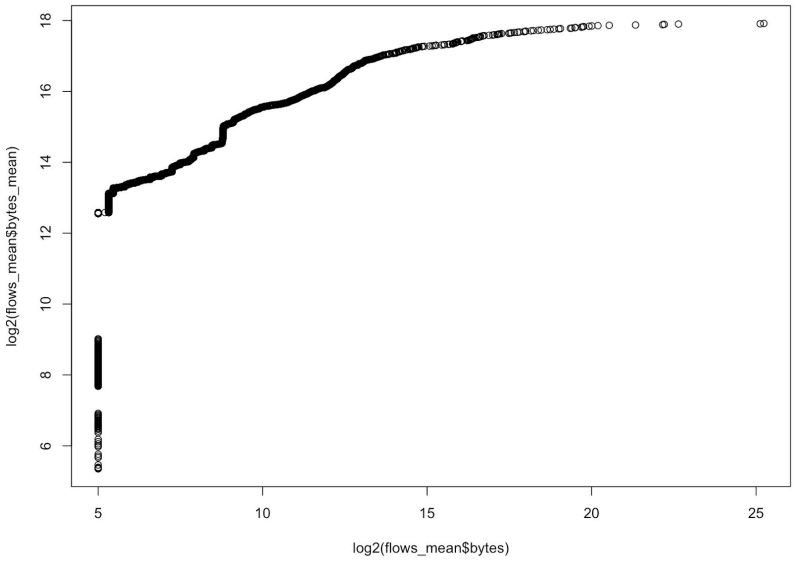
flows_mean

Filter

	packets	bytes	bytes_mean
1	1	44	44.000000
2	1	40	42.000000
3	1	40	41.333333
4	1	40	41.000000
5	1	40	40.800000
6	1	122	54.333333
7	1	40	52.285714
8	1	40	50.750000
9	1	247	72.555556
10	1	40	69.300000
11	1	40	66.636364
12	1	40	64.416667

4.3 Plot of mean estimate

```
> flows_mean <- read.csv("Assignment-5/flows_mean.csv", header = T, sep = ",")
> qqplot(log2(flows_mean$bytes),log2(flows_mean$bytes_mean))
```



4.4 Code and result of the expression for running median and the related plot

```
import pandas as pd
import numpy as np
df = pd.read_csv("/Users/yanjing/Desktop/ITMA/Assignment-5/sampling/flows.txt",
names=["packets", "bytes"], header=None)
x1_bytes = []
x2_bytes_median = []
for i in df["bytes"]:
   x1_bytes.append(i)
   x2_bytes_median.append(np.median(x1_bytes))
x2_bytes_mean_df = pd.DataFrame(x2_bytes_median)
df["bytes_median"] = x2_bytes_mean_df
df.to_csv(r"flows_median.csv",mode = 'a',index =False)
```

Just display some of the “flows_median.csv” file as follows, the detailed info can be seen in AS5.zip

flows_median

Filter

	packets	bytes	bytes_median
26	1	40	40
27	1	40	40
28	1	40	40
29	1	956	40
30	1	40	40
31	1	40	40
32	1	40	40
33	1	40	40
34	23	6777	40
35	1	40	40
36	1	40	40

```
> flows_median <- read.csv("Assignment-5/flows_median.csv", header = T, sep = ",")
> View(flows_median)
> qqplot(log2(flows_median$bytes),log2(flows_median$bytes_median))
```