ELEC-E7130 Assignment 1

Task 1: Process CSV data on command line

In this task you need to compute statistical values from a large (462 MiB) CSV file.

Aalto Linux computers have the course folder in /work/courses/unix/T/ELEC/E7130/. Under that folder is a directory general/trace/tstat/2017_04_11_18_00.out/ that has a file log_tcp_complete. It is a space-separated CSV file with 130 columns and 886467 records. The first line is the header. Provide following answers (in addition to description of your solution):

- How can you peek on file if it is too large to fit into memory?

The method is to split the large file into smaller ones, during this class, I have learned some examples. Such as, getting several lines or columns of the file through “awk” cmd, displaying some pages of the file through “more/less” cmd.
Also there are some other ways I have used, loading several columes or rows from a file through the pandas module when programming with python. Or using memory swap-in/swap-out related algorithms.

- Averages of columns 3, 7, 10, 17, 21, 24

Use awk cmd to calculate the averages of each column. Two points are needing attention, the 1st one is judging whether NR equals 0, and the 2rd one is the average is a floating-point number. The following is my results:

- Average of column 3 is “93.573”
- Average of column 7 is “36741.141”
- Average of column 10 is “0.492”
- Average of column 17 is “178.716”
- Average of column 21 is “214087.496”
- Average of column 24 is “2.286”

```
yanjing@v901-829 AS1 % awk '{sum+=$3} END {if(NR!=0) printf("Average = %.3f\n", sum/NR)}' log_tcp_complete
Average = 93.573
yanjing@v901-829 AS1 % awk '{sum+=$7} END {if(NR!=0) printf("Average = %.3f\n", sum/NR)}' log_tcp_complete
Average = 36741.141
yanjing@v901-829 AS1 % awk '{sum+=$10} END {if(NR!=0) printf("Average = %.3f\n", sum/NR)}' log_tcp_complete
Average = 0.492
yanjing@v901-829 AS1 % awk '{sum+=$17} END {if(NR!=0) printf("Average = %.3f\n", sum/NR)}' log_tcp_complete
Average = 178.716
yanjing@v901-829 AS1 % awk '{sum+=$21} END {if(NR!=0) printf("Average = %.3f\n", sum/NR)}' log_tcp_complete
Average = 214087.496
yanjing@v901-829 AS1 % awk '{sum+=$24} END {if(NR!=0) printf("Average = %.3f\n", sum/NR)}' log_tcp_complete
Average = 2.286
```

- Percentage of records where c10/c7 exceeds a) 0.01, b) 0.10, c) 0.20

(c10/c7 means for each line value in column 10 is divided by value in column 7)
Use awk cmd to calculate the percentage. FNR means all records of c10/c7 which is more than one, and n means the records that satisfy the condition in the question. Another point is that c7 should not equal 0.

- The percentage of records where c10/c7 exceeds 0.01 is “0.0170666”
- The percentage of records where c10/c7 exceeds 0.10 is “0.00495224”
- The percentage of records where c10/c7 exceeds 0.20 is “0.00302098”

```
yanjing@v901-829 AS1 % awk 'FNR>1 && $7!=0 && $10/$7>0.01{n++} END {print n/FNR}' log_tcp_complete
0.0170666
yanjing@v901-829 AS1 % awk 'FNR>1 && $7!=0 && $10/$7>0.10{n++} END {print n/FNR}' log_tcp_complete
0.00495224
yanjing@v901-829 AS1 % awk 'FNR>1 && $7!=0 && $10/$7>0.20{n++} END {print n/FNR}' log_tcp_complete
0.00302098
```

- Percentage of records where c24/c21 exceeds a) 0.01, b) 0.10, c) 0.20

Same solution with the question above.

- The percentage of records where c24/c21 exceeds 0.01 is “0.00242536”
- The percentage of records where c24/c21 exceeds 0.10 is “0.00056178”
- The percentage of records where c24/c21 exceeds 0.20 is “0.000448973”

```
yanjing@v901-829 AS1 % awk 'FNR>1 && $21!=0 && $24/$21>0.01{n++} END {print n/FNR}' log_tcp_complete
0.00242536
yanjing@v901-829 AS1 % awk 'FNR>1 && $21!=0 && $24/$21>0.10{n++} END {print n/FNR}' log_tcp_complete
0.00056178
yanjing@v901-829 AS1 % awk 'FNR>1 && $21!=0 && $24/$21>0.20{n++} END {print n/FNR}' log_tcp_complete
0.000448973
```

- Largest values in columns 3, 9, 17, 23, 31

Two methods to solve this problem.
The first one is using awk cmd totally, but the cmd should ignore the first line value of each column (The first line value contains the word like c_pkts_all, c_bytes_all...).
The second one is combining awk and sort cmds, no need to care about the first line value of each column.

```
yanjing@v901-829 AS1 % awk 'NR==2{max=$3;next}{max=max>$3?max:$3}END{print max}' log_tcp_complete
1700014
yanjing@v901-829 AS1 % awk 'NR==2{max=$9;next}{max=max>$9?max:$9}END{print max}' log_tcp_complete
313220528
yanjing@v901-829 AS1 % awk 'NR==2{max=$17;next}{max=max>$17?max:$17}END{print max}' log_tcp_complete
2791706
yanjing@v901-829 AS1 % awk 'NR==2{max=$23;next}{max=max>$23?max:$23}END{print max}' log_tcp_complete
3835783508
yanjing@v901-829 AS1 % awk 'NR==2{max=$31;next}{max=max>$31?max:$31}END{print max}' log_tcp_complete
72901107.834000
```

```
yanjing@v901-829 AS1 % cat log_tcp_complete| awk '{print $3}'| sort -n > note
yanjing@v901-829 AS1 % tail -n1 note
1700014
yanjing@v901-829 AS1 % cat log_tcp_complete| awk '{print $9}'| sort -n > note
yanjing@v901-829 AS1 % tail -n1 note
313220528
yanjing@v901-829 AS1 % cat log_tcp_complete| awk '{print $17}'| sort -n > note
yanjing@v901-829 AS1 % tail -n1 note
2791706
yanjing@v901-829 AS1 % cat log_tcp_complete| awk '{print $23}'| sort -n > note
yanjing@v901-829 AS1 % tail -n1 note
3835783508
yanjing@v901-829 AS1 % cat log_tcp_complete| awk '{print $31}'| sort -n > note
yanjing@v901-829 AS1 % tail -n1 note
72901107.834000
```

Task 2: Make a script to produce ping statistics

Use script (and crontab or other timing method) to send 20 pings every 10 minutes to ok1.iperf.comnet-student.eu over a period of 12 hours. You may want to synchronise this measurement with the following one.
For the answer produce (in addition to description of your solution and samples) calculated for each hour:

- Collect ping statistics

Note:

The Aalto server reacts so slow and I can not input some special string to the Ubuntu 18.04 terminal (such as @, / and so on), which affects my work seriously. Then I chose to use my own computer, I did use crontab in another linux OS and it worked well. But currently the crontab can not take effect after editing and saving on my computer with MACOS, seems it needs some extra conf related launchctl.

So I did not use crontab and used a shell for loop to make it, and then according to the results (which is saved as */Users/yanjing/Desktop/ITMA/AS1/log*) generated by the following script to finish the following questions. After running for more than 16 hours, I got the following 100 records showed as following. (Each record starts with the date info.)

```
yanjing@v901-829 AS1 % cat run1.sh
#! /bin/bash
for i in {1..100}
do
date >> /Users/yanjing/Desktop/ITMA/AS1/log
ping ok1.iperf.comnet-student.eu -c 20 >> /Users/yanjing/Desktop/ITMA/AS1/log
sleep 600
done
```

```
# cat log
...
Sun Sep 20 06:38:52 EEST 2020
PING iperf.netlab.hut.fi (195.148.124.36): 56 data bytes
64 bytes from 195.148.124.36: icmp_seq=0 ttl=60 time=312.148 ms
64 bytes from 195.148.124.36: icmp_seq=1 ttl=60 time=325.472 ms
64 bytes from 195.148.124.36: icmp_seq=2 ttl=60 time=425.519 ms
64 bytes from 195.148.124.36: icmp_seq=3 ttl=60 time=363.061 ms
Request timeout for icmp_seq 4
Request timeout for icmp_seq 5
64 bytes from 195.148.124.36: icmp_seq=6 ttl=60 time=309.678 ms
64 bytes from 195.148.124.36: icmp_seq=7 ttl=60 time=322.656 ms
64 bytes from 195.148.124.36: icmp_seq=8 ttl=60 time=270.163 ms
64 bytes from 195.148.124.36: icmp_seq=9 ttl=60 time=357.397 ms
64 bytes from 195.148.124.36: icmp_seq=10 ttl=60 time=266.656 ms
Request timeout for icmp_seq 11
64 bytes from 195.148.124.36: icmp_seq=12 ttl=60 time=301.421 ms
64 bytes from 195.148.124.36: icmp_seq=13 ttl=60 time=317.923 ms
64 bytes from 195.148.124.36: icmp_seq=14 ttl=60 time=332.910 ms
Request timeout for icmp_seq 15
Request timeout for icmp_seq 16
Request timeout for icmp_seq 17
64 bytes from 195.148.124.36: icmp_seq=18 ttl=60 time=363.503 ms

--- iperf.netlab.hut.fi ping statistics ---
20 packets transmitted, 13 packets received, 35.0% packet loss
round-trip min/avg/max/stddev = 266.656/328.347/425.519/40.596 ms
...
```

- Percentage of lost packets.

Dealing with the log file as following extracts the data needed.

The four columns in “task2-q1.csv” file describe “timestamp”, “transmitted packages”, “received packages” and “loss package” info respectively.

```
yanjing@v901-829 AS1 % echo "timestamp" > tmp1
yanjing@v901-829 AS1 % cat log | grep "EEST" | awk '{print $4}' >> tmp1
yanjing@v901-829 AS1 % echo "transmitted" > tmp2
yanjing@v901-829 AS1 % cat log | grep "transmitted" | awk '{print $1}' >> tmp2
yanjing@v901-829 AS1 % echo "received" > tmp3
yanjing@v901-829 AS1 % cat log | grep "transmitted" | awk '{print $4}' >> tmp3
yanjing@v901-829 AS1 % echo "loss" > tmp4
yanjing@v901-829 AS1 % cat log | grep "transmitted" | awk '{print $7}' >> tmp4
yanjing@v901-829 AS1 % paste -d "\t" tmp1 tmp2 tmp3 tmp4 > task2-q1.csv
yanjing@v901-829 AS1 %
yanjing@v901-829 AS1 % vim task2-q1.csv (with :%s/\t/,/g)
zsh: unknown file attribute: i
yanjing@v901-829 AS1 % head -n2 task2-q1.csv
timestamp,transmitted,received,loss
14:04:59,20,18,10.0%
```

Run the following code and get the result for each hour:

```
import pandas as pd
# Load data
filepath1 = r'/Users/yanjing/Desktop/ITMA/AS1/task2-q1.csv'
csv1 = pd.read_csv(filepath1)
# Compute the percentage of lost packages
dic1 = {}
for timestart in ["14", "15", "16", "17", "18", "19", "20", "21", "22", "23",
"00", "01", "02", "03", "04", "05", "06", "07"]:
   dic1[timestart] = {"transmitted":0, "received":0}
   for index,row in csv1.iterrows():
       timestamp = row["timestamp"]
       transmitted = row["transmitted"]
       received = row["received"]
       if timestamp.startswith(timestart):
         dic1[timestart]["transmitted"] = dic1[timestart]["transmitted"] +
int(transmitted)
         dic1[timestart]["received"] = dic1[timestart]["received"] +
int(received)
for clock in dic1:
     print("At clock %s, the percentage of lost packages is %.3f"%(clock,
1-dic1[clock]["received"]/dic1[clock]["transmitted"]))
```

```
yanjing@v901-829 AS1 % python3 task2-q1.py
At clock 14, the percentage of lost packages is 0.075
At clock 15, the percentage of lost packages is 0.183
At clock 16, the percentage of lost packages is 0.260
At clock 17, the percentage of lost packages is 0.425
At clock 18, the percentage of lost packages is 0.475
At clock 19, the percentage of lost packages is 0.442
At clock 20, the percentage of lost packages is 0.383
At clock 21, the percentage of lost packages is 0.290
At clock 22, the percentage of lost packages is 0.183
At clock 23, the percentage of lost packages is 0.033
At clock 00, the percentage of lost packages is 0.000
At clock 01, the percentage of lost packages is 0.000
At clock 02, the percentage of lost packages is 0.040
At clock 03, the percentage of lost packages is 0.200
At clock 04, the percentage of lost packages is 0.200
At clock 05, the percentage of lost packages is 0.325
At clock 06, the percentage of lost packages is 0.300
At clock 07, the percentage of lost packages is 0.300
```

- Average of successful round-trip-time (RTT) measurements.

Dealing with the log file as following extracts the data needed.
The two columns in “task2-q2.csv” file describe “timestamp” and “round-trip-avg” info.
According to the previous question, at clock 00 and 01, the percentages of lost packages are 0, so we just care about these two.

```
yanjing@v901-829 AS1 % echo "timestamp" > tmp1
yanjing@v901-829 AS1 % cat log | grep "EEST" | awk '{print $4}' >> tmp1
yanjing@v901-829 AS1 % echo "round-trip-avg" > tmp2
yanjing@v901-829 AS1 % cat log | grep round-trip | awk '{print $4}' | awk -F'/' '{print $2}' >> tmp2
yanjing@v901-829 AS1 % paste -d "\t" tmp1 tmp2 > task2-q2.csv
yanjing@v901-829 AS1 % vim task2-q2.csv (with :%s/\t/,/g)
zsh: unknown file attribute: i
yanjing@v901-829 AS1 % head -n2 task2-q2.csv
timestamp,round-trip-avg
14:04:59,324.174
```

Run the following code and get the result for each hour:

```
import pandas as pd
# Load data
filepath2 = r'/Users/yanjing/Desktop/ITMA/AS1/task2-q2.csv'
csv2 = pd.read_csv(filepath2)
# Compute the percentage of lost packages
for timestart in ["00", "01"]:
    count = 0
    sum_round_trip_avg = 0
    for index,row in csv2.iterrows():
        timestamp = row["timestamp"]
        round_trip_avg = row["round-trip-avg"]
        if timestamp.startswith(timestart):
```

```
            count = count + 1
            sum_round_trip_avg = sum_round_trip_avg + round_trip_avg
    print("At clock %s, the average of RTT is %.3f"%(timestart,
sum_round_trip_avg/count))
```

```
yanjing@v901-829 AS1 % python3 task2-q2.py
At clock 00, the average of RTT is 316.428
At clock 01, the average of RTT is 329.429
```

- Median of RTT. Lost packet is counted RTT as inf.

Dealing with the log file as following extracts the data needed.
The four columns in “task2-q3.csv” file describe “min”, “avg”, “max” and “stddev” of round-trip time info.

```
yanjing@v901-829 AS1 % echo "timestamp" > tmp1
yanjing@v901-829 AS1 % cat log | grep "EEST" | awk '{print $4}' >> tmp1
yanjing@v901-829 AS1 % echo "min/avg/max/stddev" > tmp2
yanjing@v901-829 AS1 % cat log | grep round-trip | awk '{print $4}' | awk -F'/' {print$1} >> tmp2
yanjing@v901-829 AS1 % paste -d "/" tmp1 tmp2 > task2-q3.csv
yanjing@v901-829 AS1 % vim task2-q3.csv (with %s/\//,/g)
zsh: unknown file attribute: i
yanjing@v901-829 AS1 % head -n3 task2-q3.csv
timestamp,min,avg,max,stddev
14:04:59,262.848,324.174,387.999,39.593
14:15:20,261.902,351.342,444.775,51.058
```

Run the following code and get the result for each hour:

```
import pandas as pd
import numpy as np
# Load data
filepath3 = r'/Users/yanjing/Desktop/ITMA/AS1/task2-q3.csv'
csv3 = pd.read_csv(filepath3)
# Compute the percentage of lost packages
dic1 = {}
for timestart in ["00", "01"]:
    dic1[timestart]={"rtt_min":[],"rtt_avg":[],"rtt_max":[],"rtt_stddev":[]}
    for index,row in csv3.iterrows():
        timestamp = row["timestamp"]
        rtt_min_va = float('%.3f'%row["min"])
        rtt_avg_va = float('%.3f'%row["avg"])
        rtt_max_va = float('%.3f'%row["max"])
        rtt_stddev_va = float('%.3f'%row["stddev"])
        if timestamp.startswith(timestart):
            dic1[timestart]["rtt_min"].append(rtt_min_va)
            dic1[timestart]["rtt_avg"].append(rtt_avg_va)
            dic1[timestart]["rtt_max"].append(rtt_max_va)
            dic1[timestart]["rtt_stddev"].append(rtt_stddev_va)
```

```
    for item in dic1[timestart]:
        print("At clock %s, the median of %s is %f"%(timestart, item,
np.median(dic1[timestart][item])))
```

```
yanjing@v901-829 AS1 % python3 task2-q3.py
At clock 00, the median of rtt_min is 251.396500
At clock 00, the median of rtt_avg is 317.335000
At clock 00, the median of rtt_max is 387.257500
At clock 00, the median of rtt_stddev is 43.209000
At clock 01, the median of rtt_min is 251.915000
At clock 01, the median of rtt_avg is 326.610000
At clock 01, the median of rtt_max is 396.215000
At clock 01, the median of rtt_stddev is 44.951000
```

Task 3: Measure network throughput performance

Use script (with crontab or alternative) to run iperf3 once an hour against ok1.iperf.comnet-student.eu over a period of 12 hours, 10 seconds run each direction (minimum 24 runs total). You can run it from your own computer or you can use Aalto servers.
As the iperf3 server supports only one session at time, the runs will be distributed according to following formula:

- minute: your student number modulo 60
- port: random selection of ports from 5200 to 5210 (inclusive, random for each run)

Produce an CSV file including a record for each run including columns:

- timestamp
- time of start (in hour:minute:second)
- total bytes transferred
- bitrate
- number of TCP retransmissions

Include a sample of CSV file, tell how you produced the CSV data and graph.
For the answer produce (in addition to description of your solution and sample of CSV file): Average bitrate for all measurements.

- Collect iperf3 statistics and generate a csv file

After collecting the data with the following script, I got a file (named “task3”) with several jsons objects inside. The format is like {} {} {}... , and I dealt with the file by changing it to [{},{},{}...] which named “task3_log.json”

```
yanjing@v901-829 AS1 % cat task3.sh
while true
do
iperf3 -c ok1.iperf.comnet-student.eu -p $(( $RANDOM % 11 + 5200)) --logfile task3 -J
sleep 3600
done
```

Running the following code to produce the csv file according to the “task3_log.json” provided by the previous question.

```
import csv
import json
import os
# Load date
jsonfile = open("/Users/yanjing/Desktop/ITMA/AS1/task3/task3_log.json", 'r+')
alldata = json.load(jsonfile)
# Prepare csv file and input date
os.system("echo ''> /Users/yanjing/Desktop/ITMA/AS1/task3/task3.csv")
csvfile = open("/Users/yanjing/Desktop/ITMA/AS1/task3/task3.csv","w")
writer = csv.writer(csvfile)
writer.writerow(["timestamp", "time_start", "total_bytes_transferred",
"sum_sent_bitrate", "sum_received_bitrate", "num_of_tcp_retr"])
for i in range(len(alldata)):
   writer.writerow([alldata[i]['start']['timestamp']['timesecs'],
alldata[i]['start']['timestamp']['time'], alldata[i]['end']['sum_sent']['bytes'],
alldata[i]['end']['sum_sent']['bits_per_second'],
alldata[i]['end']['sum_received']['bits_per_second'],
alldata[i]['end']['sum_sent']['retransmits']])
```

```
yanjing@v901-829 AS1 % python3 task3/task3.py
yanjing@v901-829 AS1 % head -n3 task3/task3.csv
timestamp,time_start,total_bytes_transferred,sum_sent_bitrate,sum_received_bitrate,num_of_tcp_retr
1600597486,"Sun, 20 Sep 2020 10:24:46 GMT",10249671496,8199675699.232255,8197561608.7736635,0
1600601096,"Sun, 20 Sep 2020 11:24:56 GMT",7581694840,6065310382.172133,6062437286.198896,199
```

- Minimum, median and maximum bitrate

Calculated the min/median/max bitrate through the following code, and the source date is based on the csv file generated in the previous question.

```
import pandas as pd
# Load data
filepath1 = r'/Users/yanjing/Desktop/ITMA/AS1/task3/task3.csv'
csv1 = pd.read_csv(filepath1)
# Compute the min/median/max of sum_sent-bits_per_second
print("The min of sum_sent-bits_per_second is
%f."%csv1["sum_sent_bitrate"].min())
print("The median of sum_sent-bits_per_second is
%f."%csv1["sum_sent_bitrate"].median())
print("The max of sum_sent-bits_per_second is
%f."%csv1["sum_sent_bitrate"].max())
# Compute the min/median/max of sum_received-bits_per_second
```

```
print("The min of sum_received-bits_per_second is
%f."%csv1["sum_received_bitrate"].min())
print("The median of sum_received-bits_per_second is
%f."%csv1["sum_received_bitrate"].median())
print("The max of sum_received-bits_per_second is
%f."%csv1["sum_received_bitrate"].max())
```

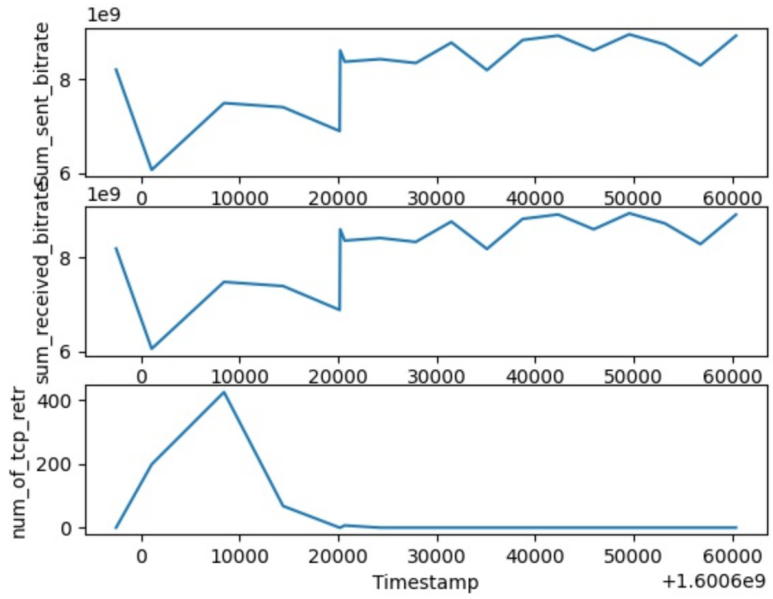
```
yanjing@v901-829 AS1 % python3 task3/task3-q1.py
The min of sum_sent-bits_per_second is 6065310382.172132.
The median of sum_sent-bits_per_second is 8396571916.567817.
The max of sum_sent-bits_per_second is 8951622314.630077.
The min of sum_received-bits_per_second is 6062437286.198896.
The median of sum_received-bits_per_second is 8394036124.356885.
The max of sum_received-bits_per_second is 8948712823.857042.
```

- Graph comparing bitrate and number of TCP retransmissions.

Draw the graph based on the three columns in the csv file.

```
import pandas as pd
from matplotlib import pyplot as plt
# Load data
filepath1 = r'/Users/yanjing/Desktop/ITMA/AS1/task3/task3.csv'
csv1 = pd.read_csv(filepath1)[['timestamp', 'sum_sent_bitrate',
'sum_received_bitrate', 'num_of_tcp_retr']]
# Draw graph
fig = plt.figure()
ax1 = fig.add_subplot(3,1,1)
ax1.set_ylabel('Sum_sent_bitrate')
ax1.set_xlabel('Timestamp')
ax1.plot(csv1['timestamp'], csv1['sum_sent_bitrate'])
ax2 = fig.add_subplot(3,1,2)
ax2.set_ylabel('sum_received_bitrate')
ax2.set_xlabel('Timestamp')
ax2.plot(csv1['timestamp'], csv1['sum_received_bitrate'])
ax2 = fig.add_subplot(3,1,3)
ax2.set_ylabel(num_of_tcp_retr)
ax2.set_xlabel('Timestamp')
ax2.plot(csv1['timestamp'], csv1['num_of_tcp_retr'])
plt.savefig("ploy1.jpg")
```

```
yanjing@v901-829 task3 % python3 task3-q2.py
```



- If you used your own computer to run iperf, describe network connectivity. Can you make any conclusions of stability based on data?

Basically, there are two points I have found.
The first one is that the lower the num_of_tcp_retr, the more stable the network is.
The second one is that the value of sum_received_bitrate/sum_sent_bitrate is closer to 1, then the network tends to be more stable.
(Also seems that the num_of_tcp_retr has some kind of negative correlation with sum_received_bitrate and sum_sent_bitrate through the graph in the previous question.)