

COMP9334 Project Report

Name: Xue WEN

zID: z5149548

1. Inter-arrival, service time and network latency distribution

(1) For inter-arrival probability distribution, I use a library of `random.expovariate` generate a series of pseudo-random numbers, which are exponentially distributed.

```
def getArrivalTime(lam, time_end):
    #random.seed(0)
    arrival_list = []
    arrival_time = 0
    while arrival_time < time_end:
        inter_arrival_time = random.expovariate(lam)
        inter_arrival_time = round(inter_arrival_time, 5)
        arrival_time = round((arrival_time + inter_arrival_time), 5)
        arrival_list.append(arrival_time)
    for i in range(len(arrival_list)):
        if arrival_list[i] > time_end:
            arrival_list.pop(i)
    arrival_list.sort()
    return arrival_list
```

Fig 1. Generate the arrival time list

In order to see the correctness of my simulation program in this part, I wrote a program named `chart.py` to plot the following figure. The data I used to plot this figure is as follow:

Parameter	Value
$\text{lam}(\lambda)$	9.72
seed	1
time_end	1000

The plot is as following:

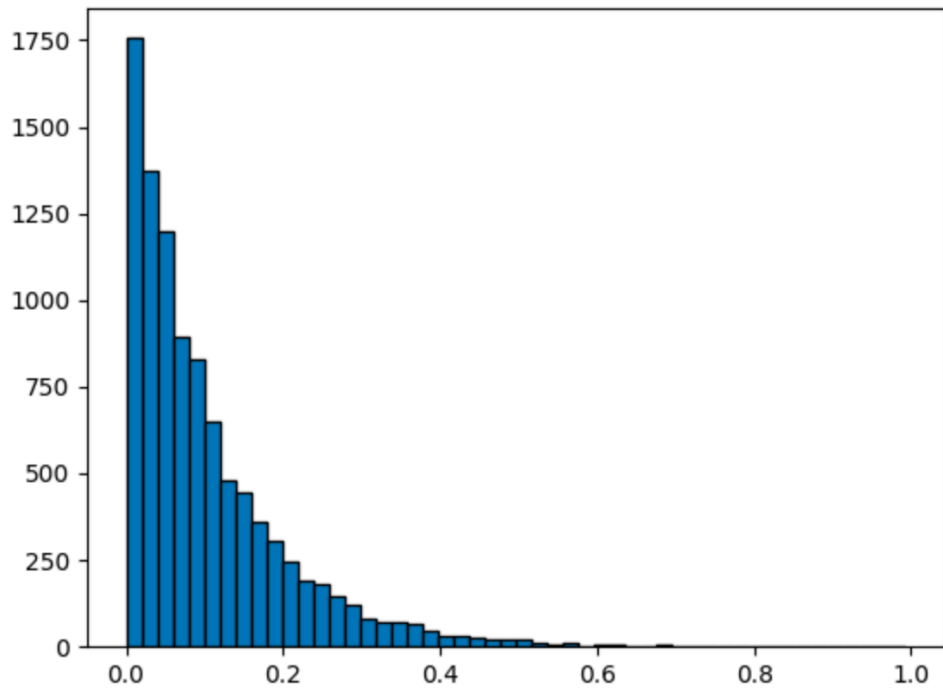


Fig 2. Histogram of exponentially distributed pseudo-random number

(2) For service time distribution, I use the following code to generate a series of pseudo-random numbers, which are following the probability density function given by the project documentation.

```
def getServiceTime(alpha1, alpha2, beta, requests_number):
    # calculate gama
    #random.seed(0)
    gama = (1.0 - beta) / ((alpha2 ** (1.0 - beta)) - (alpha1 ** (1.0 - beta)))
    service_list_random = []
    for i in range(0, requests_number):
        prob = random.random()
        prob = round(prob, 5)
        time = (prob * (1.0 - beta) / gama + alpha1 ** (1.0 - beta)) ** (1.0 / (1.0 - beta))
        service_list_random.append(round(time, 5))
    #print(len(service_list_random))
    return service_list_random
```

Fig 3. Generate the service time list

In order to see the correctness of my simulation program in this part, I wrote a program named chart.py to plot the following figure. The data I used to plot this figure is as follow:

Parameter	Value
alpha1	0.01
alpha2	0.4
beta	0.86
seed	1
requests_number	length(arrival_list)=9715

The plot is as following:

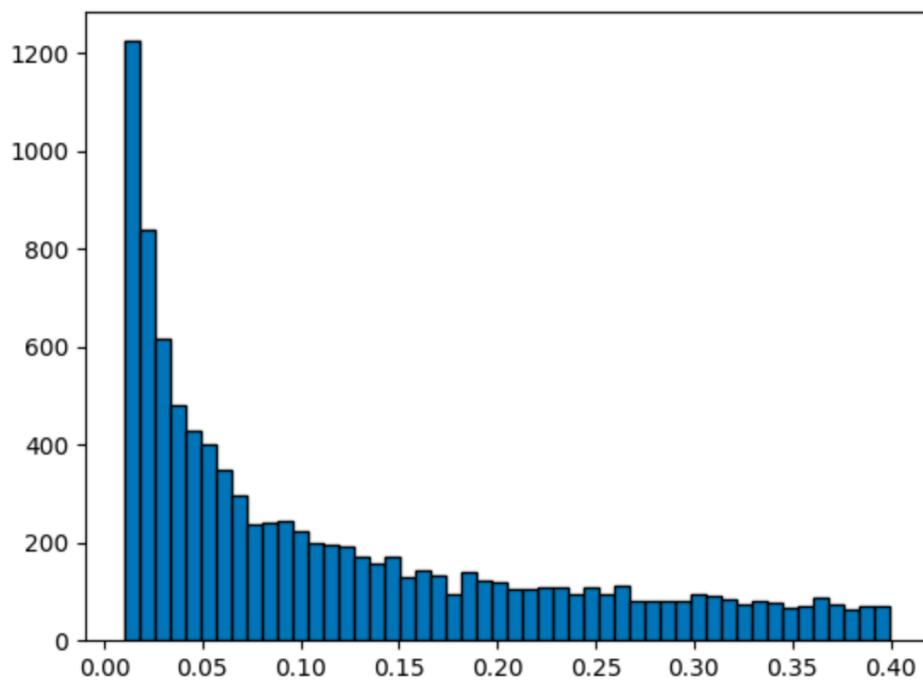


Fig 4. Histogram of service time probability distribution

(3) For network latency distribution, I use the following code to generate a series of pseudo-random numbers, which are following the uniform distribution.

```
def getNetworkLatency(v1,v2,requests_number):  
    #random.seed(0)  
    network_latency = []  
    for i in range(0, requests_number):  
        network = random.uniform(v1,v2)  
        network = round(network,5)  
        network_latency.append(network)  
    #print(len(network_latency))  
    return network_latency
```

Fig 5. Generate the network latency list

In order to see the correctness of my simulation program in this part, I wrote a program named chart.py to plot the following figure. The data I used to plot this figure is as follow:

Parameter	Value
v1	1.2
v2	1.47
seed	1
fogTimeLimit	0.2
requests_number	length(arrival_list)=9715

The plot is as following:

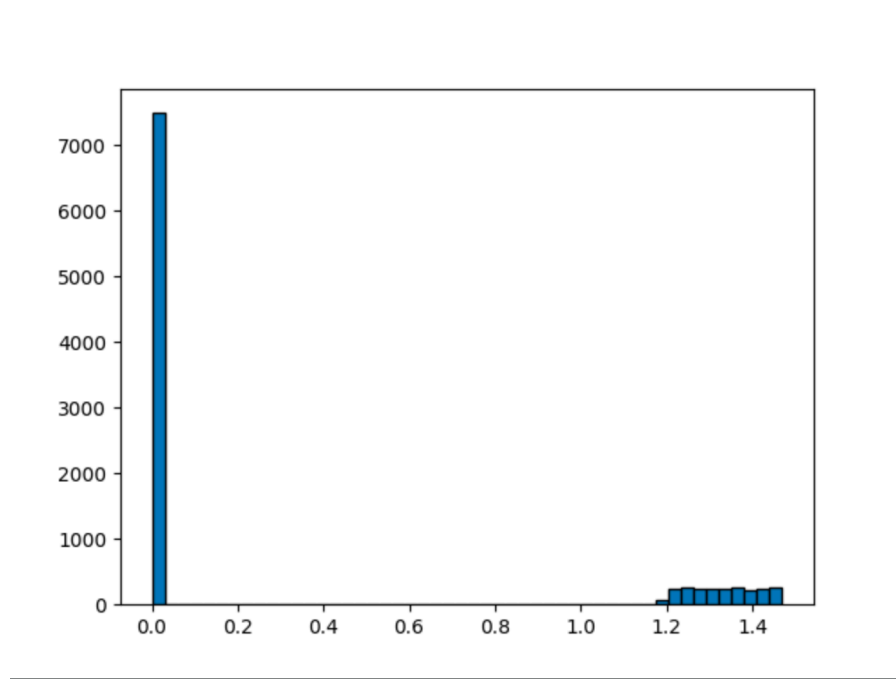


Fig 6. Histogram of uniform distribution

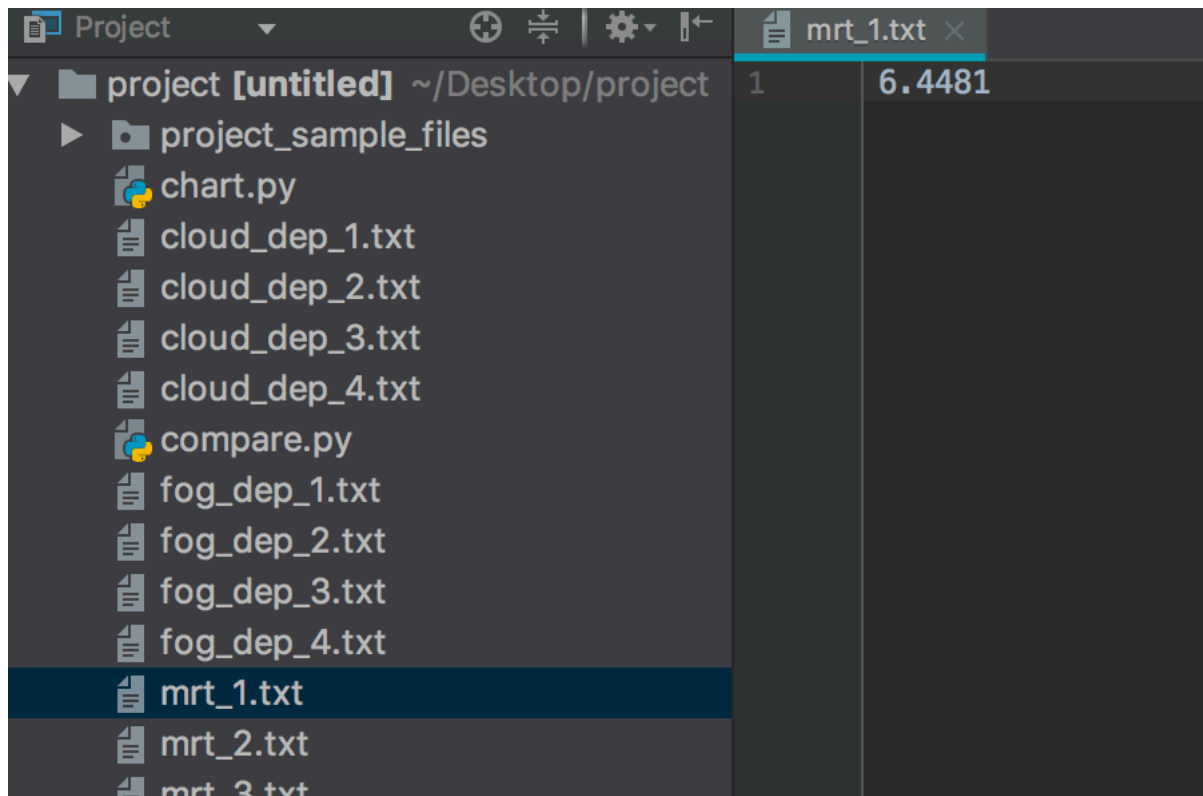
2. Simulation Code

In order to see the correctness of my simulation program, there are 3 samples as follow:

Example 1:

arrival	service	network	fogTimeLimit	fogTimeToCloudTime
1.100	4.100	1.500	2.000	0.600
6.200	5.200	1.300		
7.400	1.300	0.000		
8.300	2.000	0.000		
9.100	3.200	1.600		
10.100	4.100	1.800		

The mean response time is as following:



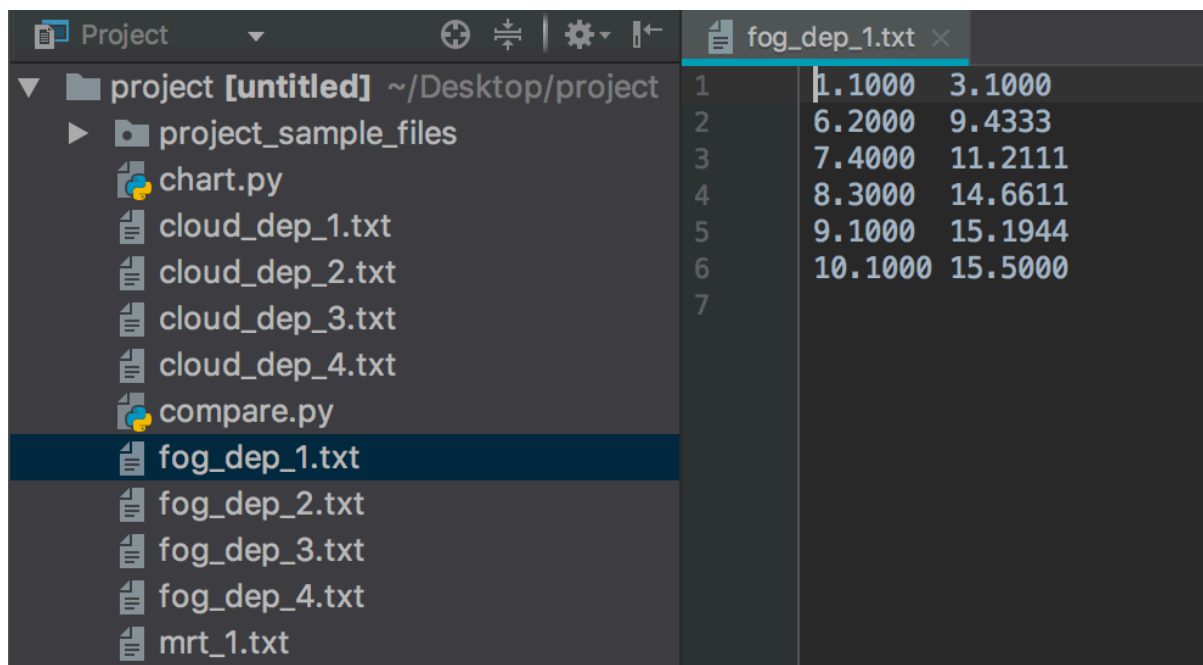
Project [untitled] ~/Desktop/project

project_sample_files

- chart.py
- cloud_dep_1.txt
- cloud_dep_2.txt
- cloud_dep_3.txt
- cloud_dep_4.txt
- compare.py
- fog_dep_1.txt
- fog_dep_2.txt
- fog_dep_3.txt
- fog_dep_4.txt
- mrt_1.txt**
- mrt_2.txt
- mrt_3.txt

	mrt_1.txt
1	6.4481

The fog departure time is as following:



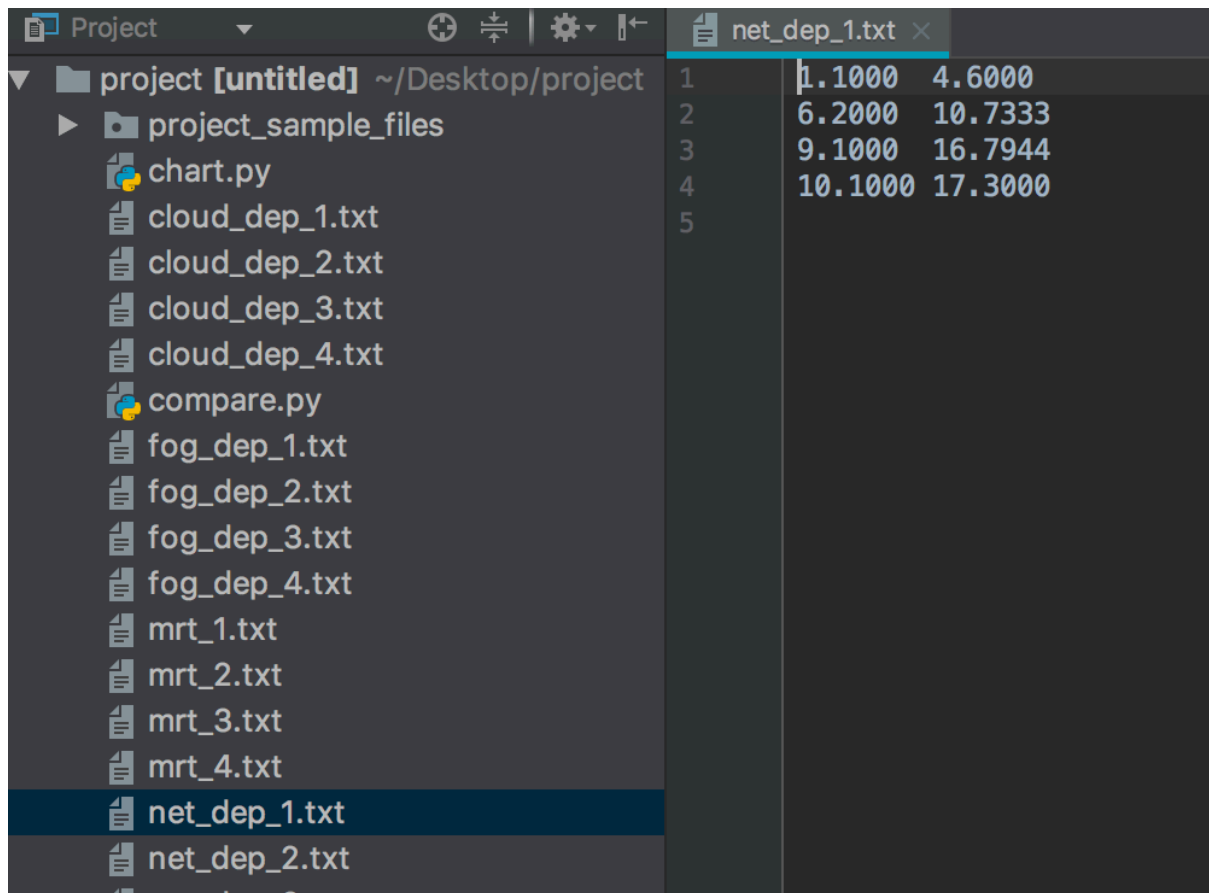
Project [untitled] ~/Desktop/project

project_sample_files

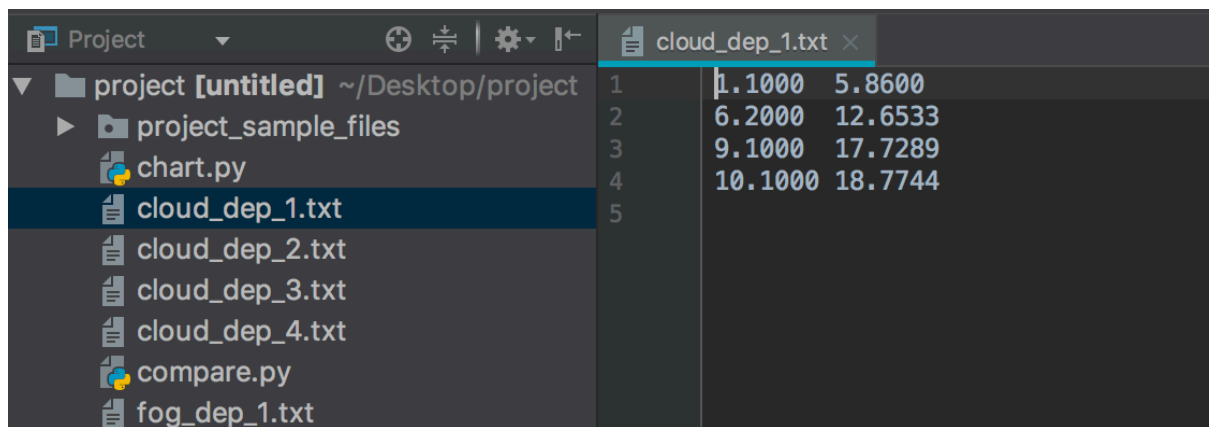
- chart.py
- cloud_dep_1.txt
- cloud_dep_2.txt
- cloud_dep_3.txt
- cloud_dep_4.txt
- compare.py
- fog_dep_1.txt**
- fog_dep_2.txt
- fog_dep_3.txt
- fog_dep_4.txt
- mrt_1.txt

	fog_dep_1.txt
1	1.1000 3.1000
2	6.2000 9.4333
3	7.4000 11.2111
4	8.3000 14.6611
5	9.1000 15.1944
6	10.1000 15.5000
7	

The network departure time is as following:



The cloud departure time is as following:

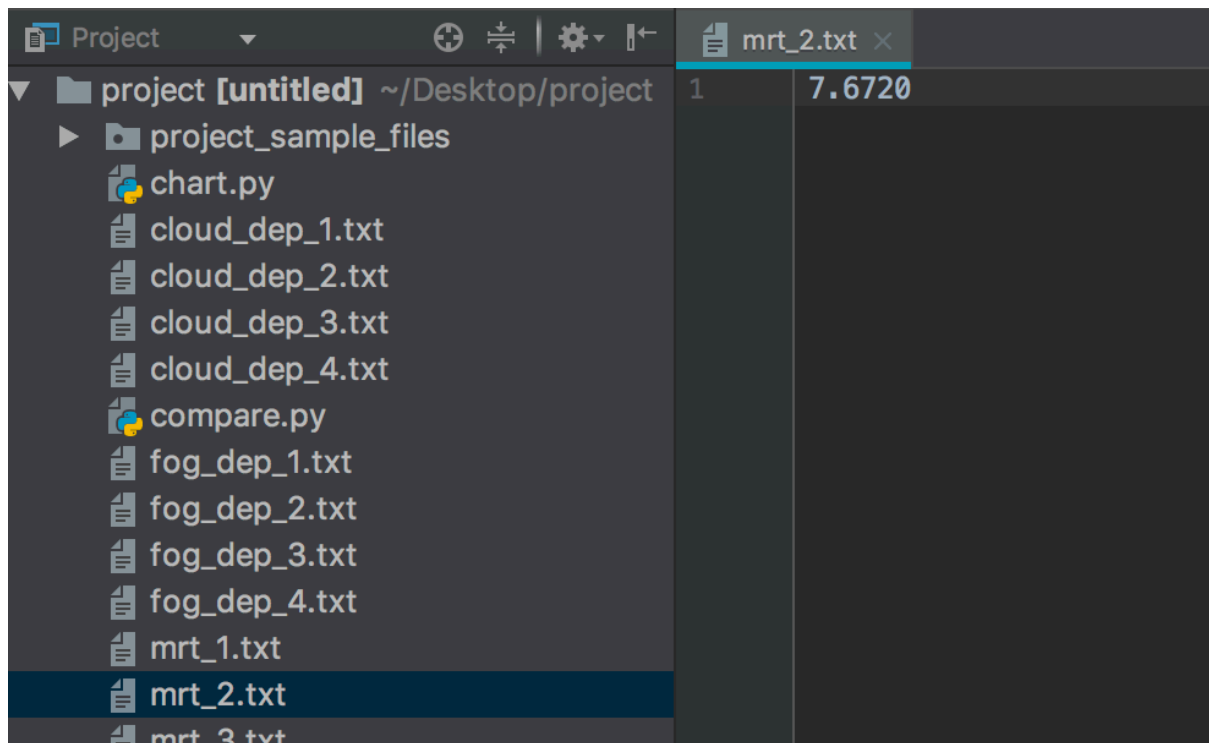


Example 2:

arrival	service	network	fogTimeLimit	fogTimeToCloudTime
1.000	3.700	1.500	2.500	0.700
2.000	5.100	1.400		

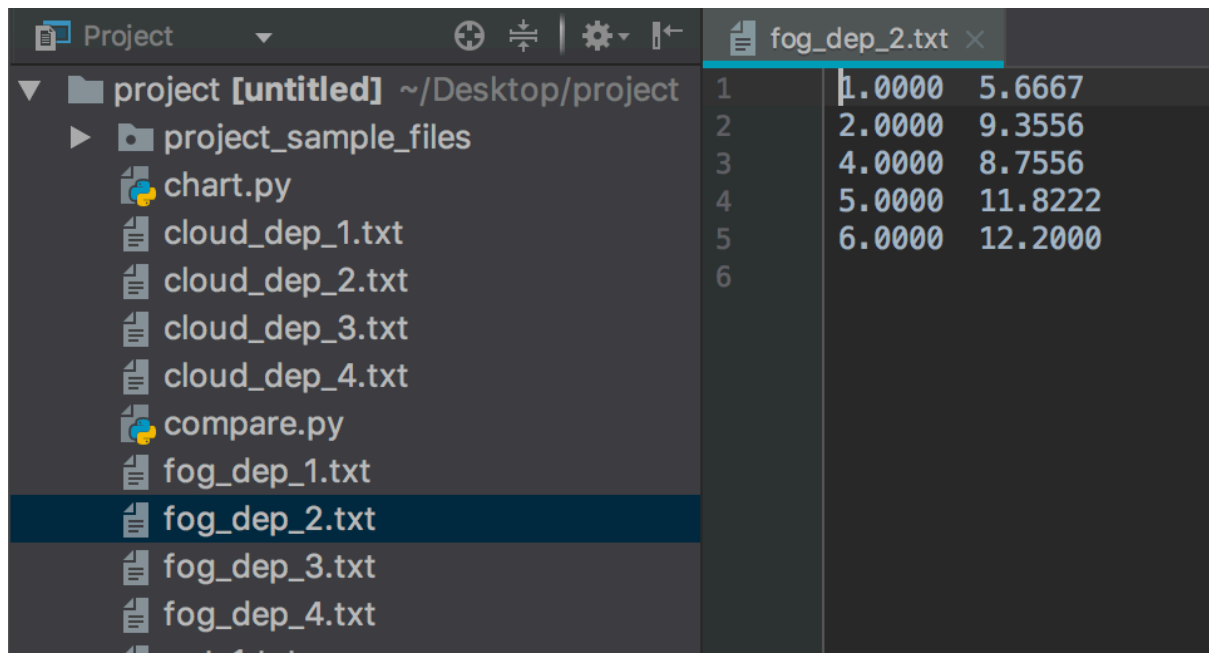
4.000	1.300	0.000		
5.000	2.400	0.000		
6.000	4.500	1.600		

The mean response time is as following:



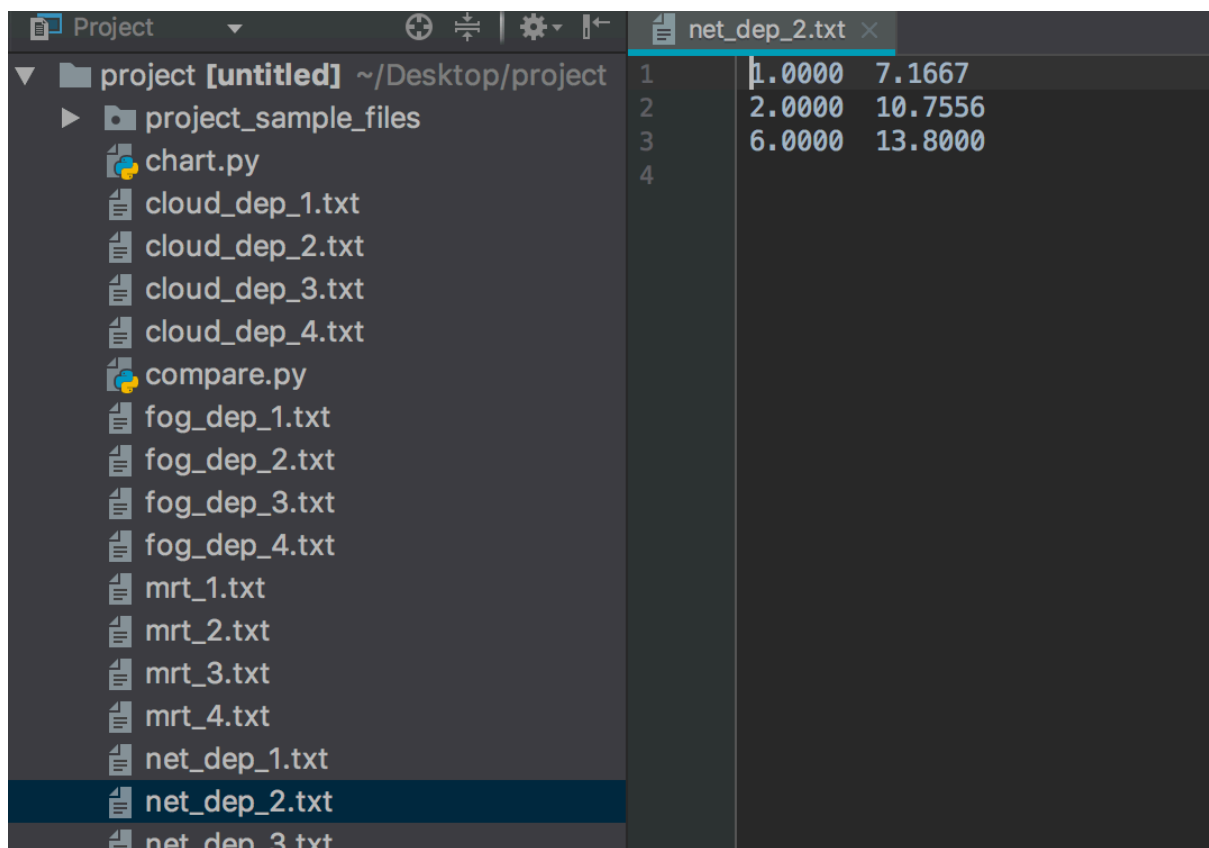
1	7.6720
---	--------

The fog departure time is as following:



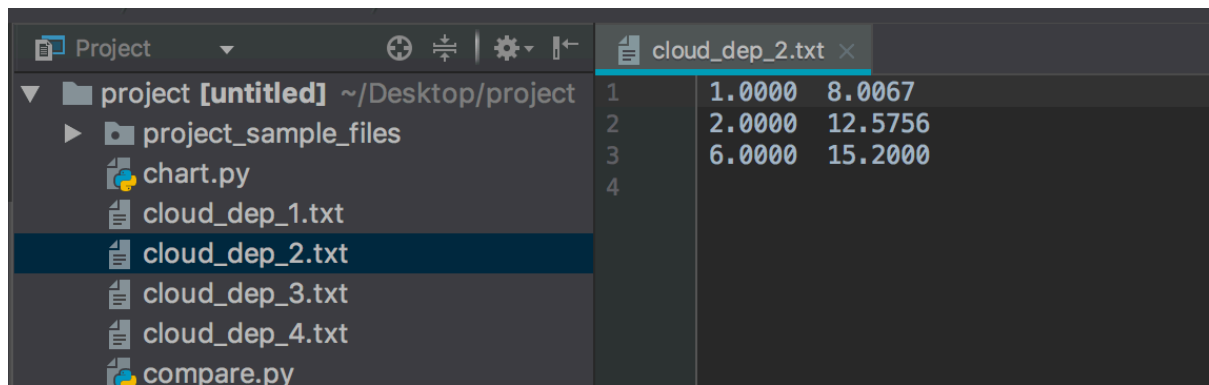
Index	Value 1	Value 2
1	1.0000	5.6667
2	2.0000	9.3556
3	4.0000	8.7556
4	5.0000	11.8222
5	6.0000	12.2000
6		

The network departure time is as following:



Index	Value 1	Value 2
1	1.0000	7.1667
2	2.0000	10.7556
3	6.0000	13.8000
4		

The cloud departure time is as following:

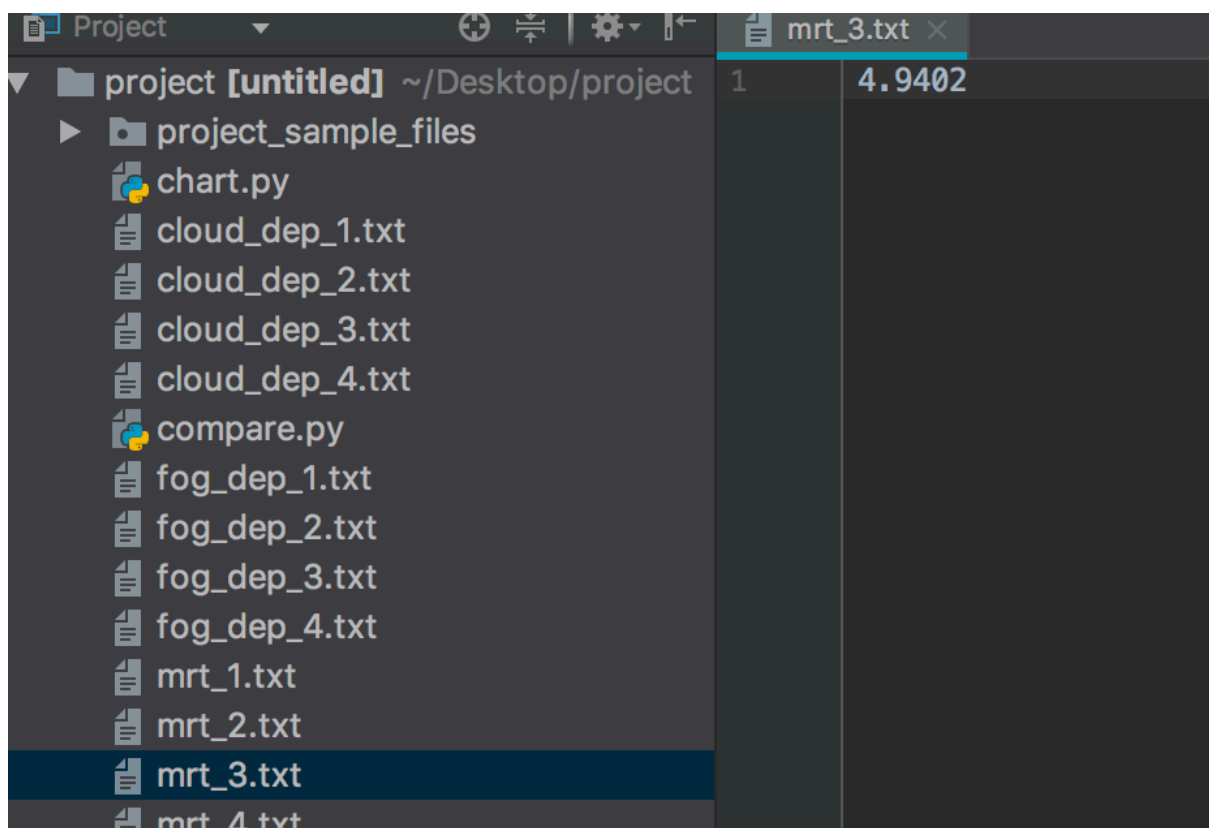


Example 3:

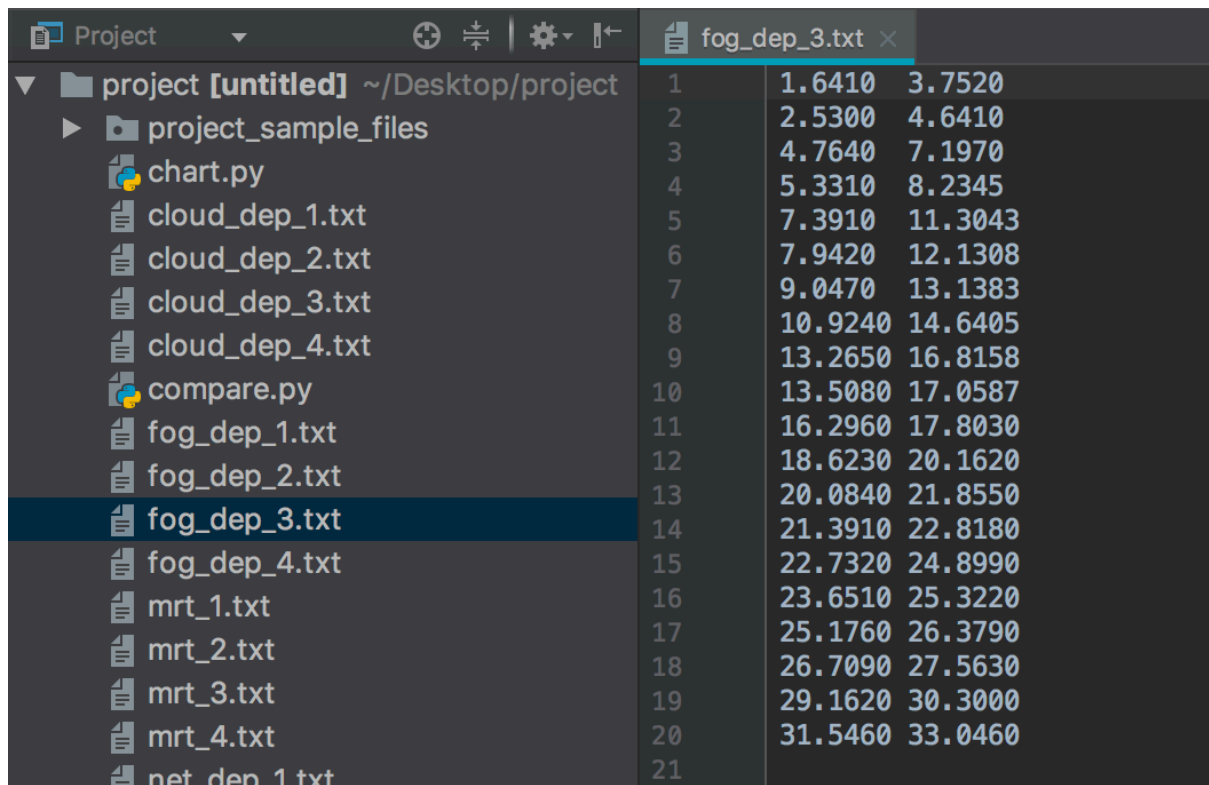
arrival	service	network	fogTimeLimit	fogTimeToCloudTime
1.641	3.222	0.622	1.500	0.800
2.530	1.893	1.847		
4.764	4.058	0.860		
5.331	2.664	0.370		
7.391	1.754	1.810		
7.942	4.695	1.959		
9.047	4.380	0.878		
10.924	2.751	0.222		
13.265	3.112	0.516		
13.508	2.935	0.817		
16.296	1.039	0.000		
18.623	1.506	0.524		
20.084	2.355	1.206		
21.391	1.152	0.000		
22.732	4.222	0.443		

23.651	0.974	0.000		
25.176	1.130	0.000		
26.709	0.854	0.000		
29.162	1.138	0.000		
31.546	2.178	1.016		

The mean response time is as following:



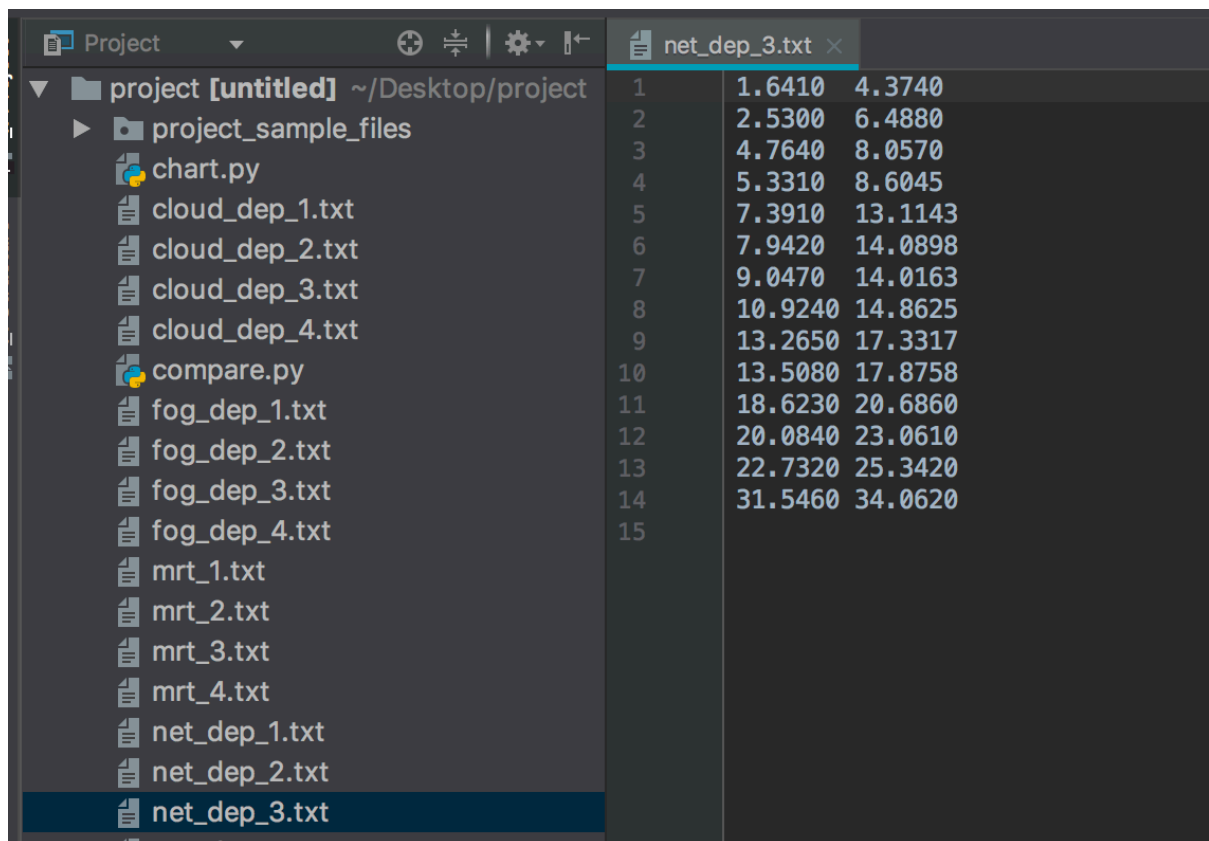
The fog departure time is as following:



The screenshot shows a code editor with a file explorer on the left and a table of data in the fog_dep_3.txt file on the right. The file explorer shows a project named 'project [untitled]' located at '~\Desktop\project'. Inside the project, there is a folder 'project_sample_files' containing several files: chart.py, cloud_dep_1.txt, cloud_dep_2.txt, cloud_dep_3.txt, cloud_dep_4.txt, compare.py, fog_dep_1.txt, fog_dep_2.txt, fog_dep_3.txt (selected), fog_dep_4.txt, mrt_1.txt, mrt_2.txt, mrt_3.txt, mrt_4.txt, and net_dep_1.txt. The table in fog_dep_3.txt has 21 rows and 3 columns. The first column contains row numbers from 1 to 21. The second and third columns contain numerical values.

1	1.6410	3.7520
2	2.5300	4.6410
3	4.7640	7.1970
4	5.3310	8.2345
5	7.3910	11.3043
6	7.9420	12.1308
7	9.0470	13.1383
8	10.9240	14.6405
9	13.2650	16.8158
10	13.5080	17.0587
11	16.2960	17.8030
12	18.6230	20.1620
13	20.0840	21.8550
14	21.3910	22.8180
15	22.7320	24.8990
16	23.6510	25.3220
17	25.1760	26.3790
18	26.7090	27.5630
19	29.1620	30.3000
20	31.5460	33.0460
21		

The network departure time is as following:



The screenshot shows a code editor with a file explorer on the left and a table of data in the net_dep_3.txt file on the right. The file explorer shows the same project structure as the first screenshot, but with net_dep_3.txt selected. The table in net_dep_3.txt has 15 rows and 3 columns. The first column contains row numbers from 1 to 15. The second and third columns contain numerical values.

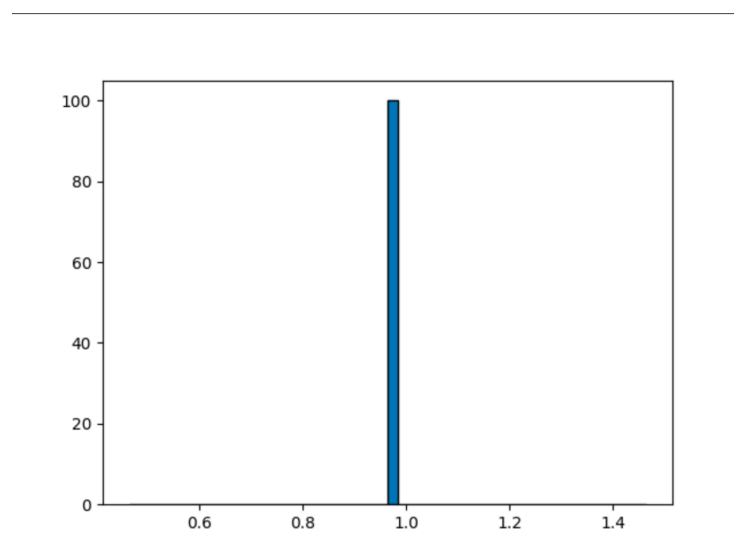
1	1.6410	4.3740
2	2.5300	6.4880
3	4.7640	8.0570
4	5.3310	8.6045
5	7.3910	13.1143
6	7.9420	14.0898
7	9.0470	14.0163
8	10.9240	14.8625
9	13.2650	17.3317
10	13.5080	17.8758
11	18.6230	20.6860
12	20.0840	23.0610
13	22.7320	25.3420
14	31.5460	34.0620
15		

The cloud departure time is as following:

Project		cloud_dep_3.txt	
▼ project [untitled] ~/Desktop/project		1	1.6410 5.7516
▶ project_sample_files		2	2.5300 6.8024
chart.py		3	4.7640 11.0346
cloud_dep_1.txt		4	5.3310 10.4669
cloud_dep_2.txt		5	7.3910 13.3175
cloud_dep_3.txt		6	7.9420 22.3195
cloud_dep_4.txt		7	9.0470 21.4626
compare.py		8	10.9240 18.0843
fog_dep_1.txt		9	13.2650 22.2626
fog_dep_2.txt		10	13.5080 22.2514
fog_dep_3.txt		11	18.6230 20.7100
fog_dep_4.txt		12	20.0840 23.7450
mrt_1.txt		13	22.7320 27.5196
		14	31.5460 34.6044
		15	

3. Reproducible

In my simulation program, given a fixed seed, my program will give identical outputs. In order to prove that, I would like to run a simulation 100 times with the same seed(1), and plot the mean response time in the figure below.



From the figure, we can see that the mean response times are the same over 100 simulations. Thus, given the same seed, my simulation program will always provide the same output.

4. Find suitable value for fogTimeLimit

(1) Find mean response time and suitable value of fogTimeLimit

The parameters are below:

$\alpha 1$	$\alpha 2$	β	λ	$v1$	$v2$	fogTimeToCloudTime	seed	time_end
0.01	0.4	0.86	9.72	1.2	1.47	0.6	1	10000

fogTimeLimit	Mean Response Time
0	1.5827
0.10	0.8283
0.11	0.8284
0.12	0.8398
0.13	0.8672
0.14	0.9121
0.15	0.9864
0.16	1.0975
0.17	1.264
0.18	1.5483
0.19	2.1048

0.20	3.2986
------	--------

From the above table, we can see that when fogTimeLimit is 0.1 or 0.11, the mean response time is better. In order to determine which one is better, I have done the following.

$\alpha 1$	$\alpha 2$	β	λ	$v1$	$v2$	fogTimeToCloudTlme	seed	time_end
0.01	0.4	0.86	9.72	1.2	1.47	0.6	1	20000

fogTimeLimit	Mean Response Time
0.10	0.8227
0.11	0.82

$\alpha 1$	$\alpha 2$	β	λ	$v1$	$v2$	fogTimeToCloudTlme	seed	time_end
0.01	0.4	0.86	9.72	1.2	1.47	0.6	1	30000

fogTimeLimit	Mean Response Time
0.10	0.8238
0.11	0.822

$\alpha 1$	$\alpha 2$	β	λ	$v1$	$v2$	fogTimeToCloudTlme	seed	time_end
0.01	0.4	0.86	9.72	1.2	1.47	0.6	1	40000

fogTimeLimit	Mean Response Time
0.10	0.8238
0.11	0.8219

From above we can see that the value of fogTimeLimit that gives the best system response time is 0.11.

(2) Transient Removals

In this part, I use seed(1), seed(10), seed(20) to generate the figure of mean response time.

seed(1):

$\alpha 1$	$\alpha 2$	β	λ	$v1$	$v2$
0.01	0.4	0.86	9.72	1.2	1.47
fogTimeToCloudTlme	seed	time_end	fogTimeLimit		
0.6	1	1000	0.11		

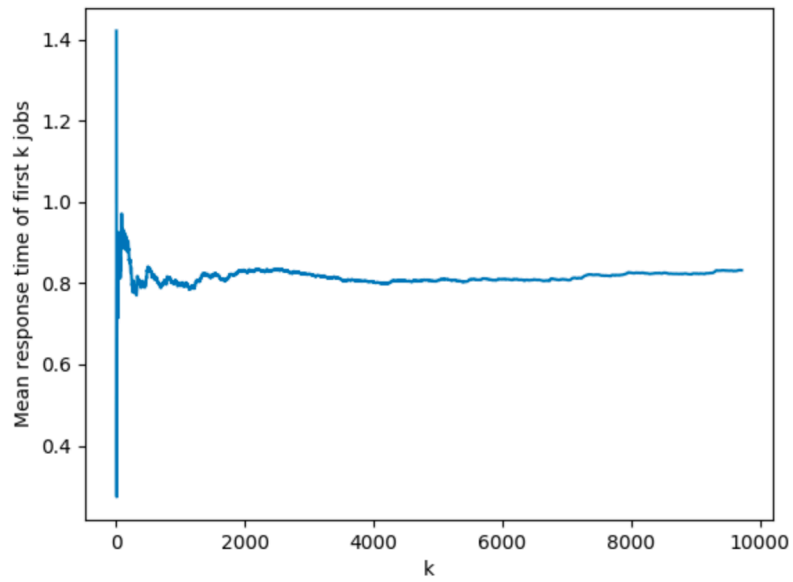


Fig 7. Mrt of k jobs when fogTimeLimit=0.11 (seed=1)

seed(10):

$\alpha 1$	$\alpha 2$	β	λ	$v1$	$v2$
0.01	0.4	0.86	9.72	1.2	1.47
fogTimeToCloudTlme	seed	time_end	fogTimeLimit		
0.6	10	1000	0.11		

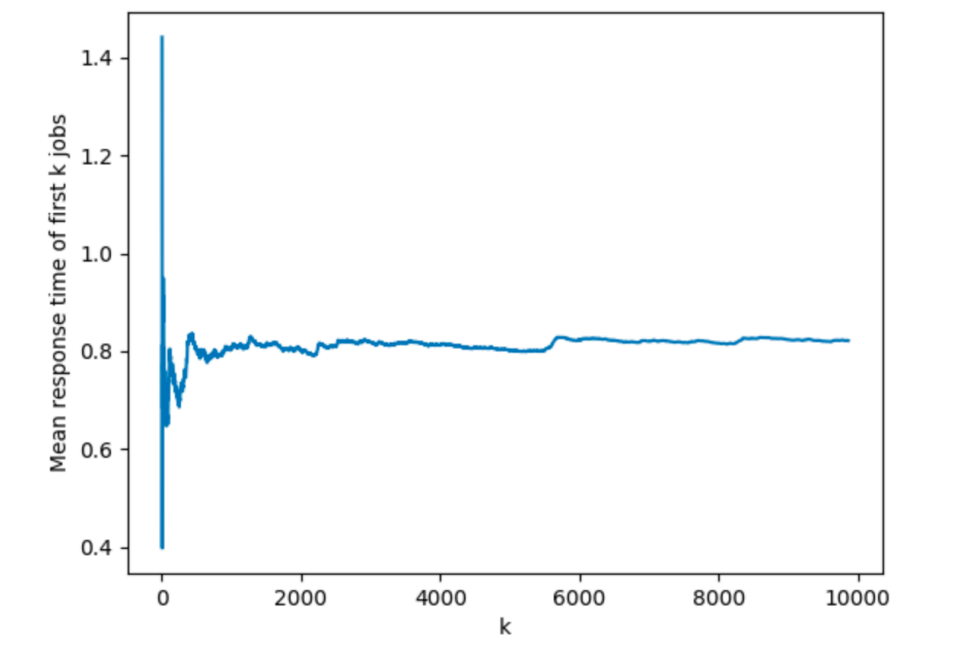


Fig 8. Mrt of k jobs when fogTimeLimit=0.11 (seed=10)

seed(20)

$\alpha 1$	$\alpha 2$	β	λ	$v1$	$v2$
0.01	0.4	0.86	9.72	1.2	1.47
fogTimeToCloudTlme	seed	time_end	fogTimeLimit		
0.6	20	1000	0.11		

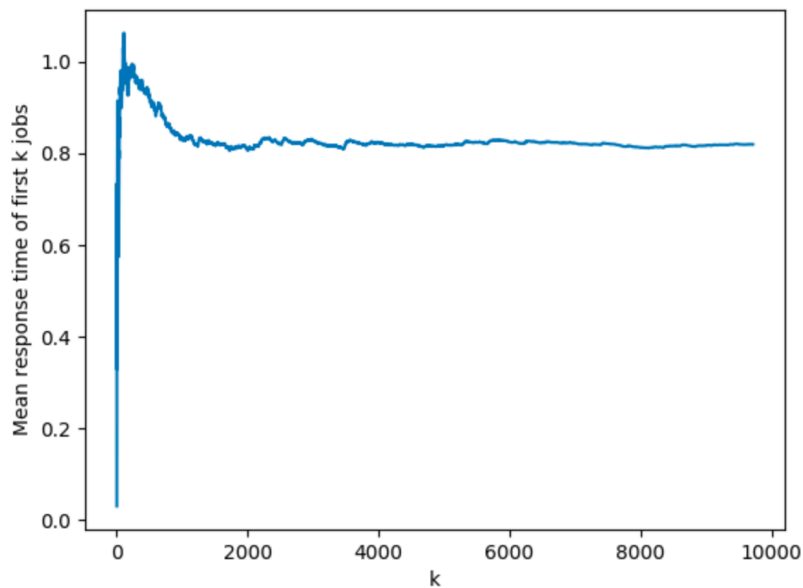


Fig 9. Mrt of k jobs when fogTimeLimit=0.11 (seed=20)

The early part of the simulation displays transient. The later part of the simulation converges or fluctuates around the steady state value. Since we are interested in the steady state value, we should not use the transient part of the data to compute the steady state value. We should remove the transient part and only use the steady state part to compute the mean. One method to identify the transient part is to use visual inspection

It can be seen from the line chart that first 2000 jobs can be removed.

And the figure after transient removals are below:

```
seed(1):
```

```
fogTimeLimit = 0.11
```

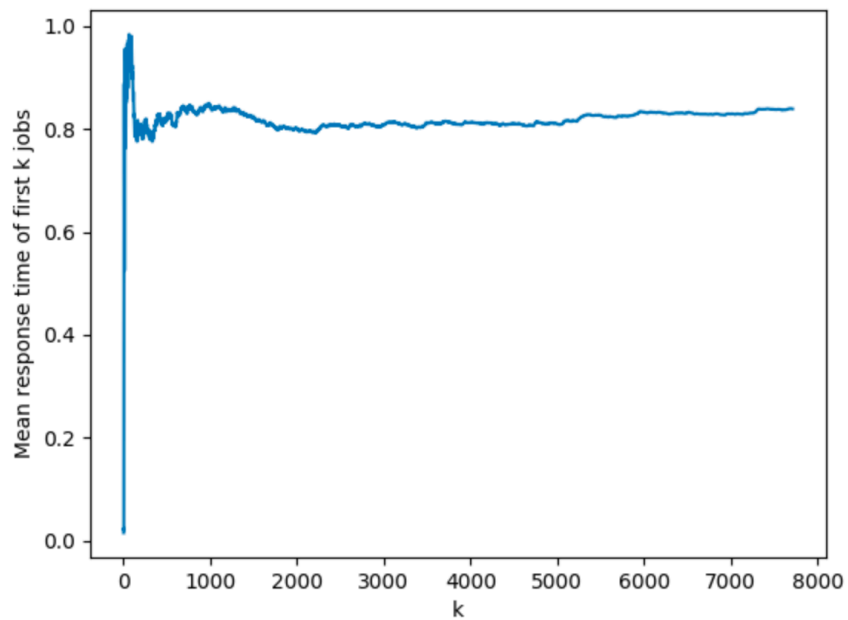


Fig 10. Transient removal when fogTimeLimit=0.11 (seed=1)

`seed(10):`

`fogTimeLimit = 0.11`

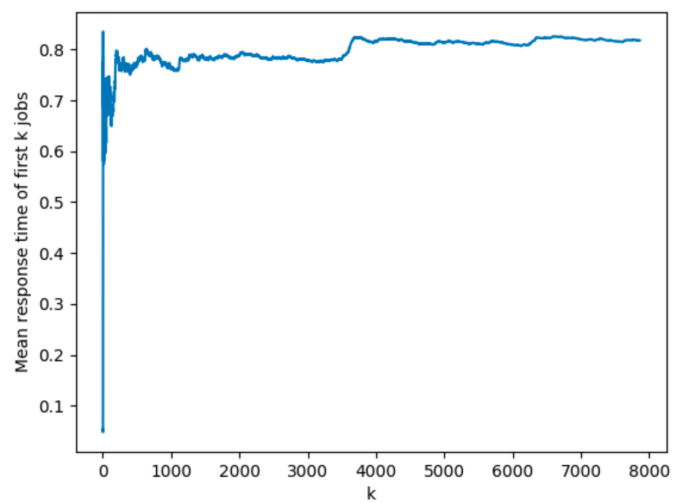


Fig 11. Transient removal when fogTimeLimit=0.11 (seed=10)

seed(20):

fogTimeLimit = 0.11

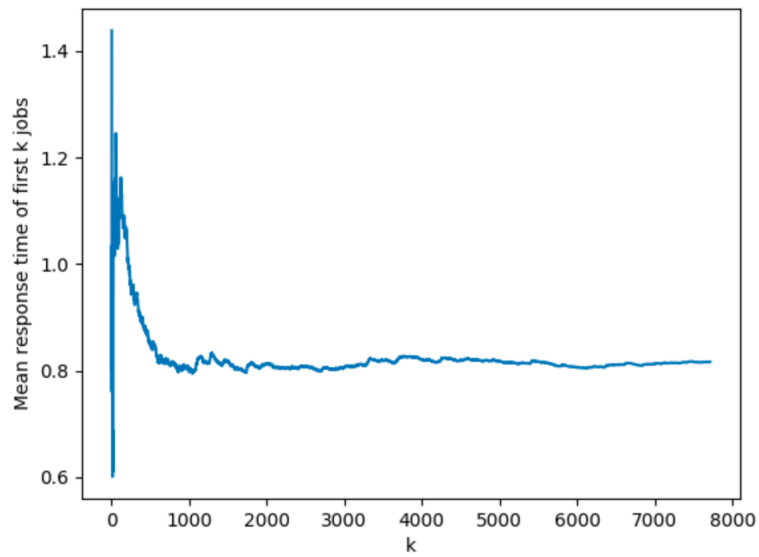


Fig 12. Transient removal when fogTimeLimit=0.11 (seed=20)

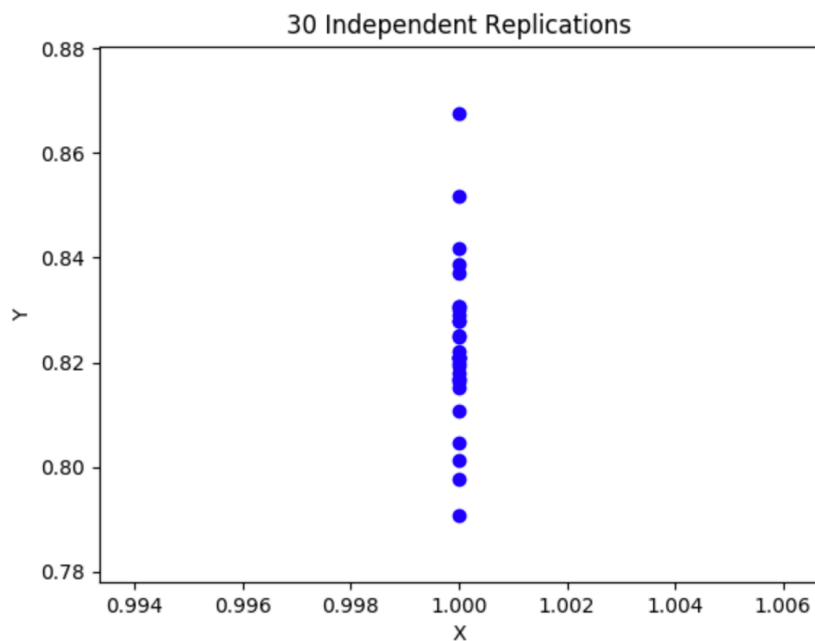
(3) 30 independent replications

$\alpha 1$	$\alpha 2$	β	λ	$v1$	$v2$
0.01	0.4	0.86	9.72	1.2	1.47
fogTimeToCloudTlme	time_end	fogTimeLimit	jobs remove		
0.6	1000	0.11	2000		

seed	Mean response time
1	0.8387
2	0.82
3	0.8194
4	0.8168
5	0.822
6	0.828
7	0.8045
8	0.8418
9	0.8252
10	0.8179
11	0.8208
12	0.828
13	0.8371
14	0.8013
15	0.8152
16	0.8301
17	0.7976
18	0.8307
19	0.8169
20	0.8164
21	0.8517
22	0.7907

23	0.8211
24	0.8306
25	0.8209
26	0.8209
27	0.825
28	0.8677
29	0.8107
30	0.8291

fogTimeLimit = 0.11



The blue circles show the estimated mean response time from the 30 independent experiments.

(4) Computing the confidence interval

In each replication, after removing the transient part and compute an estimate of the mean steady state response time
For fogTimeLimit = 0.11, the mean steady state response time is equal to 0.8232266666666669

Let us call the estimate from the kth replication, $T(k)$

$$\hat{T} = \frac{\sum_{i=1}^n T(i)}{n}$$

the sample standard deviation

$$\hat{S} = \sqrt{\frac{\sum_{i=1}^n (\hat{T} - T(i))^2}{n - 1}}$$

```
import statistics

y = [0.8387, 0.82, 0.8194, 0.8168, 0.822, 0.828, 0.8045, 0.8418, 0.8252, 0.8179, 0.8208, 0.828,
      0.8371, 0.8013, 0.8152, 0.8301, 0.7976, 0.8307, 0.8169, 0.8164, 0.8517, 0.7907, 0.8211,
      0.8306, 0.8209, 0.8209, 0.825, 0.8677, 0.8107, 0.8291]

mean_mrt = statistics.mean(y)
s = statistics.stdev(y)
print(mean_mrt)
print(s)
```

From the code above, we can get that

For fogTimeLimit = 0.11

$T = 0.8232266666666667$ (the sample arithmetic mean)

$S = 0.01529216462279212$ (the sample standard deviation)

There is a probability $(1 - \alpha)$ that the mean response time that you want to estimate lies in the interval

$$\left[\hat{T} - t_{n-1, 1-\frac{\alpha}{2}} \frac{\hat{S}}{\sqrt{n}}, \hat{T} + t_{n-1, 1-\frac{\alpha}{2}} \frac{\hat{S}}{\sqrt{n}} \right]$$

- The sample mean of ($n = 30$) replications for fogTimeLimit = 0.11 is 0.8232
- The sample standard deviation of 30 replications for fogTimeLimit = 0.11 is 0.01529216462279212
- If we want to compute the 95% confidence interval, $\alpha = 0.05$
- Since we did 30 independent replications and want 95% confidence interval, we use $t(29, 0975)$

From the t-distribution table, the value of 2.0452, the 95% confidence interval for fogTimeLimit = 0.11 is

$$[0.8232 - 2.0452 * (0.0153 / \sqrt{30}), 0.8232 + 2.0452 * (0.0153 / \sqrt{30})]$$

Using python to compute this equation, we can get:

fogTimeLimit = 0.11:

```

# fogTimeLimit = 0.11
a1 = 0.8232 - 2.0452*(0.0153/(30**((1/2)))
print(a1)

b1 = 0.8232 + 2.0452*(0.0153/(30**((1/2)))
print(b1)

```

```

0.8174869689094912
0.8289130310905088

```

$\alpha 1$	$\alpha 2$	β	λ	$v1$	$v2$
0.01	0.4	0.86	9.72	1.2	1.47
fogTimeToCloudTlme	time_end	fogTimeLimit	jobs remove		
0.6	1000	0.11	2000		

There is a 95% probability that the true mean response time that we want to estimate is in the interval [0.8175, 0.8289] under a conditions from the above table.