# 1.通过使用pdb调试，经过大量工作后，我得到了这个：

说明不是interval太大导致的new\_data长度过小

大量工作指的是：

load\_public\_data.py:

在调试data = torch.cat(data,dim)

在当前位置打了断点

len(data) = 7

data[0].size = torch.Size([1, 6, 90])

data[1].size = torch.Size([1, 2000, 90])

data[2].size = torch.Size([1, 8, 90])

data[3].size = torch.Size([1, 11, 90])

data[4].size = torch.Size([1, 2000, 90])

data[5].size = torch.Size([1, 11, 90])

data[6].size = torch.Size([1, 4, 90])

进度条问题：

0-1 ok

之后打印出!!!!,

然后进度条会断开，新出一个2/7

2-3 ok

之后打印出!!!!，

然后再换行打印出！！！！

再新开一个进度条4/7，一直到6/7才会有！！！！，而且出两次（第二次是换行打印）

7/7是新开的进度条，没有！！！！

打印！！！！说明数据长度不足2000！

、现在问题转化为为什么五组数据的长度不足2000

对于第一个数据（第二维度为6）

What's len(data[len(time\_stamp)-1]) is: 90

What's len(new\_data) is:6

也就是说，第二维度实际上是new\_data的长度。（后来我弄到第三个也是8行90列，更印证了data[i]的torch.size()的第二维度就是这里的数字）

(Pdb) time\_stamp[0]

22.737

(Pdb) time\_stamp[len(time\_stamp)-1]

6.7094

(Pdb) print(intervel)

-0.0080138

0/7 len(time\_stamp): 7111

temp\_list = 6571行，90列

len(average\_list(temp\_list))

90

new\_data是一个6行90列的列表，其中倒数第二列是average\_list!!!

证明最后一列是new\_data.append(data[len(time\_stamp)-1])加进去的！

1/7 len(time\_stamp): 7111

2/7 len(time\_stamp): 7200

现在写第二个数据，也就是头一个为2000 的数据的特征

len(new\_data) 4633

print(intervel)

-0.0036865000000000005

(Pdb)

猜测可能有点问题

print(len(temp\_list))

1

(Pdb) print(temp\_list)

[[11.937, 14.237, 13.964, 13.845, 14.215, 14.303, 14.833, 14.148, 13.797, 13.964, 13.597, 13.964, 13.024, 12.628, 12.936, 12.432, 13.441, 13.597, 13.388, 12.906, 14.346, 15.525, 16.582, 15.717, 17.313, 18.624, 16.683, 17.976, 15.166, 16.248, 17.056, 17.815, 17.901, 18.535, 16.915, 17.747, 17.844, 16.915, 16.607, 17.324, 17.825, 19.422, 18.325, 18.402, 17.576, 18.452, 19.693, 20.436, 20.323, 20.561, 20.587, 21.57, 22.3, 22.174, 22.735, 24.527, 22.131, 23.238, 20.561, 15.442, 18.078, 19.617, 20.84, 21.16, 20.703, 21.88, 22.044, 21.611, 21.679, 21.726, 21.5, 20.954, 20.236, 19.023, 19.146, 19.251, 20.066, 20.94, 20.42, 21.138, 21.773, 22.501, 24.082, 22.906, 23.238, 25.26, 22.891, 24.391, 21.667, 10.482]]

(Pdb) print(len(temp\_list[0]))

90

(Pdb) print(time\_stamp)

[17.96,......, 10.587]

(Pdb)

intervel = (time\_stamp[len(time\_stamp)-1] - time\_stamp[0]) / 2000

第三个数据：

len(new\_data)

8

print(intervel)

-0.008505549999999999

print(len(temp\_list))

6390

print(len(temp\_list[0]))

90

print(len(time\_stamp))

7200

print(time\_stamp[0])

21.647

第四个数据：

time\_stamp[0]

20.843

time\_stamp[len(time\_stamp)-1]

13.421

intervel

-0.0037110000000000003

print(len(temp\_list))

5470

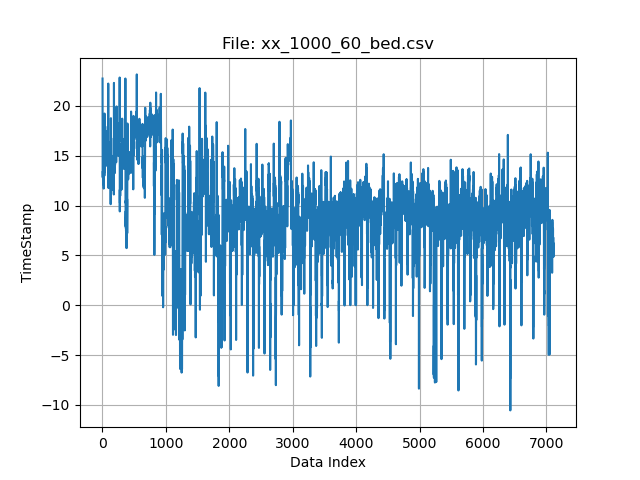
print(len(temp\_list[0]))

90

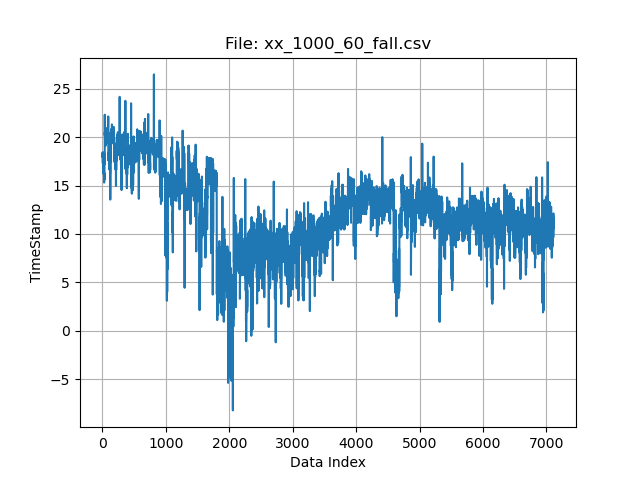
应该全走if分支--》if分支应该是len(temp\_list) != 0触发

# 2.通过对时间戳的绘图，结果是：

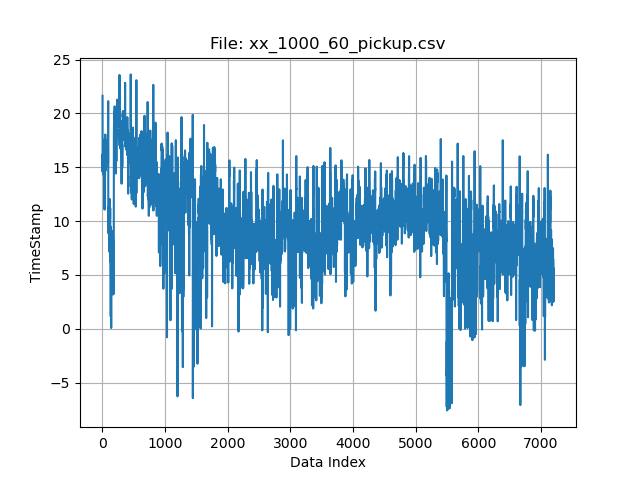
图一：



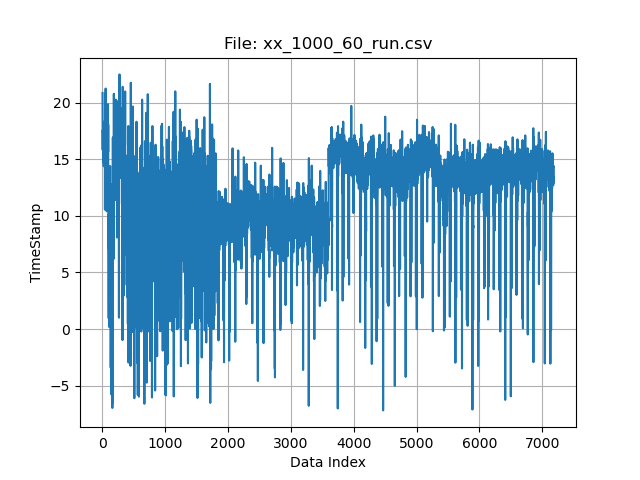
图二：



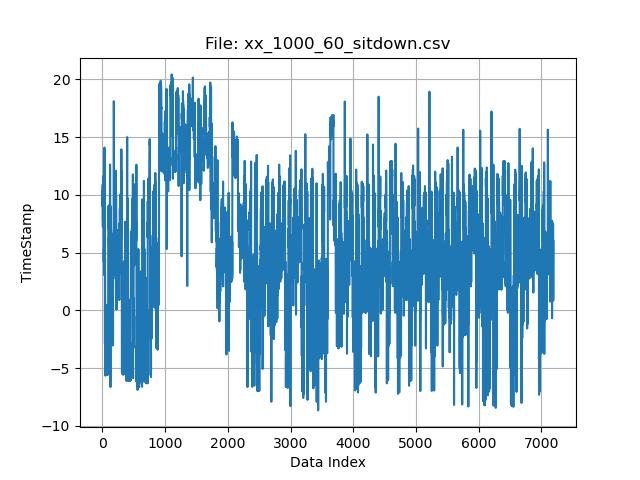
图三：



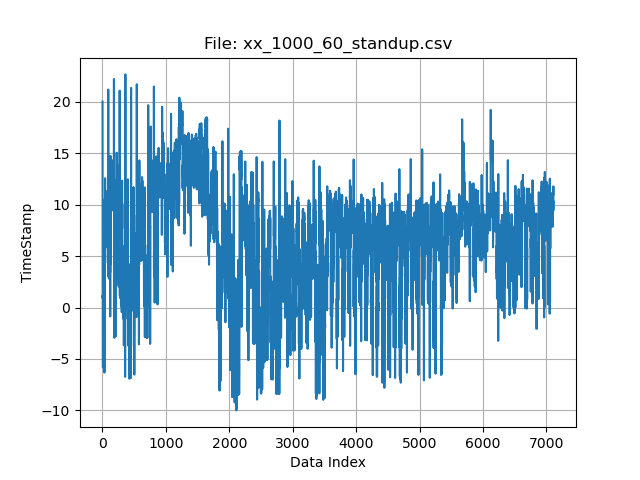
图四：



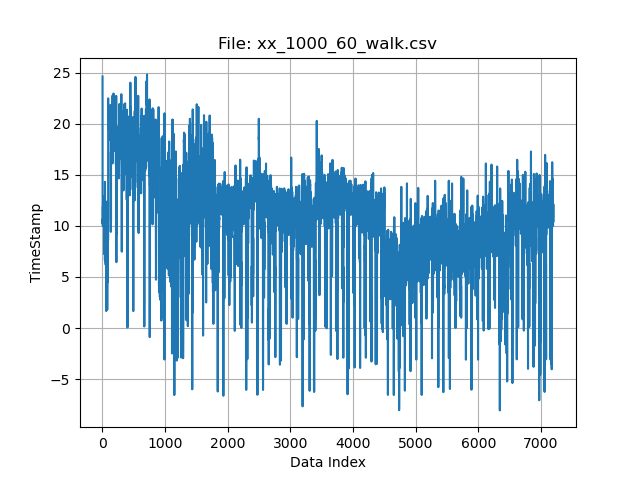
图五：



图六：



图七：

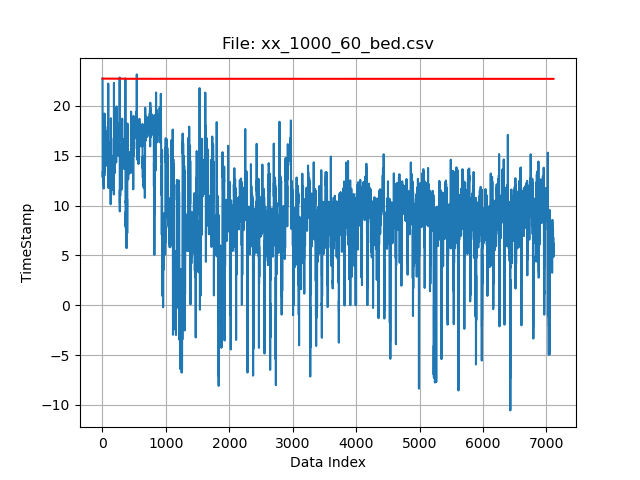


# 接下来我会添加一条线，这条线代表了在每个数据上的cur\_range所在的红线，

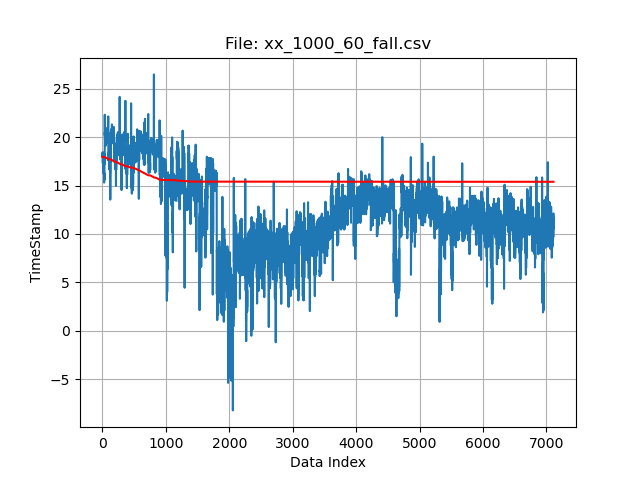
如果数据不过cur\_range的话，连if time\_stamp[i] > cur\_range:分支都不会进，cur\_range = cur\_range + intervel

这句话也不会导致range更新，从而导致new\_data的append操作减少，从而导致最终得到的数据量不足2000

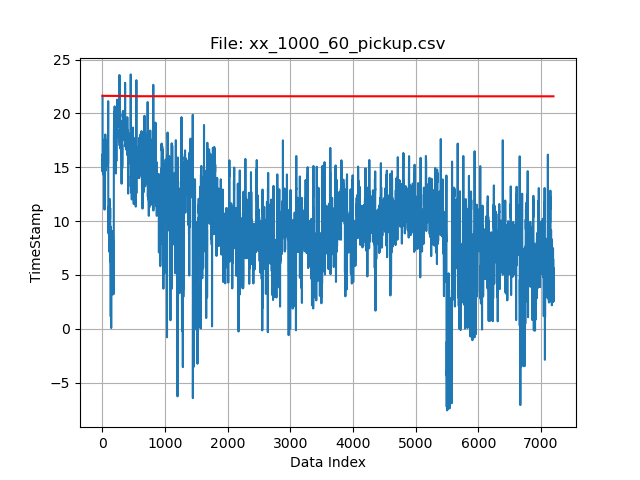
第一个图的ALL:



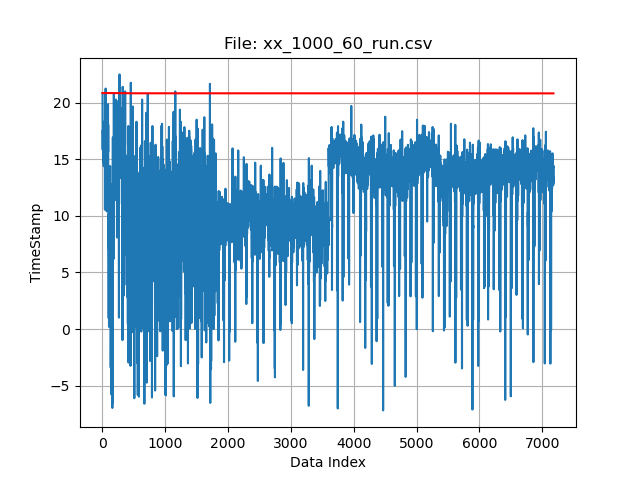
图二的：



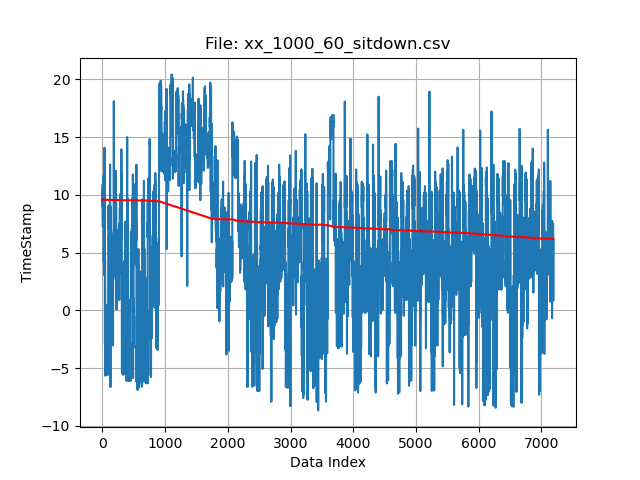
图三：



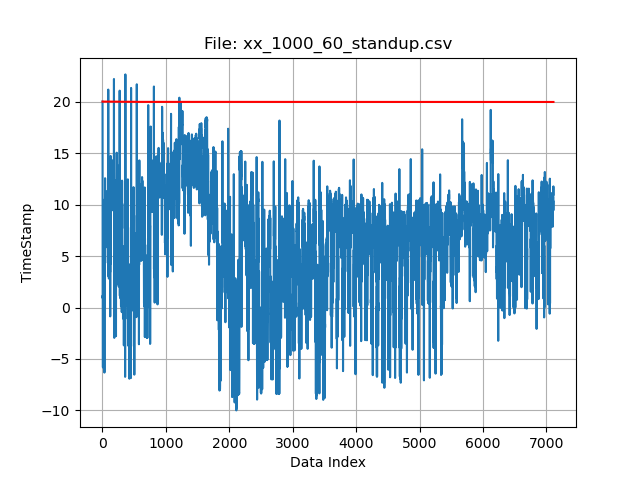
图四



图五：



图六：



图七：

