## homework1

### homwork1 experiment #1

code : homework1\_ex1.md

spark data output : /user/huangjingying/homework1/ex1

explanation for code :

1. average traffic consumption of each user and each location:

map and reduceByKey (key is user/location, do (x+y)/2 for traffic consumption)

1. user distribution in terms of locations :

map and countByKey(key is location, count number of users)

1. traffic consumption distribution in terms of location :

map and reduceByKey (key is location, add traffic consumption)

1. traffic consumption distribution in terms of users :

map and reduceByKey (key is user , add traffic consumption)

### homwork1 experiment #2

spark code : hw1\_ex2\_sum\_traffic\_volumns\_per\_hour\_of\_one\_bs-\*.md

python code : hw1\_ex2\_plot\_traffic\_volums\_eachhour.py

spark data output : /user/huangjingying/homework1/ex2 AND homwork1/ex2

python output : homework/ex2

1. select the top 3 BSs(base station) with the largest traffic:

sortBy for the result of traffic consumption distribution in terms of location and get top 3 locations with the largest traffic : (u'00309', 625502403937), (u'01077', 628462271341), (u'00332', 645735112712)

*this code is in homework1\_ex1 annotated by “for homework2 0 select the top 3 BSs(base station) with the largest traffic”*

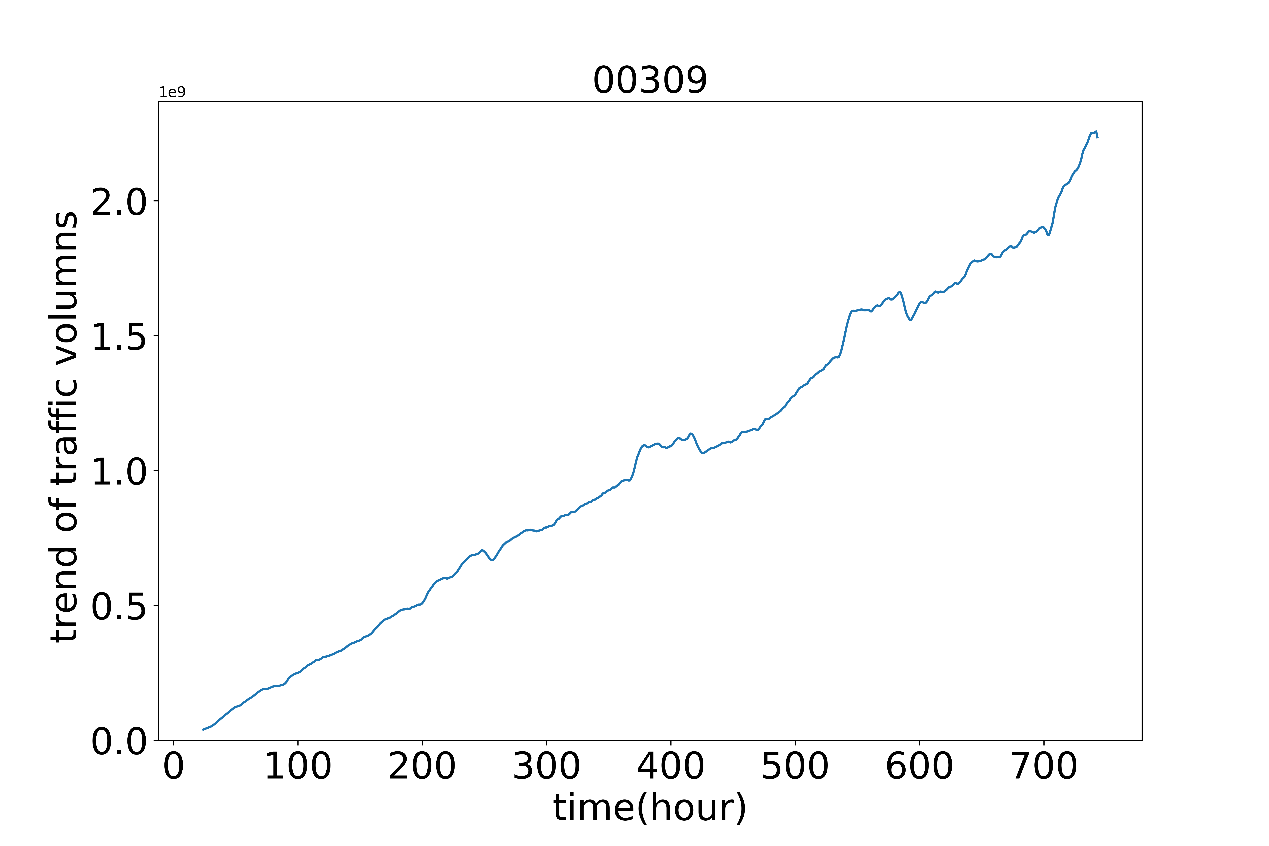
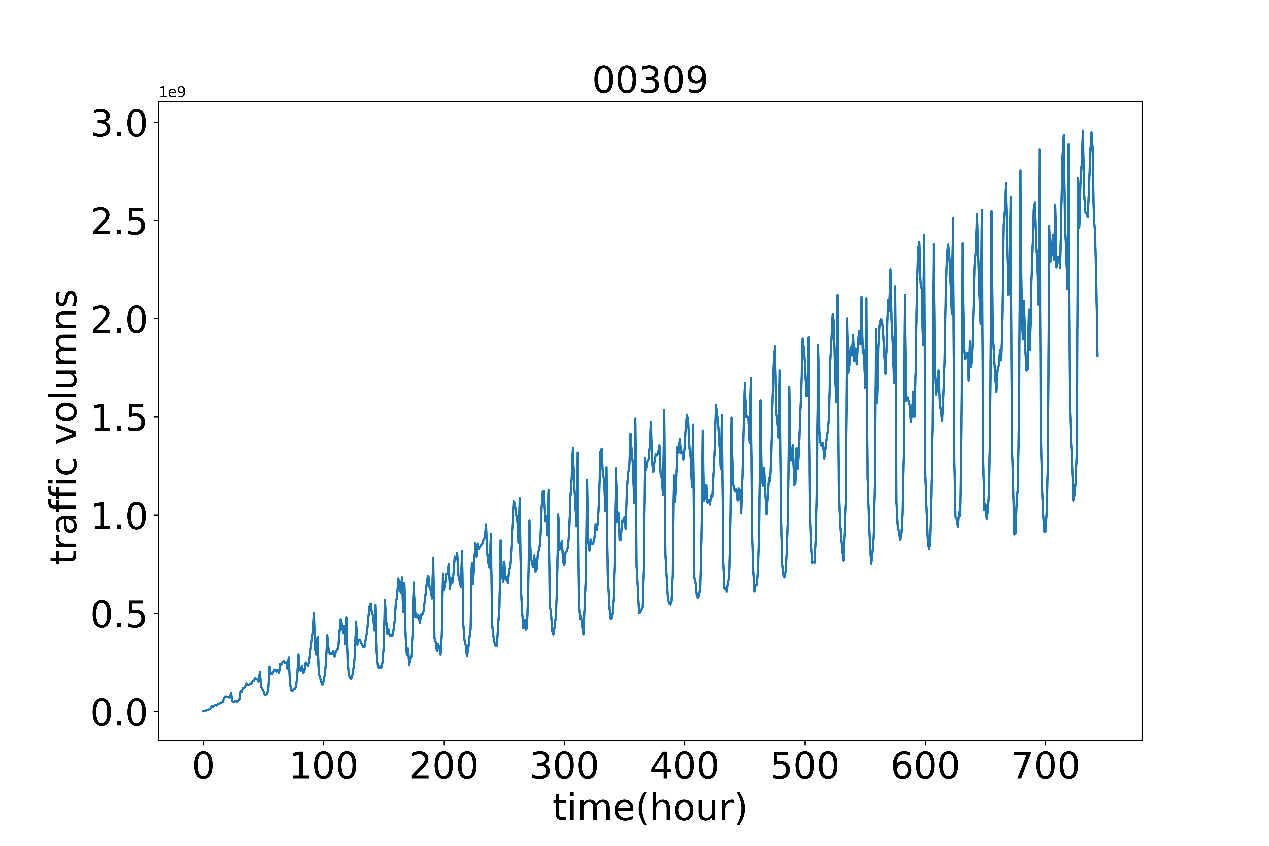
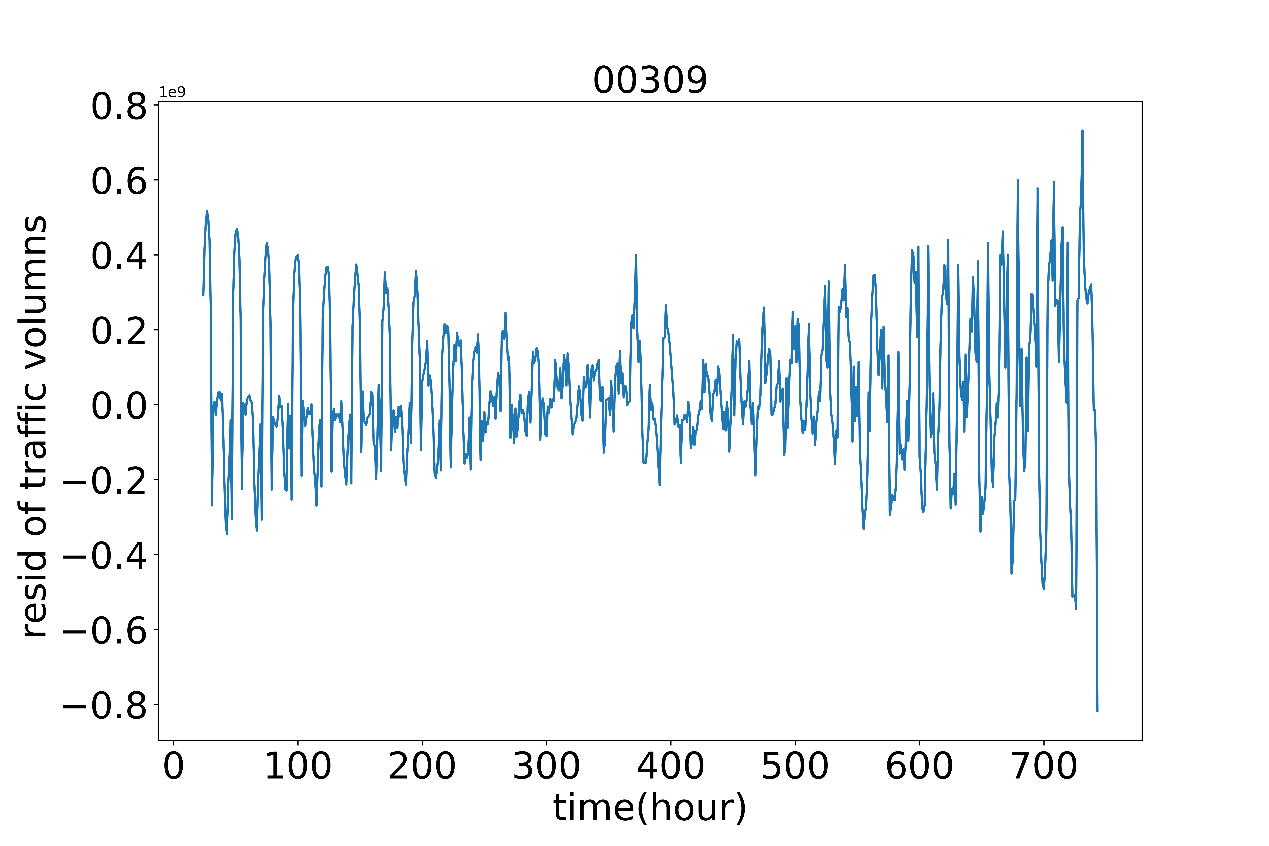
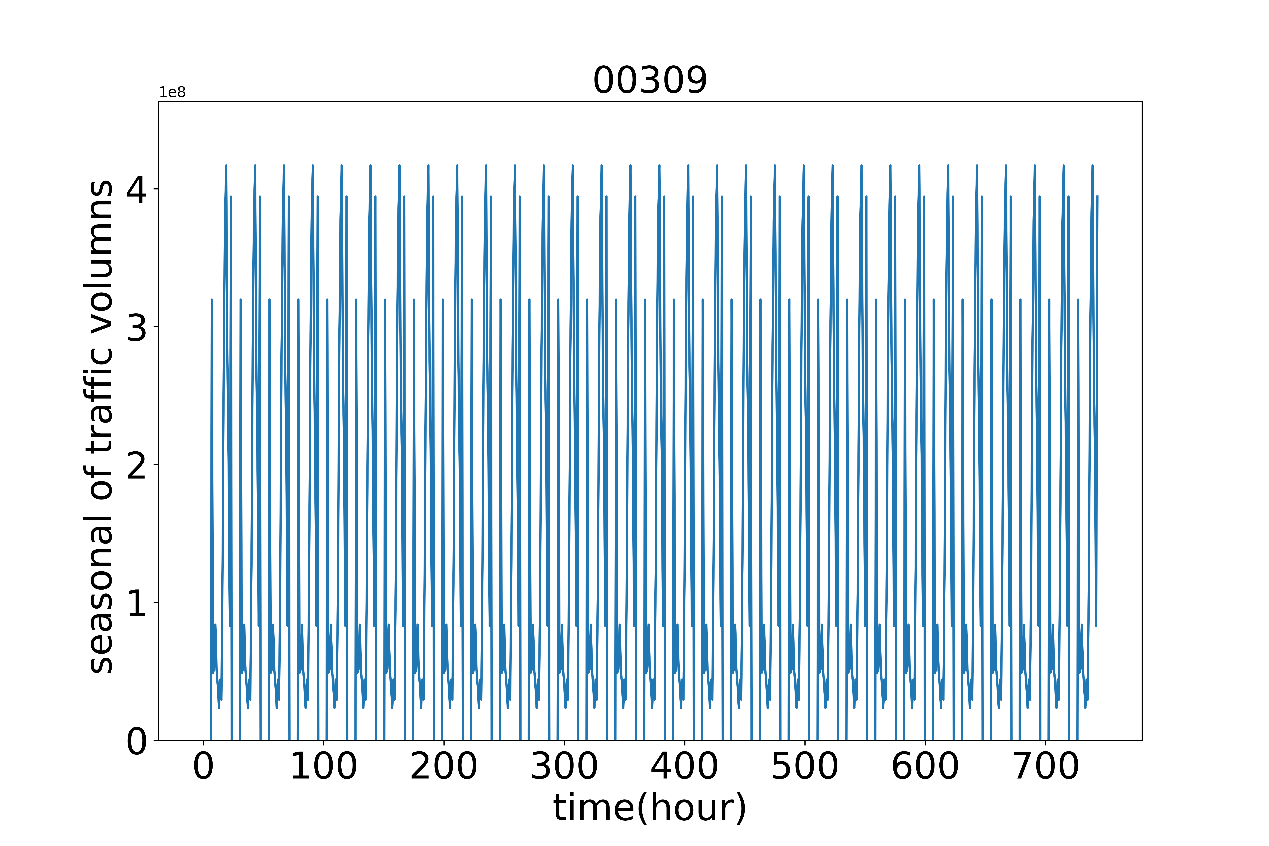
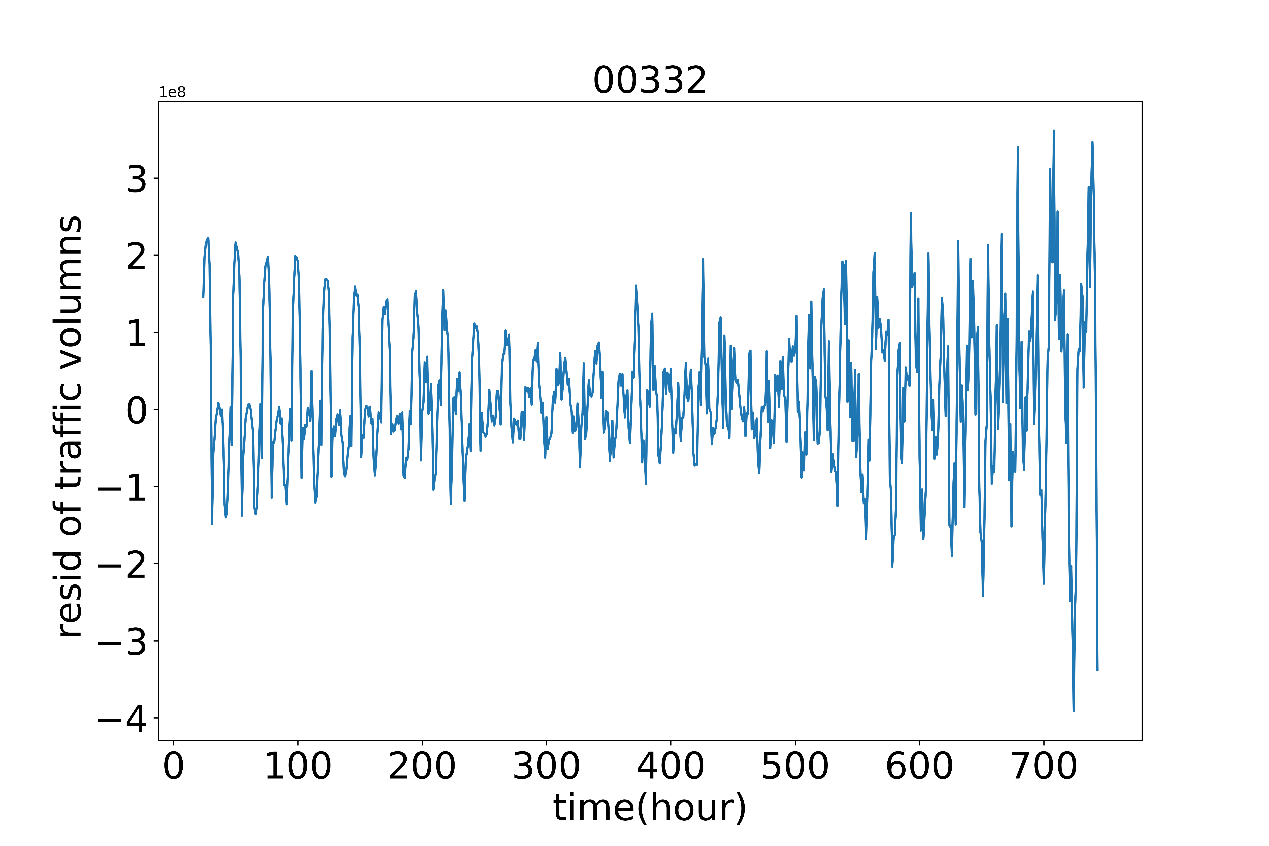
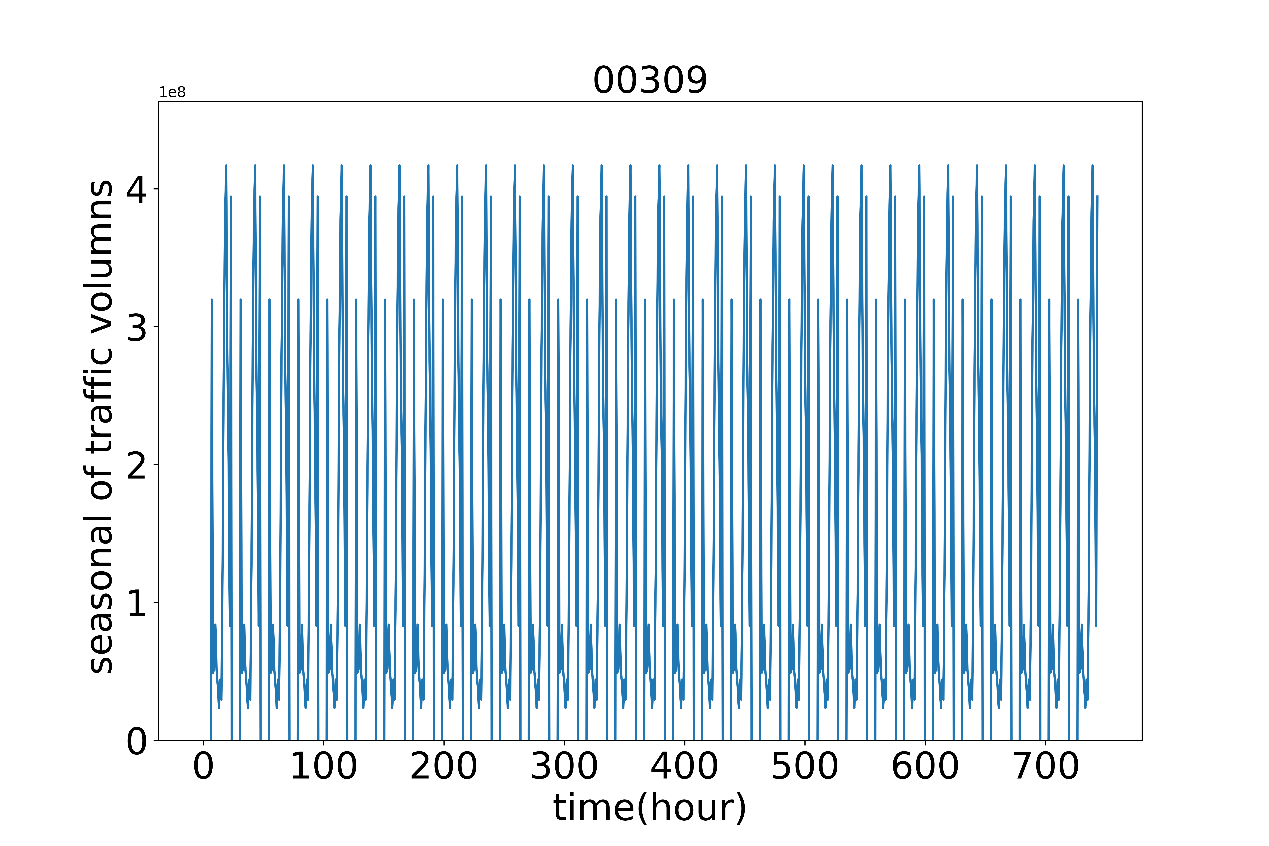
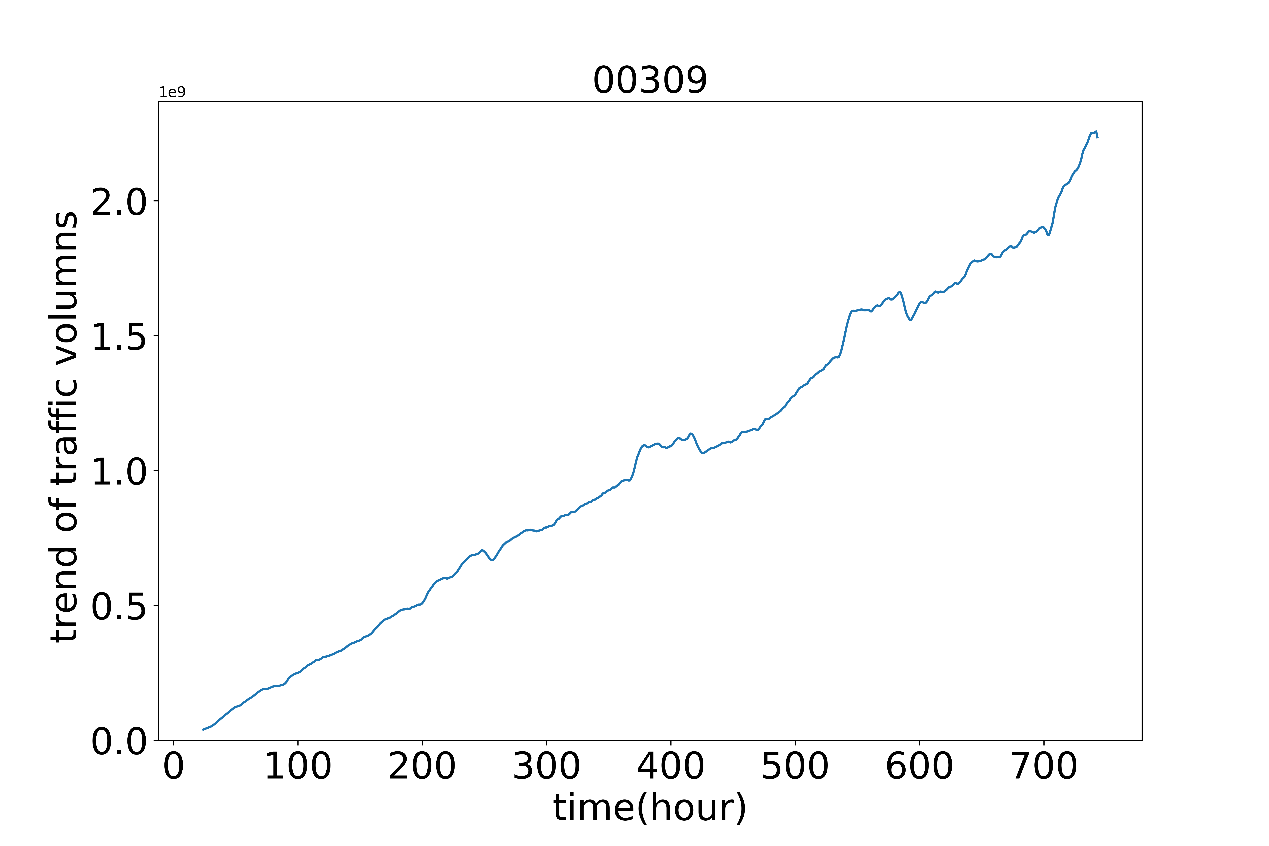
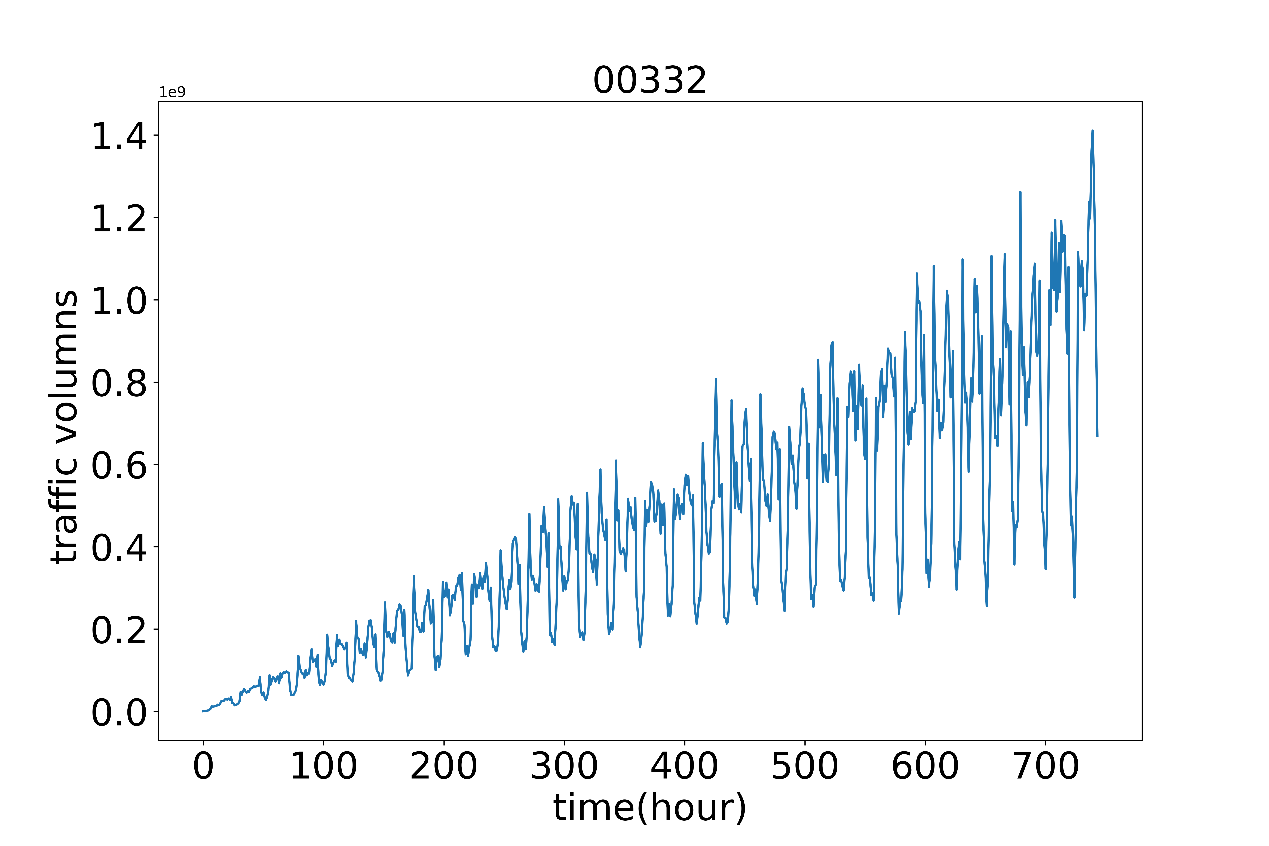
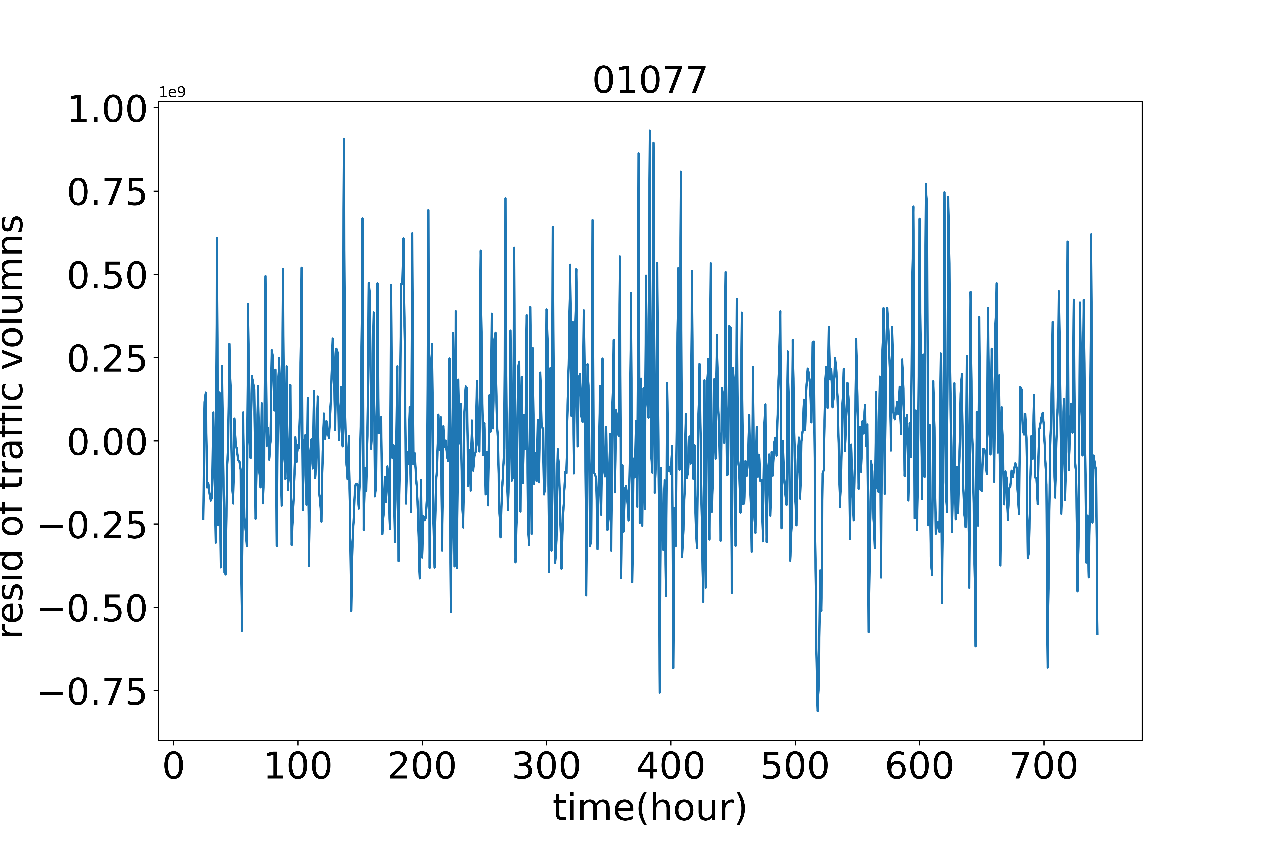
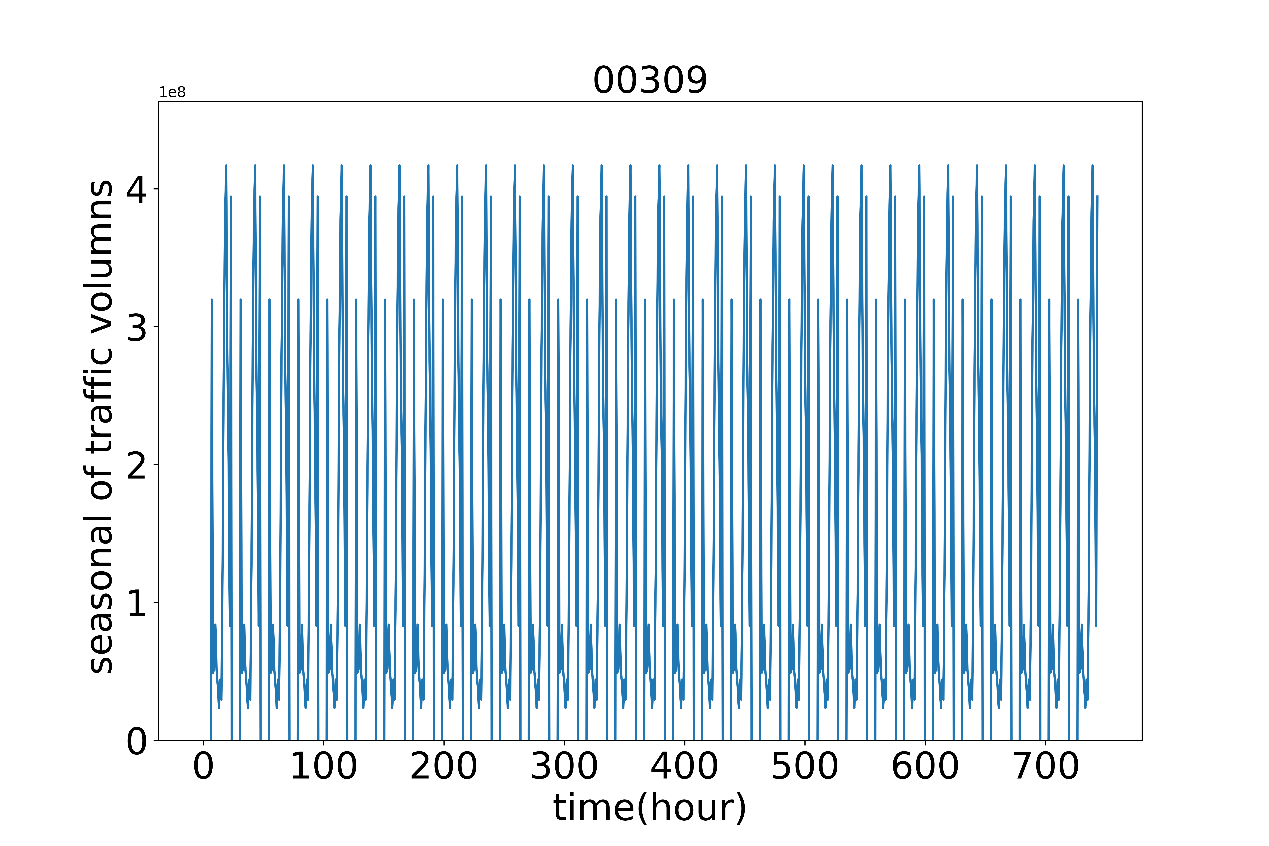
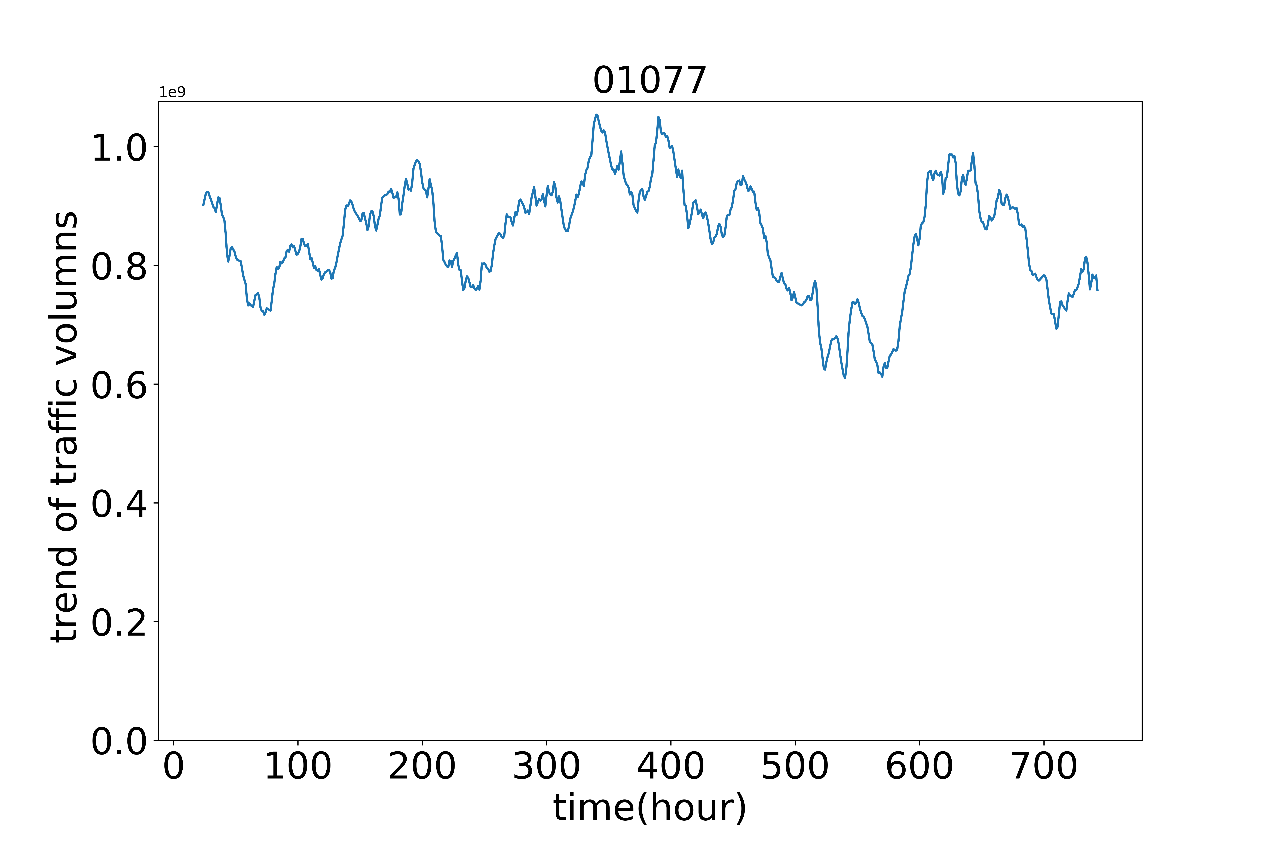
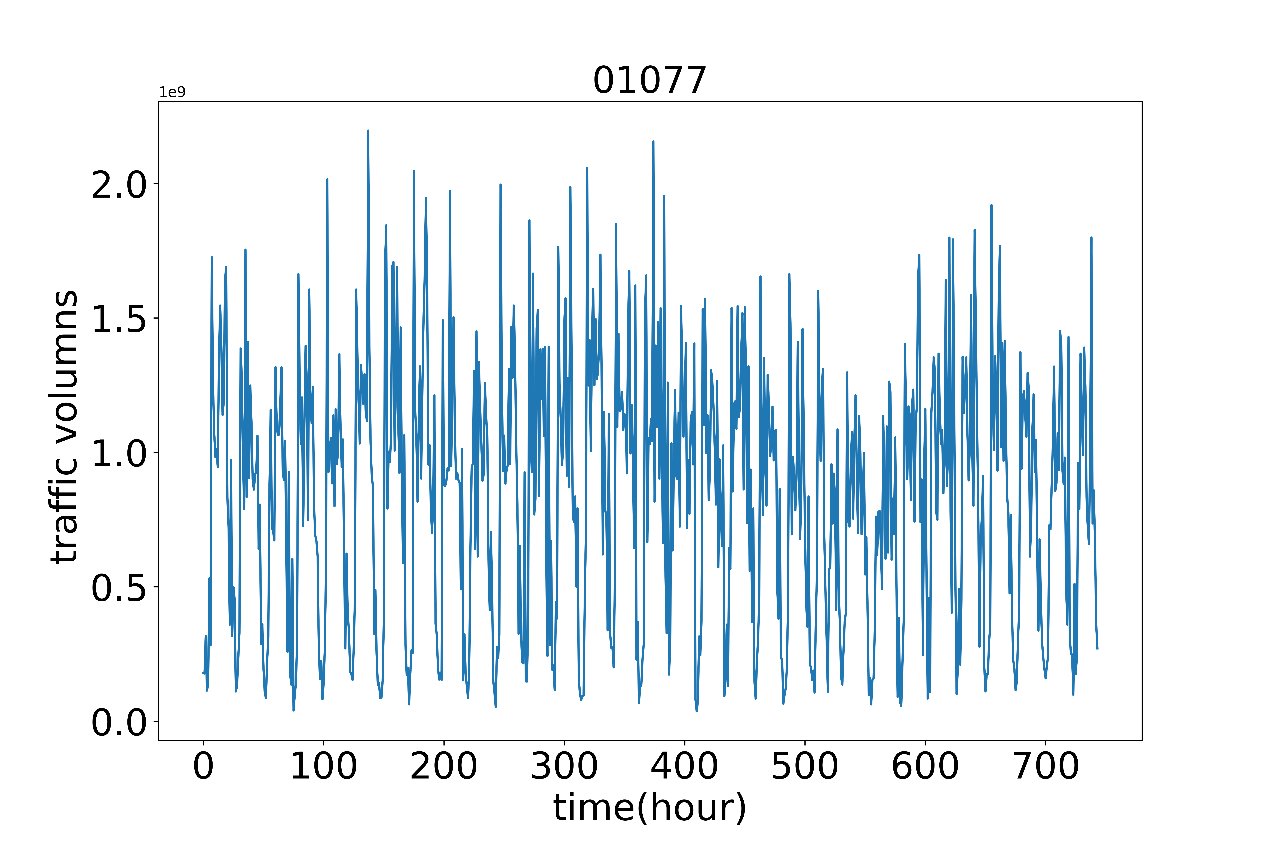
1. get the traffic volume of the BSs in time\_bins(1hour) :

filter by location , map and reduceByKey (key is start time/3600)

1. trend component , periodical component ,residual componnent :

do by seasonal\_decompose

1. plot : title is location ID

### homwork1 experiment #3

# construct concact graph

spark code : hw1\_ex3\_counts\_edges.md

spark data output : /user/huangjingying/homework1/ex3/graph\_edge\_weight\_file\_correct/correct\_file/hw1\_ex3\_countsof\_edges\_per\_hour\_allBSs\_corrected.txt

1 map and groupByKey(key is BS and start time/3600 ,group users) ,

2 map again and duplicate users and sort users and turn each two users to tuple for each key,

3 flatmap for each tuple,

4 countByValue() ,get weight

result table :

|  |  |  |
| --- | --- | --- |
| 'use\_ID' | 'use\_ID' | 'Weight' |
| 000001 | 000002 | 21 |
| 000001 | 000003 | 5 |
| … | … | … |

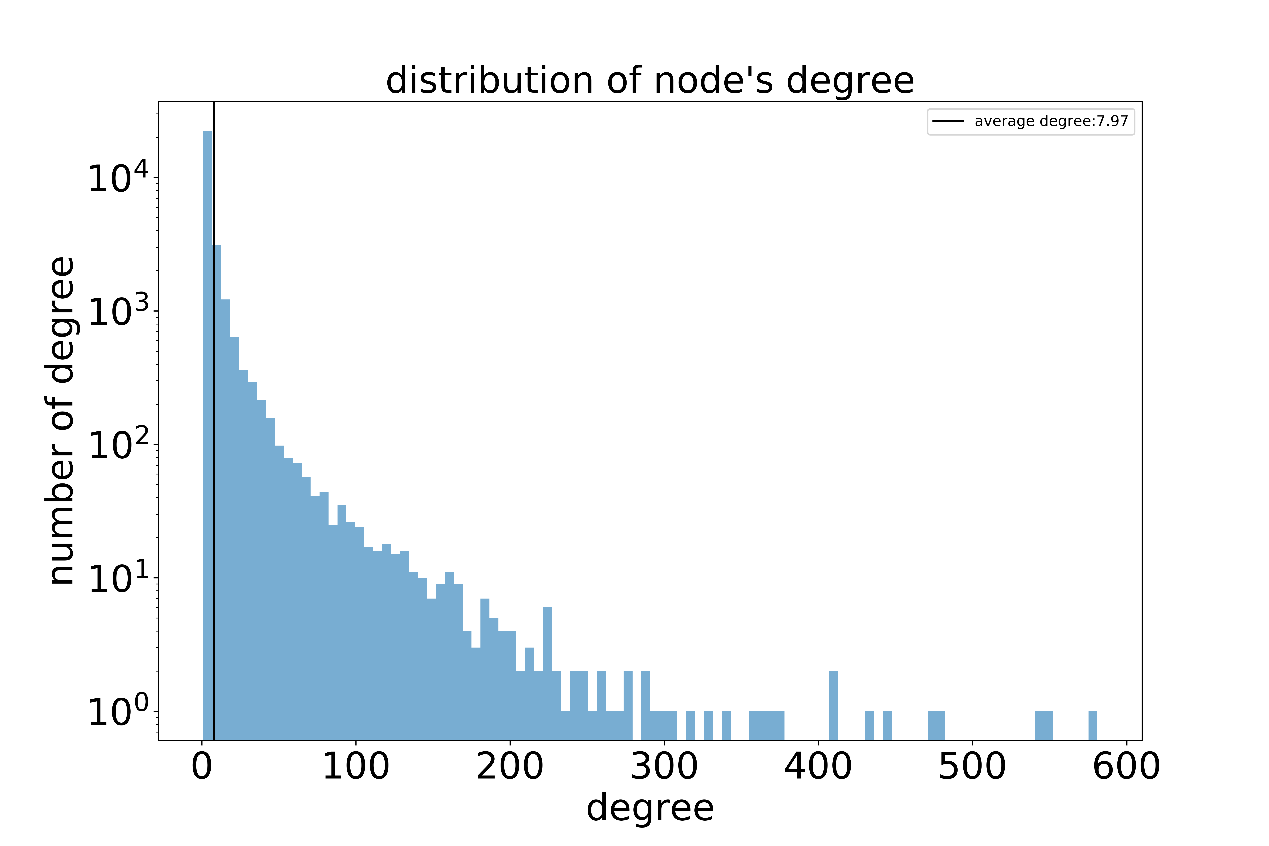
AND

# graph analysis

python code : hw1\_ex3\_get\_graph\_matrix.py

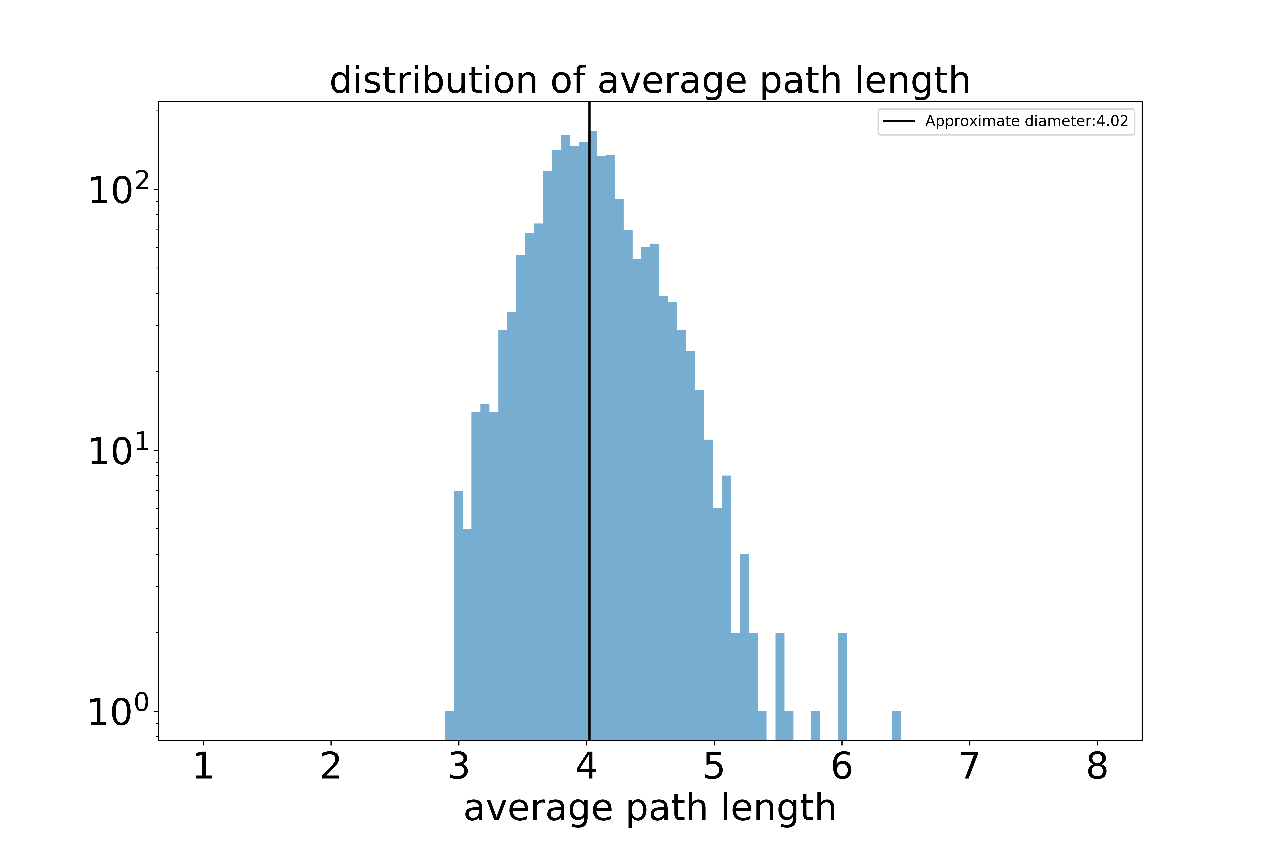
python data output : homwork1/ex3

1. get number of nodes : concat two columns of use\_ID and drop duplicate and count row number , **nodes number is #116575**
2. get number of edges/links : count row number of result table
3. average\_degree : get a pivot\_table , index is one column of use\_ID, values(degree) is the other columns of use\_ID , **average\_degree is : #7.97**

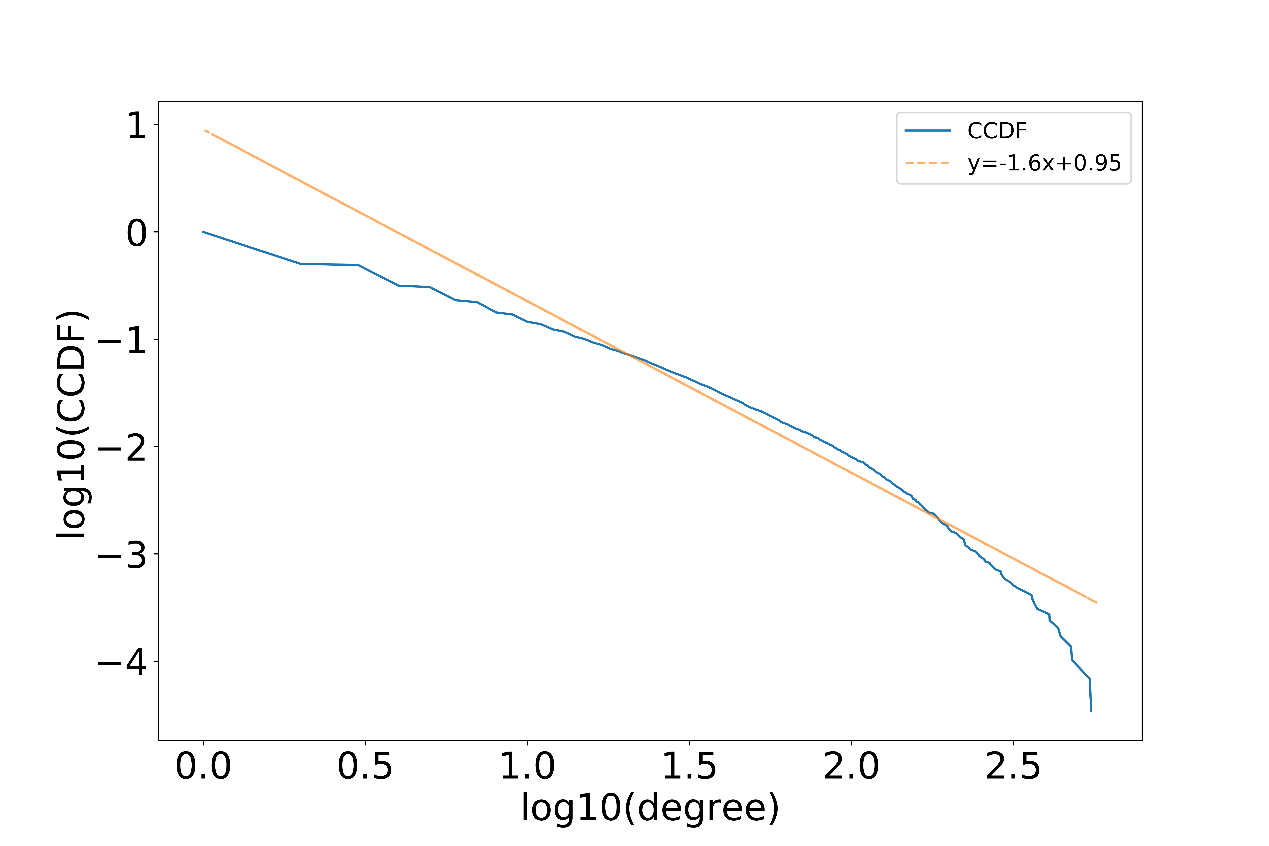


1. graph diameter & average path length (randomly select 2000 user):

for each node, merge(join) result table on user\_id , and remove nodes that already turn up, until all node include or no new node turn up. **graph diameter is #4.02**



1. plot the complementary cumulative distribution function of degree and fit



1. caculate the clustering coefficient of the 5 top users with the largest node degree

目标节点的邻居节点的个数N;

N个邻居节点间存在边的个数为M;

聚类系数=M/(N(N-1) \*2

|  |  |  |
| --- | --- | --- |
| user\_id | user\_degree | clustering\_coefficient |
| 000009 | 581.0 | 0.00851089085405662 |
| 131916 | 549.0 | 0.008156834589765067 |
| 122166 | 544.0 | 0.010392969342433106 |
| 049993 | 479.0 | 0.009425144783850595 |
| 000002 | 476.0 | 0.007704555506413091 |