Homework 0

group1_0416246_王彥茹_report

1. Extract data

First, import taipower data

```
create table power(
create_date timestamp,
create_time TIME,
south_supply float,
south_usage float,
center_supply float,
center_usage float,
north_supply float,
north_usage float,
east_supply float,
east_usage float,
primary key(create_date, create_time)
);
```

for .csv files

```
load data local infile 'C:\\Users\\User\\Desktop\\taipower\\2017-09-23.csv'
into table power
fields terminated by ',' enclosed by '' escaped by ''
lines terminated by '\n'
(@col1, @col2, @col3, @col4, @col5, @col6, @col7, @col8, @col9)
set create_date='2017-09-23', create_time=@col1, north_supply=@col2, north_usage=@col3,
center_supply=@col4, center_usage=@col5, south_supply=@col6, south_usage=@col7,
east_supply=@col8, east_usage=@col9;
```

for .json file >> convert it to .csv with online convertor before importing

```
load data local infile 'C:\\Users\\User\\Downloads\\power.csv'
into table power
fields terminated by ',' enclosed by '' escaped by ''
lines terminated by '\r\n'
(@col1, @col2, @col3, @col4, @col5, @col6, @col7, @col8, @col9, @col10, @col11, @col12, @col13, @col14, @col15)
set create_date=@col3, center_supply=@col4, center_usage=@col5, east_supply=@col6, east_usage=@col7,
north_supply=@col8, north_usage=@col9, south_supply=@col10, south_usage=@col11,
create_time=@col12;
```

Second, import weather data

```
create table weather(
locationName varchar(20),
obsTime timestamp,
elementName varchar(20),
value  varchar(30));
```

```
LOAD XML LOCAL INFILE 'C:\\Users\\User\\Desktop\\C-B0024-002.xml'
INTO TABLE weather
ROWS IDENTIFIED BY '<weatherElement>';
```

locationName	obsTime	elementName	value
BANQIAO.板橋	2016-07-04 13:00:00	日照時數	0.2
BANQIAO.板橋	2016-07-04 14:00:00	測站氣壓	1006.8
BANQIAO.板橋	2016-07-04 14:00:00	溫度	33.7
BANQIAO.板橋	2016-07-04 14:00:00	相對濕度	55
BANQIAO.板橋	2016-07-04 14:00:00	風速	2.0
BANQIAO.板橋	2016-07-04 14:00:00	風向	北北東.NNE
BANQIAO.板橋	2016-07-04 14:00:00	降水量	0.0
BANQIAO.板橋	2016-07-04 14:00:00	日照時數	0.3
BANQIAO.板橋	2016-07-04 15:00:00	測站氣壓	1006.4

Delete the datas which is not about temperature

```
delete from weather
where elementName != '溫度';
```

locationName	obsTime	value
BANQIAO.板橋	2016-07-03 01:00:00	25.3
BANQIAO.板橋	2016-07-03 02:00:00	25.1
BANQIAO.板橋	2016-07-03 03:00:00	25.1
BANQIAO.板橋	2016-07-03 04:00:00	25
BANQIAO.板橋	2016-07-03 05:00:00	24.8
BANQIAO.板橋	2016-07-03 06:00:00	25
BANQIAO.板橋	2016-07-03 07:00:00	26.7
BANQIAO.板橋	2016-07-03 08:00:00	28.7
BANQIAO.板橋	2016-07-03 09:00:00	30.6

Change the type of the value column so that comparing is easier when doing sql

```
alter table weather
modify column value float;
```

Delete the datas which is not possible

```
delete from weather
where value < -20.0;
```

2. SQL

a.

changing the red box part to get other results

```
select Max(south_usage) as max_south_usage,  Max(south_supply) as max_south_supply, date(create_date) as date
from power
where date(create_date) >= date'2016-10-01'
&& date(create_date) <= date'2017-06-30'
group by date(create_date)
order by date(create_date) ASC;
```

north

usage	supply	date
1099.4	943	2016/10/1
1031.6	872.9	2016/10/2
1275.4	1150.5	2016/10/3
1263	1139.6	2016/10/4
1282.9	1150.8	2016/10/5
1295.1	1099.9	2016/10/6
1222.4	1108.1	2016/10/7
1041.2	890.7	2016/10/8
896.2	739.8	2016/10/9
939.2	837.1	2016/10/10

center

usage	supply	date
763.5	869.9	2016/10/1
738.5	828.8	2016/10/2
820	950.2	2016/10/3
850.6	972.5	2016/10/4
835.6	966	2016/10/5
835.6	988.2	2016/10/6
859	960.3	2016/10/7
756.9	889.5	2016/10/8
684.7	856.8	2016/10/9
711.8	879.5	2016/10/10

south

usage	supply	date
918.3	1045.2	2016/10/1
889.3	989.5	2016/10/2
1038.7	1083	2016/10/3
1059	1095.8	2016/10/4
1079.4	1109	2016/10/5
1042.9	1146.8	2016/10/6
997.1	1069.6	2016/10/7
903.7	980	2016/10/8
869.6	872.8	2016/10/9
892.9	927.6	2016/10/10
1052.8	1059.9	2016/10/11

east

usage	supply	date
41.8	13.6	2016/10/1
41.7	13.5	2016/10/2
46.5	14	2016/10/3
47.6	13.8	2016/10/4
46.6	14.8	2016/10/5
44.7	13.4	2016/10/6
43.1	13.9	2016/10/7
44.5	14.2	2016/10/8
43.8	14.1	2016/10/9
44.6	13.9	2016/10/10
45.9	14.1	2016/10/11

Full result: https://drive.google.com/open?id=0B79D7HB4zSNocWFUZHl6dlEtNUk

b. locationName


```
select distinct locationName
from weather;
```

c.

changing the red box part to get other results

```
select max(value) as max_temperature, date(obsTime)
from weather
where locationName='TAIPEI,臺北'
&& date(obsTime) != '0000-00-00'
&& date(obsTime) >= date'2016-10-01'
&& date(obsTime) <= date'2017-06-30'
group by date(obsTime)
order by date(obsTime) asc;
```

choose 台北 represent north

temperatur	date
32.6	2016/10/1
34.8	2016/10/2
32.5	2016/10/3
32.3	2016/10/4
30.6	2016/10/5
33.1	2016/10/6
27.1	2016/10/7
31.3	2016/10/8
24.2	2016/10/9
23.9	2016/10/10
27.5	2016/10/11

choose 台中 represent north

temperatur	date
31.1	2016/10/1
32.9	2016/10/2
31.3	2016/10/3
31.4	2016/10/4
31.7	2016/10/5
30.2	2016/10/6
28.2	2016/10/7
28.5	2016/10/8
27.5	2016/10/9
28.4	2016/10/10

choose 台南 represent south

temperatur	date
30.4	2016/10/1
32.3	2016/10/2
31.1	2016/10/3
30.3	2016/10/4
31.8	2016/10/5
30.8	2016/10/6
31.5	2016/10/7
26.5	2016/10/8
25.9	2016/10/9
28.6	2016/10/10

choose 台東 represent east

temperatur	date
30.4	2016/10/1
30.6	2016/10/2
31.5	2016/10/3
31.6	2016/10/4
30.8	2016/10/5
28.6	2016/10/6
26	2016/10/7
24.7	2016/10/8
24.9	2016/10/9
28.3	2016/10/10

Full result: https://drive.google.com/open?id=0B79D7HB4zSNocWFUZHl6dlEtNUk

d.

highest temperature

```
1  select value as highest_temp, locationName, obsTime
2  from weather
3  where date(weather.obsTime) >= date'2016-10-01'
4  && date(weather.obsTime) <= date'2017-06-30'
5  && weather.value >= all
6  (select w2.value
7   from weather w2
8   where date(w2.obsTime) >= date'2016-10-01' && date(w2.obsTime) <= date'2017-06-30');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

highest_temp	locationName	obsTime
37.5	DAWU.大武	2017-06-24 12:00:00

lowest temperature

```
select value as lowest_temp, locationName, obsTime
from weather
where date(weather.obsTime) >= date'2016-10-01'
&& date(weather.obsTime) <= date'2017-06-30'
&& weather.value <= all
(select w2.value
 from weather w2
 where date(w2.obsTime) >= date'2016-10-01' && date(w2.obsTime) <= date'2017-06-30');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

lowest_temp	locationName	obsTime
-10.2	YUSHAN.玉山	2017-02-13 04:00:00
-10.2	YUSHAN.玉山	2017-02-13 05:00:00
-10.2	YUSHAN.玉山	2017-03-02 06:00:00
-10.2	YUSHAN.玉山	2017-03-27 06:00:00

3.

Using left join to combine supply, usage and temperature two tables

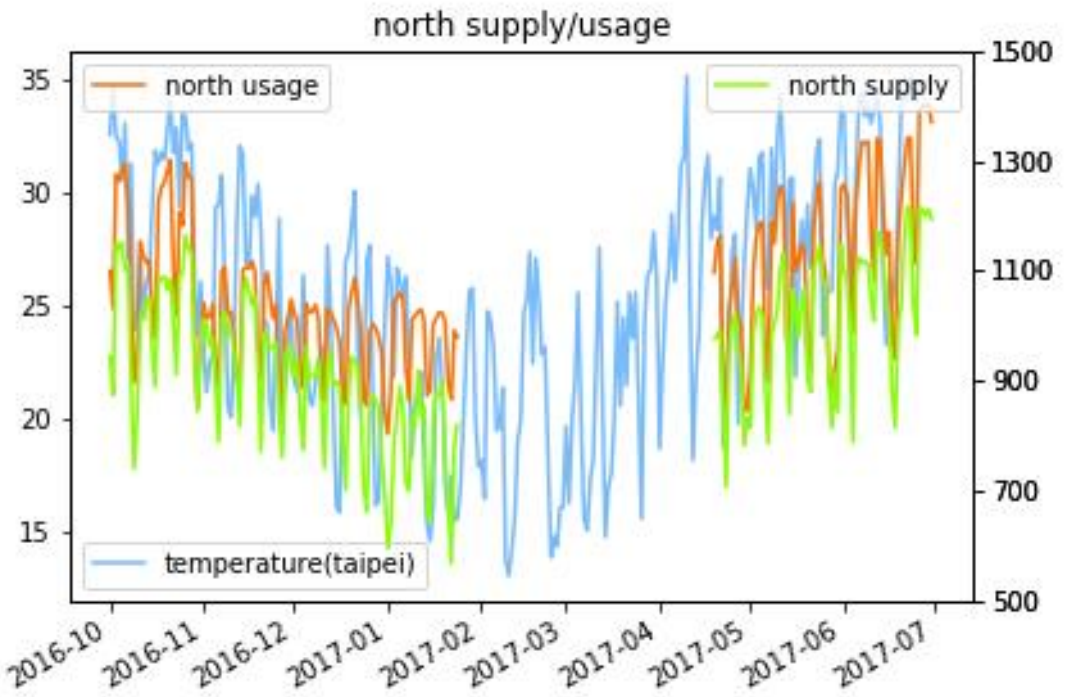
```
select max_east_usage, max_east_supply, max_temperature, w_date
from
(select date(w.obsTime) as w_date, max(w.value) as max_temperature
 from weather w
 where date(w.obsTime) >= date'2016-10-01'
 && date(w.obsTime) <= date'2017-06-30'
 && date(w.obsTime) != '0000-00-00'
 && w.locationName='TAITUNG,臺東'
 group by date(w.obsTime)
 order by date(w.obsTime) ASC) t1

 left join

 (select date(create_date) as p_date, Max(p.east_usage) as max_east_usage, Max(p.east_supply) as max_east_supply
  from power p
  where date(p.create_date) >= date'2016-10-01'
  && date(p.create_date) <= date'2017-06-30'
  group by date(p.create_date)) t2

  on t1.w_date = t2.p_date;
```

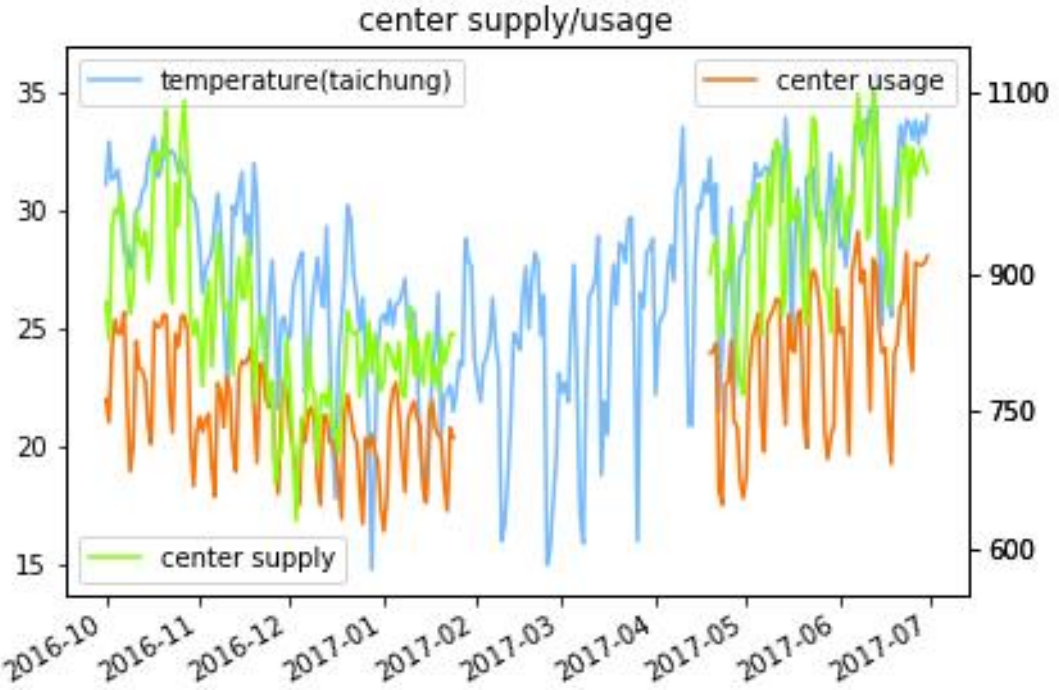
Use read_csv to get the data and put them into pandas.DataFrame also transform the data type to ‘float’ and ‘datetime’

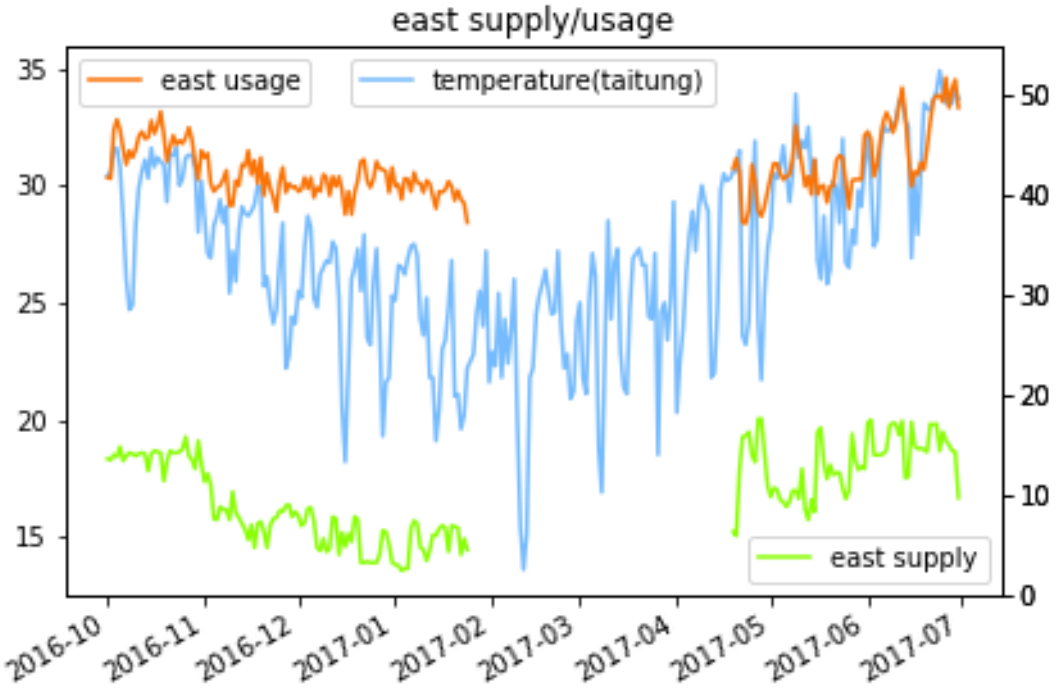
```
import numpy as np
import pandas as pd
import datetime

file = 'south.csv'

raw_data = pd.read_csv(file,header=0, delimiter=',')
data = np.array(raw_data)

df = pd.DataFrame(data, columns = ['usage', 'supply','temp','date'])
df['supply'] = df['supply'].apply(pd.to_numeric)
df['usage'] = df['usage'].apply(pd.to_numeric)
df['temp'] = df['temp'].apply(pd.to_numeric)
df['date'] = df['date'].apply(pd.to_datetime)
```

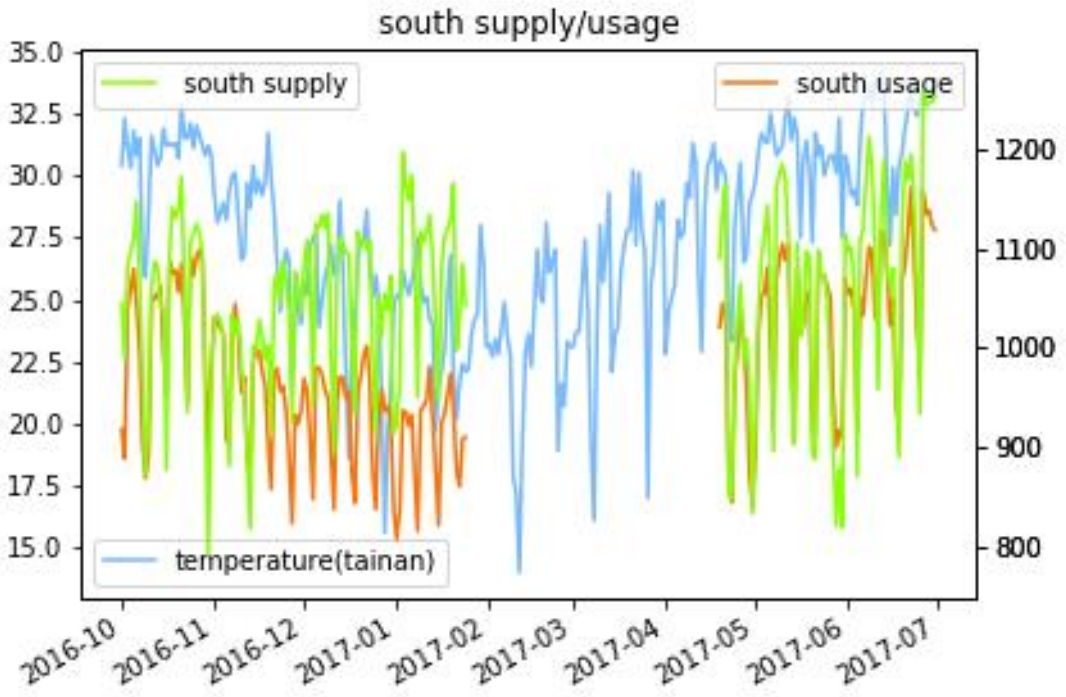
Using matplotlib.pyplot to draw the figure.
Use ax1.twinx() to let different lines share the same x-axis
Set the proper ticks and the limits to make the figure looks better

```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

fig, ax1 = plt.subplots()

ax1.plot(df['date'], df['temp'],label="temperature(taipei)",color='xkcd:sky blue')

line_usage = ax1.twinx()
line_supply = ax1.twinx()

line_usage.plot(df['date'], df['usage'], label="north usage",color='xkcd:orange')
line_supply.plot(df['date'], df['supply'],label="north supply",color='xkcd:lime green')

plt.xlabel('date')
plt.title('north supply/usage')
plt.setp(line_supply, yticks=[500, 700, 900, 1100, 1300, 1500])
plt.setp(line_usage, yticks=[500, 700, 900, 1100, 1300, 1500])
line_supply.set_ylim(500, 1500)
line_usage.set_ylim(500, 1500)

fig.autofmt_xdate()
ax1.legend()
line_usage.legend()
line_supply.legend()
plt.legend(loc="best")
fig.tight_layout()
plt.show()
fig.savefig('north.png', dpi=fig.dpi)
```




center supply/usage
temperature(taichung)
center usage
center supply
35
30
25
20
15
1100
900
750
600
2016-10
2016-11
2016-12
2017-01
2017-02
2017-03
2017-04
2017-05
2017-06
2017-07


east supply/usage
east usage
temperature(taitung)
east supply
35
30
25
20
15
50
40
30
20
10
0
2016-10
2016-11
2016-12
2017-01
2017-02
2017-03
2017-04
2017-05
2017-06
2017-07



Observation: There is positive correlation between supply, usage, and temperature.Also, north usage is more than its supply even though north has almost the highest supply. Although east has more usage than supply, it has the smallest usage at the same time.

4. using center supply and center usage to do the following tasks

D(Q,C) using distance.euclidean from scipy

```
#original dst(Q,C)
from scipy.spatial import distance
dst = distance.euclidean(Q_pre,C_pre)
print(dst)
```

```
1767.00748442
```

Importing data into python using read_csv
Then, convert it into pandas.DataFrame

```
import numpy as np
import pandas as pd
import datetime

file = 'center.csv'

raw_data = pd.read_csv(file,header=0, delimiter=',')
data = np.array(raw_data)

df = pd.DataFrame(data, columns = ['usage', 'supply','temp','date'])
df['supply'] = df['supply'].apply(pd.to_numeric)
df['usage'] = df['usage'].apply(pd.to_numeric)
df['temp'] = df['temp'].apply(pd.to_numeric)
df['date'] = df['date'].apply(pd.to_datetime)
```

Drawing figure using matplotlib.pyplot

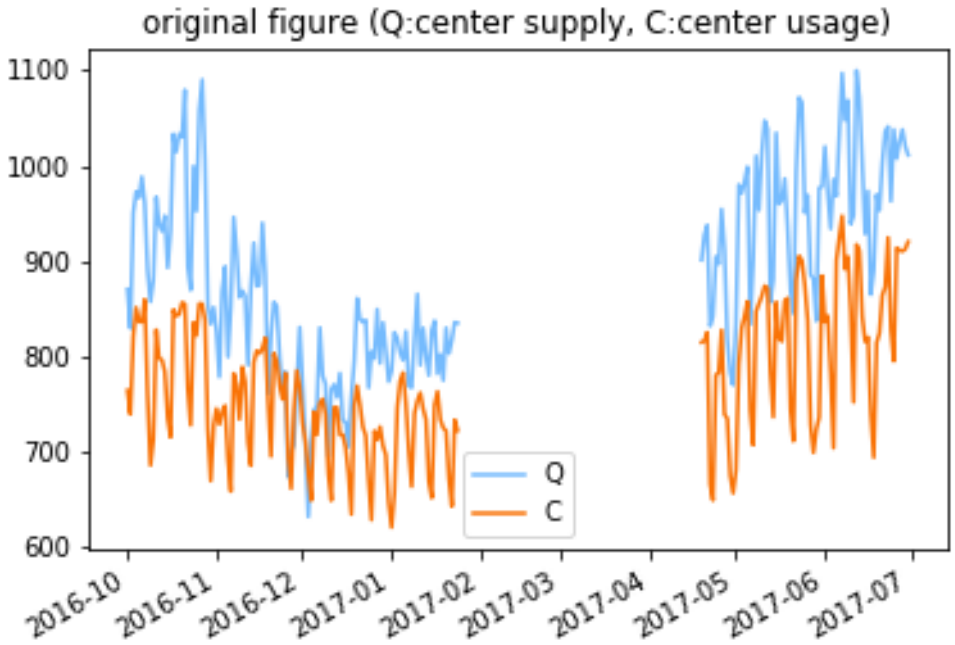
```
import matplotlib.pyplot as plt

fig, ax1 = plt.subplots()

ax1.plot(df['date'], Q,label="Q",color='xkcd:sky blue')
ax1.plot(df['date'], C,label="C",color='xkcd:orange')
plt.title('original figure (Q:center supply, C:center usage)')
ax1.legend()
fig.autofmt_xdate()
plt.show()
fig.savefig('original.png', dpi=fig.dpi)
```

Since there are some lost data in taipower , hence I deal with it using two separate array to store data , one stores ‘nan’ if there should be data but loss, the other just stores every value with no ‘nan’s in it. It much easier to call some function such as np.mean(), np.std(), signal.detrend() and don’t have to deal with the ‘nan’s. But have to do some conversion to draw the figure.

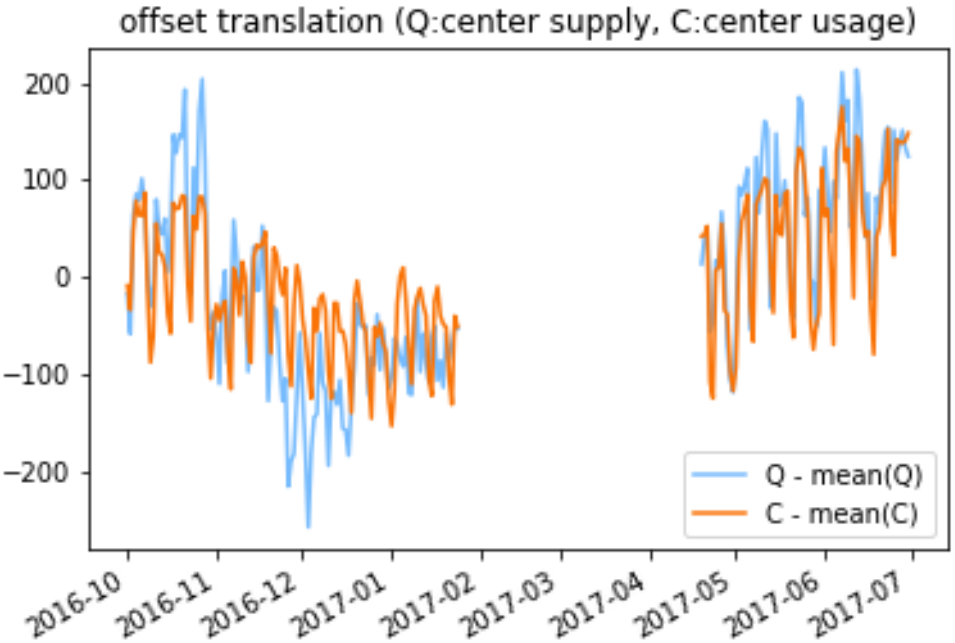
```
for i in range(0, len(Q_trans)):
    if Q_trans[i]>0:
        Q_trans[i] = Q_trans[i]-Q_mean
    else:
        Q_trans[i] = np.nan
for i in range(0, len(C)):
    if C_trans[i]>0:
        C_trans[i] = C_trans[i]-C_mean
    else:
        C_trans[i] = np.nan
```

```
for i in range(0, len(Q_sm2)):
    if i<=116:
        Q_sm2[i] = Q_d2[i]
    elif i>200:
        Q_sm2[i] = Q_d2[i-84]
    else:
        Q_sm2[i] = np.nan
for i in range(0, len(C_sm2)):
    if i<=116:
        C_sm2[i] = C_d2[i]
    elif i>200:
        C_sm2[i] = C_d2[i-84]
    else:
        C_sm2[i] = np.nan
```

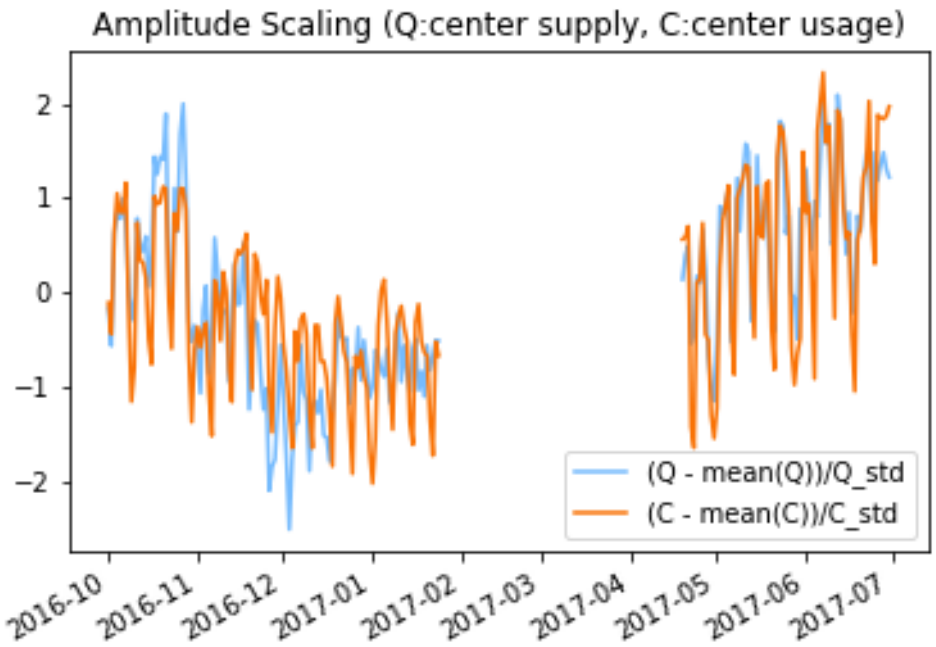
Original: D(Q,C) = 1767.00748442



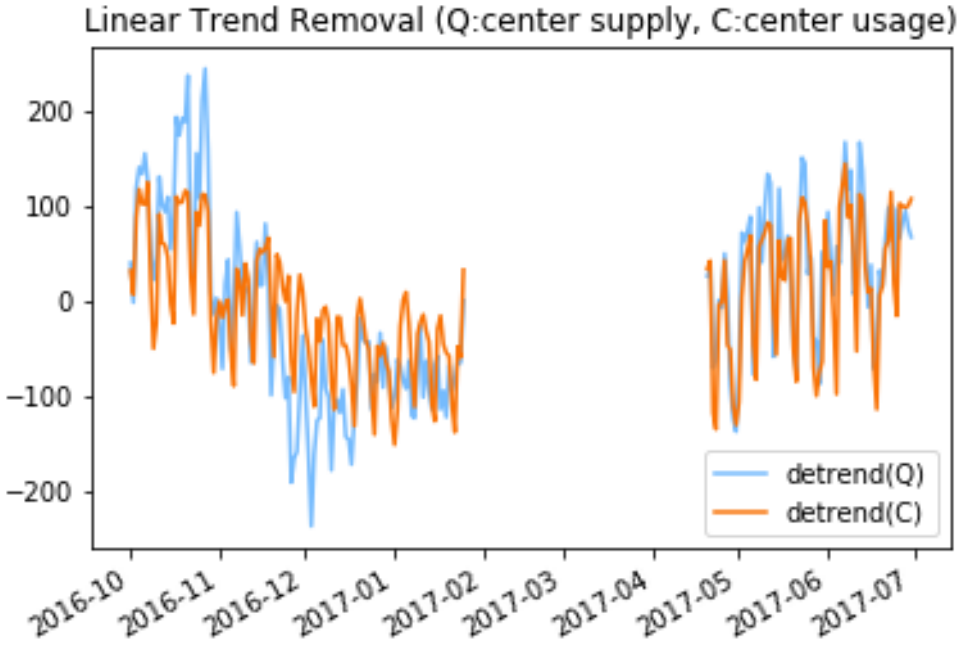
a. Offset translation: D(Q,C) = 791.628662212



b. Amplitude Scaling: D(Q,C) = 8.03557872136



c. Linear Trend Removal: D(Q,C) = 780.4740664315586



d. Noise reduction: D(Q,C) = 1704.51067624