solar panel_2 current

January 14, 2021

This data has been gathered at two solar power plants in India over a 34 day period. It has two pairs of files - each pair has one power generation dataset and one sensor readings dataset. The power generation datasets are gathered at the inverter level - each inverter has multiple lines of solar panels attached to it. The sensor data is gathered at a plant level - single array of sensors optimally placed at the plant.

0.0.1 Provenance

Sources

Power generation and sensor data gathered from two solar power plants

Collection methodology

Power generation and sensor data gathered at 15 minutes intervals over a 34 day period. Generation data collected at inverter level, while the sensor data is at the plant level. ***

0.0.2 Columns

Plant 1&2 Generation data @Inverter level

```
DATE_TIME- Date and time for each observation.
           Observations recorded at 15 minute intervals.

PLANT_ID - this will be common for the entire file.

SOURCE_KEY - Source key in this file stands for the inverter id.
             changed to Inverter id)

DC_POWER - Amount of DC power generated by the inverter (source_key)
           in this 15 minute interval. Units - kW.

AC_POWER - Amount of AC power generated by the inverter (source_key)
           in this 15 minute interval. Units - kW.

DAILY_YIELD - Daily yield is a cumulative sum of power generated
              on that day, till that point in time.

TOTAL_YIELD - This is the total yield for the inverter till that
```

```
                        point in time.
```

Plant 1&2 Weather sensor data @Plant level

```
DATE_TIME- Date and time for each observation.
           Observations recorded at 15 minute intervals.

PLANT_ID - this will be common for the entire file.

SOURCE_KEY - Stands for the sensor panel id. This will be
             common for the entire file because there's
             only one sensor panel for the plant.

AMBIENT_TEMPERATURE - This is the ambient temperature at
                      the plant.

MODULE_TEMPERATURE - There's a module (solar panel) attached
                     to the sensor panel. This is the
                     temperature reading for that module.
```

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mlp
import seaborn as sns
import numpy as np
import stat as st
```

[2]:
```
gen_1 = pd.read_csv('Plant_1_Generation_Data.csv',delimiter=',')
gen_2 = pd.read_csv('Plant_2_Generation_Data.csv',delimiter=',')

p1 = pd.read_csv('Plant_1_Weather_Sensor_Data.csv',delimiter=',')
p2 = pd.read_csv('Plant_2_Weather_Sensor_Data.csv',delimiter=',')

gen_1.rename(columns={'SOURCE_KEY':'INVERTER_ID'},inplace =True)
gen_2.rename(columns={'SOURCE_KEY':'INVERTER_ID'},inplace =True)
```

1 Understanding the data

1.1 Generation data

[3]:
```
gen_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 68778 entries, 0 to 68777
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   DATE_TIME    68778 non-null  object
 1   PLANT_ID     68778 non-null  int64
```

```
 2   INVERTER_ID  68778 non-null  object
 3   DC_POWER     68778 non-null  float64
 4   AC_POWER     68778 non-null  float64
 5   DAILY_YIELD  68778 non-null  float64
 6   TOTAL_YIELD  68778 non-null  float64
dtypes: float64(4), int64(1), object(2)
memory usage: 3.7+ MB
```

[4]:
```
print('gen1 # of Inverters:',gen_1['INVERTER_ID'].nunique())
print('gen2 # of Inverters:',gen_2['INVERTER_ID'].nunique())
```

```
gen1 # of Inverters: 22
gen2 # of Inverters: 22
```

[5]:
```
gen_1[['DATE_TIME','PLANT_ID','INVERTER_ID','DC_POWER']].head(23)
```

[5]:

	DATE_TIME	PLANT_ID	INVERTER_ID	DC_POWER
0	15-05-2020 00:00	4135001	1BY6WEcLGh8j5v7	0.0
1	15-05-2020 00:00	4135001	1IF53ai7Xc0U56Y	0.0
2	15-05-2020 00:00	4135001	3PZuoBAID5Wc2HD	0.0
3	15-05-2020 00:00	4135001	7JYdWkrLSPkdwr4	0.0
4	15-05-2020 00:00	4135001	McdE0feGgRqW7Ca	0.0
5	15-05-2020 00:00	4135001	VHMLBKoKgIrUVDU	0.0
6	15-05-2020 00:00	4135001	WRmjgnKYAwPKWDb	0.0
7	15-05-2020 00:00	4135001	ZnxXDlPa8U1GXgE	0.0
8	15-05-2020 00:00	4135001	ZoEaEvLYb1n2sOq	0.0
9	15-05-2020 00:00	4135001	adLQvlD726eNBSB	0.0
10	15-05-2020 00:00	4135001	bvBOhCH3iADSZry	0.0
11	15-05-2020 00:00	4135001	iCRJl6heRkivqQ3	0.0
12	15-05-2020 00:00	4135001	ih0vzX44oOqAx2f	0.0
13	15-05-2020 00:00	4135001	pkci93gMrogZuBj	0.0
14	15-05-2020 00:00	4135001	rGa61gmuvPhdLxV	0.0
15	15-05-2020 00:00	4135001	sjndEbLyjtCKgGv	0.0
16	15-05-2020 00:00	4135001	uHbuxQJl8lW7ozc	0.0
17	15-05-2020 00:00	4135001	wCURE6d3bPkepu2	0.0
18	15-05-2020 00:00	4135001	z9Y9gH1T5YWrNuG	0.0
19	15-05-2020 00:00	4135001	zBIq5rxdHJRwDNY	0.0
20	15-05-2020 00:00	4135001	zVJPv84UY57bAof	0.0
21	15-05-2020 00:15	4135001	1BY6WEcLGh8j5v7	0.0
22	15-05-2020 00:15	4135001	1IF53ai7Xc0U56Y	0.0

In the previous two cells, I was able to find out how many inverters each generator had.

```
gen1 # of Inverters: 22
gen2 # of Inverters: 22
```

Looking at the structure of 'DATE_TIME' in the previous cell, you can see that for each time interval there is meant to be a row for each unique inverter. eg 22 **unique** inverters at '2020-05-15 00:00:00'. However, this is not true. There are only 21 rows of DATE_TIME '2020-05-15 00:00:00'.

It appears that an inverter did not log any data for this interval, I wonder how often this has happened throughout the data set. This error might also explain the difference in data set entries between gen_1 and gen_2.

---

[6]:
```
gen_2[['DATE_TIME','PLANT_ID','INVERTER_ID','DC_POWER']].head(23)
```

[6]:
```
              DATE_TIME  PLANT_ID      INVERTER_ID  DC_POWER
0   2020-05-15 00:00:00   4136001  4UPUqMRk7TRMgml       0.0
1   2020-05-15 00:00:00   4136001  81aHJ1q11NBPMrL       0.0
2   2020-05-15 00:00:00   4136001  9kRcWv60rDACzjR       0.0
3   2020-05-15 00:00:00   4136001  Et9kgGMDl729KT4       0.0
4   2020-05-15 00:00:00   4136001  IQ2d7wF4YD8zU1Q       0.0
5   2020-05-15 00:00:00   4136001  LYwnQax7tkwH5Cb       0.0
6   2020-05-15 00:00:00   4136001  LlT2YUhhzqhg5Sw       0.0
7   2020-05-15 00:00:00   4136001  Mx2yZCDsyf6DPfv       0.0
8   2020-05-15 00:00:00   4136001  NgDl19wMapZy17u       0.0
9   2020-05-15 00:00:00   4136001  PeE6FRyGXUgsRhN       0.0
10  2020-05-15 00:00:00   4136001  Qf4GUc1pJu5T6c6       0.0
11  2020-05-15 00:00:00   4136001  Quc1TzYxW2pYoWX       0.0
12  2020-05-15 00:00:00   4136001  V94E5Ben1TlhnDV       0.0
13  2020-05-15 00:00:00   4136001  WcxssY2VbP4hApt       0.0
14  2020-05-15 00:00:00   4136001  mqwcsP2rE7J0TFp       0.0
15  2020-05-15 00:00:00   4136001  oZ35aAeoifZaQzV       0.0
16  2020-05-15 00:00:00   4136001  oZZkBaNadn6DNKz       0.0
17  2020-05-15 00:00:00   4136001  q49J1IKaHRwDQnt       0.0
18  2020-05-15 00:00:00   4136001  rrq4fwE8jgrTyWY       0.0
19  2020-05-15 00:00:00   4136001  vOuJvMaM2sgwLmb       0.0
20  2020-05-15 00:00:00   4136001  xMbIugepa2P7lBB       0.0
21  2020-05-15 00:00:00   4136001  xoJJ8DcxJEcupym       0.0
22  2020-05-15 00:15:00   4136001  4UPUqMRk7TRMgml       0.0
```

Here we go, my suspicions about the missing data seem to be true. Above in Gen_2 all 22 inverters have data entries for the DATE_TIME 2020-05-15 00:00:00. because of the disparity of entries between gen_1 and gen_2 *1,080* i will investigate the missing data further, but first, I will convert 'DATE_TIME' column to Dtype 'date_time'.

[7]:
```
gen_1.tail(1)
```

[7]:
```
              DATE_TIME  PLANT_ID      INVERTER_ID  DC_POWER  AC_POWER  \
68777  17-06-2020 23:45   4135001  zVJPv84UY57bAof       0.0       0.0

       DAILY_YIELD  TOTAL_YIELD
68777       5910.0    7363272.0
```

gen_1, DATE_TIME format: day-month-year 24H:Minute

[8]:
```
gen_2.tail(1)
```

[8]:
```
                 DATE_TIME  PLANT_ID      INVERTER_ID  DC_POWER  AC_POWER  \
67697  2020-06-17 23:45:00   4136001  xoJJ8DcxJEcupym       0.0       0.0

       DAILY_YIELD  TOTAL_YIELD
67697       4316.0  209335741.0
```

gen_2, DATE_TIME format: year-month-day 24Hour:Minute:second

[9]:
```
gen_1['DATE_TIME'] = pd.to_datetime(gen_1['DATE_TIME'],format ='%d-%m-%Y %H:%M')
gen_2['DATE_TIME'] = pd.to_datetime(gen_2['DATE_TIME'],format ='%Y-%m-%d %H:%M:
 ↪%S')
```

2 Missing inverter data

Investigating missing entires.

[10]:
```
print('Gen_1 unique inverters')
print('\n')
inv_freq1 = gen_1['INVERTER_ID'].value_counts()
print(inv_freq1)
print('\n')
print('inverters:',inv_freq1.count())
print('68778 entries')
print('Confirming count matches',inv_freq1.sum())
```

```
Gen_1 unique inverters


bvBOhCH3iADSZry    3155
1BY6WEcLGh8j5v7    3154
VHMLBKoKgIrUVDU    3133
7JYdWkrLSPkdwr4    3133
ZnxXDlPa8U1GXgE    3130
ih0vzX44oOqAx2f    3130
wCURE6d3bPkepu2    3126
z9Y9gH1T5YWrNuG    3126
iCRJl6heRkivqQ3    3125
uHbuxQJl8lW7ozc    3125
pkci93gMrogZuBj    3125
McdE0feGgRqW7Ca    3124
rGa61gmuvPhdLxV    3124
zVJPv84UY57bAof    3124
sjndEbLyjtCKgGv    3124
ZoEaEvLYb1n2sOq    3123
adLQvlD726eNBSB    3119
zBIq5rxdHJRwDNY    3119
1IF53ai7Xc0U56Y    3119
WRmjgnKYAwPKWDb    3118
```

```
3PZuoBAID5Wc2HD    3118
YxYtjZvoooNbGkE    3104
Name: INVERTER_ID, dtype: int64


inverters: 22
68778 entries
Confirming count matches 68778
```

[11]:
```
print('Gen_2 unique inverters')
print('\n')
inv_freq2 = gen_2['INVERTER_ID'].value_counts()
print(inv_freq2)
print('\n')
print('inverters:',inv_freq2.count())
print('67698 entries')
print('Confirming count matches',inv_freq2.sum())
```

```
Gen_2 unique inverters


LlT2YUhhzqhg5Sw    3259
PeE6FRyGXUgsRhN    3259
81aHJ1q11NBPMrL    3259
V94E5Ben1TlhnDV    3259
LYwnQax7tkwH5Cb    3259
9kRcWv60rDACzjR    3259
WcxssY2VbP4hApt    3259
rrq4fwE8jgrTyWY    3259
xoJJ8DcxJEcupym    3259
vOuJvMaM2sgwLmb    3259
q49J1IKaHRwDQnt    3259
oZZkBaNadn6DNKz    3259
oZ35aAeoifZaQzV    3195
4UPUqMRk7TRMgml    3195
Et9kgGMDl729KT4    3195
Quc1TzYxW2pYoWX    3195
Qf4GUc1pJu5T6c6    3195
Mx2yZCDsyf6DPfv    3195
mqwcsP2rE7J0TFp    2355
IQ2d7wF4YD8zU1Q    2355
NgDl19wMapZy17u    2355
xMbIugepa2P7lBB    2355
Name: INVERTER_ID, dtype: int64


inverters: 22
67698 entries
```

```
Confirming count matches 67698
```

Immediately it's apparent that the issue of missing inverter data is larger than I had initially thought. My initial theory was that a few culprit inverters were not functioning properly, causing the disparity in data entries. However, it seems that most if not all the inverters are missing at least some data.

To understand the extent of the problem we need to know how many data entries there should be for a 100% functional inverter.

"*Collection methodology Power generation and sensor data gathered at 15 minutes intervals over 34 days*"

According to this, there should be 4 intervals per hour for each inverter. With 24 hours in a day for 34 days, equals a total of 816 hours.

$816 * 4 = 3,264$ **intervals of 15 minutes.**

None of the inverters matches this number, However, most are close enough except 4. these 4 inverters from **gen_2** are far below 3,264 .

```
mqwcsP2rE7J0TFp    2355
NgDl19wMapZy17u    2355
IQ2d7wF4YD8zU1Q    2355
xMbIugepa2P71BB    2355
```

2.1 Inverter reliability %

[12]:
```
gen1_inv_activation = (inv_freq1/3264)*100
gen1_inv_activation = round(gen1_inv_activation,1)

print('Gen1 Top 5 inverters','\n', gen1_inv_activation.head(5))
print('Sample mean reliability of gen_1 inverters',round(gen1_inv_activation.
 ↪mean(),1))
```

```
Gen1 Top 5 inverters
 bvBOhCH3iADSZry    96.7
1BY6WEcLGh8j5v7    96.6
VHMLBKoKgIrUVDU    96.0
7JYdWkrLSPkdwr4    96.0
ZnxXDlPa8U1GXgE    95.9
Name: INTER_ID, dtype: float64
Sample mean reliability of gen_1 inverters 95.8
```

These inverters from gen_1 had the least amount of missing data. There is also the mean reliability of this sample of data.

[13]:
```
gen2_inv_activation = (inv_freq2/3264)*100
gen2_inv_activation = round(gen2_inv_activation,1)
```

```
print('Gen2 Top 5 inverters','\n', gen2_inv_activation.head(5))
print('Sample mean reliability of gen_1 inverters w␣
↪outliers',round(gen2_inv_activation.mean(),1))
print('Sample mean reliability of gen_1 inverters w/o␣
↪outliers',round(gen2_inv_activation[:18].mean(),1))
```

```
Gen2 Top 5 inverters
 LlT2YUhhzqhg5Sw    99.8
PeE6FRyGXUgsRhN    99.8
81aHJ1q11NBPMrL    99.8
V94E5Ben1TlhnDV    99.8
LYwnQax7tkwH5Cb    99.8
Name: INVERTER_ID, dtype: float64
Sample mean reliability of gen_1 inverters w outliers 94.3
Sample mean reliability of gen_1 inverters w/o outliers 99.2
```

These inverters from gen_1 had the least amount of missing data.

[14]:
```
def groupby_inv_date(df,freq,fillna=False):
    gb = df.groupby(['INVERTER_ID',pd.
↪Grouper(freq=freq,key='DATE_TIME')])['INVERTER_ID'].count()
    gb_org = gb.unstack().transpose()

    if fillna == True:
        gb_org_cleaned = gb_org.fillna(0)
        return gb_org_cleaned

    return gb_org
```

[15]:
```
pctactive_24hg1 = groupby_inv_date(gen_1,'24h',True)
pctactive_24hg2 = groupby_inv_date(gen_2,'24h',True)

pctactive_24hg1 = pctactive_24hg1/96
pctactive_24hg2 = pctactive_24hg2/96

#percentage of inverters active by day.
fig,axes = plt.subplots(3,1,figsize=(15,10))
fig.suptitle('Inverter mean reliability by day')
ax1 = sns.lineplot(ax=axes[0],data=pctactive_24hg1,legend=False,marker='o')
ax2 = sns.lineplot(ax=axes[1],data=pctactive_24hg2,legend=False,marker='o')
ax3 = sns.lineplot(ax=axes[2],data=pctactive_24hg2,legend=False,marker='o')
ax3.set_ylim(0.9)
ax3.margins(x=0.05,y=-0.25)
ax1.set_title('gen 1 - fig1')
ax2.set_title('gen 2 - fig2')
ax3.set_title('gen 2 zoomed - fig3')
ax1.set_xlabel('')
ax2.set_xlabel('')
```

```
ax1.set_ylabel('mean reliability')
ax2.set_ylabel('mean reliability')
ax3.set_ylabel('mean reliability')

plt.show()
```

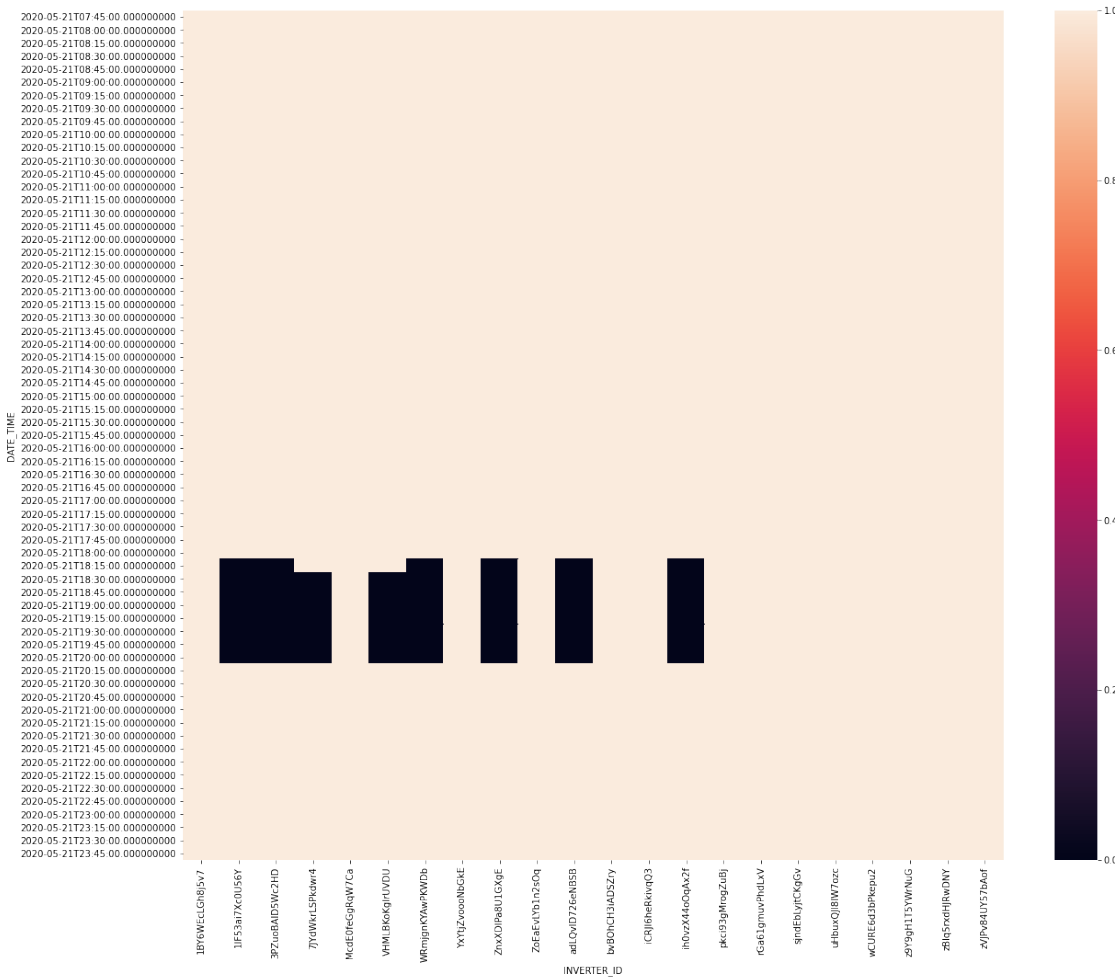
I think what stands out are the two valleys in fig_1. Both valleys involve all 22 inverters, with the first valley reaching its minima at around 05-21, with the second occurring around 05-29.

The markers indicate as to the rate of change in reliability

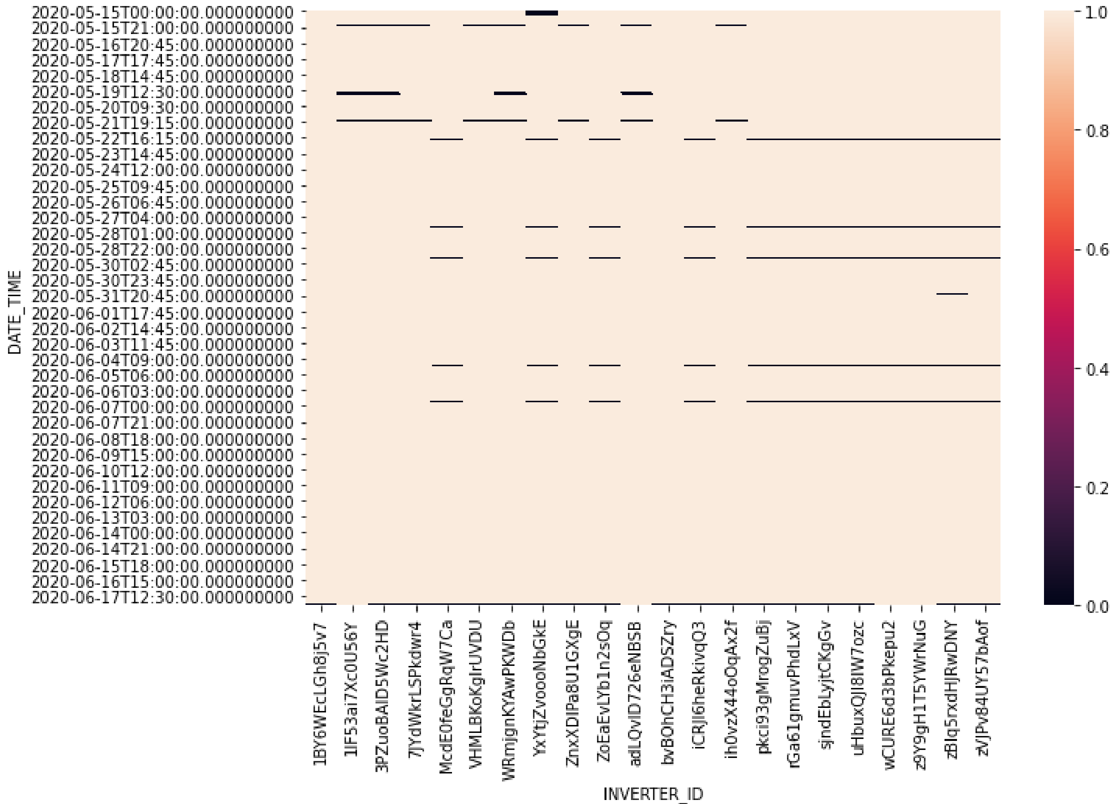
2.1.1 gen_1 what happened at 05-21 and 05-29 ?

Between the 18th and 20th of May the reliability of the inverters fell from above 0.95 to below 0.7. i want to try and find out a closer time frame and see if there is any indication as to why.

[25]:
```
inv_1c = groupby_inv_date(gen_1,'15T',True)
f, ax = plt.subplots(figsize=(20, 17))
f = sns.heatmap(inv_1c.loc['2020-05-21 00:00:00':'2020-05-21 23:45:00'])
```



```
[412]: f, ax = plt.subplots(figsize=(20, 17))
       f = sns.heatmap(inv_1c.loc['2020-05-15 00:00:00':'2020-05-17 12:45:00'])
```



[308]:
```python
inv_1 = groupby_inv_date(gen_1,'15T',False)
inv_1.iloc[:,:5].tail()
```

[308]:
```
INVERTER_ID          1BY6WEcLGh8j5v7  1IF53ai7Xc0U56Y  3PZuoBAID5Wc2HD  \
DATE_TIME
2020-06-17 23:45:00              1.0              1.0              1.0
2020-05-25 05:30:00              NaN              1.0              NaN
2020-05-25 06:00:00              NaN              1.0              1.0
2020-05-26 18:15:00              NaN              1.0              1.0
2020-06-03 14:00:00              NaN              1.0              NaN

INVERTER_ID          7JYdWkrLSPkdwr4  McdE0feGgRqW7Ca
DATE_TIME
2020-06-17 23:45:00              1.0              1.0
2020-05-25 05:30:00              NaN              NaN
2020-05-25 06:00:00              1.0              NaN
2020-05-26 18:15:00              1.0              1.0
2020-06-03 14:00:00              NaN              NaN
```

[403]:
```python
inv_1c = groupby_inv_date(gen_1,'15T',True)
inv_2c = groupby_inv_date(gen_1,'24h',True)
```

What's interesting is that there seems to be a pattern for when the inverters did not record data. The pattern between the subplots is strikingly similar. There appear to be groupings of periods when the inverters where note recording data. because of the pattern, my initial thought is that there might have been some kind of scheduled maintenance happening on the inverters to cause them to not record at the same time.

2.1.2 Dropping outlier inverters.

If you recall the inverters from gen_2. There were 4 inverters with a reliability of 0 between the 21st and 28th, the reason for this is because those 4 inverters were offline for that week. These 4 inverters are outliers, and for this reason, I will create a copy of gen_2 without these 4 inverters so that when I analysis the power output the figures are affected.

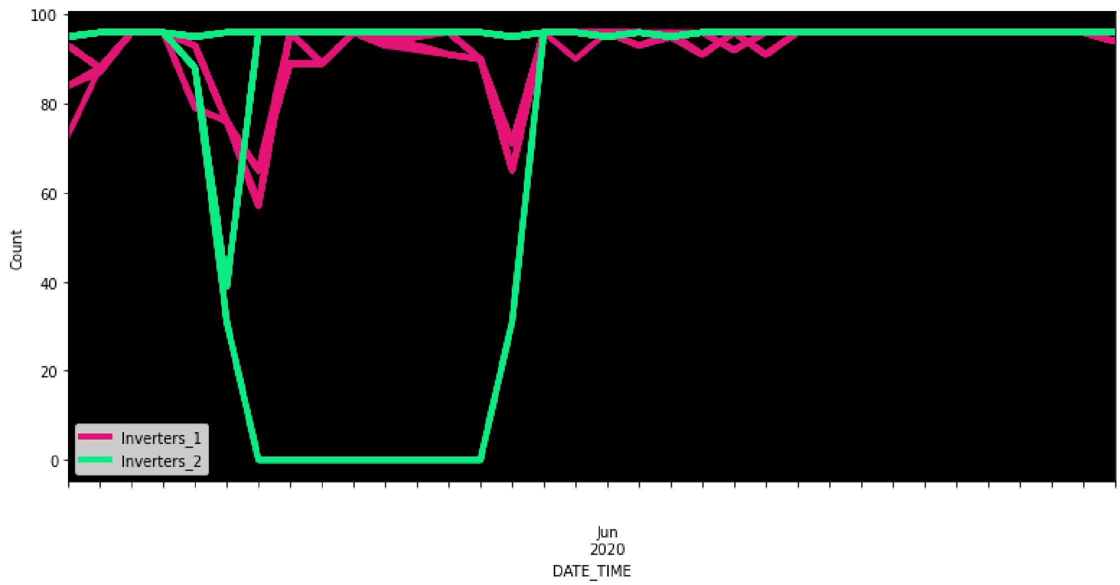
```
IQ2d7wF4YD8zU1Q
NgDl19wMapZy17u
mqwcsP2rE7J0TFp
xMbIugepa2P71BB
```

[145]:
```
mask = (gen_2['INVERTER_ID']=='IQ2d7wF4YD8zU1Q')␣
 ↪|(gen_2['INVERTER_ID']=='NgDl19wMapZy17u')|(gen_2['INVERTER_ID']=='mqwcsP2rE7J0TFp')|(gen_2
drop_index = gen_2[mask].index
clean_g2 = gen_2.drop(drop_index)
clean_g2['INVERTER_ID'].nunique()
```

[145]: 18

2.1.3 Comparison

[146]:
```
fig,ax = plt.subplots(1,1,figsize=(13,6))
inv_1.plot(ax=ax,legend=False,c='#E31475',lw=4)
inv_2.plot(ax=ax,legend=False,c='#09ED83',lw=4)

handles, labels = plt.gca().get_legend_handles_labels()
h = []
h.append(handles[0])
h.append(handles[-1])

ax.legend(h,('Inverters_1','Inverters_2'))
ax.set_ylabel('Count')
ax.set_facecolor("black")
ax.set_xticks(ticks=inv_1.index)

plt.show()
```



3 Power Output

3.1 Data error

[158]:
```
gen_1[['AC_POWER','DC_POWER']].describe()
```

[158]:

	AC_POWER	DC_POWER
count	68778.000000	68778.000000
mean	307.802752	314.742621
std	394.396439	403.645717
min	0.000000	0.000000
25%	0.000000	0.000000
50%	41.493750	42.900000
75%	623.618750	636.696429
max	1410.950000	1447.112500

[156]:
```
clean_g2[['AC_POWER','DC_POWER']].describe()
```

[156]:

	AC_POWER	DC_POWER
count	58278.000000	58278.000000
mean	236.198958	241.503113
std	358.800021	367.166035
min	0.000000	0.000000
25%	0.000000	0.000000
50%	0.000000	0.000000
75%	424.000000	432.155000
max	1385.420000	1420.933333

Above on gen_1 it seems the data has been entered incorrectly, DC power is roughly 10x that of AC power. After confirming that gen_2 dc power did not have this same error U feel confidant dividing gen_1 dc power by 10.

[157]:
```
gen_1['DC_POWER'] = gen_1['DC_POWER']/10
```

3.2 AC/DC

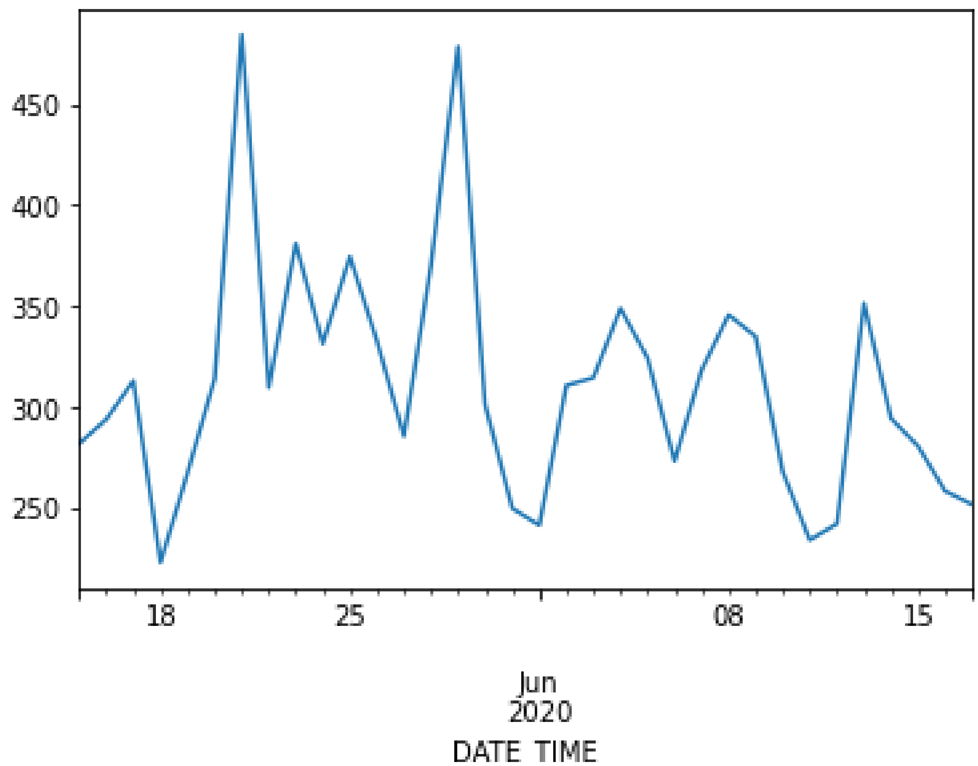
3.2.1 Max/Min power generated in 24 hours

[297]:
```
c = ['AC_POWER','DC_POWER']
max1_gb = gen_1.groupby([pd.Grouper(freq='1d',key='DATE_TIME')])[c].sum()
max2_gb = gen_2.groupby([pd.Grouper(freq='1d',key='DATE_TIME')])[c].sum()

g2_max = max2_gb.reset_index()
g1_max = max1_gb.reset_index()
g1_max = g1_max.max()
g2_max = g2_max.max()
g2_max=g2_max.rename('gen_2 max 24H output AC/DC ')
g1_max=g1_max.rename('gen_1 max 24H output AC/DC ')

print(g1_max)
print('\n')
print(g2_max)
```

```
DATE_TIME    2020-06-17 00:00:00
AC_POWER                  771576
DC_POWER                  789897
Name: gen_1 max 24H output AC/DC , dtype: object


DATE_TIME    2020-06-17 00:00:00
AC_POWER                  651438
DC_POWER                  666608
Name: gen_2 max 24H output AC/DC , dtype: object
```

[353]:
```
min_gb = gen_1.groupby([pd.Grouper(freq='1d',key='DATE_TIME')])[c].mean()
min_gb['AC_POWER'].plot()
min_gb['AC_POWER'].std()
```

[353]:
```
59.851533088359595
```



3.3 Mean AC/DC output over 24 hours

[159]:
```
columns = ['AC_POWER','DC_POWER']

g1_hour = gen_1.copy()
g1_hour.index = g1_hour['DATE_TIME']

#g2_hour = gen_2.copy()
g2_hour=clean_g2.copy()
g2_hour.index = g2_hour['DATE_TIME']

gen1_h_output = g1_hour.groupby(by=g1_hour.index.hour)[columns].mean()
gen2_h_output= g2_hour.groupby(by=g2_hour.index.hour)[columns].mean()
gen1_h_output
```

[159]:

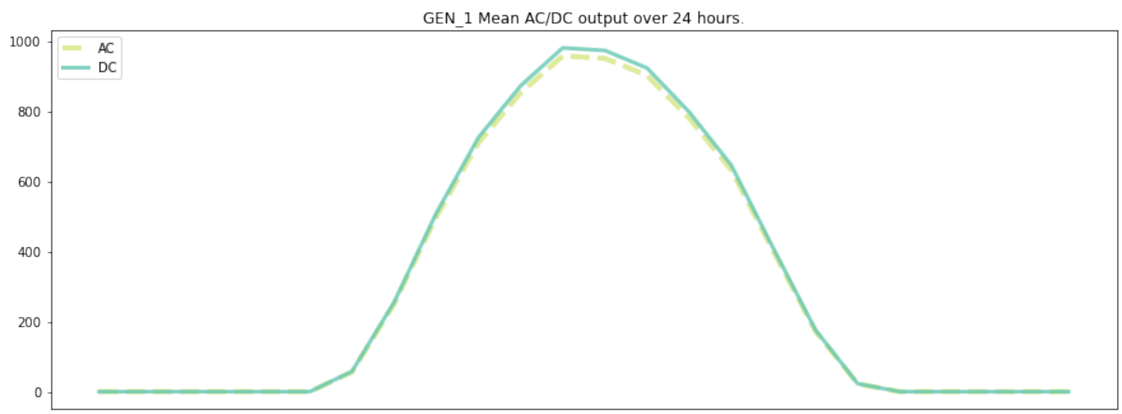
	AC_POWER	DC_POWER
DATE_TIME		
0	0.000000	0.000000
1	0.000000	0.000000
2	0.000000	0.000000
3	0.000000	0.000000
4	0.000000	0.000000

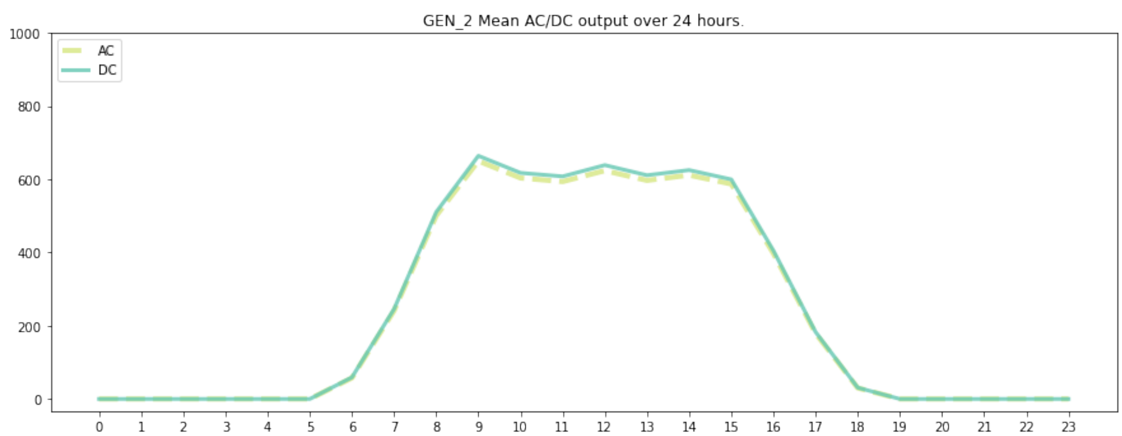
```
5            0.000000    0.000000
6           56.135778   57.811362
7          250.239163  255.138672
8          498.911000  508.862822
9          709.346945  724.739896
10         852.328529  872.041154
11         957.688308  980.556592
12         950.481939  973.128183
13         902.032936  923.237755
14         780.039030  797.701627
15         632.251989  645.788186
16         400.311990  408.243828
17         171.865596  175.636688
18          22.155258   22.886792
19           0.000000    0.000000
20           0.000000    0.000000
21           0.000000    0.000000
22           0.000000    0.000000
23           0.000000    0.000000
```

So this looks about what you would expect during low light hours there is very little power generated and as the day progresses towards midday the power output increased and then decreased as it approaches night time. However there is immediately and error in the data that stands out to me, and that is DC_POWER from first glance it appears that the dc_power value is about 10* what it should be. I say this because ac_power and dc_power should be very similar.

[160]:
```
fig = plt.figure(figsize=(15,12))
ax1 = fig.add_subplot(2,1,1)
ax2 = fig.add_subplot(2,1,2)

#D6E681
ax1.
 ↪plot(gen1_h_output['AC_POWER'],label='AC',color='#D6E681',ls='--',lw=4,alpha=0.
 ↪8)
ax1.
 ↪plot(gen1_h_output['DC_POWER'],label='DC',color='#63C7B2',ls='-',lw=3,alpha=0.
 ↪8)

ax2.
 ↪plot(gen2_h_output['AC_POWER'],label='AC',color='#D6E681',ls='--',lw=4,alpha=0.
 ↪8)
ax2.
 ↪plot(gen2_h_output['DC_POWER'],label='DC',color='#63C7B2',ls='-',lw=3,alpha=0.
 ↪8)



ax2.set_xticks(range(0,24,1))
```

```python
ax2.set_yticks([0,200,400,600,800,1000])
ax1.tick_params(axis='x',bottom=False,labelbottom=False)

ax1.legend(loc='upper left')
ax2.legend(loc='upper left')

#ax1.set_facecolor("black")
#ax2.set_facecolor("black")

ax1.set_title('GEN_1 Mean AC/DC output over 24 hours.')
ax2.set_title('GEN_2 Mean AC/DC output over 24 hours.')
plt.show()
```

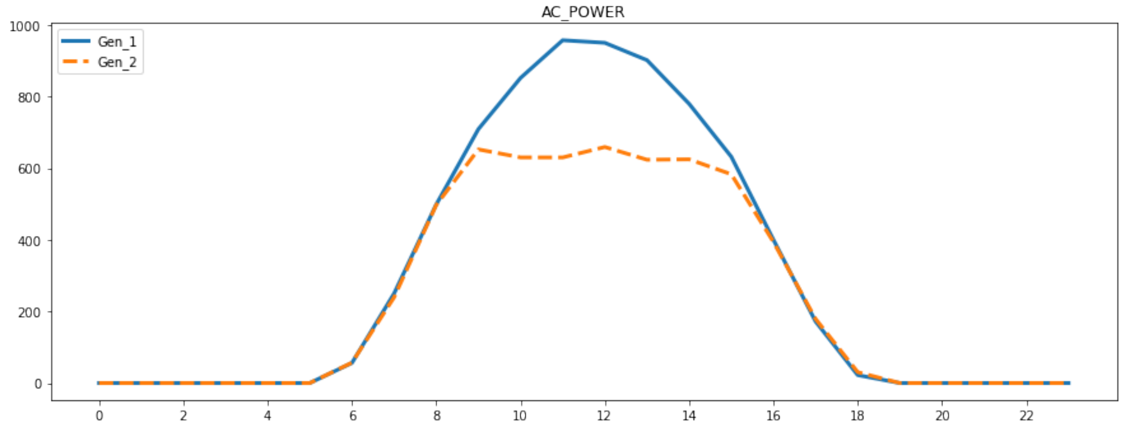


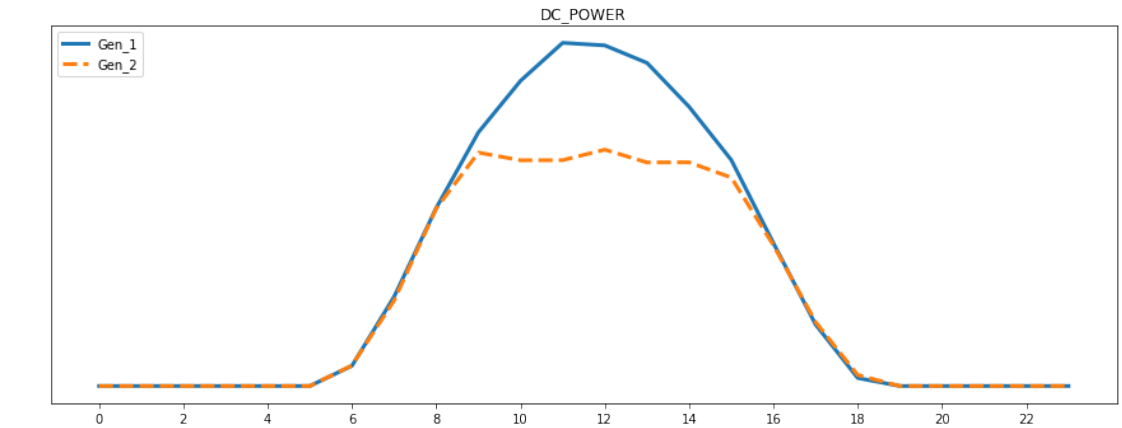


Here is the mean output per hour, over the 34 days. The graph representing gen_1 makes intuitive sense. The solar panels start to gradually generate more power as the day reaches noon and then it reverses and light levels start to drop. **gen_2** follows this path but with its top cut off.

[41]:
```python
columns = ['AC_POWER','DC_POWER']
```

```python
g1_hour = gen_1.copy()
g1_hour.index = g1_hour['DATE_TIME']
gen1_h_output = g1_hour.groupby(by=g1_hour.index.hour)[columns].mean()

fig = plt.figure(figsize=(15,12))
ax1 = fig.add_subplot(2,1,1)
ax2 = fig.add_subplot(2,1,2)

ax1.plot(gen1_h_output['AC_POWER'],label='Gen_1',lw=3)
ax1.plot(gen2_h_output['AC_POWER'],label='Gen_2',ls='--',lw=3)
ax2.plot(gen1_h_output['DC_POWER'],label='Gen_1',lw=3)
ax2.plot(gen2_h_output['DC_POWER'],label='Gen_2',ls='--',lw=3)
#D6E681
#63C7B2
ax1.set_xticks(range(0,24,2))
ax2.set_xticks(range(0,24,2))

ax2.tick_params(axis='y',left=False,labelleft=False)

ax1.legend(loc='upper left')
ax2.legend(loc='upper left')

#ax1.set_facecolor("black")
#ax2.set_facecolor("black")

ax1.set_title('AC_POWER')
ax2.set_title('DC_POWER')
plt.show()
```



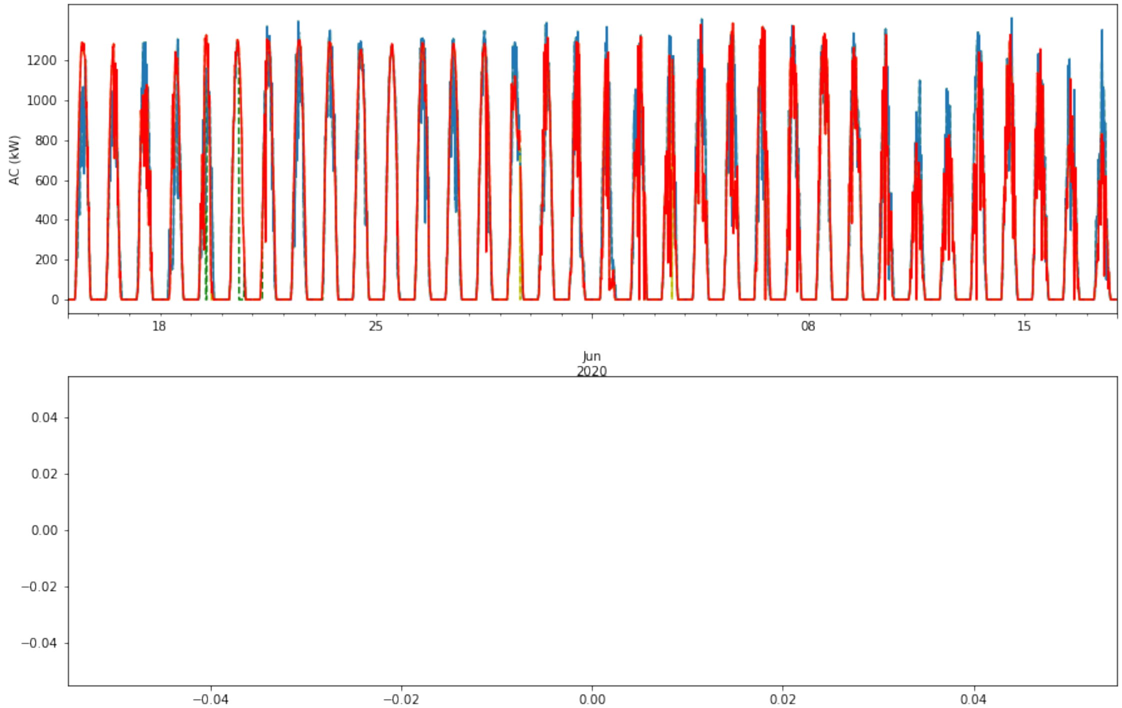


Here we see a comparison between gen_1 and gen_2, what strikes me as interesting is that both lines have a similar rise and fall in the morning and evening. it is between the hours of 8:30 and 15:30 that gen_2's output is insufficient. I wonder if during this time gen_2s solar panels are obstructed reducing sun exposer.

[214]:
```
g1_t15= gen_1.groupby(pd.Grouper(freq='15T',key='DATE_TIME'))
g2_t15= clean_g2.groupby(pd.Grouper(freq='15T',key='DATE_TIME'))

gen1_t15_ac = g1_t15['AC_POWER'].max()
gen1_t15_dc = g1_t15['DC_POWER'].max()

gen2_t15_ac = g2_t15['AC_POWER'].max()
gen2_t15_dc = g2_t15['DC_POWER'].max()

g1_day_ac = gen1_t15_ac[(gen1_t15_ac.index >='15-05-2020 00:00')&(gen1_t15_ac.
 ↪index <'2020-06-17 23:45:00')]
g1_day_ac_smoothed = g1_day_ac.fillna(0)
```

```
#g1_day_dc = gen1_t15_dc[(gen1_t15_dc.index >='15-05-2020 00:00')&(gen1_t15_dc.
↪index <'17-05-2020 23:45')]
#g1_day_dc_smoothed = g1_day_dc.fillna(0)

g2_day_ac = gen2_t15_ac[(gen1_t15_ac.index >='15-05-2020 00:00')&(gen1_t15_ac.
↪index <'2020-06-17 23:45:00')]
g2_day_ac_smoothed = g2_day_ac.fillna(0)

fig, axes = plt.subplots(2,1)


ax1 = g1_day_ac_smoothed.plot(ax=axes[0],figsize=(15,10),c='g',ls='--')
ax1 = g1_day_ac.plot(ax=axes[0],figsize=(15,10))

ax2 = g2_day_ac_smoothed.plot(ax=axes[0],figsize=(15,10),c='y',ls='--')
ax2 = g2_day_ac.plot(ax=axes[0],figsize=(15,10),c='r')
plt.plot()

ax2.set_yticks([0,200,400,600,800,1000,1200])

ax1.set_ylabel('AC (kW)')
ax2.set_ylabel('AC (kW)')
ax1.set_xlabel('')
#ax2 = g1_day_dc_smoothed.plot(ax=axes[1],figsize=(15,10),c='g',ls='--')
#ax2 = g1_day_dc.plot(ax=axes[1],figsize=(15,10))
```

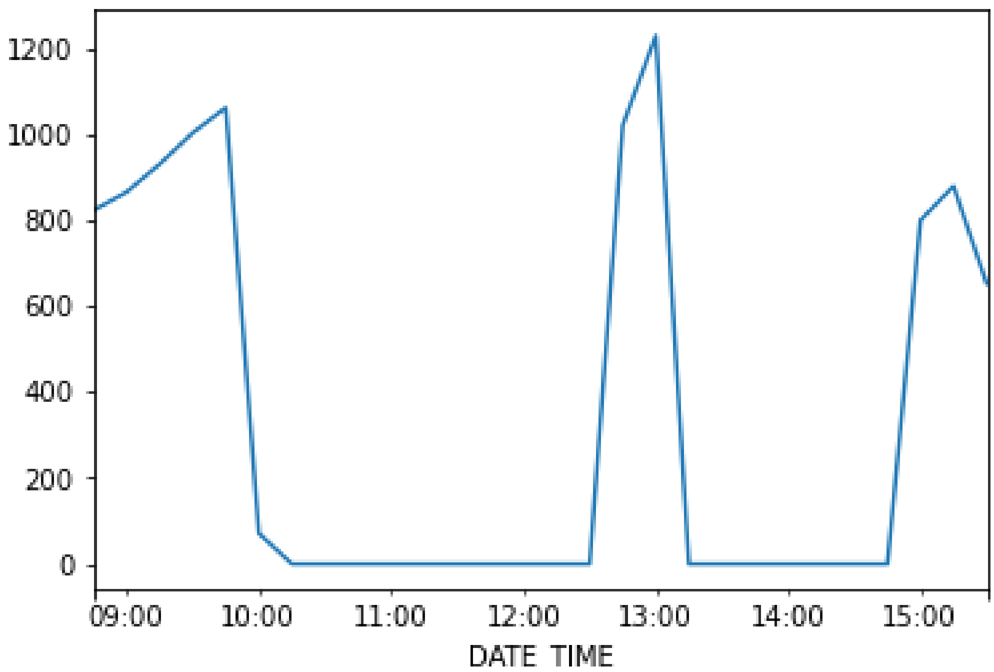
[214]: Text(0.5, 0, '')



[205]:
```
max_ac1= gen_1.groupby([pd.Grouper(freq='1d',key='DATE_TIME')])['AC_POWER'].max()
max_ac2= clean_g2.groupby([pd.Grouper(freq='1d',key='DATE_TIME')])['AC_POWER'].max()
```

[254]:
```
g1_peakrange = (clean_g2['DATE_TIME']>'2020-05-15 08:30:00')&(clean_g2['DATE_TIME']<'2020-05-15 15:45:00')
g1_peakrange = clean_g2[g1_peakrange]

list_inverters = ['4UPUqMRk7TRMgml','rq4fwE8jgrTyWY','V94E5Ben1TlhnDV','q49J1IKaHRwDQnt','LYwnQax7tkwH5Cb']

mask = g1_peakrange['INVERTER_ID']==list_inverters[0]
g1_peakrange= g1_peakrange[mask]
g1_peakrange.index=g1_peakrange['DATE_TIME']
g1_peakrange=g1_peakrange.drop(columns='DATE_TIME')
g1_peakrange['AC_POWER'].plot()
```

[254]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x23662686148>
```


1200
1000
800
600
400
200
0
09:00
10:00
11:00
12:00
13:00
14:00
15:00
DATE_TIME

[ ]: