Engineering Analytics and Machine Learning Lab 4

for Specialist Diploma in Internet of Things

Author's Name: Teo Kok Keong

1 Normalization

Lets try range scaling on the DAILYDATA_S24_201801.csv weather data. We are trying to find whether DailyRainfallTotal have any relationship with MeanTemperature. Before we even do anything, lets see how is the raw data like.

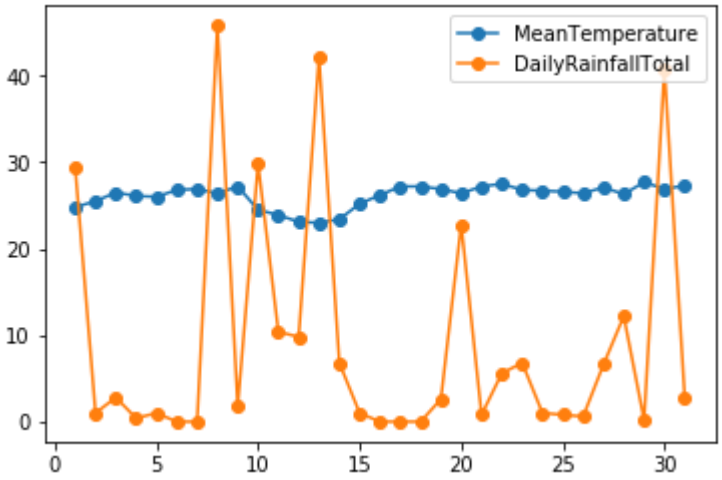
In [95]:
```
#import library required
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [96]:
```
df=pd.read_csv('DAILYDATA_S24_201801.csv')  #load the data
df.head()   #have a preview of the data
```

Out[96]:

	Station	Year	Month	Day	DailyRainfallTotal	Highest30MinRainfall	Highest60MinRain
0	Changi	2018	1	1	29.4	6.0	11.6
1	Changi	2018	1	2	1.0	0.4	0.4
2	Changi	2018	1	3	2.8	1.8	2.0
3	Changi	2018	1	4	0.4	0.2	0.2
4	Changi	2018	1	5	1.0	1.0	1.0

In [97]:
```
#let's plot scatter plot of both DailyRainfallTotal with MeanTemperature vs Da
y on the same graph
plt.plot(df.Day, df.MeanTemperature,'-o')
plt.plot(df.Day, df.DailyRainfallTotal,'-o')
plt.legend()
plt.show()
```



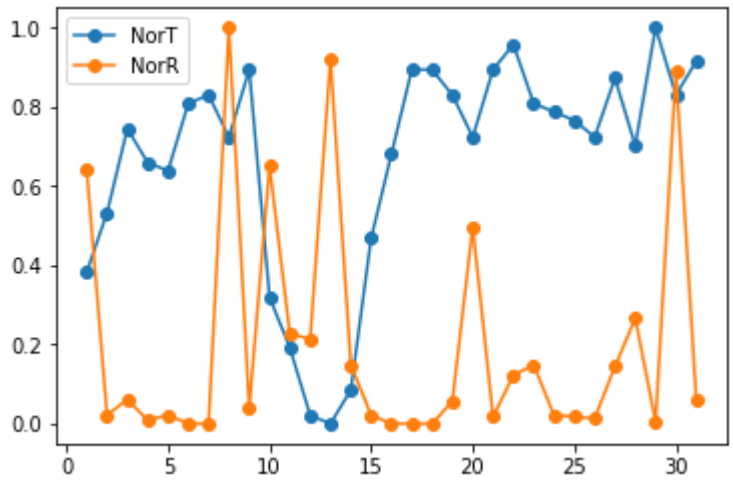
From the graph we can see that there is not much change in the mean temperature, it is nearly a straight line. We are unablet o observe any obvious relationship between the two parameters. This is because we are comparing two parameters of different scale, which is comparing apple with orange. Daily Rainfall total value varied from 0 to as high as more than 40. However, temperature only varied between the narrow band of maybe 25 to 31 or 32.

In [98]:
```
minT=min(df.MeanTemperature)   #find the min of MeanTemperature
maxT=max(df.MeanTemperature)   #find the max of MeanTemperature
df['NorT']=(df.MeanTemperature-minT)/(maxT-minT)   #Scale to 0 t 1 of Mean Tem
perature
#Scale to 0 to 1 for Daily Rainfall Total
df['NorR']=(df.DailyRainfallTotal-min(df.DailyRainfallTotal))/(max(df.DailyRai
nfallTotal)-min(df.DailyRainfallTotal))
df.head() #Preview the data again
```

Out[98]:

	Station	Year	Month	Day	DailyRainfallTotal	Highest30MinRainfall	Highest60MinRain
0	Changi	2018	1	1	29.4	6.0	11.6
1	Changi	2018	1	2	1.0	0.4	0.4
2	Changi	2018	1	3	2.8	1.8	2.0
3	Changi	2018	1	4	0.4	0.2	0.2
4	Changi	2018	1	5	1.0	1.0	1.0

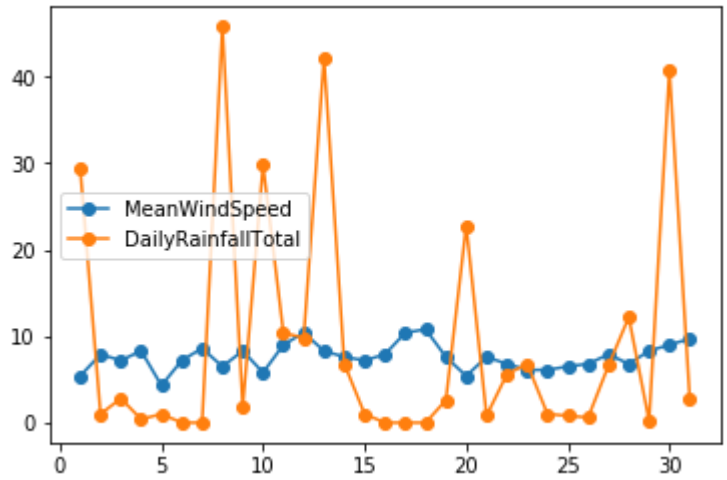
In [99]:
```
#let's plot scatter plot of both rescaled DailyRainfallTotal with MeanTemperat
ure vs Day on the same graph
plt.plot(df.Day, df.NorT,'-o')
plt.plot(df.Day, df.NorR,'-o')
plt.legend()
plt.show()
```



The relationshi become obvious for observation after rescale both data to 0 to 1.

Exercise 1

In [100]:
```
#let's plot scatter plot of both DailyRainfallTotal with MeanTemperature vs Da
y on the same graph
plt.plot(df.Day, df.MeanWindSpeed,'-o')
plt.plot(df.Day, df.DailyRainfallTotal,'-o')
plt.legend()
plt.show()
```

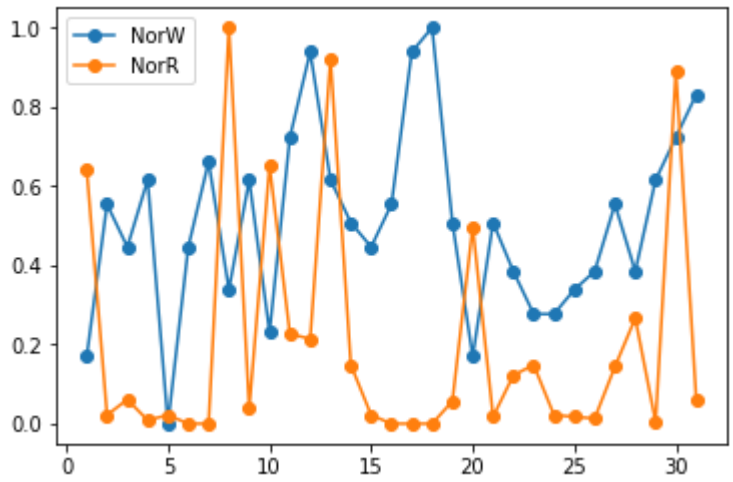


In [101]:
```
#Scale to 0 to 1 for Daily Rainfall Total
df['NorW']=(df.MeanWindSpeed-min(df.MeanWindSpeed))/(max(df.MeanWindSpeed)-min
(df.MeanWindSpeed))
df.head() #Preview the data again
```

Out[101]:

	Station	Year	Month	Day	DailyRainfallTotal	Highest30MinRainfall	Highest60MinRain
0	Changi	2018	1	1	29.4	6.0	11.6
1	Changi	2018	1	2	1.0	0.4	0.4
2	Changi	2018	1	3	2.8	1.8	2.0
3	Changi	2018	1	4	0.4	0.2	0.2
4	Changi	2018	1	5	1.0	1.0	1.0

In [102]:
```
#let's plot scatter plot of both rescaled DailyRainfallTotal with MeanTemperat
ure vs Day on the same graph
plt.plot(df.Day, df.NorW,'-o')
plt.plot(df.Day, df.NorR,'-o')
plt.legend()
plt.show()
```



Any comment?

Exercise 2

Try with DAILYDATA_S24_201801.csv with standardization transformation to find the relationship btween

- Total Daily Rainfall and MeanTemperature
- Total Daily Rainfall and MeanWindSpeed

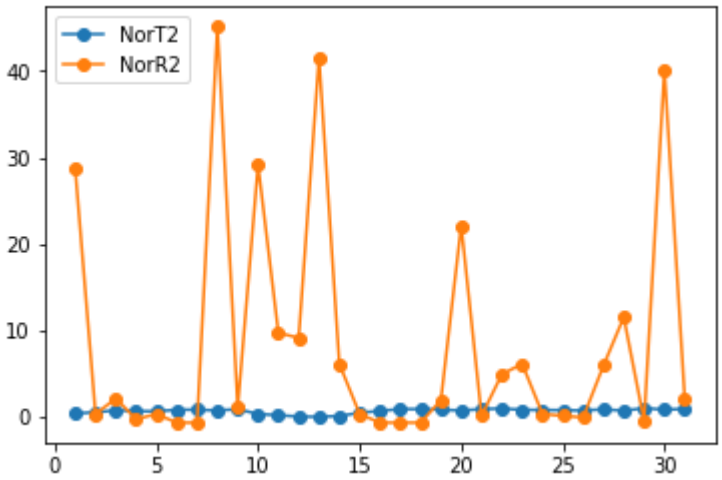
Hint: Use np.mean() and np.std() to find mean and standard deviation

In [103]:
```
meanT=np.mean(df.MeanTemperature)   #find the min of MeanTemperature
stdT=np.std(df.MeanTemperature)   #find the max of MeanTemperature
df['NorT2']=(df.MeanTemperature-minT)/(maxT-minT)   #Scale to 0 t 1 of Mean Te
mperature
#Scale to 0 to 1 for Daily Rainfall Total
df['NorR2']=df.DailyRainfallTotal-np.mean(df.DailyRainfallTotal)/np.std(df.Dai
lyRainfallTotal)
df.head() #Preview the data again
```

Out[103]:

	Station	Year	Month	Day	DailyRainfallTotal	Highest30MinRainfall	Highest60MinRain
0	Changi	2018	1	1	29.4	6.0	11.6
1	Changi	2018	1	2	1.0	0.4	0.4
2	Changi	2018	1	3	2.8	1.8	2.0
3	Changi	2018	1	4	0.4	0.2	0.2
4	Changi	2018	1	5	1.0	1.0	1.0

In [104]:
```
#let's plot scatter plot of both rescaled DailyRainfallTotal with MeanTemperat
ure vs Day on the same graph
plt.plot(df.Day, df.NorT2,'-o')
plt.plot(df.Day, df.NorR2,'-o')
plt.legend()
plt.show()
```

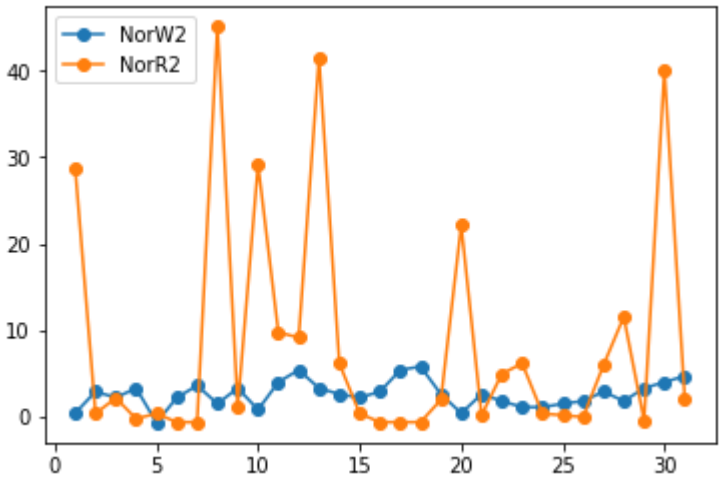


In [105]:
```
#Scale to 0 to 1 for Daily Rainfall Total
df['NorW2']=df.MeanWindSpeed-np.mean(df.MeanWindSpeed)/np.std(df.MeanWindSpeed
)
df.head() #Preview the data again
```

Out[105]:

	Station	Year	Month	Day	DailyRainfallTotal	Highest30MinRainfall	Highest60MinRain
0	Changi	2018	1	1	29.4	6.0	11.6
1	Changi	2018	1	2	1.0	0.4	0.4
2	Changi	2018	1	3	2.8	1.8	2.0
3	Changi	2018	1	4	0.4	0.2	0.2
4	Changi	2018	1	5	1.0	1.0	1.0

In [106]:
```
#Let's plot scatter plot of both rescaled DailyRainfallTotal with MeanTemperat
ure vs Day on the same graph
plt.plot(df.Day, df.NorW2,'-o')
plt.plot(df.Day, df.NorR2,'-o')
plt.legend()
plt.show()
```

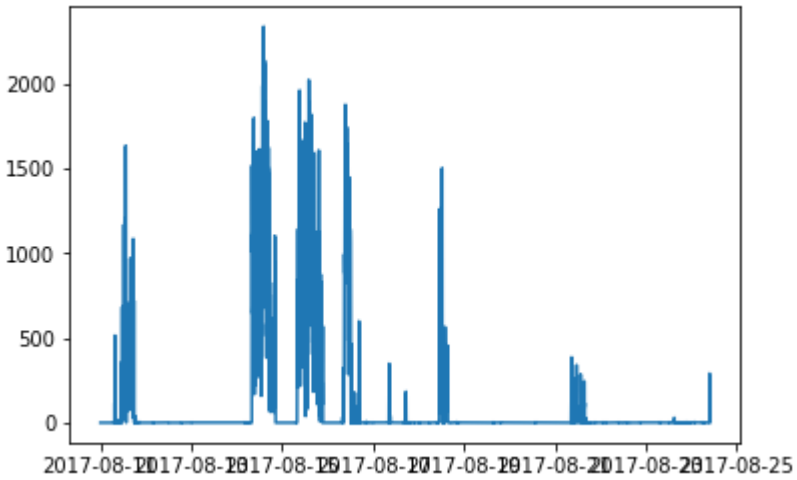


Is it as useful in this as re-scaling to 0 to 1?

2 Data Aggregration

Data aggregation is a type of data and information mining process where data is searched, gathered and presented in a report-based, summarized format to achieve specific business objectives or processes and/or conduct human analysis.

In [107]:
```
dd=pd.read_csv('remote_v3.csv')  #load the data
dd['time']=pd.to_datetime(dd['time'],format='%Y-%m-%d %H:%M:%S+08:00',utc=True
) #convert string to datetime
#tt=pd.to_datetime(dd['time'],format='%Y-%m-%d %H:%M:%S',utc=True) #convert st
ring to datetime
#dd['time']=(dd['time'].astype(np.int64)) /100000000   #convert to second
dd['onlytime']=pd.to_datetime(dd.time.dt.floor('h')).dt.time
dd.set_index('time', inplace=True)
plt.plot(dd.index,dd['value'])
plt.show()
```



In [108]:
```
dd.describe() #we could always use the describe method to understand the data
```

Out[108]:

	value
count	6441.000000
mean	89.950474
std	294.443643
min	0.000000
25%	0.000000
50%	0.000000
75%	1.000000
max	2337.000000

We load the data, convert the datetime and change to index. We also plot a graph but this tell us nothing. Since this is in a class room, we ould expect the occupancy of the venue to change according to day of week (monday, tuesday...) and timing. We do have the time but no information on the day of week. Howeverr, we have the date so we could generate a new column that indicate the day of week.

In [109]:
```
dd['dayofweek']=dd.index.dayofweek
dd.head()
```

Out[109]:

	value	onlytime	dayofweek
time			
2017-08-11 00:00:00+00:00	0.0	00:00:00	4
2017-08-11 00:03:00+00:00	0.0	00:00:00	4
2017-08-11 00:06:00+00:00	0.0	00:00:00	4
2017-08-11 00:09:00+00:00	0.0	00:00:00	4
2017-08-11 00:12:00+00:00	0.0	00:00:00	4

In [110]:
```
#this section only to illustration some aggregation method available
print("result of sum:")
print(dd.sum()) #sum  accordingly to column
print("result of mean:")
print(dd.mean())  #mean  accordingly to colum

#by default aggregation return results within each column
#by specifying the axis argument, we can instead aggregate within each row:

print(dd.mean(axis='columns').head()) #in this case does not make sense but we
 are only illustrating the feature
```

```
result of sum:
value        579371.0
dayofweek     19323.0
dtype: float64
result of mean:
value        89.950474
dayofweek     3.000000
dtype: float64
time
2017-08-11 00:00:00+00:00    2.0
2017-08-11 00:03:00+00:00    2.0
2017-08-11 00:06:00+00:00    2.0
2017-08-11 00:09:00+00:00    2.0
2017-08-11 00:12:00+00:00    2.0
dtype: float64
```

So far we have summary the data by rows or columns, most of the time we need to aggregate with GroupBy to find insight into the data.

Back to analysising the remote eye data. We probably want to groupby dayofweek and apply mean operation.

In [111]:
```
#lets group by day of  week and view it statistical properties
dd.groupby('dayofweek').describe()
```

Out[111]:

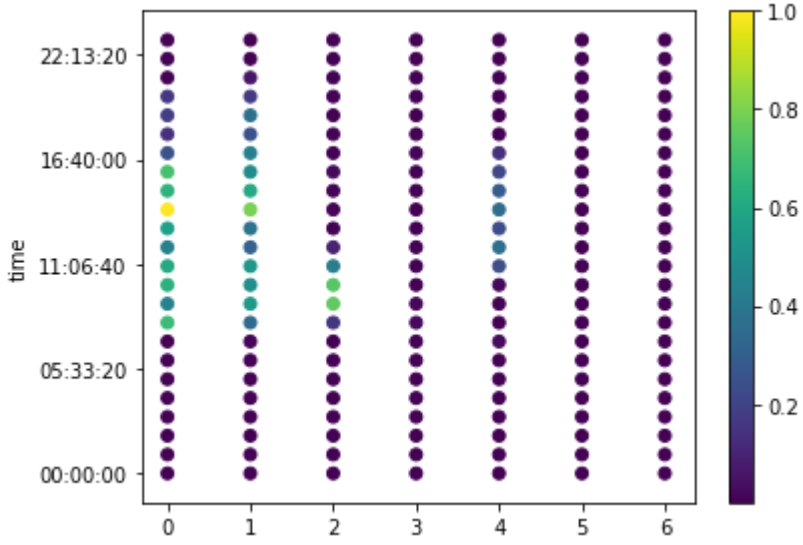
	value							
	count	**mean**	**std**	**min**	**25%**	**50%**	**75%**	**max**
dayofweek								
0	960.0	238.586458	478.290875	0.0	0.0	0.0	161.00	2337.0
1	960.0	211.870833	414.311167	0.0	0.0	0.0	188.25	2022.0
2	960.0	81.553125	300.554526	0.0	0.0	0.0	1.00	1876.0
3	681.0	2.819383	22.476418	0.0	0.0	0.0	1.00	348.0
4	960.0	69.116667	179.080594	0.0	0.0	0.0	2.00	1634.0
5	960.0	0.198958	0.455529	0.0	0.0	0.0	0.00	3.0
6	960.0	0.185417	0.436829	0.0	0.0	0.0	0.00	3.0

In [112]:
```
#we would perform aggregate on the data
gdd=dd.groupby(['dayofweek','onlytime']).aggregate([np.mean, min])
print(gdd)
print(gdd.loc[0])
```

		value	
		mean	min
dayofweek	onlytime		
0	00:00:00	0.250	0.0
	01:00:00	0.275	0.0
	02:00:00	0.200	0.0
	03:00:00	0.125	0.0
	04:00:00	0.275	0.0
	05:00:00	0.175	0.0
	06:00:00	0.350	0.0
	07:00:00	0.075	0.0
	08:00:00	602.750	0.0
	09:00:00	396.775	0.0
	10:00:00	567.500	0.0
	11:00:00	536.375	0.0
	12:00:00	385.825	0.0
	13:00:00	505.800	0.0
	14:00:00	885.925	0.0
	15:00:00	569.700	0.0
	16:00:00	626.900	0.0
	17:00:00	224.875	0.0
	18:00:00	123.200	0.0
	19:00:00	158.975	0.0
	20:00:00	138.975	0.0
	21:00:00	0.500	0.0
	22:00:00	0.050	0.0
	23:00:00	0.225	0.0
1	00:00:00	0.025	0.0
	01:00:00	0.400	0.0
	02:00:00	0.075	0.0
	03:00:00	0.250	0.0
	04:00:00	0.025	0.0
	05:00:00	0.325	0.0
...		...	...
5	18:00:00	0.200	0.0
	19:00:00	0.425	0.0
	20:00:00	0.125	0.0
	21:00:00	0.125	0.0
	22:00:00	0.050	0.0
	23:00:00	0.325	0.0
6	00:00:00	0.175	0.0
	01:00:00	0.225	0.0
	02:00:00	0.125	0.0
	03:00:00	0.300	0.0
	04:00:00	0.100	0.0
	05:00:00	0.175	0.0
	06:00:00	0.050	0.0
	07:00:00	0.200	0.0
	08:00:00	0.175	0.0
	09:00:00	0.150	0.0
	10:00:00	0.075	0.0
	11:00:00	0.275	0.0
	12:00:00	0.200	0.0
	13:00:00	0.350	0.0
	14:00:00	0.075	0.0
	15:00:00	0.375	0.0
	16:00:00	0.125	0.0

```
          17:00:00    0.175  0.0
          18:00:00    0.150  0.0
          19:00:00    0.250  0.0
          20:00:00    0.225  0.0
          21:00:00    0.200  0.0
          22:00:00    0.100  0.0
          23:00:00    0.200  0.0

[168 rows x 2 columns]
            value
             mean  min
onlytime
00:00:00    0.250  0.0
01:00:00    0.275  0.0
02:00:00    0.200  0.0
03:00:00    0.125  0.0
04:00:00    0.275  0.0
05:00:00    0.175  0.0
06:00:00    0.350  0.0
07:00:00    0.075  0.0
08:00:00  602.750  0.0
09:00:00  396.775  0.0
10:00:00  567.500  0.0
11:00:00  536.375  0.0
12:00:00  385.825  0.0
13:00:00  505.800  0.0
14:00:00  885.925  0.0
15:00:00  569.700  0.0
16:00:00  626.900  0.0
17:00:00  224.875  0.0
18:00:00  123.200  0.0
19:00:00  158.975  0.0
20:00:00  138.975  0.0
21:00:00    0.500  0.0
22:00:00    0.050  0.0
23:00:00    0.225  0.0
```

In [113]:
```
#we would perform aggregate on the data
gdd=dd.groupby(['dayofweek','onlytime']).aggregate([np.mean])

#find the max mean value to normalize the data for comparision
bb=[]

for i in range(7):
    b=max(gdd.loc[i,'value']['mean'])
    bb.append(b)

bb_max=max(bb)

fig, ax = plt.subplots()
ll=[0]*len(gdd.xs(0))

for i in range(7):
    ll=[i]*len(gdd.xs(i))
    ax=plt.scatter(ll,gdd.xs(i).index,c=gdd.loc[i,'value']['mean']/bb_max,vmax
=1)

plt.colorbar()
fig.tight_layout()
plt.show()
```

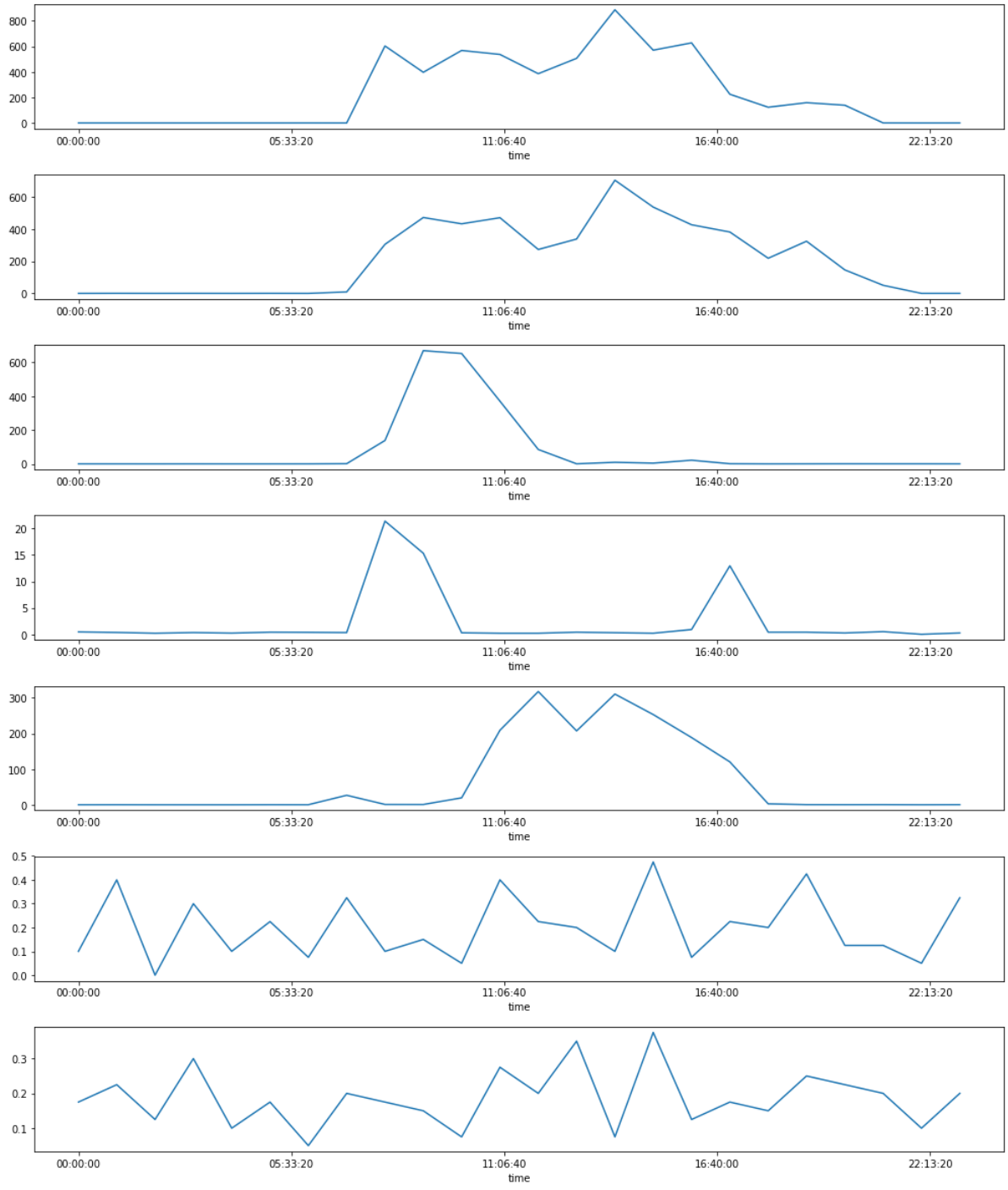


List of aggregate built-in statistic functions

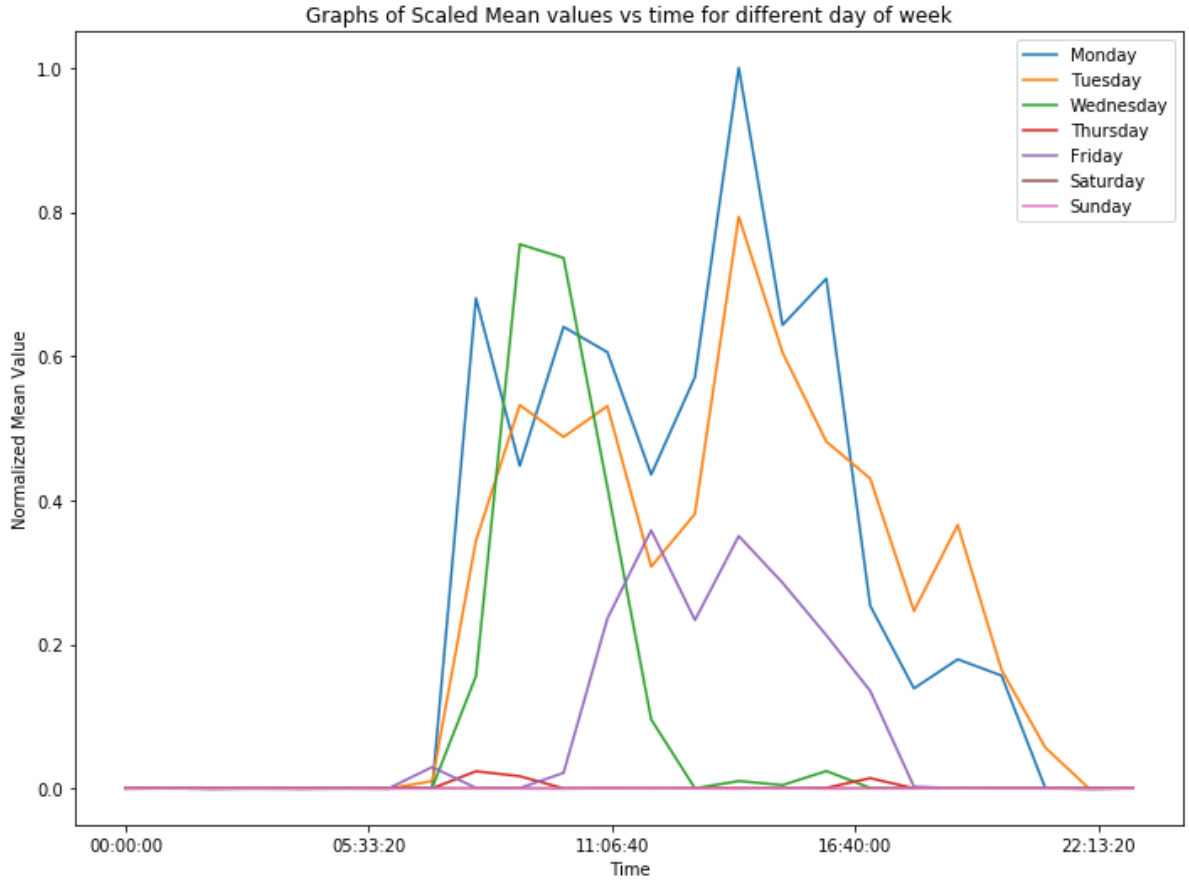
Function	Description
count	Number of non-null observations
sum	Sum of values
mean	Mean of values
mad	Mean absolute deviation
median	Arithmetic median of values
min	Minimum
max	Maximum
mode	Mode
abs	Absolute Value
prod	Product of values
std	Unbiased standard deviation
var	Unbiased variance
sem	Unbiased standard error of the mean
skew	Unbiased skewness (3rd moment)
kurt	Unbiased kurtosis (4th moment)
quantile	Sample quantile (value at %)
cumsum	Cumulative sum
cumprod	Cumulative product
cummax	Cumulative maximum
cummin	Cumulative minimum

In [114]:
```
plt.subplot(711)
fig = plt.gcf()
fig.set_size_inches(14, 16.5)

# equivalent but more general
axx=plt.subplot(7, 1, 1)


axx.plot(gdd.loc[0].index,gdd.loc[0,'value']['mean'])
axx2=plt.subplot(7,1, 2)
axx2.plot(gdd.loc[1].index,gdd.loc[1,'value']['mean'])
axx3=plt.subplot(7,1, 3)
axx3.plot(gdd.loc[2].index,gdd.loc[2,'value']['mean'])
axx4=plt.subplot(7,1, 4)
axx4.plot(gdd.loc[3].index,gdd.loc[3,'value']['mean'])
axx5=plt.subplot( 7,1, 5)
axx5.plot(gdd.loc[4].index,gdd.loc[4,'value']['mean'])
axx6=plt.subplot(7,1, 6)
axx6.plot(gdd.loc[5].index,gdd.loc[5,'value']['mean'])
axx7=plt.subplot(7,1, 7)
axx7.plot(gdd.loc[6].index,gdd.loc[6,'value']['mean'])
plt.tight_layout()
plt.show()
```

```
c:\users\teokk\appdata\local\programs\python\python35\lib\site-packages\matpl
otlib\cbook\deprecation.py:107: MatplotlibDeprecationWarning: Adding an axes
using the same arguments as a previous axes currently reuses the earlier inst
ance.  In a future version, a new instance will always be created and returne
d.  Meanwhile, this warning can be suppressed, and the future behavior ensure
d, by passing a unique label to each axes instance.
  warnings.warn(message, mplDeprecation, stacklevel=1)
```



In [115]:
```
bb=[]
for i in range(7):
    b=max(gdd.loc[i,'value']['mean'])
    bb.append(b)

bb_max=max(bb)

#plt.subplot(711)
fig = plt.gcf()
fig.set_size_inches(10, 7.5)

# equivalent but more general
#axx=plt.subplot(7, 1, 1)


plt.plot(gdd.loc[0].index,gdd.loc[0,'value']['mean']/bb_max,label='Monday')
#axx2=plt.subplot(7,1, 2)
plt.plot(gdd.loc[1].index,gdd.loc[1,'value']['mean']/bb_max,label='Tuesday')
#axx3=plt.subplot(7,1, 3)
plt.plot(gdd.loc[2].index,gdd.loc[2,'value']['mean']/bb_max,label='Wednesday')
#axx4=plt.subplot(7,1, 4)
plt.plot(gdd.loc[3].index,gdd.loc[3,'value']['mean']/bb_max,label='Thursday')
#axx5=plt.subplot( 7,1, 5)
plt.plot(gdd.loc[4].index,gdd.loc[4,'value']['mean']/bb_max,label='Friday')
#axx6=plt.subplot(7,1, 6)
plt.plot(gdd.loc[5].index,gdd.loc[5,'value']['mean']/bb_max,label='Saturday')
#axx7=plt.subplot(7,1, 7)
plt.plot(gdd.loc[6].index,gdd.loc[6,'value']['mean']/bb_max,label='Sunday')
plt.xlabel('Time')
plt.ylabel('Normalized Mean Value')
plt.title('Graphs of Scaled Mean values vs time for different day of week')
plt.legend()
plt.tight_layout()
plt.show()
```



Exercise 3

Load the iris dataset from sklearn (code would be provided below), the data set would be saved in variable data. Thare are two sets of data, data.data (data itself) and data.target (the bred code).

- use data.data and data.target to found one dataframe
- Compute mean, max, min of each parameter for each bred type (0,1,2,). Which aggregated value would be more suitable to classified the data according to the target.

In [116]:
```
from sklearn.datasets import load_iris


data=load_iris()
targetname=['Setosa','Versicolour','Virginica']

df=pd.DataFrame(data.data)

df.columns=['sepallen','sepalwidth','pedallen','pedalwidth']
df.head()
```

Out[116]:

	sepallen	sepalwidth	pedallen	pedalwidth
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [117]:
```
df['target']=data.target
df.head()
```

Out[117]:

	sepallen	sepalwidth	pedallen	pedalwidth	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [118]:
```
dfg=df.groupby(['target']).aggregate([np.mean,max,min])
dfg.head()
```

Out[118]:

	sepallen			sepalwidth			pedallen			pedalwidth		
	mean	**max**	**min**	**mean**	**max**	**min**	**mean**	**max**	**min**	**mean**	**max**	**min**
target												
0	5.006	5.8	4.3	3.418	4.4	2.3	1.464	1.9	1.0	0.244	0.6	0.1
1	5.936	7.0	4.9	2.770	3.4	2.0	4.260	5.1	3.0	1.326	1.8	1.0
2	6.588	7.9	4.9	2.974	3.8	2.2	5.552	6.9	4.5	2.026	2.5	1.4

Exercise 4

oad the amazonaws credit data from this site https://rodeo-tutorials.s3.amazonaws.com/data/credit-data-non-null.csv (https://rodeo-tutorials.s3.amazonaws.com/data/credit-data-non-null.csv)

- Slice a subset of data that consist of 'serious_dlqin2yrs', 'age', 'monthly_income'
- Compute the mean parameters for each serious_dlqin2yrs category of the subset

In [119]:
```
df4 = pd.read_csv("https://rodeo-tutorials.s3.amazonaws.com/data/credit-data-non-null.csv")
df4.head()
```

Out[119]:

	serious_dlqin2yrs	revolving_utilization_of_unsecured_lines	age	number_ 59_days_past_due_
0	1	0.766127	45	2
1	0	0.957151	40	0
2	0	0.658180	38	1
3	0	0.233810	30	0
4	0	0.907239	49	1

In [120]:
```
sub=df4[['serious_dlqin2yrs', 'age', 'monthly_income']]
sub.head()
```

Out[120]:

	serious_dlqin2yrs	age	monthly_income
0	1	45	9120.0
1	0	40	2600.0
2	0	38	3042.0
3	0	30	3300.0
4	0	49	63588.0

In [121]:
```
sub.groupby("serious_dlqin2yrs").aggregate([np.mean])
```

Out[121]:

	age	monthly_income
	mean	**mean**
serious_dlqin2yrs		
0	52.751375	5473.758555
1	45.926591	4746.613006

In [122]:
```
sub.groupby("age").aggregate([np.mean])
```

Out[122]:

	serious_dlqin2yrs	monthly_income
	mean	mean
age		
0	0.000000	6000.000000
21	0.071038	848.972678
22	0.082949	1079.919355
23	0.109204	1441.219969
24	0.120098	1758.556373
25	0.126967	2233.391396
26	0.123219	2468.860855
27	0.124066	2757.591181
28	0.131410	3145.467949
29	0.105170	3454.682726
30	0.107899	3741.888487
31	0.106477	4100.286555
32	0.113659	4174.910732
33	0.109870	4627.940598
34	0.097448	4728.585151
35	0.107302	4982.472841
36	0.099622	5001.846995
37	0.090044	5264.909163
38	0.089320	5314.164956
39	0.093740	5484.727151
40	0.085354	5545.587132
41	0.094170	5730.400064
42	0.093770	5818.328358
43	0.086035	6148.702930
44	0.073467	7264.474196
45	0.081097	6106.648772
46	0.087237	6011.406031
47	0.082280	6227.561710
48	0.075145	6062.616658
49	0.081574	6481.098775

	serious_dlqin2yrs	monthly_income
	mean	mean
age		
...	...	...
76	0.020287	4040.559594
77	0.016379	3860.838035
78	0.022770	3943.656546
79	0.022426	4084.933741
80	0.021689	3936.068493
81	0.011628	3754.430233
82	0.029366	4204.760433
83	0.019531	4892.496094
84	0.016667	3292.770833
85	0.018634	4132.331263
86	0.014742	3987.565111
87	0.022409	3753.535014
88	0.025559	3385.990415
89	0.032609	3154.525362
90	0.015152	3564.611111
91	0.032468	3457.383117
92	0.000000	3218.924731
93	0.011494	3563.873563
94	0.021277	6523.914894
95	0.022222	2462.555556
96	0.000000	4052.111111
97	0.000000	2497.470588
98	0.000000	2009.666667
99	0.222222	4652.555556
101	0.333333	1516.333333
102	0.000000	3339.000000
103	0.000000	1672.666667
105	0.000000	1600.000000
107	0.000000	0.000000

	serious_dlqin2yrs	monthly_income
	mean	mean
age		
109	0.000000	0.000000

86 rows × 2 columns

3 Filter data

A data frames columns can be queried with a boolean expression. Every frame has the module query() as one of its objects members.

We start by importing pandas, numpy and creating a dataframe:

In [123]:
```
import pandas as pd
import numpy as np

data = {'name': ['Alice', 'Bob', 'Charles', 'David', 'Eric'],
        'year': ['01-01-2017', '02-12-2017', '03-03-2017', '23-04-2017', '30-0
3-2017'],
        'salary': [40000, 24000, 31000, 20000, 30000]}

df = pd.DataFrame(data)
df.year=pd.to_datetime(df['year'],format='%d-%m-%Y',utc=True)
print(df)
print(df.info())
```

```
      name  salary                      year
0    Alice   40000 2017-01-01 00:00:00+00:00
1      Bob   24000 2017-12-02 00:00:00+00:00
2  Charles   31000 2017-03-03 00:00:00+00:00
3    David   20000 2017-04-23 00:00:00+00:00
4     Eric   30000 2017-03-30 00:00:00+00:00
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 3 columns):
name      5 non-null object
salary    5 non-null int64
year      5 non-null datetime64[ns, UTC]
dtypes: datetime64[ns, UTC](1), int64(1), object(1)
memory usage: 200.0+ bytes
None
```

Filter by Query

A data frames columns can be queried with a boolean expression. Every frame has the module query() as one of its objects members.

We start by importing pandas, numpy and creating a dataframe:

In [124]:
```
df_filtered = df.query('salary>30000')
print(df_filtered)
```

```
      name  salary                       year
0    Alice   40000 2017-01-01 00:00:00+00:00
2  Charles   31000 2017-03-03 00:00:00+00:00
```

In [125]:
```
df_filtered = df.query('salary<=30000')
print(df_filtered)
```

```
    name  salary                       year
1    Bob   24000 2017-12-02 00:00:00+00:00
3  David   20000 2017-04-23 00:00:00+00:00
4   Eric   30000 2017-03-30 00:00:00+00:00
```

In [126]:
```
df_filtered = df.query('salary==30000')
print(df_filtered)
```

```
   name  salary                       year
4  Eric   30000 2017-03-30 00:00:00+00:00
```

In [127]:
```
df_filtered = df.query('salary>=2000 and salary<=30000')
print(df_filtered)
```

```
    name  salary                       year
1    Bob   24000 2017-12-02 00:00:00+00:00
3  David   20000 2017-04-23 00:00:00+00:00
4   Eric   30000 2017-03-30 00:00:00+00:00
```

Filter by indexing

In [128]:
```
df_filtered = df[(df.salary >= 30000) & (df.year >'01-01-2017')]
print(df_filtered)
```

```
      name  salary                       year
2  Charles   31000 2017-03-03 00:00:00+00:00
4     Eric   30000 2017-03-30 00:00:00+00:00
```

In [129]:
```
df_filtered = df[(df.year >= '01-01-2007') & (df.year > '01-06-2017')]
print(df_filtered)
```

```
      name  salary                      year
1      Bob   24000 2017-12-02 00:00:00+00:00
2  Charles   31000 2017-03-03 00:00:00+00:00
3    David   20000 2017-04-23 00:00:00+00:00
4     Eric   30000 2017-03-30 00:00:00+00:00
```

Filter by Pandas Groupby

In [130]:
```
df1 = pd.DataFrame( {
    "Name" : ["Alice", "Ada", "Mallory", "Mallory", "Billy" , "Mallory"] ,
    "City" : ["Sydney", "Sydney", "Paris", "Sydney", "Sydney", "Paris"]} )
print(df1)
```

```
     City     Name
0  Sydney    Alice
1  Sydney      Ada
2   Paris  Mallory
3  Sydney  Mallory
4  Sydney    Billy
5   Paris  Mallory
```

In [131]:
```
print(df1.groupby(["City"])[['Name']].count())
```

```
        Name
City
Paris      2
Sydney     4
```

Exercise 5

Load the bike sharing hourly data file bike_sharing_hourly.xlsx file. Please filter to obtain data frame that:

- contain only on holiday
- contain only season 1 and 3
- contain only data from 2011-01-01 to 2011-010-05

Note: Below contact code to load excel file to Pandas dataframe Note: Install xlrd package for excel support, on anaconda use command "command conda install xlrd" for installation withhout virtual environment use pip with command "pip install xlrd

In [132]:
```
file='bike_sharing_hourly.xlsx'
dfx = pd.read_excel(file, sheet_name='bike_sharing_hourly')
print(dfx.head())
print(dfx.info())
```

```
   instant     dteday  season  yr  mnth  hr  holiday  weekday  workingday  \
0        1 2011-01-01       1   0     1   0        0        6           0
1        2 2011-01-01       1   0     1   1        0        6           0
2        3 2011-01-01       1   0     1   2        0        6           0
3        4 2011-01-01       1   0     1   3        0        6           0
4        5 2011-01-01       1   0     1   4        0        6           0

   weathersit  temp   atemp   hum  windspeed  casual  registered  cnt
0           1  0.24  0.2879  0.81        0.0       3          13   16
1           1  0.22  0.2727  0.80        0.0       8          32   40
2           1  0.22  0.2727  0.80        0.0       5          27   32
3           1  0.24  0.2879  0.75        0.0       3          10   13
4           1  0.24  0.2879  0.75        0.0       0           1    1
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17379 entries, 0 to 17378
Data columns (total 17 columns):
instant       17379 non-null int64
dteday        17379 non-null datetime64[ns]
season        17379 non-null int64
yr            17379 non-null int64
mnth          17379 non-null int64
hr            17379 non-null int64
holiday       17379 non-null int64
weekday       17379 non-null int64
workingday    17379 non-null int64
weathersit    17379 non-null int64
temp          17379 non-null float64
atemp         17379 non-null float64
hum           17379 non-null float64
windspeed     17379 non-null float64
casual        17379 non-null int64
registered    17379 non-null int64
cnt           17379 non-null int64
dtypes: datetime64[ns](1), float64(4), int64(12)
memory usage: 2.3 MB
None
```

In [133]:
```
filtered=dfx.query('holiday==1')
print(filtered)
```

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday
372	373	2011-01-17	1	0	1	0	1	1	0
373	374	2011-01-17	1	0	1	1	1	1	0
374	375	2011-01-17	1	0	1	2	1	1	0
375	376	2011-01-17	1	0	1	3	1	1	0
376	377	2011-01-17	1	0	1	4	1	1	0
377	378	2011-01-17	1	0	1	5	1	1	0
378	379	2011-01-17	1	0	1	6	1	1	0
379	380	2011-01-17	1	0	1	7	1	1	0
380	381	2011-01-17	1	0	1	8	1	1	0
381	382	2011-01-17	1	0	1	9	1	1	0
382	383	2011-01-17	1	0	1	10	1	1	0
383	384	2011-01-17	1	0	1	11	1	1	0
384	385	2011-01-17	1	0	1	12	1	1	0
385	386	2011-01-17	1	0	1	13	1	1	0
386	387	2011-01-17	1	0	1	14	1	1	0
387	388	2011-01-17	1	0	1	15	1	1	0
388	389	2011-01-17	1	0	1	16	1	1	0
389	390	2011-01-17	1	0	1	17	1	1	0
390	391	2011-01-17	1	0	1	18	1	1	0
391	392	2011-01-17	1	0	1	19	1	1	0
392	393	2011-01-17	1	0	1	20	1	1	0
393	394	2011-01-17	1	0	1	21	1	1	0
394	395	2011-01-17	1	0	1	22	1	1	0
395	396	2011-01-17	1	0	1	23	1	1	0
1157	1158	2011-02-21	1	0	2	0	1	1	0
1158	1159	2011-02-21	1	0	2	1	1	1	0
1159	1160	2011-02-21	1	0	2	2	1	1	0
1160	1161	2011-02-21	1	0	2	3	1	1	0

1161	1162	2011-02-21	1	0	2	4	1	1	0
1162	1163	2011-02-21	1	0	2	5	1	1	0
...	...	...	...	..	...	..	...	...	...
16439	16440	2012-11-22	4	1	11	17	1	4	0
16440	16441	2012-11-22	4	1	11	18	1	4	0
16441	16442	2012-11-22	4	1	11	19	1	4	0
16442	16443	2012-11-22	4	1	11	20	1	4	0
16443	16444	2012-11-22	4	1	11	21	1	4	0
16444	16445	2012-11-22	4	1	11	22	1	4	0
16445	16446	2012-11-22	4	1	11	23	1	4	0
17212	17213	2012-12-25	1	1	12	0	1	2	0
17213	17214	2012-12-25	1	1	12	1	1	2	0
17214	17215	2012-12-25	1	1	12	2	1	2	0
17215	17216	2012-12-25	1	1	12	4	1	2	0
17216	17217	2012-12-25	1	1	12	5	1	2	0
17217	17218	2012-12-25	1	1	12	6	1	2	0
17218	17219	2012-12-25	1	1	12	7	1	2	0
17219	17220	2012-12-25	1	1	12	8	1	2	0
17220	17221	2012-12-25	1	1	12	9	1	2	0
17221	17222	2012-12-25	1	1	12	10	1	2	0
17222	17223	2012-12-25	1	1	12	11	1	2	0
17223	17224	2012-12-25	1	1	12	12	1	2	0
17224	17225	2012-12-25	1	1	12	13	1	2	0
17225	17226	2012-12-25	1	1	12	14	1	2	0
17226	17227	2012-12-25	1	1	12	15	1	2	0
17227	17228	2012-12-25	1	1	12	16	1	2	0
17228	17229	2012-12-25	1	1	12	17	1	2	0
17229	17230	2012-12-25	1	1	12	18	1	2	0

17230	17231	2012-12-25	1	1	12	19	1	2	0	
17231	17232	2012-12-25	1	1	12	20	1	2	0	
17232	17233	2012-12-25	1	1	12	21	1	2	0	
17233	17234	2012-12-25	1	1	12	22	1	2	0	
17234	17235	2012-12-25	1	1	12	23	1	2	0	

	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
372	2	0.20	0.1970	0.47	0.2239	1	16	17
373	2	0.20	0.1970	0.44	0.1940	1	15	16
374	2	0.18	0.1667	0.43	0.2537	0	8	8
375	2	0.18	0.1818	0.43	0.1940	0	2	2
376	2	0.18	0.1970	0.43	0.1343	1	2	3
377	2	0.18	0.1970	0.43	0.1642	0	1	1
378	2	0.18	0.1818	0.43	0.1940	0	5	5
379	2	0.16	0.1818	0.50	0.1343	4	9	13
380	2	0.16	0.1515	0.47	0.2239	3	30	33
381	2	0.16	0.1515	0.47	0.2239	8	39	47
382	2	0.16	0.1515	0.50	0.2537	7	50	57
383	2	0.16	0.1515	0.55	0.1940	9	55	64
384	2	0.18	0.1970	0.47	0.1343	10	70	80
385	2	0.18	0.1970	0.47	0.1343	13	80	93
386	2	0.18	0.2121	0.43	0.1045	12	74	86
387	2	0.20	0.2121	0.47	0.1642	21	72	93
388	2	0.20	0.2121	0.47	0.1642	6	76	82
389	1	0.20	0.1970	0.51	0.1940	4	67	71
390	2	0.18	0.1667	0.55	0.2537	7	85	92
391	3	0.18	0.1818	0.59	0.1940	2	58	60
392	3	0.16	0.1515	0.80	0.1940	4	29	33
393	3	0.16	0.1515	0.80	0.1940	3	24	27
394	3	0.14	0.1212	0.93	0.2537	0	13	13
395	3	0.16	0.1364	0.86	0.2836	1	3	4
1157	2	0.34	0.3030	0.42	0.3284	7	30	37
1158	2	0.34	0.3030	0.42	0.3284	2	11	13
1159	2	0.34	0.3030	0.42	0.3284	1	3	4
1160	2	0.34	0.3030	0.42	0.2985	2	3	5
1161	1	0.32	0.3182	0.45	0.1642	1	0	1
1162	2	0.34	0.3636	0.36	0.0000	1	2	3
...	...	...	...	...	...	...	...	...
16439	1	0.44	0.4394	0.35	0.0000	66	43	109
16440	1	0.40	0.4091	0.43	0.0000	16	54	70
16441	1	0.36	0.3788	0.76	0.0000	13	31	44
16442	1	0.34	0.3636	0.71	0.0000	15	37	52
16443	1	0.34	0.3485	0.61	0.0896	7	39	46
16444	1	0.32	0.3485	0.61	0.0000	8	36	44
16445	1	0.30	0.3333	0.70	0.0000	8	28	36
17212	3	0.24	0.2576	0.93	0.0896	3	10	13
17213	2	0.26	0.2576	0.87	0.1642	0	13	13
17214	2	0.26	0.2576	0.87	0.1642	0	7	7
17215	2	0.24	0.2576	0.87	0.0896	0	1	1
17216	2	0.22	0.2273	0.93	0.1343	2	1	3
17217	2	0.22	0.2273	0.93	0.1343	1	6	7
17218	2	0.22	0.2273	0.93	0.1642	0	6	6

17219	2	0.24	0.2879	0.87	0.0000	1	10	11
17220	2	0.24	0.2576	0.87	0.0000	7	21	28
17221	1	0.28	0.3182	0.81	0.0000	11	21	32
17222	1	0.30	0.3182	0.75	0.0896	43	43	86
17223	1	0.32	0.3333	0.76	0.0896	62	52	114
17224	1	0.40	0.4091	0.50	0.3284	75	46	121
17225	1	0.38	0.3939	0.46	0.2985	58	68	126
17226	1	0.36	0.3333	0.50	0.2537	51	56	107
17227	2	0.36	0.3333	0.50	0.2537	48	38	86
17228	2	0.32	0.3030	0.57	0.2537	16	34	50
17229	2	0.32	0.3030	0.66	0.2537	20	23	43
17230	2	0.32	0.3030	0.66	0.2239	16	20	36
17231	2	0.32	0.3030	0.66	0.2836	11	29	40
17232	2	0.30	0.2879	0.65	0.1940	8	26	34
17233	2	0.30	0.3030	0.70	0.1642	3	16	19
17234	2	0.28	0.2727	0.65	0.2537	4	26	30

[500 rows x 17 columns]

In [134]:
```
filter=dfx[(dfx.season==1) | (dfx.season==3)]
print(filter)
```

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday
0	1	2011-01-01	1	0	1	0	0	6	0
1	2	2011-01-01	1	0	1	1	0	6	0
2	3	2011-01-01	1	0	1	2	0	6	0
3	4	2011-01-01	1	0	1	3	0	6	0
4	5	2011-01-01	1	0	1	4	0	6	0
5	6	2011-01-01	1	0	1	5	0	6	0
6	7	2011-01-01	1	0	1	6	0	6	0
7	8	2011-01-01	1	0	1	7	0	6	0
8	9	2011-01-01	1	0	1	8	0	6	0
9	10	2011-01-01	1	0	1	9	0	6	0
10	11	2011-01-01	1	0	1	10	0	6	0
11	12	2011-01-01	1	0	1	11	0	6	0
12	13	2011-01-01	1	0	1	12	0	6	0
13	14	2011-01-01	1	0	1	13	0	6	0
14	15	2011-01-01	1	0	1	14	0	6	0
15	16	2011-01-01	1	0	1	15	0	6	0
16	17	2011-01-01	1	0	1	16	0	6	0
17	18	2011-01-01	1	0	1	17	0	6	0
18	19	2011-01-01	1	0	1	18	0	6	0
19	20	2011-01-01	1	0	1	19	0	6	0
20	21	2011-01-01	1	0	1	20	0	6	0
21	22	2011-01-01	1	0	1	21	0	6	0
22	23	2011-01-01	1	0	1	22	0	6	0
23	24	2011-01-01	1	0	1	23	0	6	0
24	25	2011-01-02	1	0	1	0	0	0	0
25	26	2011-01-02	1	0	1	1	0	0	0
26	27	2011-01-02	1	0	1	2	0	0	0
27	28	2011-01-02	1	0	1	3	0	0	0

28	29	2011-01-02	1	0	1	4	0	0	0
29	30	2011-01-02	1	0	1	6	0	0	0
...	...	...	...	..	...	..	...	...	...
17349	17350	2012-12-30	1	1	12	18	0	0	0
17350	17351	2012-12-30	1	1	12	19	0	0	0
17351	17352	2012-12-30	1	1	12	20	0	0	0
17352	17353	2012-12-30	1	1	12	21	0	0	0
17353	17354	2012-12-30	1	1	12	22	0	0	0
17354	17355	2012-12-30	1	1	12	23	0	0	0
17355	17356	2012-12-31	1	1	12	0	0	1	1
17356	17357	2012-12-31	1	1	12	1	0	1	1
17357	17358	2012-12-31	1	1	12	2	0	1	1
17358	17359	2012-12-31	1	1	12	3	0	1	1
17359	17360	2012-12-31	1	1	12	4	0	1	1
17360	17361	2012-12-31	1	1	12	5	0	1	1
17361	17362	2012-12-31	1	1	12	6	0	1	1
17362	17363	2012-12-31	1	1	12	7	0	1	1
17363	17364	2012-12-31	1	1	12	8	0	1	1
17364	17365	2012-12-31	1	1	12	9	0	1	1
17365	17366	2012-12-31	1	1	12	10	0	1	1
17366	17367	2012-12-31	1	1	12	11	0	1	1
17367	17368	2012-12-31	1	1	12	12	0	1	1
17368	17369	2012-12-31	1	1	12	13	0	1	1
17369	17370	2012-12-31	1	1	12	14	0	1	1
17370	17371	2012-12-31	1	1	12	15	0	1	1
17371	17372	2012-12-31	1	1	12	16	0	1	1
17372	17373	2012-12-31	1	1	12	17	0	1	1
17373	17374	2012-12-31	1	1	12	18	0	1	1

17374	17375	2012-12-31	1	1	12	19	0	1	1	
17375	17376	2012-12-31	1	1	12	20	0	1	1	
17376	17377	2012-12-31	1	1	12	21	0	1	1	
17377	17378	2012-12-31	1	1	12	22	0	1	1	
17378	17379	2012-12-31	1	1	12	23	0	1	1	

	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	0.24	0.2879	0.81	0.0000	3	13	16
1	1	0.22	0.2727	0.80	0.0000	8	32	40
2	1	0.22	0.2727	0.80	0.0000	5	27	32
3	1	0.24	0.2879	0.75	0.0000	3	10	13
4	1	0.24	0.2879	0.75	0.0000	0	1	1
5	2	0.24	0.2576	0.75	0.0896	0	1	1
6	1	0.22	0.2727	0.80	0.0000	2	0	2
7	1	0.20	0.2576	0.86	0.0000	1	2	3
8	1	0.24	0.2879	0.75	0.0000	1	7	8
9	1	0.32	0.3485	0.76	0.0000	8	6	14
10	1	0.38	0.3939	0.76	0.2537	12	24	36
11	1	0.36	0.3333	0.81	0.2836	26	30	56
12	1	0.42	0.4242	0.77	0.2836	29	55	84
13	2	0.46	0.4545	0.72	0.2985	47	47	94
14	2	0.46	0.4545	0.72	0.2836	35	71	106
15	2	0.44	0.4394	0.77	0.2985	40	70	110
16	2	0.42	0.4242	0.82	0.2985	41	52	93
17	2	0.44	0.4394	0.82	0.2836	15	52	67
18	3	0.42	0.4242	0.88	0.2537	9	26	35
19	3	0.42	0.4242	0.88	0.2537	6	31	37
20	2	0.40	0.4091	0.87	0.2537	11	25	36
21	2	0.40	0.4091	0.87	0.1940	3	31	34
22	2	0.40	0.4091	0.94	0.2239	11	17	28
23	2	0.46	0.4545	0.88	0.2985	15	24	39
24	2	0.46	0.4545	0.88	0.2985	4	13	17
25	2	0.44	0.4394	0.94	0.2537	1	16	17
26	2	0.42	0.4242	1.00	0.2836	1	8	9
27	2	0.46	0.4545	0.94	0.1940	2	4	6
28	2	0.46	0.4545	0.94	0.1940	2	1	3
29	3	0.42	0.4242	0.77	0.2985	0	2	2
...	...	...	...	...	...	...	...	...
17349	2	0.24	0.2121	0.44	0.2985	12	113	125
17350	1	0.34	0.3636	0.61	0.0000	16	86	102
17351	1	0.22	0.1970	0.47	0.3284	9	63	72
17352	1	0.20	0.2121	0.51	0.1642	5	42	47
17353	1	0.20	0.1970	0.55	0.1940	6	30	36
17354	1	0.20	0.1970	0.51	0.2239	10	39	49
17355	1	0.18	0.1818	0.55	0.1940	4	30	34
17356	1	0.18	0.1818	0.55	0.1940	6	13	19
17357	1	0.16	0.1667	0.59	0.1642	3	8	11
17358	1	0.16	0.1818	0.59	0.1045	0	1	1
17359	1	0.14	0.1667	0.69	0.1045	0	3	3
17360	1	0.16	0.1515	0.64	0.1940	0	9	9
17361	1	0.16	0.1667	0.64	0.1642	0	40	40
17362	1	0.16	0.1818	0.64	0.1343	2	83	85

17363	1	0.14	0.1515	0.69	0.1343	9	187	196
17364	2	0.18	0.2121	0.64	0.1045	13	144	157
17365	2	0.20	0.2121	0.69	0.1343	33	87	120
17366	2	0.22	0.2273	0.60	0.1940	43	114	157
17367	2	0.24	0.2273	0.56	0.1940	52	172	224
17368	2	0.26	0.2576	0.44	0.1642	38	165	203
17369	2	0.28	0.2727	0.45	0.2239	62	185	247
17370	2	0.28	0.2879	0.45	0.1343	69	246	315
17371	2	0.26	0.2576	0.48	0.1940	30	184	214
17372	2	0.26	0.2879	0.48	0.0896	14	150	164
17373	2	0.26	0.2727	0.48	0.1343	10	112	122
17374	2	0.26	0.2576	0.60	0.1642	11	108	119
17375	2	0.26	0.2576	0.60	0.1642	8	81	89
17376	1	0.26	0.2576	0.60	0.1642	7	83	90
17377	1	0.26	0.2727	0.56	0.1343	13	48	61
17378	1	0.26	0.2727	0.65	0.1343	12	37	49

[8738 rows x 17 columns]

In [135]:
```
filter=dfx[(dfx.dteday>='2011-01-01') & (dfx.dteday<='2011-01-04')]
print(filter)
```

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	\
0	1	2011-01-01	1	0	1	0	0	6	0	
1	2	2011-01-01	1	0	1	1	0	6	0	
2	3	2011-01-01	1	0	1	2	0	6	0	
3	4	2011-01-01	1	0	1	3	0	6	0	
4	5	2011-01-01	1	0	1	4	0	6	0	
5	6	2011-01-01	1	0	1	5	0	6	0	
6	7	2011-01-01	1	0	1	6	0	6	0	
7	8	2011-01-01	1	0	1	7	0	6	0	
8	9	2011-01-01	1	0	1	8	0	6	0	
9	10	2011-01-01	1	0	1	9	0	6	0	
10	11	2011-01-01	1	0	1	10	0	6	0	
11	12	2011-01-01	1	0	1	11	0	6	0	
12	13	2011-01-01	1	0	1	12	0	6	0	
13	14	2011-01-01	1	0	1	13	0	6	0	
14	15	2011-01-01	1	0	1	14	0	6	0	
15	16	2011-01-01	1	0	1	15	0	6	0	
16	17	2011-01-01	1	0	1	16	0	6	0	
17	18	2011-01-01	1	0	1	17	0	6	0	
18	19	2011-01-01	1	0	1	18	0	6	0	
19	20	2011-01-01	1	0	1	19	0	6	0	
20	21	2011-01-01	1	0	1	20	0	6	0	
21	22	2011-01-01	1	0	1	21	0	6	0	
22	23	2011-01-01	1	0	1	22	0	6	0	
23	24	2011-01-01	1	0	1	23	0	6	0	
24	25	2011-01-02	1	0	1	0	0	0	0	
25	26	2011-01-02	1	0	1	1	0	0	0	
26	27	2011-01-02	1	0	1	2	0	0	0	
27	28	2011-01-02	1	0	1	3	0	0	0	
28	29	2011-01-02	1	0	1	4	0	0	0	
29	30	2011-01-02	1	0	1	6	0	0	0	
..	...	...	...	..	...	..	...	...	...	
62	63	2011-01-03	1	0	1	17	0	1	1	
63	64	2011-01-03	1	0	1	18	0	1	1	
64	65	2011-01-03	1	0	1	19	0	1	1	
65	66	2011-01-03	1	0	1	20	0	1	1	
66	67	2011-01-03	1	0	1	21	0	1	1	
67	68	2011-01-03	1	0	1	22	0	1	1	
68	69	2011-01-03	1	0	1	23	0	1	1	
69	70	2011-01-04	1	0	1	0	0	2	1	
70	71	2011-01-04	1	0	1	1	0	2	1	
71	72	2011-01-04	1	0	1	2	0	2	1	
72	73	2011-01-04	1	0	1	4	0	2	1	
73	74	2011-01-04	1	0	1	5	0	2	1	
74	75	2011-01-04	1	0	1	6	0	2	1	
75	76	2011-01-04	1	0	1	7	0	2	1	
76	77	2011-01-04	1	0	1	8	0	2	1	
77	78	2011-01-04	1	0	1	9	0	2	1	
78	79	2011-01-04	1	0	1	10	0	2	1	
79	80	2011-01-04	1	0	1	11	0	2	1	
80	81	2011-01-04	1	0	1	12	0	2	1	
81	82	2011-01-04	1	0	1	13	0	2	1	
82	83	2011-01-04	1	0	1	14	0	2	1	
83	84	2011-01-04	1	0	1	15	0	2	1	
84	85	2011-01-04	1	0	1	16	0	2	1	
85	86	2011-01-04	1	0	1	17	0	2	1	
86	87	2011-01-04	1	0	1	18	0	2	1	

87	88	2011-01-04	1	0	1	19	0	2	1
88	89	2011-01-04	1	0	1	20	0	2	1
89	90	2011-01-04	1	0	1	21	0	2	1
90	91	2011-01-04	1	0	1	22	0	2	1
91	92	2011-01-04	1	0	1	23	0	2	1

	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	0.24	0.2879	0.81	0.0000	3	13	16
1	1	0.22	0.2727	0.80	0.0000	8	32	40
2	1	0.22	0.2727	0.80	0.0000	5	27	32
3	1	0.24	0.2879	0.75	0.0000	3	10	13
4	1	0.24	0.2879	0.75	0.0000	0	1	1
5	2	0.24	0.2576	0.75	0.0896	0	1	1
6	1	0.22	0.2727	0.80	0.0000	2	0	2
7	1	0.20	0.2576	0.86	0.0000	1	2	3
8	1	0.24	0.2879	0.75	0.0000	1	7	8
9	1	0.32	0.3485	0.76	0.0000	8	6	14
10	1	0.38	0.3939	0.76	0.2537	12	24	36
11	1	0.36	0.3333	0.81	0.2836	26	30	56
12	1	0.42	0.4242	0.77	0.2836	29	55	84
13	2	0.46	0.4545	0.72	0.2985	47	47	94
14	2	0.46	0.4545	0.72	0.2836	35	71	106
15	2	0.44	0.4394	0.77	0.2985	40	70	110
16	2	0.42	0.4242	0.82	0.2985	41	52	93
17	2	0.44	0.4394	0.82	0.2836	15	52	67
18	3	0.42	0.4242	0.88	0.2537	9	26	35
19	3	0.42	0.4242	0.88	0.2537	6	31	37
20	2	0.40	0.4091	0.87	0.2537	11	25	36
21	2	0.40	0.4091	0.87	0.1940	3	31	34
22	2	0.40	0.4091	0.94	0.2239	11	17	28
23	2	0.46	0.4545	0.88	0.2985	15	24	39
24	2	0.46	0.4545	0.88	0.2985	4	13	17
25	2	0.44	0.4394	0.94	0.2537	1	16	17
26	2	0.42	0.4242	1.00	0.2836	1	8	9
27	2	0.46	0.4545	0.94	0.1940	2	4	6
28	2	0.46	0.4545	0.94	0.1940	2	1	3
29	3	0.42	0.4242	0.77	0.2985	0	2	2
..	...	...	...	...	...	...	...	...
62	1	0.24	0.2273	0.30	0.2239	11	146	157
63	1	0.24	0.2576	0.32	0.1045	9	148	157
64	1	0.20	0.2576	0.47	0.0000	8	102	110
65	1	0.20	0.2273	0.47	0.1045	3	49	52
66	1	0.18	0.1970	0.64	0.1343	3	49	52
67	1	0.14	0.1515	0.69	0.1343	0	20	20
68	1	0.18	0.2121	0.55	0.1045	1	11	12
69	1	0.16	0.1818	0.55	0.1045	0	5	5
70	1	0.16	0.1818	0.59	0.1045	0	2	2
71	1	0.14	0.1515	0.63	0.1343	0	1	1
72	1	0.14	0.1818	0.63	0.0896	0	2	2
73	1	0.12	0.1515	0.68	0.1045	0	4	4
74	1	0.12	0.1515	0.74	0.1045	0	36	36
75	1	0.12	0.1515	0.74	0.1343	2	92	94
76	1	0.14	0.1515	0.69	0.1642	2	177	179
77	1	0.16	0.1515	0.64	0.2239	2	98	100
78	2	0.16	0.1364	0.69	0.3284	5	37	42
79	1	0.22	0.2121	0.51	0.2985	7	50	57
80	1	0.22	0.2273	0.51	0.1642	12	66	78

81	1	0.24	0.2273	0.56	0.1940	18	79	97
82	1	0.26	0.2576	0.52	0.2239	9	54	63
83	1	0.28	0.2727	0.52	0.2537	17	48	65
84	1	0.30	0.2879	0.49	0.2537	15	68	83
85	1	0.28	0.2727	0.48	0.2239	10	202	212
86	1	0.26	0.2576	0.48	0.1940	3	179	182
87	1	0.24	0.2576	0.48	0.1045	2	110	112
88	1	0.24	0.2576	0.48	0.1045	1	53	54
89	1	0.22	0.2727	0.64	0.0000	0	48	48
90	1	0.22	0.2576	0.64	0.0896	1	34	35
91	1	0.20	0.2273	0.69	0.0896	2	9	11

[92 rows x 17 columns]