Assignment 02

YuGuobin 12332284

1. Significant earthquakes since 2150 B.C.

1.1. [5 points] Compute the total number of deaths caused by earthquakes since 2150 B.C. in each country, and then print the top ten countries along with the total number of deaths.

a) First, read the csv data, using "delimiter='\t'" to specify the TAB as the separatorart.

```
Sig_Eqs= pd.read_csv('earthquakes-2023-11-01_21-34-28_+0800.tsv', delimiter='\t')
Sig_Eqs.head()
```

	Search Parameters	Id	Year	Mo	Dy	Hr	Mn	Sec	Tsu	Vol	...	Total Missing	Total Missing Description	Total Injuries	Total Injuries Description	Total Damage ($Mil)	Total Damage Description	Total Houses Destroyed	De Desc
0	[]	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	NaN	1.0	-2150.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2	NaN	2.0	-2000.0	NaN	NaN	NaN	NaN	NaN	1.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	NaN	3.0	-2000.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	1.0	NaN	
4	NaN	5877.0	-1610.0	NaN	NaN	NaN	NaN	NaN	3.0	1351.0	...	NaN	NaN	NaN	NaN	NaN	3.0	NaN	

5 rows × 49 columns

b) Group the "countries" using "groupby" and select the "Total Deaths" field for summation statistics, and finally arrange them in order from largest to smallest, showing the top 10.

```
Sig_Eqs.groupby(['Country'])['Total Deaths'].sum().sort_values(ascending=False).head(10)
```

```
Country
CHINA          2041929.0
TURKEY          995648.0
IRAN            758650.0
SYRIA           437700.0
ITALY           422679.0
JAPAN           356083.0
HAITI           323776.0
AZERBAIJAN      310119.0
INDONESIA       282819.0
ARMENIA         189000.0
Name: Total Deaths, dtype: float64
```

1.2. [10 points] Compute the total number of earthquakes with magnitude larger than 6.0 (use column Mag as the magnitude) worldwide each year, and then plot the time series. Do you observe any trend? Explain why or why not?

a) First, "Mag" is filtered and located, and then grouped according to "Year".

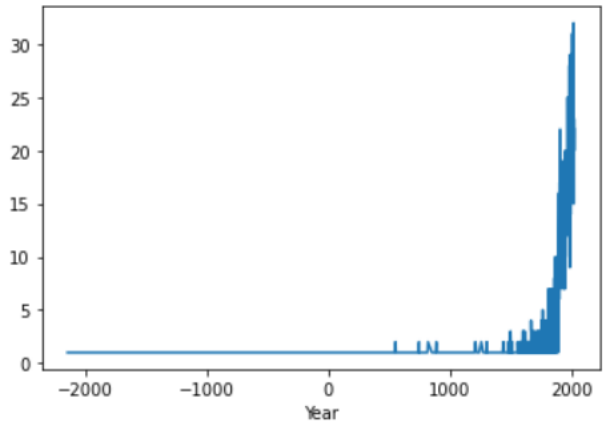
Finally, the "count" method is used to count the number of each group.

b) It is known from the following reference link that the "count" method is used to count the value that does not contain NaN

The total number of earthquakes greater than 6.0 is on the rise, mainly due to the development of human civilization and science and technology in modern times, resulting in an explosive increase in the total number of earthquakes observed and recorded.

```
Sig_Eqs.loc[Sig_Eqs['Mag']>6.0].groupby(Sig_Eqs['Year']).count()['Mag'].plot()
```

```
<AxesSubplot:xlabel='Year'>
```



I got inspired by reading:

https://www.cnblogs.com/zknublx/p/12048410.html

1.3. [10 points] Write a function CountEq_LargestEq that returns both (1) the total number of earthquakes since 2150 B.C. in a given country AND (2) the date of the largest earthquake ever happened in this country. Apply CountEq_LargestEq to every country in the file, report your results in a descending order.

a) First, define a function "CountEq_LargestEq" that takes the argument "country_name".

b) Select data with the same Country name based on the country field, save the data in country_sel, and calculate its length.

c) Use the "max ()" function to select the data with the largest "Mag" in "country_sel" and save it to the "largest" variable.

d) Get the date in "largest".

e) Next, create a "result_df" dataframe variable and write the country name, the total number of earthquakes, and the maximum earthquake date into the dataframe. Finally, result_df is returned.

f) Next, use the "unique ()" method to get the unique value for the country.

g) Define an empty dataframe as "results_df" to save the result.

h) Finally, write a for loop that executes the "CountEq_LargestEq" function for each country, adding the return value to a new line of "results_df" each time.

```
def CountEq_LargestEq(country_name):
    country_sel = Sig_Eqs[Sig_Eqs['Country'] == country_name]
    total = len(country_sel)

    largest = country_sel[country_sel['Mag'] == country_sel['Mag'].max()]
    largest_date = largest[['Year','Mo','Dy']]
    result_df = pd.DataFrame({'country': [country_name], 'count': [total], 'largestdate': [largest_date],
                              })
    return result_df

countries = Sig_Eqs['Country'].unique()
results_df = pd.DataFrame(columns=[])

for country in countries:
    country_df = CountEq_LargestEq(country)
    results_df = pd.concat([results_df, country_df])

results_df = results_df.sort_values('count', ascending=False)
results_df
```

	country	count	largestdate
0	CHINA	620	Year Mo Dy 982 1668.0 7.0 25.0
0	JAPAN	414	Year Mo Dy 5743 2011.0 3.0 11.0
0	INDONESIA	411	Year Mo Dy 5341 2004.0 12.0 26.0
0	IRAN	384	Year Mo Dy 238 856.0 12.0 22.0
0	TURKEY	335	Year Mo Dy 3413 1939.0 12.0 2...
...	...	...	...
0	KIRIBATI	1	Year Mo Dy 2622 1905.0 6.0 30.0
0	PALAU	1	Year Mo Dy 2838 1914.0 10.0 23.0
0	CENTRAL AFRICAN REPUBLIC	1	Year Mo Dy 2964 1921.0 9.0 16.0
0	LIBYA	1	Year Mo Dy 3926 1963.0 2.0 21.0
0	NaN	0	Empty DataFrame Columns: [Year, Mo, Dy] Index: []

157 rows × 3 columns

2. Wind speed in Shenzhen during the past 10 years

2.1. [10 points] Plot monthly averaged wind speed as a function of the observation time. Is there a trend in monthly averaged wind speed within the past 10 years?

a) Filter the data: First of all, copy the two columns "DATA" and "WND" from the "2281305.csv" to a new file and name it "wind.xlsl", where

"WND" is separated by comma into 5 parameters: "direction", "directionquality", "type", "speed", and "speedquality", and then the value of "speed==9999" is filtered out and deleted, this series of work is completed in excel.

b) First, read in "wind.xlsx" and name it “winddata”:

```
winddata = pd.read_excel('wind.xlsx')
winddata.head(10)
```

	DATE	direction	directionquality	type	speed	speedquality
0	2010-01-02T00:00:00	40.0	1.0	N	20.0	1.0
1	2010-01-02T01:00:00	999.0	9.0	V	10.0	1.0
2	2010-01-02T02:00:00	999.0	9.0	C	0.0	1.0
3	2010-01-02T03:00:00	140.0	1.0	N	10.0	1.0
4	2010-01-02T04:00:00	300.0	1.0	N	40.0	1.0
5	2010-01-02T05:00:00	320.0	1.0	N	50.0	1.0
6	2010-01-02T06:00:00	270.0	1.0	N	10.0	1.0
7	2010-01-02T07:00:00	350.0	1.0	N	30.0	1.0
8	2010-01-02T08:00:00	360.0	1.0	N	30.0	1.0
9	2010-01-02T09:00:00	40.0	1.0	N	30.0	1.0

c) Get the year and month of “winddata” respectively and add them to the new columns named "year" and "mon":

```
winddata['year'] = pd.to_datetime(winddata['DATE']).dt.year
winddata['mon'] = pd.to_datetime(winddata['DATE']).dt.month
winddata.head(10)
```

	DATE	direction	directionquality	type	speed	speedquality	year	mon
0	2010-01-02T00:00:00	40.0	1.0	N	20.0	1.0	2010.0	1.0
1	2010-01-02T01:00:00	999.0	9.0	V	10.0	1.0	2010.0	1.0
2	2010-01-02T02:00:00	999.0	9.0	C	0.0	1.0	2010.0	1.0
3	2010-01-02T03:00:00	140.0	1.0	N	10.0	1.0	2010.0	1.0
4	2010-01-02T04:00:00	300.0	1.0	N	40.0	1.0	2010.0	1.0
5	2010-01-02T05:00:00	320.0	1.0	N	50.0	1.0	2010.0	1.0
6	2010-01-02T06:00:00	270.0	1.0	N	10.0	1.0	2010.0	1.0
7	2010-01-02T07:00:00	350.0	1.0	N	30.0	1.0	2010.0	1.0
8	2010-01-02T08:00:00	360.0	1.0	N	30.0	1.0	2010.0	1.0
9	2010-01-02T09:00:00	40.0	1.0	N	30.0	1.0	2010.0	1.0

d) Group the year and month, and average the "speed", use the "unstack ()" method so that the groups are not stacked, and divide by the magnification factor 10:

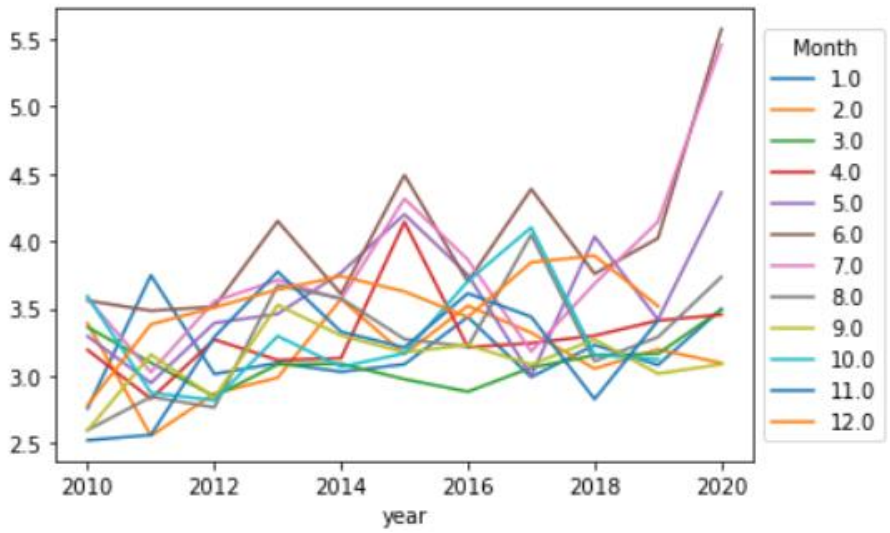
```
monthly_avg = winddata.groupby(['year', 'mon'])['speed'].mean().unstack()/10
monthly_avg
```

mon	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	11.0	12.0
year												
2010.0	2.756267	3.388060	3.360700	3.191341	3.293640	3.558333	3.575472	2.595430	2.593315	3.589744	2.519553	2.780619
2011.0	3.746289	2.552906	3.096076	2.828452	2.945873	3.482566	3.025538	2.842318	3.156944	2.873144	2.559722	3.381081
2012.0	3.012129	2.867626	2.846995	3.268802	3.391129	3.512500	3.552561	2.764151	2.837500	2.816464	3.287500	3.506720
2013.0	3.094213	2.985119	3.083445	3.115438	3.459184	4.149049	3.711382	3.672131	3.525762	3.294479	3.773250	3.640000
2014.0	3.028777	3.569492	3.088957	3.130252	3.765191	3.607066	3.567054	3.574642	3.297895	3.065856	3.326338	3.739130
2015.0	3.084763	3.159193	2.973737	4.142857	4.199187	4.491137	4.315152	3.271617	3.173867	3.166160	3.209424	3.625510
2016.0	3.435794	3.519423	2.880705	3.211134	3.750253	3.699367	3.862104	3.216049	3.228330	3.702840	3.611702	3.435138
2017.0	2.989680	3.319507	3.059717	3.241851	3.000000	4.386994	3.180061	4.047867	3.082340	4.101420	3.437302	3.842480
2018.0	3.228514	3.052868	3.152130	3.297441	4.033708	3.756785	3.683179	3.107472	3.269412	3.154959	2.825166	3.891481
2019.0	3.079108	3.195286	3.162023	3.409853	3.416327	4.022059	4.145749	3.286448	3.014973	3.121150	3.408957	3.516860
2020.0	3.496532	3.094081	3.489234	3.453306	4.362198	5.575800	5.459140	3.733608	3.085019	NaN	NaN	NaN

e) Finally, plot with "plot ()" method and add legend:

```
ax = monthly_avg.plot(legend=False)
ax.legend(loc='center left', bbox_to_anchor=(1, 0.5), title='Month')
```

```
<matplotlib.legend.Legend at 0x1d7de1144c0>
```



f) It can be seen from the figure that the wind speed of each month in Shenzhen has shown an overall upward trend in the past 10 years, especially the wind speed in June and July has increased most significantly in recent years.

I got inspired by reading:

https://www.jianshu.com/p/6c3d34e1677c

3. Explore a data set

3.1. [5 points] Load the csv, XLS, or XLSX file, and clean possible data points with missing values or bad quality.

a) The visibility data is extracted from the column of “VIS” in the “2281305.csv” and saved as “vis. Xlsx”. The “VIS” is divided into “VIS”, “DQC”, “VC”, and “QVC” by comma. Refer to page 10 of the user guide in question 2 (POS 79-87). Then delete “VIS==999999” and “VC==9 data”.

b) Load the visibility data in and name it "visdata":

```
visdata = pd.read_excel('vis.xlsx')
visdata.head(10)
```

	DATE	VIS	DQC	VC	QVC
0	2010-01-02T00:00:00	4000.0	1.0	N	1.0
1	2010-01-02T01:00:00	2600.0	1.0	N	1.0
2	2010-01-02T02:00:00	2600.0	1.0	N	1.0
3	2010-01-02T03:00:00	5000.0	1.0	N	1.0
4	2010-01-02T04:00:00	2100.0	1.0	N	1.0
5	2010-01-02T05:00:00	1800.0	1.0	N	1.0
6	2010-01-02T06:00:00	3000.0	1.0	N	1.0
7	2010-01-02T07:00:00	2000.0	1.0	N	1.0
8	2010-01-02T08:00:00	4600.0	1.0	N	1.0
9	2010-01-02T09:00:00	2000.0	1.0	N	1.0

3.2. [5 points] Plot the time series of a certain variable.

a) First get the year, month and hour data from the "DATA" field and name them "year", "mon" and "hour" respectively:

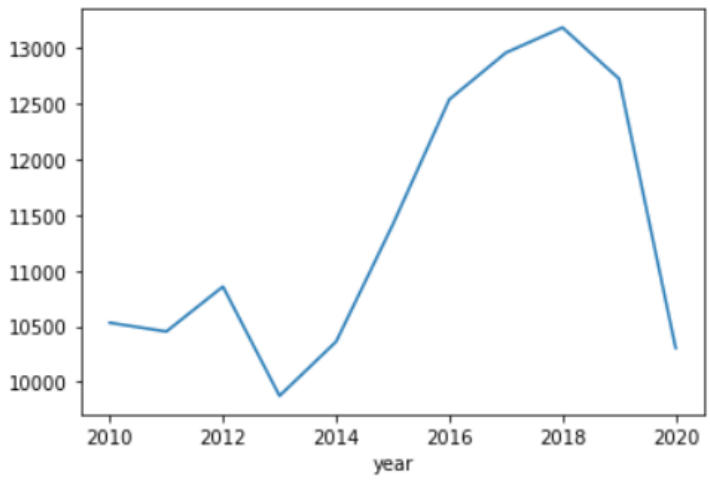
```
visdata['year'] = pd.to_datetime(visdata['DATE']).dt.year
visdata['mon'] = pd.to_datetime(visdata['DATE']).dt.month
visdata['hour'] = pd.to_datetime(visdata['DATE']).dt.hour
visdata.head(10)
```

	DATE	VIS	DQC	VC	QVC	year	mon	hour
0	2010-01-02T00:00:00	4000.0	1.0	N	1.0	2010.0	1.0	0.0
1	2010-01-02T01:00:00	2600.0	1.0	N	1.0	2010.0	1.0	1.0
2	2010-01-02T02:00:00	2600.0	1.0	N	1.0	2010.0	1.0	2.0
3	2010-01-02T03:00:00	5000.0	1.0	N	1.0	2010.0	1.0	3.0
4	2010-01-02T04:00:00	2100.0	1.0	N	1.0	2010.0	1.0	4.0
5	2010-01-02T05:00:00	1800.0	1.0	N	1.0	2010.0	1.0	5.0
6	2010-01-02T06:00:00	3000.0	1.0	N	1.0	2010.0	1.0	6.0
7	2010-01-02T07:00:00	2000.0	1.0	N	1.0	2010.0	1.0	7.0
8	2010-01-02T08:00:00	4600.0	1.0	N	1.0	2010.0	1.0	8.0
9	2010-01-02T09:00:00	2000.0	1.0	N	1.0	2010.0	1.0	9.0

b) Then the “visdata” is grouped according to the year and the mean of the "VIS" is calculated:

```
yearVisAvg = visdata.groupby(['year'])['VIS'].mean().plot()
yearVisAvg
```

```
<AxesSubplot:xlabel='year'>
```

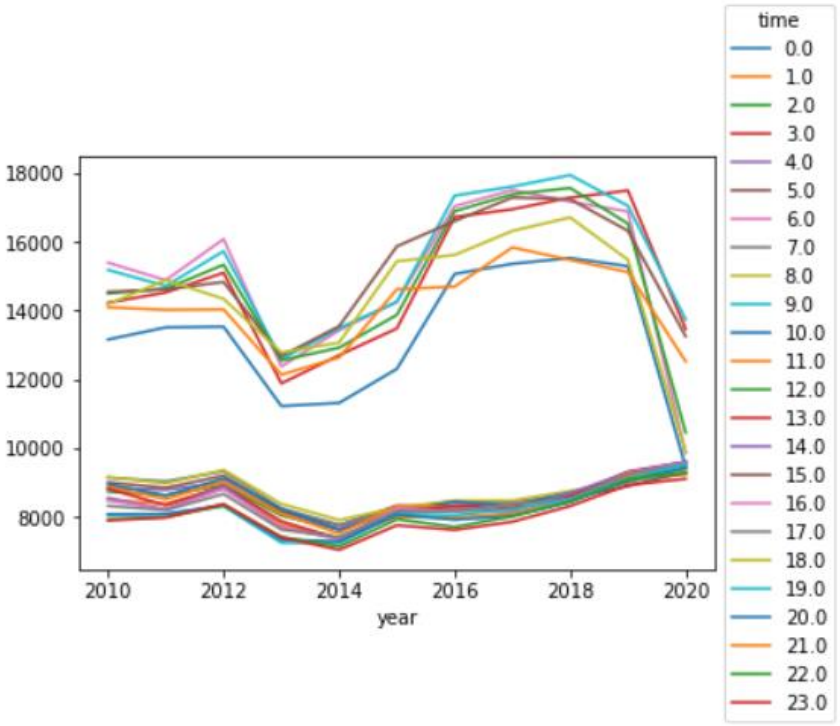


3.3. [5 points] Conduct at least 5 simple statistical checks with the variable, and report your findings.

a) First, we group the "visdata" according to years and hours and calculate their mean values, and then plotted by year, with each series representing a different time of day:

```
hourVisAve = visdata.groupby(['year', 'hour'])['VIS'].mean().unstack()
ax = hourVisAve.plot(legend=False)
ax.legend(loc='center left', bbox_to_anchor=(1, 0.5), title='time')
```

```
<matplotlib.legend.Legend at 0x1d7e04f1580>
```

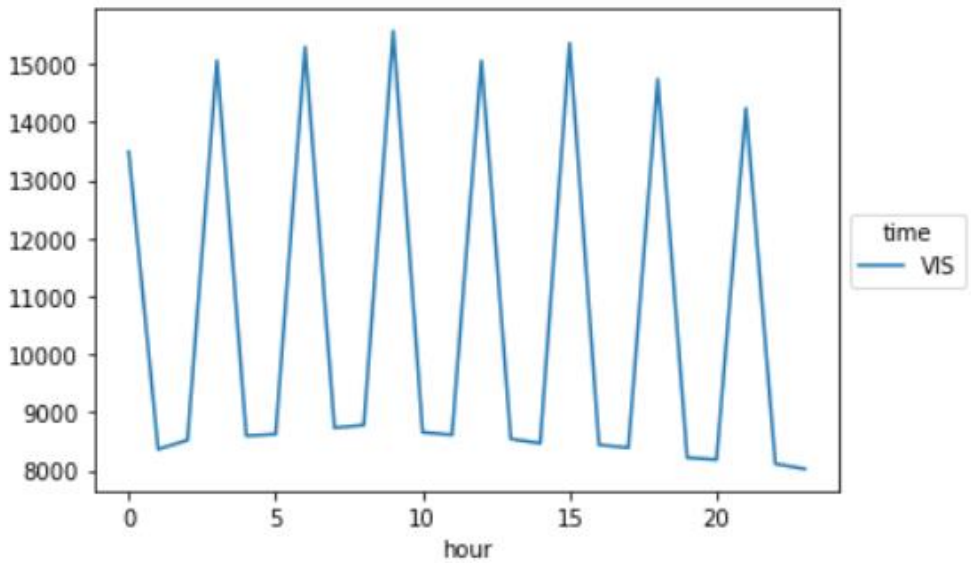


We found that the horizontal visibility of Shenzhen in the past 10 years was overall rising before 2018, and it declined rapidly after 2018, but there are different trends at different times of day.

b) In order to explore whether visibility changes throughout the day, "visdata" is grouped according to "hour" and its average is calculated. It is interesting that visibility varied periodically throughout the day, but it was clear that the high values differed so much from the low values that we suspected the high values were wrong:

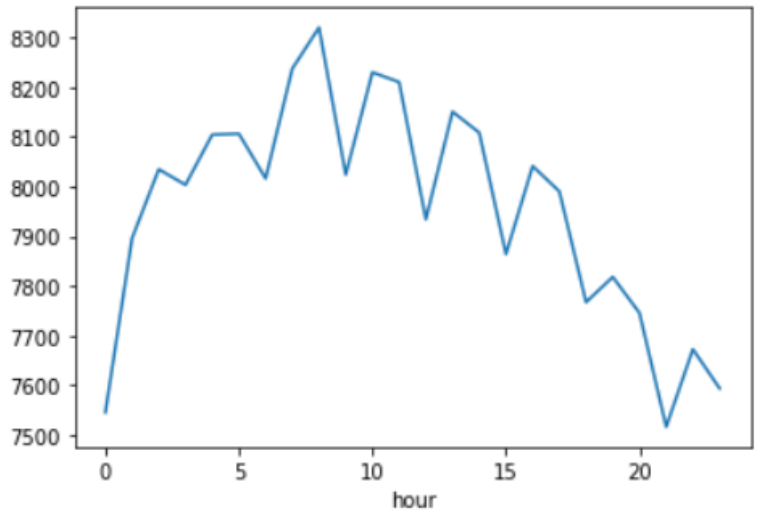
```
hourVisAve = visdata.groupby(['hour'])['VIS'].mean()
ax = hourVisAve.plot(legend=False)
```

```
<matplotlib.legend.Legend at 0x1d7e0a5a940>
```



c) Therefore, we screened the data less than 10000, re-counted and plotted it. We found that visibility in Shenzhen varied significantly throughout the day, being the highest at 7 and 8 am, then gradually decreasing, and being the lowest at night:

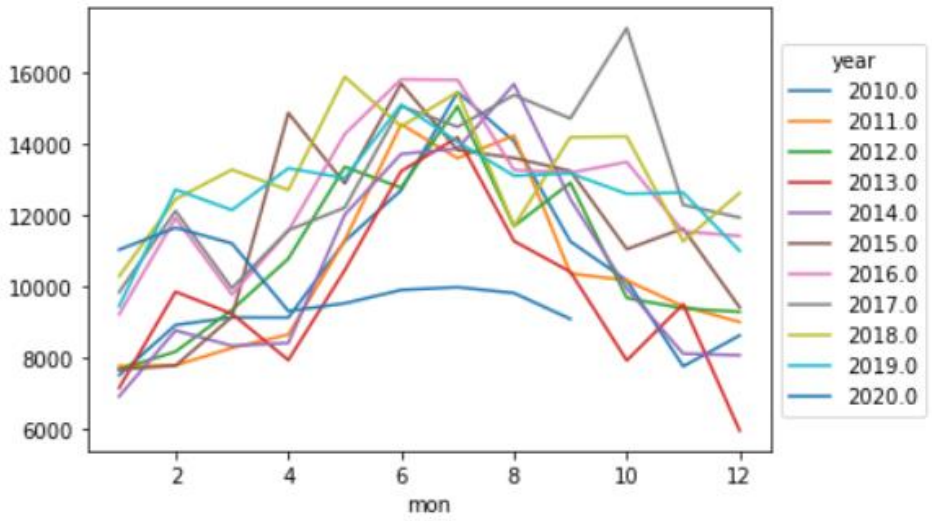
```
hourVisAve = visdata[visdata['VIS']<10000].groupby(['hour'])['VIS'].mean()
ax = hourVisAve.plot(legend=False)
```



d) We grouped them by month, year, and calculated their average, and then plotted them. We found that the visibility in Shenzhen was significantly higher in June and July, which may be related to the high wind speed during this period:

```
monVisAve = visdata.groupby(['mon','year'])['VIS'].mean().unstack()
ax = monVisAve.plot(legend=False)
ax.legend(loc='center left', bbox_to_anchor=(1, 0.5), title='year')
```

```
<matplotlib.legend.Legend at 0x1e35bd3de50>
```



e) We can also calculate the standard deviation of visibility each month, and it can still be found that the value in June and July is the smallest, indicating that the data in this period is the most stable:

```
hourVisStd = visdata[visdata['VIS']<10000].groupby(['mon'])['VIS'].std()
ax = hourVisStd.plot(legend=False)
```