Assignment 2

ESE5023

Computing and Programming for Environmental Research

Instructor:
Lei Zhu

Author:
Gong Guoqing
121332198

1. Significant earthquakes since 2150 B.C.

In [19]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
eq_data = pd.read_csv('Sig_Eqs.tsv',sep = '\t')
plt.rcParams['figure.dpi'] = 100
```

读取文件

1.1

In [20]:
```
#1.1
```

In [21]:
```
df_td = eq_data.groupby(['Country']).sum()
df_td
df_td2 = df_td.sort_values('Total Deaths',ascending=False).head(10)['Total Deaths']
print(df_td2)
df_td3 = df_td.sum()['Total Deaths']
print(df_td3)
```

将国家数据进行分类，合并到 df_td 中；
将 df_td 中的数据按 Total_Death 进行降序排列，取前 10 位输出；
将 df_td 中的 Total_Death 数据相加，输出。

```
Country
CHINA          2041784.0
TURKEY          867654.0
IRAN            758638.0
ITALY           359064.0
JAPAN           355137.0
SYRIA           337700.0
HAITI           323770.0
AZERBAIJAN      310119.0
INDONESIA       280351.0
ARMENIA         189000.0
Name: Total Deaths, dtype: float64
7162533.0
```

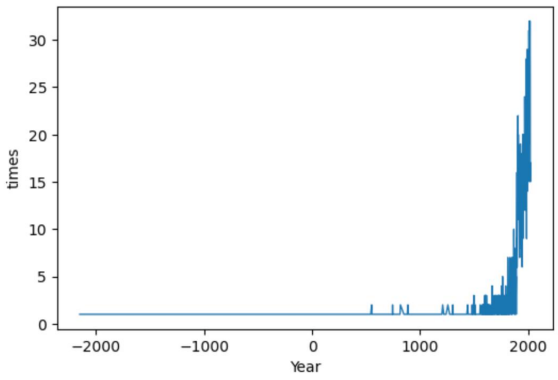
1.2

In [54]:
```
mag_df = eq_data[eq_data['Mag']>6.0].count()
print(mag_df['Mag'])
eq_data[eq_data['Mag']>6].groupby('Year').count()['Country'].plot(lw=1,ylabel = 'times')

#趋势：1500年之后6.0震级的地震上升趋势明显
#原因可能是之前没有记录
```

```
2870
```

将 eq_data 中 Mag>6.0 的次数输出给 mag_df;
将 Mag>6.0 的地震次数按年份进行分类，输出。



1.3

```
In [25]: def CountEq_LargestEq(country):

             CountEq_df = len(eq_data[eq_data['Country'] == country])

             LargestEq = eq_data[eq_data['Country'] == country].sort_values('Mag',ascending = False).iloc[0][['Year','Mo','Dy','
             date = str(LargestEq['Year'])[:-2]+'-'+str(LargestEq['Mo'])[:-2].zfill(2)+'-'+str(LargestEq['Dy'])[:-2].zfill(2)

             return CountEq_df,date
```

定义函数 CountEq_LargestEq；
CountEq_df 算的是地震次数；
LargestEq 算的是最大地震出现的日期；
Date 是将 LargestEq 算出的日期进行一个格式排版；
将函数返回给 CountEq_df, date。

```
In [26]: country_list = eq_data['Country'].unique()

         arr_eqdata= np.full((country_list.shape[0]-1,2),np.nan)
```

country_list 找出国家的总量；
arr_eqdata 建立一个行数为国家总量，列数为 2 的空表。

```
In [27]: df_0 = pd.DataFrame(arr_eqdata,index=country_list[1:],columns=['count','date of biggest mag'])
```

df_0 是建立一个 index = country，column 为‘count’和‘date of biggest mag’的表格。

```
In [28]: for country in country_list[1:]:
             df_0.loc[country,['count','date of biggest mag']]=CountEq_LargestEq(country)
```

建立一个循环，将 eq_data 中的地震次数和最大地震出现的日期加入新建的表格 df_0 中。

```
In [29]: df_0.sort_values('count',ascending=False)
```

Out[29]:

	count	date of biggest mag
CHINA	610.0	1668-07-25
JAPAN	409.0	2011-03-11
INDONESIA	401.0	2004-12-26
IRAN	380.0	856-12-22
TURKEY	330.0	1912-08-09
...	...	...
NORWAY	1.0	1819-08-31
CENTRAL AFRICAN REPUBLIC	1.0	1921-09-16
PALAU	1.0	1914-10-23
KIRIBATI	1.0	1905-06-30
COMOROS	1.0	2018-05-15

将 df_0 中的地震次数按照降序排列，得到表格 df_0。

2. Wind speed in Shenzhen during the past 10 years

```
In [17]: import pandas as pd
         import numpy as np
         from matplotlib import pyplot as plt
         df_input = pd.read_csv('2281305.csv')
```

读取文件

```
In [24]: df_wind = pd.DataFrame(np.full((len(df_input),1),np.nan),index = df_input['DATE'],columns = ['speed rate'])
         for i in range(len(df_wind)):
             df_wind.iloc[i,0] = int(df_input['WND'][i][8:12])
```

df_wind 是建立的一个新的 index = df_input [‘DATE’]的表格，建立新的一列[‘speed rate’]；将 df_input[‘WND’]中的 speed rate 导入新的表格中。

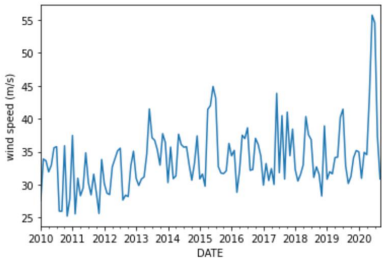
```
In [28]: arr_wind = df_wind['speed rate'].values
         arr_wind[arr_wind==9999] = np.nan
         df_wind['speed rate'] = arr_wind
```

检查错误数据（9999），将错误数据赋值为空值（NAN）。

```
In [29]: df_wind.index = pd.to_datetime(df_wind.index)
```

利用 to_datetime() 将时间格式以 yyyy-mm-dd 的形式输出。

```
In [33]: def plot10year():
             df_wind.resample('m').mean()['speed rate'].plot(ylabel = 'wind speed (m/s)')
         plot10year()
```



利用 resample() 将过去 10 年的数据以月份平均后输出。

3. Explore a data set

	Province	Year	PIRR	AIRR	WCI	IRR WUI	revised IRR WUI	Ratio of industrial water recycling	Ratio of industrial water evaporated	IND WUI	revised IND WUI
0	Anhui	1975	418.943076	29930.904227	0.019706	775.065446	775.065446	0.000000	0.306688	0.310811	0.310811
1	Anhui	1976	463.369634	18491.148830	0.020940	746.094373	717.345493	0.000000	0.290317	0.296136	0.289606
2	Anhui	1977	468.093808	21995.614134	0.022175	761.651117	737.127359	0.000000	0.273163	0.279693	0.265661
3	Anhui	1978	542.566592	14670.005881	0.023410	767.420515	798.875848	0.000000	0.315550	0.212044	0.194575
4	Anhui	1979	622.801535	19082.387416	0.024406	771.687823	828.967468	0.007621	0.243994	0.187960	0.168401
5	Anhui	1980	442.283199	24558.553760	0.025694	772.328081	805.825334	0.029787	0.268395	0.186694	0.161364
6	Anhui	1981	514.166045	18936.521926	0.027116	770.727524	848.370289	0.051952	0.245218	0.151795	0.124125
7	Anhui	1982	484.141343	24520.929043	0.028605	770.610234	865.906829	0.074118	0.186361	0.139954	0.111741
8	Anhui	1983	515.032473	28502.246768	0.029525	763.106834	864.673887	0.094618	0.200634	0.129950	0.101491
9	Anhui	1984	539.189414	26915.205421	0.030897	787.900081	871.947653	0.118272	0.199282	0.113588	0.090604
10	Anhui	1985	486.498970	25208.945986	0.032436	811.652674	917.671318	0.141927	0.212055	0.106881	0.089935
11	Anhui	1986	536.859427	19702.315623	0.033540	822.158200	923.603472	0.165581	0.220466	0.080212	0.067177
12	Anhui	1987	457.809114	25421.629445	0.034644	795.838393	906.138505	0.189236	0.221841	0.058901	0.050428
13	Anhui	1988	586.513457	18918.404486	0.035463	765.276345	862.097041	0.212890	0.204998	0.049495	0.044004
14	Anhui	1989	477.484214	23640.381171	0.036684	748.025376	824.615461	0.236544	0.238356	0.049416	0.043203

我所采用的数据如表所示（部分）

3.1

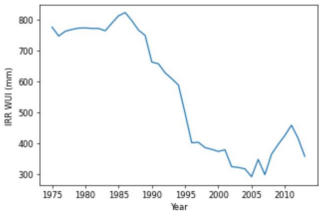
```
In [96]: import pandas as pd
         import numpy as np
         from matplotlib import pyplot as plt
         df = pd.read_excel('Data for identifying the drivers for sectoral WUI.xlsx')
         plt.rcParams['figure.dpi'] = 60
```

读取数据

3.2

```
In [114]: #3.2
```

```
In [115]: df1 = df.groupby(['Year']).sum()['IRR WUI'].plot(ylabel = 'IRR WUI (mm) ')
```



按年份分类，输出每年的 IRR WUI 的总量

3.3

```
In [117]: #mean value
          meam_WUI_prov = df.mean()['IRR WUI']
          print('The mean value of IRR WUI is: ',meam_WUI_prov)
```

```
The mean value of IRR WUI is:  561.2856105764963
```

算 IRR WUI 的平均值

(以下几种计算方法从参考文献中获得启发)

```
In [118]: #Shapiro-Wilk Test
          from scipy.stats import shapiro
          stat, p = shapiro(df['IRR WUI'])
          print('stat=%.3f, p=%.3f' % (stat, p))
          if p > 0.05:
              print('Probably Gaussian')
          else:
              print('Probably not Gaussian')
```

```
stat=0.842, p=0.000
Probably not Gaussian
```

Shapiro-Wilk Test

```
In [119]: #Chi-Squared Test
          from scipy.stats import chi2_contingency
          stat, p, dof, expected = chi2_contingency(df['IRR WUI'])
          print('stat=%.3f, p=%.3f' % (stat, p))
          if p > 0.05:
              print('Probably independent')
          else:
              print('Probably dependent')
```

```
stat=0.000, p=1.000
Probably independent
```

Chi-Squared Test

In [120]:
```
#Augmented Dickey-Fuller Unit Root Test
from statsmodels.tsa.stattools import adfuller
stat, p, lags, obs, crit, t = adfuller(df['IRR WUI'])
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
    print('Probably not Stationary')
else:
    print('Probably Stationary')
```

```
stat=-0.610, p=0.869
Probably not Stationary
```

Augmented Dickey-Fuller Unit Root Test

In [121]:
```
#D'Agostino's K^2 Test
from scipy.stats import normaltest
stat, p = normaltest(df['IRR WUI'])
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
    print('Probably Gaussian')
else:
    print('Probably not Gaussian')
```

```
stat=81.608, p=0.000
Probably not Gaussian
```

D'Agostino's K^2 Test

Reference：

https://pandas.pydata.org/docs/reference/api/pandas.to_datetime.html
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.resample.html
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.shapiro.html
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.chi2_contingency.html
https://www.statsmodels.org/dev/generated/statsmodels.tsa.stattools.adfuller.html
https://machinelearningmastery.com/a-gentle-introduction-to-normality-tests-in-python/