In [1]:

```
import numpy as py
import pandas as pd
import matplotlib as mpl
###############     1.1       #########
sig=pd.read_csv('sig_eqs.tsv',sep='\t')
sig2=sig[['Country','Deaths']]
sig2=sig2.groupby(['Country']).sum().sort_values('Deaths',ascending=False)
sig2.head(20)
###############     1.2       #########
sig3=sig[['Year','Ms']]
sig3=sig3.dropna()
sig3['num']=1
sig3=sig3.groupby(['Year']).sum(['num']).reset_index()
sig3.plot(x='Year',y='num')
#绘制出来的图像似乎有地震愈来愈频发的趋势，但是我认为这个结论是不合理的。
#因为随着近现代观测手段的完善，记录到的地震本就比以前多，而且关于Ms的数据有大量缺
###############    1.3        ###########
country_name=sig['Country']
country_name = country_name.dropna()
country_name = country_name.drop_duplicates().reset_index(drop=True)
country_list = country_name.tolist()
c_l_sig=sig[['Year','Country','Mo','Dy','Hr','Mn','Sec','Location Name','Latitud
#使用Mag作为衡量地震大小的标准
largest_eq=pd.DataFrame()
for i in range(len(country_name)):
    country_sig=c_l_sig.loc[c_l_sig['Country']==country_list[i]]
    eq_num=len(country_sig)
    country_sig=country_sig.sort_values('Mag',ascending=False)
    eq_largestMag=country_sig.iloc[0,10]
    new_row=country_sig.loc[country_sig['Mag']==eq_largestMag]
    largest_eq=pd.concat([largest_eq,new_row], ignore_index=True)
largest_eq['Mag'].astype(int)
largest_eq.sort_values(by='Mag',ascending=False)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[1], line 5
      3 import matplotlib as mpl
      4 ###############     1.1       #########
----> 5 sig=pd.read_csv('sig_eqs.tsv',sep='\t')
      6 sig2=sig[['Country','Deaths']]
      7 sig2=sig2.groupby(['Country']).sum().sort_values('Deaths',ascending=Fals
e)

File E:\anaconda\Lib\site-packages\pandas\io\parsers\readers.py:1026, in read_csv
(filepath_or_buffer, sep, delimiter, header, names, index_col, usecols, dtype, en
gine, converters, true_values, false_values, skipinitialspace, skiprows, skipfoot
er, nrows, na_values, keep_default_na, na_filter, verbose, skip_blank_lines, pars
e_dates, infer_datetime_format, keep_date_col, date_parser, date_format, dayfirs
t, cache_dates, iterator, chunksize, compression, thousands, decimal, linetermina
tor, quotechar, quoting, doublequote, escapechar, comment, encoding, encoding_err
ors, dialect, on_bad_lines, delim_whitespace, low_memory, memory_map, float_preci
sion, storage_options, dtype_backend)
   1013 kwds_defaults = _refine_defaults_read(
   1014     dialect,
   1015     delimiter,
   (...)
   1022     dtype_backend=dtype_backend,
   1023 )
   1024 kwds.update(kwds_defaults)
-> 1026 return _read(filepath_or_buffer, kwds)

File E:\anaconda\Lib\site-packages\pandas\io\parsers\readers.py:620, in _read(fil
epath_or_buffer, kwds)
    617 _validate_names(kwds.get("names", None))
    619 # Create the parser.
--> 620 parser = TextFileReader(filepath_or_buffer, **kwds)
    622 if chunksize or iterator:
    623     return parser

File E:\anaconda\Lib\site-packages\pandas\io\parsers\readers.py:1620, in TextFile
Reader.__init__(self, f, engine, **kwds)
   1617     self.options["has_index_names"] = kwds["has_index_names"]
   1619 self.handles: IOHandles | None = None
-> 1620 self._engine = self._make_engine(f, self.engine)

File E:\anaconda\Lib\site-packages\pandas\io\parsers\readers.py:1880, in TextFile
Reader._make_engine(self, f, engine)
   1878     if "b" not in mode:
   1879         mode += "b"
-> 1880 self.handles = get_handle(
   1881     f,
   1882     mode,
   1883     encoding=self.options.get("encoding", None),
   1884     compression=self.options.get("compression", None),
   1885     memory_map=self.options.get("memory_map", False),
   1886     is_text=is_text,
   1887     errors=self.options.get("encoding_errors", "strict"),
   1888     storage_options=self.options.get("storage_options", None),
   1889 )
   1890 assert self.handles is not None
   1891 f = self.handles.handle

File E:\anaconda\Lib\site-packages\pandas\io\common.py:873, in get_handle(path_or
```

```
_buf, mode, encoding, compression, memory_map, is_text, errors, storage_options)
    868 elif isinstance(handle, str):
    869     # Check whether the filename is to be opened in binary mode.
    870     # Binary mode does not support 'encoding' and 'newline'.
    871     if ioargs.encoding and "b" not in ioargs.mode:
    872         # Encoding
--> 873         handle = open(
    874             handle,
    875             ioargs.mode,
    876             encoding=ioargs.encoding,
    877             errors=errors,
    878             newline="",
    879         )
    880     else:
    881         # Binary mode
    882         handle = open(handle, ioargs.mode)

FileNotFoundError: [Errno 2] No such file or directory: 'sig_eqs.tsv'
```

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib as mpl
bw=pd.read_csv('Baoan_Weather_1998_2022.csv')
bw['YEAR']=bw['DATE'].str.split('-').str[0]
bw['MONTH']=bw['DATE'].str.split('-').str[1]
bw['DAY']=bw['DATE'].str.split('-').str[2]
bull=bw['TMP'].str.contains('\+')
indexes=np.where(bw['TMP']=='+9999,9')[0].tolist()
b=len(bw)
for i in range(b):
    bw.loc[i,'TMP']=bw.loc[i,'TMP'].replace('+','').replace(',','')
#indexes=[int(i) for i in indexes]
bw['TMP']=bw['TMP'].astype(int)
for i in indexes:
    bw.loc[i,'TMP']=bw.loc[i-1,'TMP']
    #如果出现连续的99999，线性插值会出现不合理的值，于是选择最近邻插值

bw2=bw.groupby(['YEAR','MONTH']).mean('TMP')
bw2['TMP']=bw2['TMP']/100


bw2['TMP'].plot()
```

In [ ]:
```
import numpy as py
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
###############     3.1       #########
df = pd.read_csv('ibtracs.ALL.list.csv',
                 usecols=range(17),
                 skiprows=[1, 2],
                 parse_dates=['ISO_TIME'],
                 na_values=['NOT_NAMED', 'NAME'])
df.head()
df['WMO_WIND']=pd.to_numeric(df['WMO_WIND'].str.replace(' ', ''), errors='coerce
df2=df.sort_values(['WMO_WIND'],ascending=False)
df2=df2.groupby(['SID']).first().reset_index()
top20 = df2.sort_values(by='WMO_WIND', ascending=False).head(20)
print(top20['NAME'])
###############     3.2       #########
```

```
df3=top20[['NAME','WMO_WIND']]
df3['WMO_WIND']=df3['WMO_WIND'].astype(int)
df3.plot(x='NAME',y='WMO_WIND',kind='bar')
###############     3.3       #########
num=df.groupby(['BASIN']).count()['SID']
num.plot(kind='bar')
###############     3.4       #########
x=df['LON'].astype(int)
y=df['LAT'].astype(int)
plt.figure(figsize=(8, 6))
hb = plt.hexbin(x, y, gridsize=100, cmap='viridis')
plt.colorbar(hb, label='log10(N)')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('The number')
plt.show()
###############     3.5       #########
mangkhut=df[(df['NAME'] == 'MANGKHUT')&(df['SEASON'] ==2018)]
mangkhut_lon = mangkhut['LON']
mangkhut_lat = mangkhut['LAT']
plt.figure(figsize=(10, 6))
plt.scatter(mangkhut_lon, mangkhut_lat, color='blue', s=60)
plt.title('Typhoon Mangkhut (2018) Track')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
###############     3.6       #########
df_1970 = df[df['SEASON'] >= 1970]
df_wp_ep = df_1970[(df_1970['BASIN'] == 'WP') | (df_1970['BASIN'] == 'EP')]
df_wp_ep.head()
###############     3.7       #########
df_wp_ep2=df_wp_ep.copy()
df_wp_ep2['Date'] = pd.to_datetime(df_wp_ep2['ISO_TIME'])
df_wp_ep2.set_index('Date', inplace=True)
perday = df_wp_ep2.resample('D').size()
perday.plot()
plt.title('Number of Datapoints per Day')
plt.xlabel('Date')
plt.ylabel('Number of Datapoints')
plt.show()
###############     3.8       #########
df_wp_ep3=df_wp_ep.copy()
df_wp_ep3['Date'] = pd.to_datetime(df_wp_ep3['ISO_TIME'])
df_wp_ep3.set_index('Date', inplace=True)
df_wp_ep3['dayofyear'] = df_wp_ep3.index.dayofyear
day_sum=df_wp_ep3.groupby([df_wp_ep3.index.year, 'dayofyear']).size()
day_ave=day_sum.groupby('dayofyear').mean()
day_ave.plot()
plt.title('Climatology of Datapoint Counts by Day of Year')
plt.xlabel('Day of Year')
plt.ylabel('Ave Number of Datapoints')
plt.show()
###############     3.9       #########
piancha=day_sum-day_ave
piancha = piancha.fillna(0)
piancha.plot()
plt.title('Anomaly of Daily Counts from Climatology')
plt.xlabel('Day of Year')
plt.ylabel('Anomaly')
plt.show()
```

```
###############     3.10       #########
pc2=piancha.copy()
pc2=pc2.groupby('Date').sum()
pc2.plot()
plt.title('Annual Resampled Anomaly of Hurricane Activity')
plt.xlabel('Year')
plt.ylabel('Total Anomaly')
plt.show()
```

In [ ]:

```
import netCDF4 as nc
import numpy as py
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
df=pd.read_csv('tsunamis.tsv',sep='\t')
df = df.dropna(how='all')
df = df.drop('Search Parameters',axis=1)
df = df.drop_duplicates()
##########  4.2#######
df2=df[['Year','Maximum Water Height (m)']]
df2=df2.dropna()
df2['num']=1
df2=df2.groupby(['Year']).sum(['num']).reset_index()
df2.plot(x='Year',y='num')
##########  4.3  #######
#最大波高大于5米的海啸数的时间序列
df3=df2.copy()
df3['Maximum Water Height (m)'].astype(int)
df3=df2[df2['Maximum Water Height (m)']>=5]
df3['num']=1
df3=df3.groupby(['Year']).sum(['num']).reset_index()
df3.plot(x='Year',y='num')
#发生海啸最多的国家
df4=df.copy()
df4 = df4.dropna(how='all')
df4['num']=1
df4=df4.groupby(['Country']).sum().sort_values('num',ascending=False)
print(df4['num'])
#发生海啸最多的月份
df5=df.copy()
df5=df5.dropna(how='all')
df5['num']=1
df5=df5.groupby(['Mo']).sum().sort_values('num',ascending=False)
print(df5['num'])

#海啸造成的死亡人数
df6=df.copy()
df6=df6.dropna(subset=['Deaths'])
df6=df6['Deaths']
df6.sum()
#每年因海啸死亡的人数
df7=df[['Year','Deaths']]
df7=df7.dropna(subset=['Deaths'])
df7=df7.groupby(['Year']).sum()
df7.plot()
```