1.1

[7]:
```python
Sig_Eqs = pd.read_csv('earthquakes-2024-10-28_14-56-33_+0800.tsv', sep='\t')
Sig_Eqs.head()
```

[7]:

	Search Parameters	Year	Mo	Dy	Hr	Mn	Sec	Tsu	Vol	Location Name	...	Total Missing	Total Missing Description	Total Injuries	Total Injuries Description	Total Damage ($Mil)	Total Damage Description	H Des
0	[]	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	
1	NaN	-2150.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	JORDAN: BAB-A-DARAA,AL-KARAK	...	NaN	NaN	NaN	NaN	NaN	NaN	
2	NaN	-2000.0	NaN	NaN	NaN	NaN	NaN	1.0	NaN	SYRIA: UGARIT	...	NaN	NaN	NaN	NaN	NaN	NaN	
3	NaN	-2000.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	TURKMENISTAN: W	...	NaN	NaN	NaN	NaN	NaN	1.0	
4	NaN	-1610.0	NaN	NaN	NaN	NaN	NaN	3.0	1351.0	GREECE: THERA ISLAND (SANTORINI)	...	NaN	NaN	NaN	NaN	NaN	3.0	

5 rows × 39 columns

[9]:
```python
#把国家的信息提取出来
Sig_Eqs['Country'] = Sig_Eqs['Location Name'].str.split(":").str[0]
Sig_Eqs.head()
```

[9]:

	Search Parameters	Year	Mo	Dy	Hr	Mn	Sec	Tsu	Vol	Location Name	...	Total Missing Description	Total Injuries	Total Injuries Description	Total Damage ($Mil)	Total Damage Description	Total Houses Destroyed	D
0	[]	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	
1	NaN	-2150.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	JORDAN: BAB-A-DARAA,AL-KARAK	...	NaN	NaN	NaN	NaN	NaN	NaN	
2	NaN	-2000.0	NaN	NaN	NaN	NaN	NaN	1.0	NaN	SYRIA: UGARIT	...	NaN	NaN	NaN	NaN	NaN	NaN	
3	NaN	-2000.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	TURKMENISTAN: W	...	NaN	NaN	NaN	NaN	1.0	NaN	
4	NaN	-1610.0	NaN	NaN	NaN	NaN	NaN	3.0	1351.0	GREECE: THERA ISLAND (SANTORINI)	...	NaN	NaN	NaN	NaN	3.0	NaN	

5 rows × 40 columns

[11]:
```python
total_death = Sig_Eqs.groupby('Country')['Deaths'].sum().reset_index()
print(total_death.sort_values('Deaths', ascending=False)[0:20])
```

```
          Country     Deaths
58          CHINA  2075947.0
319        TURKEY  1148745.0
140          IRAN   995410.0
148         ITALY   498418.0
295         SYRIA   369224.0
119         HAITI   323478.0
23     AZERBAIJAN   317219.0
152         JAPAN   278607.0
17        ARMENIA   191890.0
146        ISRAEL   160120.0
233      PAKISTAN   145080.0
82        ECUADOR   135496.0
143          IRAQ   120200.0
323  TURKMENISTAN   117412.0
241          PERU   101461.0
248      PORTUGAL    83547.0
104        GREECE    80482.0
56          CHILE    64270.0
131         INDIA    61960.0
298        TAIWAN    57152.0
```

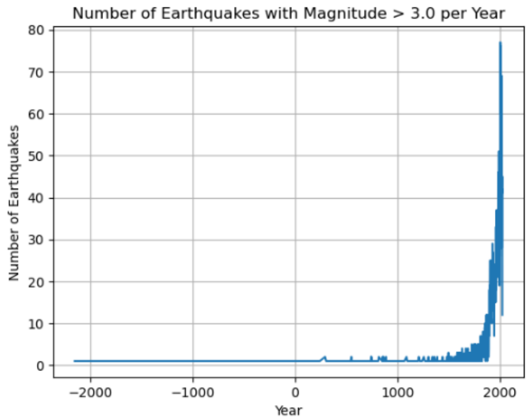
#由于台湾属于中国，需要对列表进行修改

[43]:
```python
#由于台湾属于中国，需要对列表进行修改
Sig_Eqs.loc[Sig_Eqs['Country'] == 'TAIWAN', 'Country'] = 'CHINA'

total_death = total_death.groupby('Country')['Deaths'].sum().reset_index()
print(total_death.sort_values('Deaths', ascending=False)[0:20])
```

```
          Country     Deaths
58          CHINA  2133099.0
318        TURKEY  1148745.0
140          IRAN   995410.0
148         ITALY   498418.0
295         SYRIA   369224.0
119         HAITI   323478.0
23     AZERBAIJAN   317219.0
152         JAPAN   278607.0
17        ARMENIA   191890.0
146        ISRAEL   160120.0
233      PAKISTAN   145080.0
82        ECUADOR   135496.0
143          IRAQ   120200.0
322  TURKMENISTAN   117412.0
241          PERU   101461.0
248      PORTUGAL    83547.0
104        GREECE    80482.0
56          CHILE    64270.0
131         INDIA    61960.0
317       TUNISIA    48013.0
```

1.2

```
[11]: large_earthquakes = Sig_Eqs[Sig_Eqs['Mag'] > 3.0]
total_earthquakes = large_earthquakes['Year'].value_counts().sort_index()
total_earthquakes.plot(kind='line')
plt.title('Number of Earthquakes with Magnitude > 3.0 per Year')
plt.xlabel('Year')
plt.ylabel('Number of Earthquakes')
plt.grid()
plt.show()
```



1000 年后地震发生的数量变多，特别是 2000 年后，
这有可能是由于地质与人类活动影响导致，
也可能是随着社会发展，能够统计到地震的技术发展，更多的地震被人类观测统计到。
1.3

```
[19]: #1.3 [10 points]
#Write a function CountEq_LargestEq that returns (1) the total number of earthquakes since 2150 B.C. in a given country
#AND (2) date and location 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.
```

```
[21]: def CountEq_LargestEq(country_name, Sig_Eqs):
    country_eqs = Sig_Eqs[Sig_Eqs['Country'] == country_name]
    if not country_eqs.empty:
        earthquake_counts = len(country_eqs)
        max_index = country_eqs['Mag'].idxmax()        # 找到最大震级的索引
        max_date = Sig_Eqs.loc[max_index, 'Year']
        max_location = Sig_Eqs.loc[max_index, 'Location Name']
    return earthquake_counts, max_date, max_location
country_name = input("请输入查找的国家名: ")
earthquake_counts, max_date, max_location = CountEq_LargestEq(country_name, Sig_Eqs)
print(f"{country_name}有史以来发生发生的地震总数为: {earthquake_counts}, 发生最大地震的日期为: {max_date}, 位置为: {max_location}")
```

结果示例：

```
请输入查找的国家名:  ITALY
ITALY有史以来发生发生的地震总数为: 332, 发生最大地震的日期为: 1915.0, 位置为: ITALY:  MARSICA, AVEZZANO, ABRUZZI
```

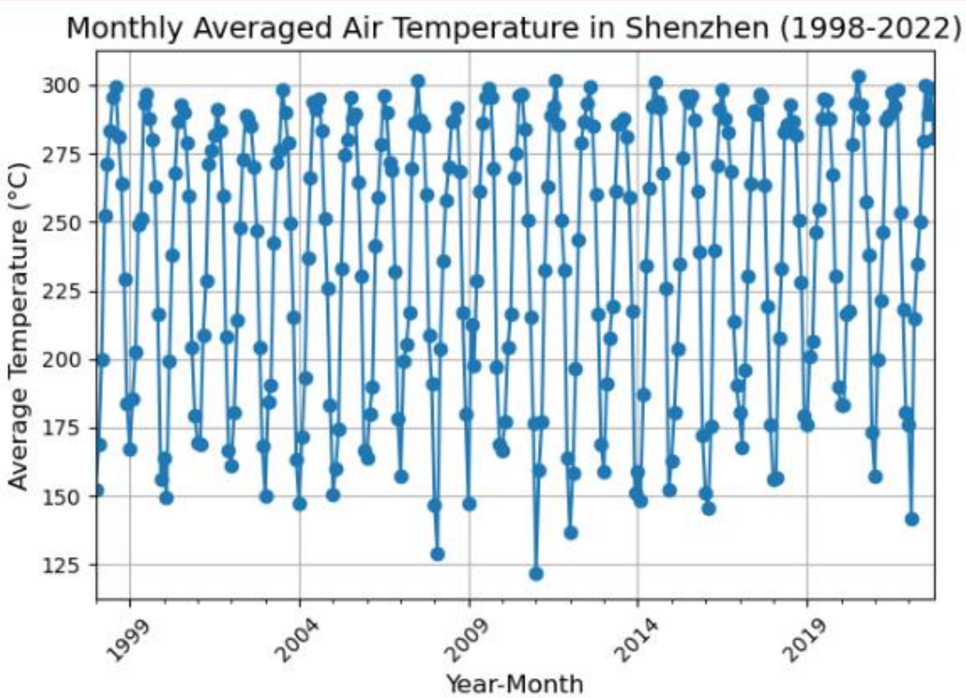
```
请输入查找的国家名:  CHINA
CHINA有史以来发生发生的地震总数为: 721, 发生最大地震的日期为: 1668.0, 位置为: CHINA:  SHANDONG PROVINCE
```

2. 在过去 25 年中，月平均气温变化随年份趋势相对稳定

[249]:
```python
import pandas as pd

data = pd.read_csv('E:/课程/ESE_COURSE/作业2/Baoan_Weather_1998_2022.csv',
                  #sep=',',
                  header=0,
                  parse_dates=['DATE'],  # Parse the date column as datetime
                  na_values=['', 'NaN'],  # Specify additional strings to recognize as NA
                  skipinitialspace=True,  # Skip spaces after delimiter
                 low_memory=False)

print(data.head())
```

```
       STATION                DATE  SOURCE REPORT_TYPE CALL_SIGN  \
0  59493099999 1998-01-01 00:00:00       4       SY-MT     ZGSZ
1  59493099999 1998-01-01 01:00:00       4       FM-15     ZGSZ
2  59493099999 1998-01-01 02:00:00       4       FM-15     ZGSZ
3  59493099999 1998-01-01 03:00:00       4       SY-MT     ZGSZ
4  59493099999 1998-01-01 04:00:00       4       FM-15     ZGSZ

  QUALITY_CONTROL           AA1  AA2  AA3    AG1  ... REPORT_TYPE.1  SA1  \
0            V020  06,0000,9,1  NaN  NaN  0,000  ...          SY-MT  NaN
1            V020           NaN  NaN  NaN  0,999  ...          FM-15  NaN
2            V020           NaN  NaN  NaN  0,999  ...          FM-15  NaN
3            V020           NaN  NaN  NaN  0,000  ...          SY-MT  NaN
4            V020           NaN  NaN  NaN  0,999  ...          FM-15  NaN

       SLP SOURCE.1      TMP  UA1  UG1          VIS WG1            WND
0  10184,1        4  +0186,1  NaN  NaN  008000,1,N,1 NaN  040,1,N,0040,1
1  99999,9        4  +0220,1  NaN  NaN  003300,1,N,1 NaN  130,1,N,0020,1
2  99999,9        4  +0240,1  NaN  NaN  003500,1,N,1 NaN  110,1,N,0020,1
```

[250]:
```python
# 过滤数据，只保留有效的温度值
data[['TMP', 'QUA']] = data['TMP'].str.split(',', expand=True)
data = data[(data['TMP'] != '+9999') & (data['QUA'].isin(['1', '5']))]

# 清理TMP，删除不需要的字符，并转换为浮点数
data['TMP'] = data['TMP'].replace({r'[,+]': ''}, regex=True).astype(float)
# 设置日期为索引
data['DATE'] = pd.to_datetime(data['DATE'])

# 计算每月的平均气温
temperature_data = data[['DATE', 'TMP']]
temperature_data['Year-Month'] = temperature_data['DATE'].dt.to_period('M')
monthly_avg_temp = temperature_data.groupby('Year-Month')['TMP'].mean()

#画图
monthly_avg_temp.plot( marker='o', linestyle='-')
plt.title('Monthly Averaged Air Temperature in Shenzhen (1998-2022)', fontsize=14)
plt.xlabel('Year-Month', fontsize=12)
plt.ylabel('Average Temperature (°C)', fontsize=12)
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



3.

[57]:
```
#3. Global collection of hurricanes
#The International Best Track Archive for Climate Stewardship (IBTrACS) project is the most complete global collection of tropical cyclones available.
#It merges recent and historical tropical cyclone data from multiple agencies to create a unified, publicly available, best-track dataset that improves i
#IBTrACS was developed collaboratively with all the World Meteorological Organization (WMO) Regional Specialized Meteorological Centers, as well as other

#In this problem set, we will use all storms available in the IBTrACS record since 1842.
#Download the file ibtracs.ALL.list.v04r00.csv, move the .csv file to your working directory.
#Read Column Variable Descriptions for variables in the file.
#Examine the first few lines of the file.

#Below we provide an example to load the file as a pandas dataframe.
#Think about the options being used and why, and modify when necessary.
```

[59]:
```
df = pd.read_csv('E:/课程/ESE_COURSE/作业2/ibtracs.ALL.list.v04r00.csv',
                 usecols=range(17),
                 skiprows=[1, 2],
                 parse_dates=['ISO_TIME'],
                 na_values=['NOT_NAMED', 'NAME'],
                low_memory=False)
df.head()
```

[59]:

	SID	SEASON	NUMBER	BASIN	SUBBASIN	NAME	ISO_TIME	NATURE	LAT	LON	WMO_WIND	WMO_PRES	WMO_AGENCY	TRACK_TYPE
	1842298N11080	1842	1	NI	BB	NaN	1842-10-25 06:00:00	NR	10.8709	79.8265				main
	1842298N11080	1842	1	NI	BB	NaN	1842-10-25 09:00:00	NR	10.8431	79.3524				main
	1842298N11080	1842	1	NI	BB	NaN	1842-10-25 12:00:00	NR	10.8188	78.8772				main
	1842298N11080	1842	1	NI	BB	NaN	1842-10-25 15:00:00	NR	10.8000	78.4000				main
	1842298N11080	1842	1	NI	AS	NaN	1842-10-25 18:00:00	NR	10.7884	77.9194				main

3.1

[61]:
```
#3.1 [5 points] Group the data on Storm Identifie (SID), report names (NAME) of the 10 largest hurricanes according to wind speed (WMO_WIND).
```

[63]:
```
df['WMO_WIND'] = pd.to_numeric(df['WMO_WIND'], errors='coerce')
largest_hurricanes = df.groupby('SID').agg({'WMO_WIND': 'max', 'NAME': 'first'}).sort_values('WMO_WIND', ascending=False)
top_10 = largest_hurricanes.head(10)
top_10
```

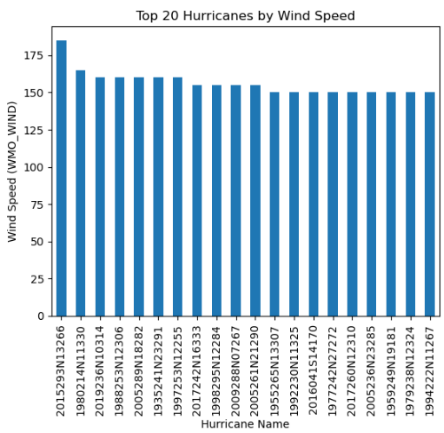
[63]:

SID	WMO_WIND	NAME
2015293N13266	185.0	PATRICIA
1980214N11330	165.0	ALLEN
2019236N10314	160.0	DORIAN
1988253N12306	160.0	GILBERT
2005289N18282	160.0	WILMA
1935241N23291	160.0	None
1997253N12255	160.0	LINDA
2017242N16333	155.0	IRMA
1998295N12284	155.0	MITCH
2009288N07267	155.0	RICK

3.2

[65]:
```
#3.2 [5 points] Make a bar chart of the wind speed (WMO_WIND) of the 20 strongest-wind hurricanes.
```

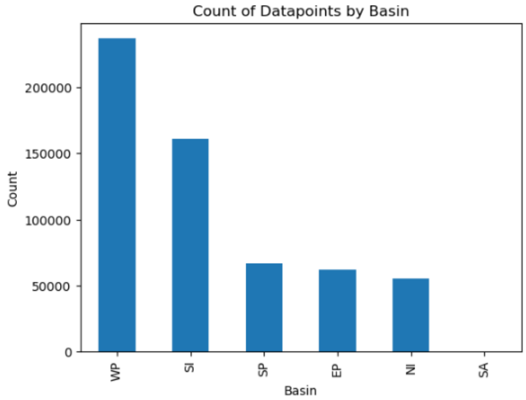
[67]:
```
top_20 = largest_hurricanes.head(20)
top_20['WMO_WIND'].plot(kind='bar')
plt.title('Top 20 Hurricanes by Wind Speed')
plt.xlabel('Hurricane Name')
plt.ylabel('Wind Speed (WMO_WIND)')
plt.show()
```



3.3

```
[69]: #3.3 [5 points] Plot the count of all datapoints by Basin as a bar chart.
```

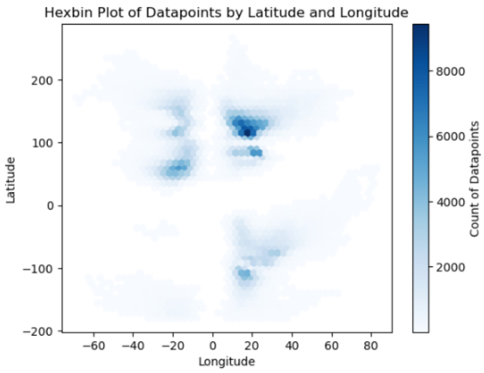
```
[71]: df['BASIN'].value_counts().plot(kind='bar')
      plt.title('Count of Datapoints by Basin')
      plt.xlabel('Basin')
      plt.ylabel('Count')
      plt.show()
```



3.4

```
[73]: #3.4 [5 points] Make a hexbin plot of the location of datapoints in Latitude and Longitude.
```

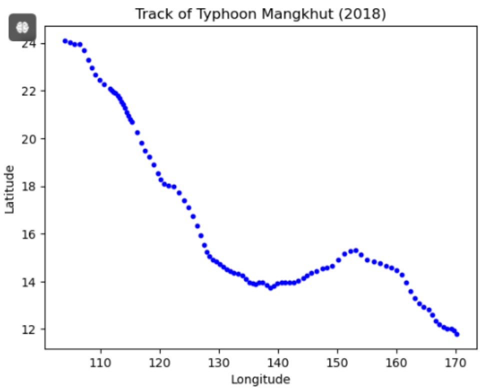
```
[75]: plt.hexbin(df['LAT'], df['LON'], gridsize=50, cmap='Blues', mincnt=1)
      plt.colorbar(label='Count of Datapoints')
      plt.xlabel('Longitude')
      plt.ylabel('Latitude')
      plt.title('Hexbin Plot of Datapoints by Latitude and Longitude')
      plt.show()
```



3.5

```
[77]: #3.5 [5 points] Find Typhoon Mangkhut (from 2018) and plot its track as a scatter plot.
```

```
[79]: # 筛选2018年的台风“山竹”
      mangkhut = df[(df['NAME'] == 'MANGKHUT') & (df['SEASON'] == 2018)]
      plt.scatter(mangkhut['LON'], mangkhut['LAT'], c='blue', s=10)
      plt.xlabel('Longitude')
      plt.ylabel('Latitude')
      plt.title('Track of Typhoon Mangkhut (2018)')
      plt.show()
```



3.6

```
[81]: #3.6 [5 points] Create a filtered dataframe that contains only data since 1970 from the Western North Pacific ("WP") and Eastern North Pacific ("EP") Bas
      #Use this for the rest of the problem set.
```

```
[83]: # 筛选1970年以后的数据
      filtered_df = df[(df['SEASON'] >= 1970) & (df['BASIN'].isin(['WP', 'EP']))]
      filtered_df
```

[83]:

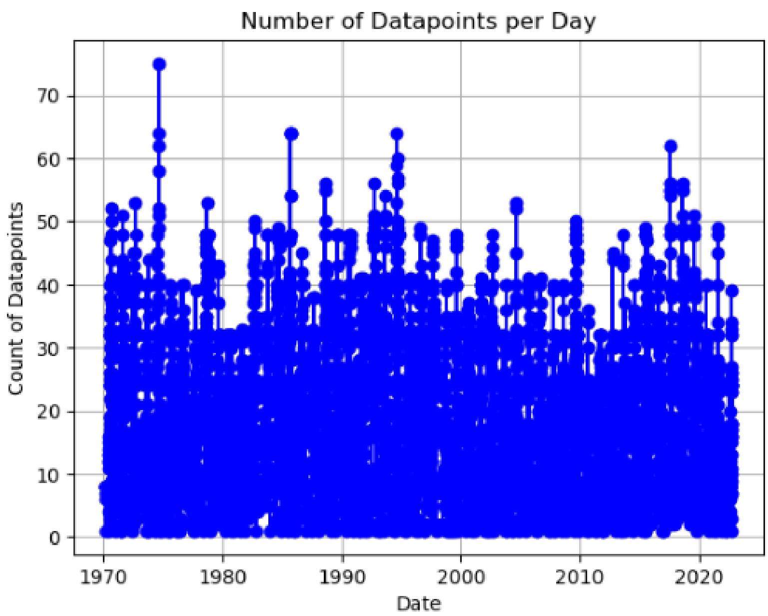
	SID	SEASON	NUMBER	BASIN	SUBBASIN	NAME	ISO_TIME	NATURE	LAT	LON	WMO_WIND	WMO_PRES	WMO_AGENCY	T
350393	1970050N07151	1970	22	WP	MM	NANCY	1970-02-19 00:00:00	TS	7.00000	151.400	NaN	1006	tokyo	
350394	1970050N07151	1970	22	WP	MM	NANCY	1970-02-19 03:00:00	TS	7.24752	151.205	NaN			
350395	1970050N07151	1970	22	WP	MM	NANCY	1970-02-19 06:00:00	TS	7.50000	151.000	NaN	1002	tokyo	
350396	1970050N07151	1970	22	WP	MM	NANCY	1970-02-19 09:00:00	TS	7.75747	150.772	NaN			
350397	1970050N07151	1970	22	WP	MM	NANCY	1970-02-19 12:00:00	TS	8.00000	150.500	NaN	998	tokyo	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
707084	2022275N10316	2022	76	EP	MM	JULIA	2022-10-10 15:00:00	TS	13.99570	-90.294	NaN			P
707085	2022275N10316	2022	76	EP	MM	JULIA	2022-10-10 18:00:00	NR	14.50000	-91.000	NaN			P
707173	2022286N15151	2022	80	WP	MM	NaN	2022-10-12 12:00:00	NR	15.20000	151.300	NaN			P
707174	2022286N15151	2022	80	WP	MM	NaN	2022-10-12 15:00:00	NR	15.05000	151.325	NaN			P
707175	2022286N15151	2022	80	WP	MM	NaN	2022-10-12 18:00:00	NR	14.90000	151.350	NaN			P

176352 rows × 17 columns

3.7

```
[85]: #3.7 [5 points] Plot the number of datapoints per day
```

```
[87]: # 将ISO_TIME转换为日期
      filtered_df['ISO_DATE'] = pd.to_datetime(filtered_df['ISO_TIME']).dt.date
      # 按天统计数据点数量
      daily_counts = filtered_df['ISO_DATE'].value_counts().sort_index()
      # 创建图表
      plt.plot(daily_counts.index, daily_counts.values, marker='o', linestyle='-', color='blue')
      plt.xlabel('Date')
      plt.ylabel('Count of Datapoints')
      plt.title('Number of Datapoints per Day')
      plt.grid(True)
      plt.show()
```

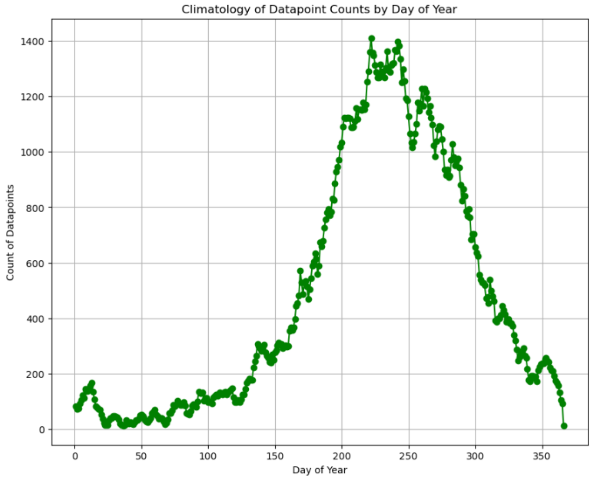


3.8

```
[88]: #3.8 [5 points] Calculate the climatology of datapoint counts as a function of day of year.
      #The day of year is the sequential day number starting with day 1 on January 1st.
```

```
[91]: # 将ISO_DATE转换为年日
      filtered_df['DAY_OF_YEAR'] = pd.to_datetime(filtered_df['ISO_DATE']).dt.dayofyear
      datapoint_counts = filtered_df.groupby(['DAY_OF_YEAR']).size()

      # 计算每天的平均数据点数量以得到气候学数据
      climatology = datapoint_counts.groupby(datapoint_counts.index).mean()
      print(climatology)

      # 创建图表
      plt.figure(figsize=(10, 8))
      plt.plot(climatology.index, climatology.values, marker='o', linestyle='-', color='green')
      plt.xlabel('Day of Year')
      plt.ylabel('Count of Datapoints')
      plt.title('Climatology of Datapoint Counts by Day of Year')
      plt.grid(True)
      plt.show()
```

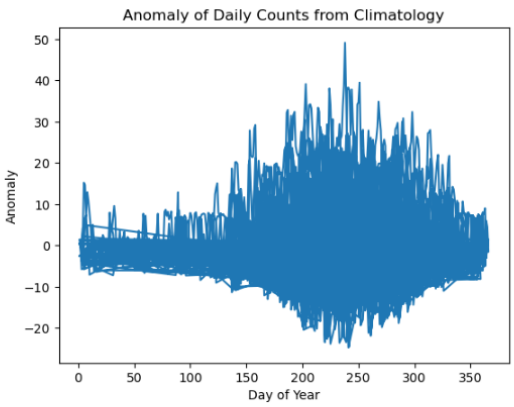
```
DAY_OF_YEAR
1       83.0
2       72.0
3       74.0
4       93.0
5      105.0
       ...
362    158.0
363    132.0
364    104.0
365     93.0
366     13.0
Length: 366, dtype: float64
```



3.9

```
[93]: #3.9 [5 points] Calculate the anomaly of daily counts from the climatology.
```

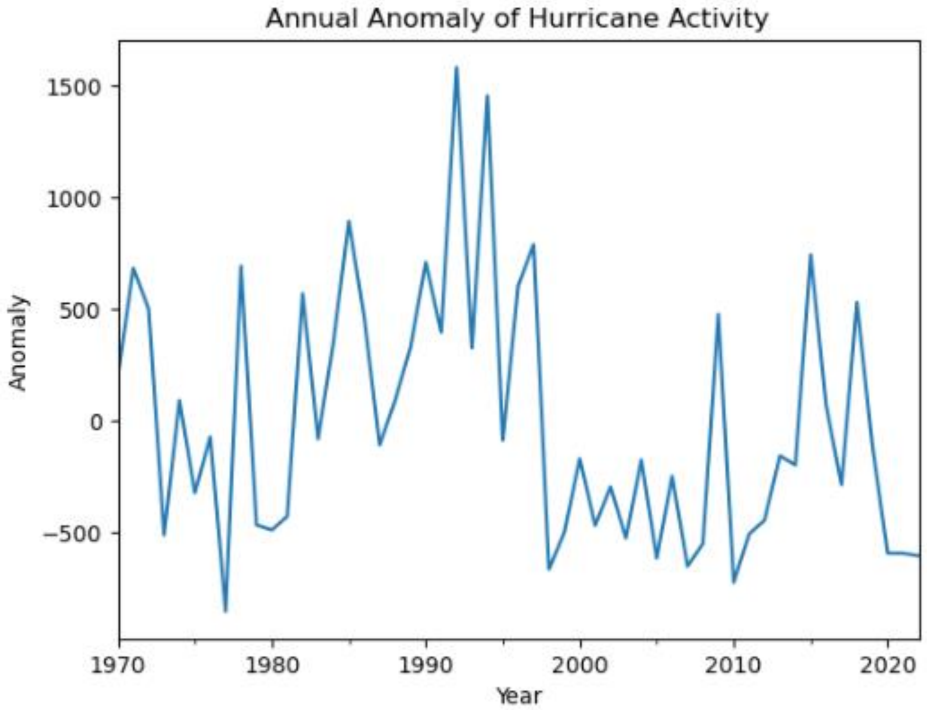
```
[95]: daily_counts = filtered_df['ISO_TIME'].dt.date.value_counts().sort_index()
      daily_counts_df = daily_counts.reset_index()
      daily_counts_df.columns = ['date', 'count']
      daily_counts_df['day_of_year'] = pd.to_datetime(daily_counts_df['date']).dt.dayofyear
      climatology = daily_counts_df.groupby('day_of_year')['count'].mean()
      daily_counts_df = daily_counts_df.set_index('day_of_year').join(climatology.rename('climatology'), on='day_of_year', how='left')
      daily_counts_df['anomaly'] = daily_counts_df['count'] - daily_counts_df['climatology']
      daily_counts_df['anomaly'].plot()
      plt.title('Anomaly of Daily Counts from Climatology')
      plt.xlabel('Day of Year')
      plt.ylabel('Anomaly')
      plt.show()
```



3.10 从图中看 1990~1995 年份的飓风活动异常突出

```
[96]: #3.10 [5 points] Resample the anomaly timeseries at annual resolution and plot.
      #So which years stand out as having anomalous hurricane activity
```

```
[99]: daily_counts_df['date'] = pd.to_datetime(daily_counts_df['date'])
      daily_counts_df.set_index('date', inplace=True)
      annual_anomalies = daily_counts_df['anomaly'].resample('Y').sum()

      annual_anomalies.plot()
      plt.title('Annual Anomaly of Hurricane Activity')
      plt.xlabel('Year')
      plt.ylabel('Anomaly')
      plt.show()
```



4. 由于在网站上下载的是 txt 文件，先把文件按内容格式提取成为转化成 csv 文件

[101]:
```
#4. Explore a data set
#Browse the National Centers for Environmental Information (NCEI) or Advanced Global Atmospheric Gases Experiment (AGAGE) website.
#Search and download a data set you are interested in. You are also welcome to use data from your group in this problem set.
#But the data set should be in csv, XLS, or XLSX format, and have temporal information.
```

[255]:
```
# 读取TXT文件内容
with open('global_mean_md.txt', 'r') as file:
    lines = file.readlines()

# 处理数据，转换为CSV格式
csv_lines = []
for line in lines:
    # 数据列之间使用制表符或多个空格分隔，分割每一行的数据
    columns = line.strip().split()
    # 将分割后的数据用逗号连接，形成CSV格式的一行
    csv_line = ','.join(columns) + '\n'
    csv_lines.append(csv_line)

# 将转换后的数据写入CSV文件
with open('global_mean_md.csv', 'w') as csv_file:
    csv_file.writelines(csv_lines)

print(f"文件已成功转换并保存为: {csv_file_path}")
```
```
文件已成功转换并保存为: global_mean_md.csv
```

4.1

[105]:
```
#4.1 [5 points] Load the csv, XLS, or XLSX file, and clean possible data points with missing values or bad quality.
```

[257]:
```
data = pd.read_csv('global_mean_md.csv',
                   usecols=range(17),
                   skiprows=[1, 2],
                   na_values=['NOT_NAMED', 'NAME'],
                   low_memory=False)
data
```

[257]:

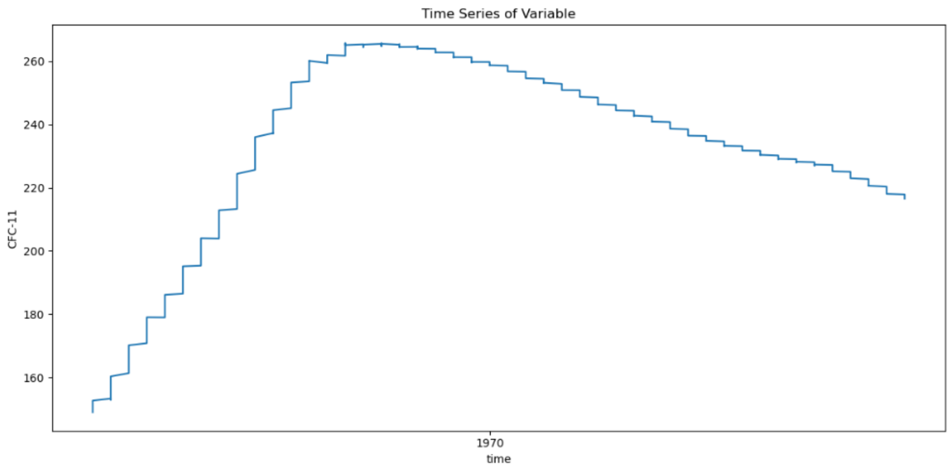
	time	MM	YYYY	CFC-11	---	CFC-12	---.1	CFC-113	---.2	CH3CCl3	---.3	CCl4	---.4	N2O	---.5	CH4	---.6
0	1978.708	9	1978	148.925	5.412	271.445	9.476	0.0	0.0	69.845	8.760	89.393	1.479	299.889	0.490	0.000	0.000
1	1978.792	10	1978	149.670	5.189	273.775	9.159	0.0	0.0	70.845	8.261	88.964	1.188	300.557	0.582	0.000	0.000
2	1978.875	11	1978	150.647	4.904	276.780	7.185	0.0	0.0	71.945	7.771	88.941	0.657	300.598	0.467	0.000	0.000
3	1978.958	12	1978	152.607	5.623	280.265	8.520	0.0	0.0	74.768	9.833	90.429	1.471	300.255	0.528	0.000	0.000
4	1979.042	1	1979	153.245	4.635	283.425	8.146	0.0	0.0	74.405	8.975	91.107	1.915	301.152	0.463	0.000	0.000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
533	2023.125	2	2023	217.640	0.329	490.787	0.288	-99.0	-99.0	1.049	0.033	73.970	0.415	336.866	0.315	1915.755	39.237
534	2023.208	3	2023	217.426	0.321	490.525	0.269	-99.0	-99.0	1.038	0.034	73.898	0.403	336.909	0.370	1918.005	40.217
535	2023.292	4	2023	217.103	0.376	490.002	0.287	-99.0	-99.0	1.032	0.037	73.820	0.435	336.913	0.385	1916.914	38.491
536	2023.375	5	2023	216.826	0.351	489.524	0.232	-99.0	-99.0	1.021	0.035	73.717	0.493	336.933	0.435	1912.802	32.088
537	2023.458	6	2023	216.568	0.335	489.022	0.246	-99.0	-99.0	1.012	0.034	73.602	0.524	337.045	0.474	1915.006	28.981

538 rows × 17 columns

4.2

[127]:
```
#4.2 [5 points] Plot the time series of a certain variable.
```

[259]:
```
data['time'] = pd.to_datetime(data['time'])  # Ensure date is in datetime format
plt.figure(figsize=(12, 6))
plt.plot(data['time'], data['CFC-11'])
plt.title('Time Series of Variable')
plt.xlabel('time')
plt.ylabel('CFC-11')
plt.tight_layout()
plt.show()
```



4.3

[131]:
```
#4.3 [5 points] Conduct at least 5 simple statistical checks with the variable, and report your findings.
```

[261]:
```
mean_cf11 = data['CFC-11'].mean()
median_cf11 = data['CFC-11'].median()
max_cf11 = data['CFC-11'].max()
min_cf11 = data['CFC-11'].min()
std_cf11 = data['CFC-11'].std()

print(f"Mean of CFC-11: {mean_cf11}")
print(f"Median of CFC-11: {median_cf11}")
print(f"Maximum of CFC-11: {max_cf11}")
print(f"Minimum of CFC-11: {min_cf11}")
print(f"Standard Deviation of CFC-11: {std_cf11}")
```

```
Mean of CFC-11: 233.99916356877324
Median of CFC-11: 238.9685
Maximum of CFC-11: 265.743
Minimum of CFC-11: 148.925
Standard Deviation of CFC-11: 28.299559667156423
```