

# Assignment 02

基本的环境准备

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import matplotlib as mpl
5 import datetime
6 from io import StringIO
7 import matplotlib.dates as mdates
8 import matplotlib.units as munits
9 converter = mdates.ConciseDateConverter()
10 munits.registry[np.datetime64] = converter
11 munits.registry[datetime.date] = converter
12 munits.registry[datetime.datetime] = converter
```

## 1. Significant earthquakes since 2150 B.C.

The Significant Earthquake Database contains information on destructive earthquakes from 2150 B.C. to the present. On the top left corner, select all columns and download the entire significant earthquake data file in .tsv format by clicking the Download TSV File button. Click the variable name for more information. Read the file (e.g., earthquakes-2025-10-29\_15-11-32\_+0800.tsv) as an object and name it Sig\_Eqs.

### 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.

依据题目要求，对下载的数据表格进行处理，主要需要从 Location Name 中提取地震发生所在的国家，官网的说明中，其结构为 Country State City，但在实际的数据中存在多种格式，因此这里通过分隔符 : 和 ; 对所在国家进行初步提取。对初步提取的结果，通过 AI 和人工修正的方式决定其准确所在国家。另外，由于一场地震可能涉及多个国家，但并没有额外的数据可以用于衡量每个国家的损失大小，因此这里进行平均分配。

```
1 Sig_Eqs = pd.read_table("./data/earthquakes-2025-10-29_20-30-02_+0800.tsv") # 读取数据库
2 Sig_Eqs = Sig_Eqs.iloc[1:] # 跳过第一行
3 Sig_Eqs.reset_index(inplace=True) # 重置索引
4
5 def get_base_from_location(location: str) -> list:
6     """
7     # Parameters:
8     location, str    : Location Name in earthquakes dataset
9     # Returns:
10    List of countrys
11    # Note
12    依照数据库的规则 (country: region) 进行基本的区分，将给定位置的国家提取出来。
13    实际会有一些例外情况，例如：
14    - Barbados, saint vincent, dominica, antigua (四个国家，但并未通过 `;` 分隔)
```

```

15     - Armenia-azerbaijan-iran (三个国家, 但通过连字符连接, 而非分号分隔)
16     - Boso, japan: Japan city (国家名后置, 且不使用冒号分隔国家和地区)
17     - Boston and salem, massachusetts (没有给出所在国家, 仅给了州或县, 在美国、加拿大、墨西哥、日本等地常出现)
18     - East mediterranean sea (没有所在国家)
19     """
20
21     country_list = []
22     location = location.strip() # 去除多余的空格
23     if ";" in location: # 使用 ; 分隔文本, 返回列表
24         location_list = location.split(';')
25     else:
26         location_list = [location]
27     for location_tmp in location_list: # 对列表进行循环, 去除 : 后的内容, 并去除多余的空格
28         if len(location_tmp) > 0: # 排除为空的情况
29             if ":" in location_tmp:
30                 country_list += [location_tmp.split(':')[0].strip()]
31             else:
32                 country_list += [location_tmp.strip()]
33
34     # 获取每个位置的名称, 并依据最基本的规则获取所有可能的国家 (存在很大误差)
35     df = pd.DataFrame()
36     for i in Sig_Eqs.index:
37         location = Sig_Eqs.loc[i, "Location Name"]
38         countrys = get_base_from_location(location)
39         for country in countrys:
40             if not country in df.index:
41                 df.loc[country, "Location Name"] = location
42     df.to_csv("./data/PS2_country.csv")
43
44     # 将 PS2_country.csv 输入 AI, 然后对 AI 回答的结果进行人工校对, 得到最终的对照表 earthquakes-
45     # country.xlsx
46     # 根据对照表获取一份字典 country_info_dict, 对应 get_base_from_location 输出的位置和实际所在的国家
47     country_info = pd.read_excel("./data/earthquakes-country.xlsx")
48     country_info = country_info.set_index("Name")
49     country_info_dict = {x.upper(): country_info.loc[x, "Country"].split(", ") for x in
50     country_info.index}
51
52     def get_country_from_location(location: str) -> list:
53         """
54         # Parameters:
55             location, str    : Location Name in earthquakes dataset
56         # Returns:
57             List of countrys
58         # Note
59         依据修正后的对照表对国家名称进行匹配
60         """
61
62         country_list = []
63         location = location.strip()
64         if ";" in location:
65             location_list = location.split(';')

```

```

63     else:
64         location_list = [location]
65     for location_tmp in location_list:
66         location_tmp = location_tmp.strip()
67         if len(location_tmp) > 0:
68             try:
69                 if ":" in location_tmp:
70                     country_list += country_info_dict[location_tmp.split(':')[0].strip()]
71             else:
72                 country_list += country_info_dict[location_tmp.strip()]
73     except:
74         print(location, '|', location_tmp)
75         raise IndexError
76     return country_list
77
78 Sig_Eqs_list = [] # 用于保存一场地震对应多个国家的情景，最终会被合并到原列表中
79 num_list = ["Deaths", "Missing", "Injuries", "Damage ($Mil)", "Houses Destroyed",
80 "Houses Damaged", "Total Deaths", "Total Missing", "Total Injuries", "Total Damage
81 ($Mil)", "Total Houses Destroyed", "Total Houses Damaged"] # 将同一场地震分配到不同国家,
82 需要计算的列
83 index_other = len(Sig_Eqs) # 用于确认被分出来的行具体所在行
84 Sig_Eqs_old = Sig_Eqs.copy() # 备份用于其他分析
85 for index, row in Sig_Eqs.iterrows():
86     country_list = get_country_from_location(row["Location Name"]) # 获取地震对应国家
87     n_country = len(country_list)
88     Sig_Eqs.loc[index, "Country"] = country_list[0] # 修改原始记录为第一个国家
89     Sig_Eqs.loc[index, num_list] /= n_country # 将地震损失/伤亡平均分配到每个国家
90     if len(country_list) > 1: # 如果国家数大于 1，则增加新的条目
91         for country_tmp in country_list[1:]:
92             df_tmp = Sig_Eqs.loc[index].copy()
93             df_tmp["Country"] = country_tmp # 修改新增条目的所在国家
94             df_tmp.name = index_other # 更新新增条目的所在行
95             Sig_Eqs_list.append(df_tmp) # 将新增条目加入列表
96             index_other += 1 # 递增所在行
97
98 Sig_Eqs = pd.concat([Sig_Eqs, pd.concat(Sig_Eqs_list, axis=1).T]) # 合并新增条目到原表格

```

按国家对死亡人数进行分组求和，并按照死亡人数排序得到前 10 个死亡人数最多的国家：

```

1  >>> Sig_Eqs.loc[:, ["Total Deaths",
2      "Country"]].groupby("Country").sum().sort_values("Total Deaths",
3      ascending=False).head(10)
4      Total Deaths
5      Country
6      China      2159641.0
7      Turkey     993261.0
8      Iran       828692.0
9      Syria      434409.0
10     Italy      423280.0
11     Haiti      323782.0
12     Japan      319294.0
13     Azerbaijan 318501.0
14     Indonesia   282898.0
15     Pakistan    154277.0

```

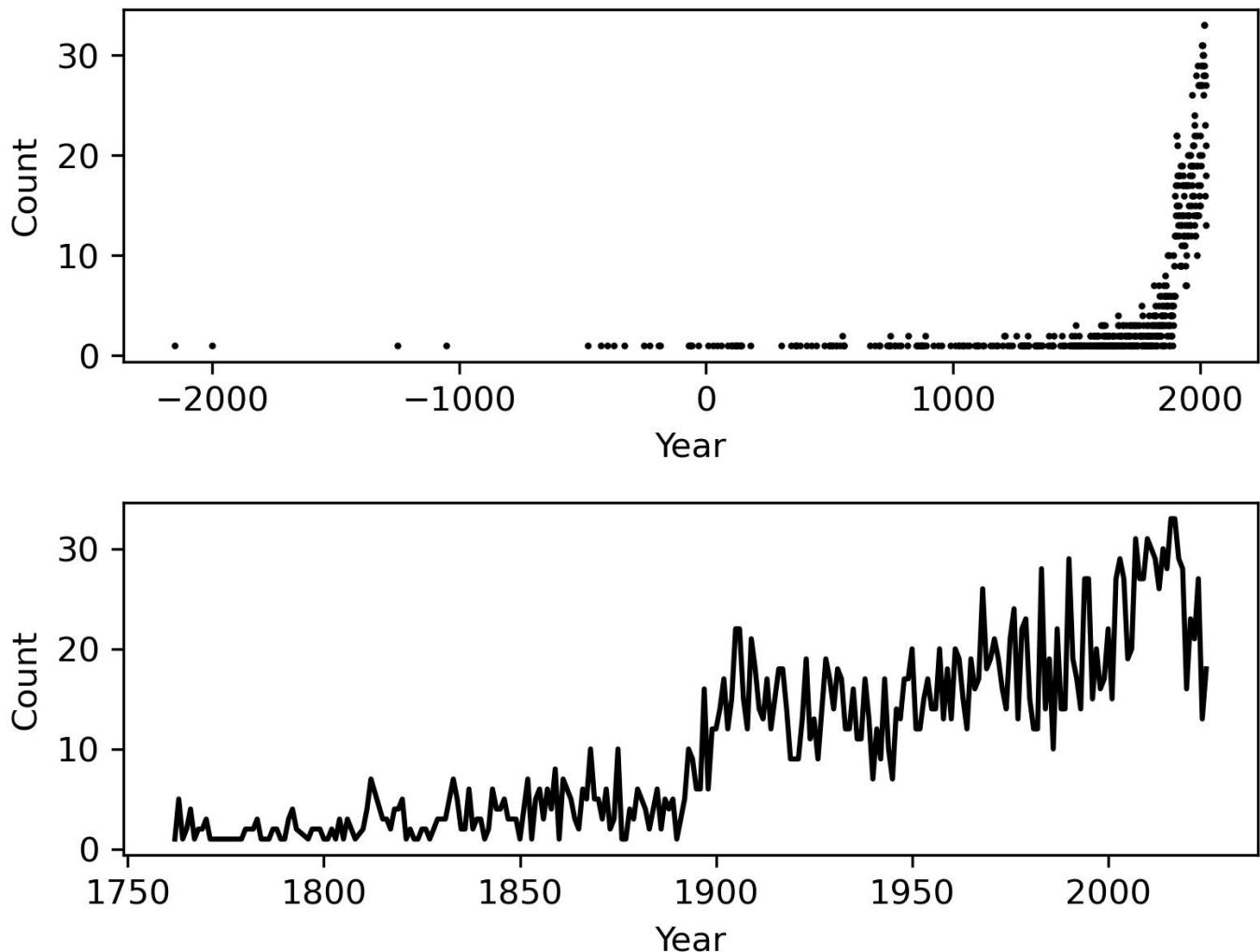
## 1.2

[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?

```

1  df = Sig_Eqs_old.loc[Sig_Eqs_old["Mag"]>6, ["Year", "Mag"]].groupby(["Year"]).count()
2  # 统计震级 6 级以上的地震数
3  x, y = np.array(df.index), np.array(df.loc[:, "Mag"])
4  fig = plt.figure(figsize=[6, 4.5], dpi=300)
5  spec = fig.add_gridspec(ncols=1, nrows=2, hspace=0.4)
6  # 绘制公元前 2150 年至今的趋势图
7  ax = fig.add_subplot(spec[0])
8  ax.plot(x, y, "ko", ms=1)
9  ax.set(xlabel="Year", ylabel="Count")
10 # 绘制近 250 年的趋势图
11 ax = fig.add_subplot(spec[1])
12 ax.plot(x[-250:], y[-250:], "k-", ms=1)
13 ax.set(xlabel="Year", ylabel="Count")
14 fig.savefig("./images/PS2_1_figure1.jpg", bbox_inches="tight")

```



从整体的趋势看，在公元前的地震记录非常稀疏，直到近几百年地震记录逐渐增加，这应当是由技术手段的改进导致对地震的探测能力增强，以往不能探测到的地震被探测到了，并不能代表地震数目的显著增加。绘制近 250 年的数据显示，6 级以上地震数目在 1900 年左右有显著的增加（官网在 Location Name 字段的说明下也说了 1900 年之前缺少仪器定位），然后长期稳定在每年 15 次左右。

### 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.

```

1  def CountEq(df: pd.DataFrame) -> int:
2      """
3          目的是为了计算地震的数目，当然这里写的并不严谨，因为之前按照国家对同一场地震进行了分配。不过考虑到最终计算时会对国家进行循环，因此在这里并无影响。
4      """
5      return len(df)
6
7  def LargestEq(df: pd.DataFrame) -> str:
8      """
9          计算最大地震所在的日期，并返回日期字符串。如果没有具体月份则设为当年 1 月，如果没有具体日则设为当月 1 日。

```

```

10     """
11     if not all(df["Mag"].isna()): # 如果不是所有的震级都是 NaN, 则获取对应的年月日, 并返回字符串
12         largest_row = df.loc[df["Mag"].idxmax(skipna=True)]
13         year = int(largest_row["Year"])
14         month = int(largest_row["Mo"]) if not np.isnan(largest_row["Mo"]) else 1
15         day = int(largest_row["Dy"]) if not np.isnan(largest_row["Dy"]) else 1
16         largest_date = f"{year:+05d}-{month:02d}-{day:02d}"
17     else: # 否则返回 None
18         largest_date = None
19     return largest_date
20
21 def CountEq_LargestEq(df):
22     return CountEq(df), LargestEq(df)
23
24 # 创建一个新的表格保存分析结果
25 Eq_info = pd.DataFrame()
26 for country in Sig_Eqs["Country"].unique(): # 对于所有国家循环, 计算地震总数和最大地震日期
27     Eq_info.loc[country, ["CountEq", "LargestEqDate"]] =
28     CountEq_LargestEq(Sig_Eqs.loc[Sig_Eqs["Country"]==country])

```

将结果按降序排列如下：

```

1 >>> print(Eq_info.sort_values(by="CountEq", ascending=False)) # 按照地震总数降序排列
2
3      CountEq  LargestEqDate
4
5      China        735  +1950-08-15
6      Japan        423  +2011-03-11
7  Indonesia        418  +2004-12-26
8      Iran         398  +0856-12-22
9      Turkey        375  +1939-12-26
10     ...
11     ...
12     ...
13     ...
14
15 [157 rows x 2 columns]

```

## 2. Wind speed in Shenzhen from 2010 to 2020

In this problem set, we will examine how wind speed changes in Shenzhen during the past 10 years, we will take a look at the hourly weather data measured at the BaoAn International Airport. The data set is from NOAA Integrated Surface Dataset. Download the file 2281305.zip, where the number 2281305 is the site ID. Extract the zip file, you should see a file named 2281305.csv. Save the .csv file to your working directory.

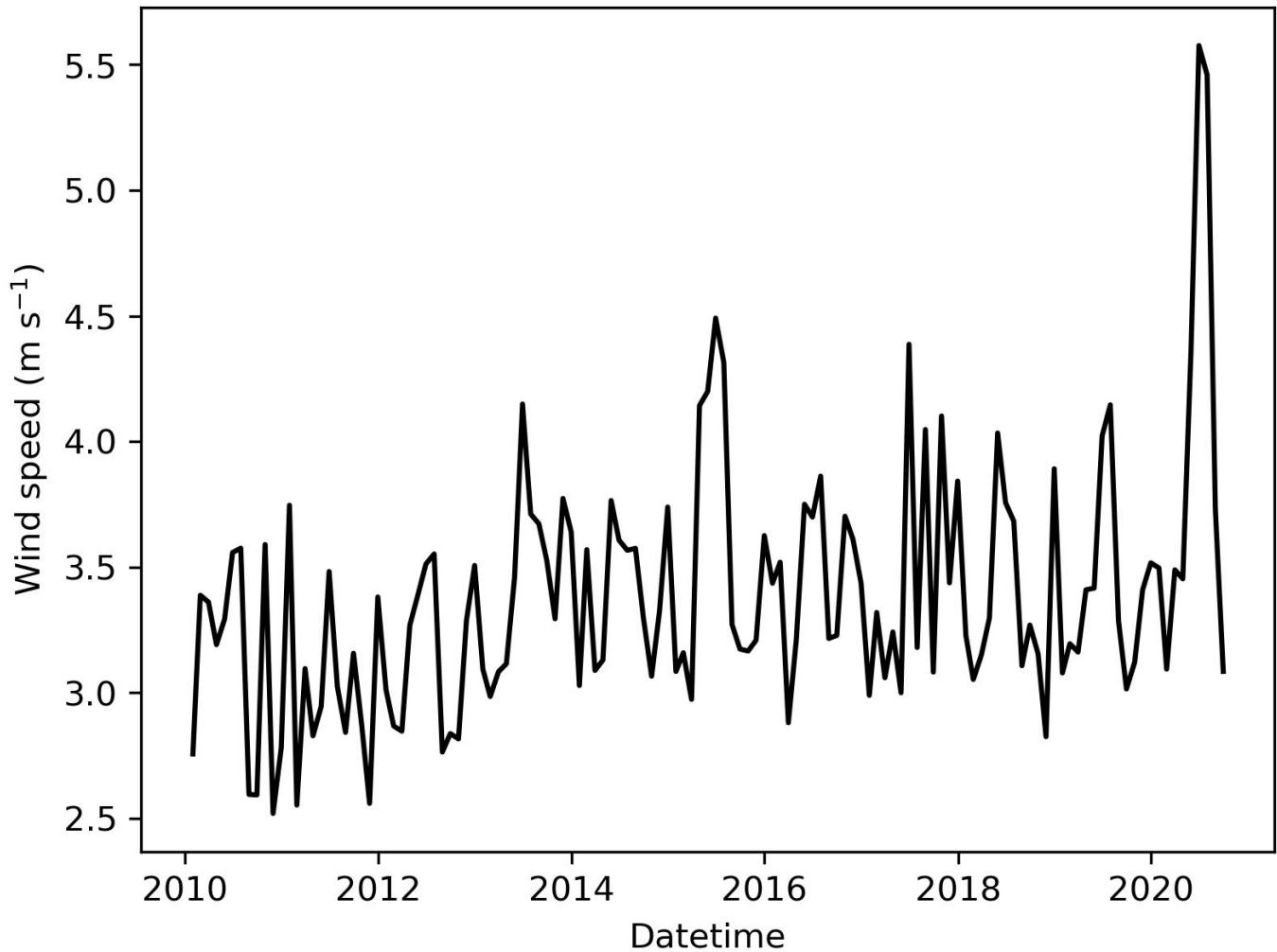
Read page 8-9 (POS 61-70) of the comprehensive user guide for the detailed format of the wind data. Explain how you filter the data in your report.

在 User Guide 中描述了风速相关数据的说明，其中所有数字为 9 的则表示数据存在问题。依据网络搜索得到的结果，在数据中风相关信息保存在列 WND 中，分别为"风向角度,风向质控,观测类型,风速,风速质控"，其中风速需要除以 10 转化为米每秒，并保留质控标记为 0、1、2、4、5、6、9 的结果（排除被怀疑或者错误的结果），并排除观测类型为 9 且风速不为 0 的结果（观测类型为 9 且风速不为 0 表示缺失，观测类型为 9 且风速为 0 表示静风状态）。

[10 points] Plot monthly averaged wind speed as a function of the observation time. Is there a trend in monthly averaged wind speed from 2010 to 2020?

```
1 df = pd.read_csv("./data/2281305.csv", parse_dates=[ "DATE" ], na_values=999).loc[:, ["STATION", "DATE", "WND"]]
2 wind_csv = StringIO("\n".join(df[ "WND" ].to_list()))
3 df_wind = pd.read_csv(wind_csv, names=[ "DIRECTION", "DIRECTION QUALITY", "OBSERVATION
TYPE", "SPEED", "SPEED QUALITY" ], na_values=[ 999, 9999 ])
4 df_wind[ "SPEED" ] /= 10
5 df_wind = pd.concat([df.loc[:, "DATE"], df_wind], axis=1)
6 qc_speed = df_wind[ "SPEED QUALITY" ].isin([ 0, 1, 2, 4, 5, 6, 9 ])
7 qc_obs_type = (df_wind[ "OBSERVATION TYPE" ]!="9") | ((df_wind[ "OBSERVATION TYPE" ]=="9") &
(df_wind[ "SPEED" ]==0))
8 df_wind_speed_qc = df_wind.loc[qc_obs_type&qc_speed, [ "DATE", "SPEED" ]]
9 df_wind_speed_qc.set_index("DATE", inplace=True)
10 df_wind_speed_monthly = df_wind_speed_qc.resample("M").mean()
11 fig = plt.figure(figsize=[ 6, 4.5 ], dpi=300)
12 spec = fig.add_gridspec(ncols=1, nrows=1)
13 ax = fig.add_subplot(spec[ 0 ])
14 ax.plot(df_wind_speed_monthly.index, df_wind_speed_monthly[ "SPEED" ], "k-")
15 ax.set(xlabel="Datetime", ylabel=r"Wind speed (m s$^{-1}$)", title="BaoAn station")
16 fig.savefig("./images/PS2_2_figure1.jpg", bbox_inches="tight")
```

## BaoAn station



绘制 2010 至 2020 风速月均值如图，可以看到，相比较 2010 至 2012 年，2013 至 2020 年的平均风速有些许增加（从  $3.0$  增加至  $3.5 \text{ m s}^{-1}$  左右），且风速的极大值也有明显增加，尤其是在 2020 年最大月均风速达到了  $5.5 \text{ m s}^{-1}$ 。而在 2013 到 2020 年内部则并未观察到明显的变化趋势。

### 3. Explore a data set

Browse the CASEarth, 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.

#### 3.1

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

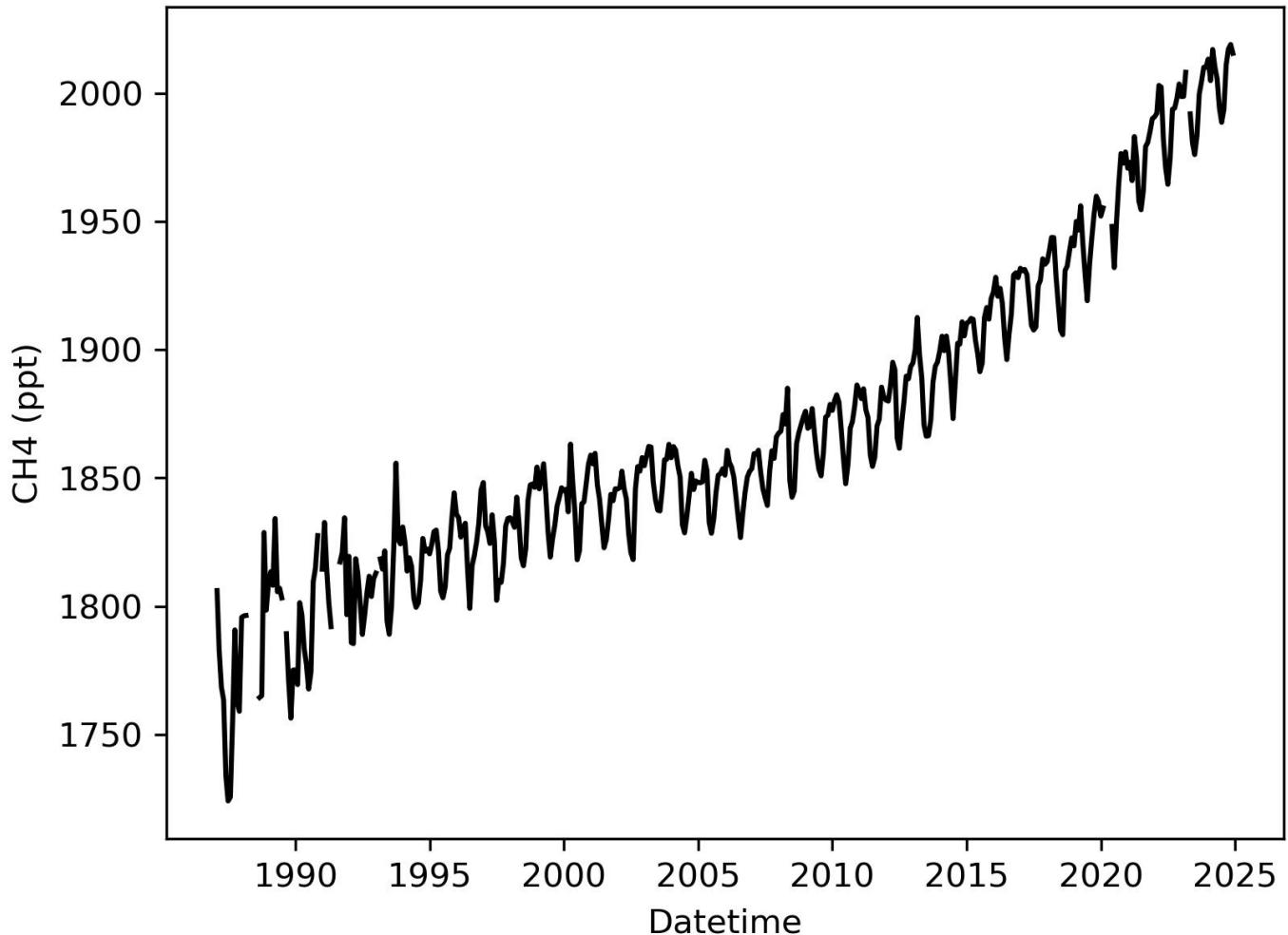
这里使用的数据集来源于 [Mace Head Atmospheric Research Station 的甲烷观测 Monthly Baseline 数据](#)，其数据可以从 AGAGE 下载得到。考虑到网站目前仅提供 nc 格式文件，因此为了满足题目要求，这里先将目标变量时间序列转换为 csv 格式文件。其原始数据的缺失值在 nc 文件中以 NaN 格式表示，在 csv 中则以空值表示（读取后依旧为 NaN）。

```
1 # 读取时间序列的表格，设置 Time 为索引，并将其解析为时间格式
2 df = pd.read_csv("./data/agage_mhd_ch4_monthly.csv", index_col=["Time"], parse_dates=[ "Time" ])
```

#### 3.2

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

```
1 fig = plt.figure(figsize=[6, 4.5], dpi=300)
2 spec = fig.add_gridspec(ncols=1, nrows=1)
3 ax = fig.add_subplot(spec[0])
4 ax.plot(df.index, df["CH4"], "k-", data=df)
5 ax.set(ylabel="CH4 (ppt)", xlabel="Datetime")
6 fig.savefig("./images/PS2_3_figure1.jpg", bbox_inches="tight")
```



绘制了 1978 至 2025 年甲烷观测结果如图。

### 3.3

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

这里对原始的时间序列数据进行了如下统计检查：

```

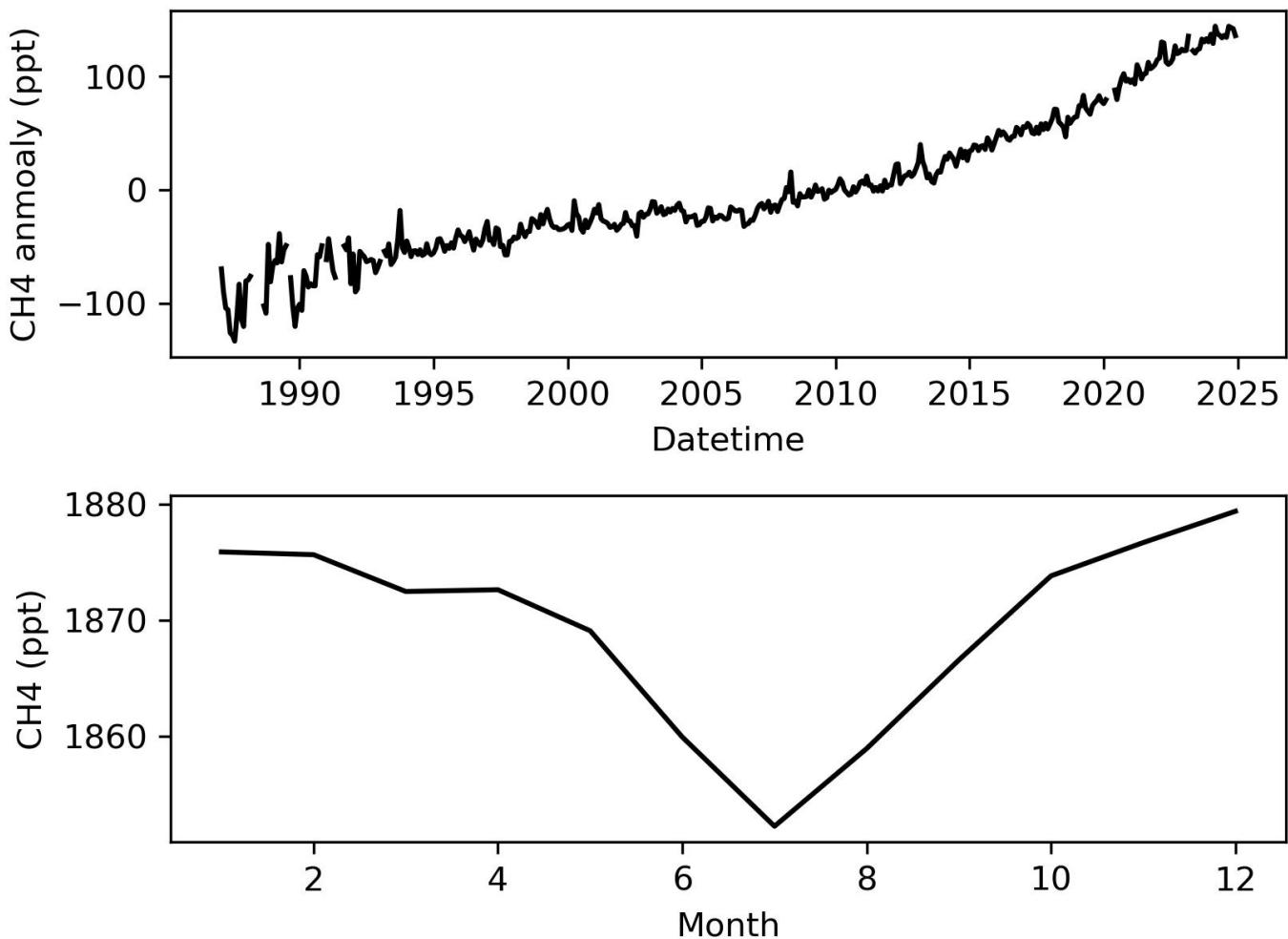
1 # 计算月份均值
2 df_monthly_mean = df.groupby(df.index.month).mean()
3 # 计算距平
4 df_anmo = df.copy()
5 df_anmo[ "Month" ] = pd.to_datetime(df_anmo.index).month
6 for i in range(1, 13):
7     df_anmo.loc[df_anmo[ "Month" ]==i, "CH4" ] -= df_anmo.loc[df_anmo[ "Month" ]==i,
8 "CH4" ].mean()
9
9 fig = plt.figure(figsize=[6, 4.5], dpi=300)
10 spec = fig.add_gridspec(ncols=1, nrows=2, hspace=0.4)
11 ax = fig.add_subplot(spec[0])
12 ax.plot(df_anmo.index, df_anmo[ "CH4" ], "k-")
13 ax.set(ylabel="CH4 anmoaly (ppt)", xlabel="Datetime")
14 ax = fig.add_subplot(spec[1])
15 ax.plot(df_monthly_mean.index, df_monthly_mean[ "CH4" ], "k-")

```

```

16 ax.set(ylabel="CH4 (ppt)", xlabel="Month")
17 fig.savefig("./images/PS2_3_figure2.jpg", bbox_inches="tight")

```



```

1 delta_CH4_from_1987 = float(df_anmo["CH4"].max() - df_anmo["CH4"].min())
2 delta_CH4_season = float(df_monthly_mean["CH4"].max() - df_monthly_mean["CH4"].min())
3 CH4_min_month = int(df_monthly_mean.index[np.argmin(df_monthly_mean["CH4"].values)])
4 CH4_max_month = int(df_monthly_mean.index[np.argmax(df_monthly_mean["CH4"].values)])
5 delta_CH4_from_1987, delta_CH4_season, CH4_min_month, CH4_max_month

```

1. 时间序列的绘图结果显示，甲烷存在明显的季节变化，为了剔除季节变化的影响，这里计算了其距平作为第一个统计量
2. 从距平我们可以获得大气甲烷浓度从 1987 年以来已经增加了 277.8 ppt，相比 1987 年增加了约 15%
3. 此外，通过对各个月份求均值可以获得甲烷的月变化曲线，可以看到每年甲烷的极差平均为 27.17 ppt
4. 另外计算了每年甲烷浓度最低的月份为 7 月，这可能和北半球夏季太阳辐射强，导致大气 ·OH 增加，更多的甲烷被氧化有关
5. 而每年甲烷浓度最高的月份为 12 月，理由同上