Mission 2

#NC文件读取

获取所有变量信息  
all\_vars\_info = dataset.variables.items()  
all\_vars\_info = list(all\_vars\_info)  
print(all\_vars\_info)

dict\_keys(['char\_year', 'char\_species', 'char\_std\_index', 'std\_lat', 'std\_lon', 'daily\_conc'])

[

('char\_year', <class 'netCDF4.\_netCDF4.Variable'>

|S1 char\_year(year, string\_length)

units: -

long\_name: Years (2013/2014/2015/2016/2017/2018/2019/2020/2021)

\_FillValue: b'\x00'

unlimited dimensions:

current shape = (9, 30)

filling on),

('char\_species', <class 'netCDF4.\_netCDF4.Variable'>

|S1 char\_species(var, string\_length)

units: ug/m3 for all

long\_name: Species (PM25/PM10/SO2/NO2/O3/Daily\_O3\_8h/CO/MDA8)

\_FillValue: b'\x00'

unlimited dimensions:

current shape = (8, 30)

filling on),

('char\_std\_index', <class 'netCDF4.\_netCDF4.Variable'>

|S1 char\_std\_index(std, string\_length)

units: -

long\_name: Station Index

\_FillValue: b'\x00'

unlimited dimensions:

current shape = (1674, 30)

filling on),

('std\_lat', <class 'netCDF4.\_netCDF4.Variable'>

float32 std\_lat(std)

units: -

long\_name: Station Latitude

unlimited dimensions:

current shape = (1674,)

filling on, default \_FillValue of 9.969209968386869e+36 used),

('std\_lon', <class 'netCDF4.\_netCDF4.Variable'>

float32 std\_lon(std)

units: -

long\_name: Station Longitude

unlimited dimensions:

current shape = (1674,)

filling on, default \_FillValue of 9.969209968386869e+36 used),

('daily\_conc', <class 'netCDF4.\_netCDF4.Variable'>

float32 daily\_conc(year, day1year\_full, var, std)

units: ug/m3 for all

long\_name: Daily Observed Concs (PM25/PM10/SO2/NO2/O3/Daily\_O3\_8h/CO/MDA8)

\_FillValue: -999.0

unlimited dimensions:

current shape = (9, 366, 8, 1674)（9年，366天，8种物质，1674个站点）

filling on)]

共1654个站点

# 数据处理代码

import numpy as np  
import netCDF4 as nc  
  
import matplotlib as mpl  
import matplotlib.pyplot as plt  
import matplotlib.ticker as mticker  
from matplotlib.colors import LinearSegmentedColormap, ListedColormap  
  
import cartopy  
import cartopy.crs as ccrs  
import cartopy.feature as cfeature  
import cartopy.mpl.ticker as cticker  
from cartopy.mpl.ticker import LongitudeFormatter, LatitudeFormatter  
from cartopy.feature import NaturalEarthFeature  
  
  
import geopandas as gpd  
import pandas as pd  
from shapely import geometry  
import os  
  
  
  
file = r'C:\Users\28166\Desktop\huanke peixun\mission 2\mission 2 data-DailyConc\_China\_CNEMC\_EachStation\_2013to2021\_nc\Daily\_OBS\_Concs4EachStation\_Y2013toY2021.nc'  
# 以nc文件对象格式打开  
dataset = nc.Dataset(file)  
# 所有变量名  
all\_vars = dataset.variables.keys()  
print(all\_vars)  
# dict\_keys(['char\_year', 'char\_species', 'char\_std\_index', 'std\_lat', 'std\_lon', 'daily\_conc'])  
# 或者可以直接print(dataset.variables.keys())  
  
# 获取单独的一个变量的数据  
years = dataset.variables['char\_year'][:]  
species = dataset.variables['char\_species'][:]  
std\_index = dataset.variables['char\_std\_index'][:]  
lats = dataset.variables['std\_lat'][:]  
lons = dataset.variables['std\_lon'][:]  
concs = dataset.variables['daily\_conc'][:,:,0,:]  
# 搞清楚cons中四个维度9，366，8，1674分别指什么，然后按需去切片，是冒号不是分号  
  
# 转换成数组（有数值的量转化成数组，没有数值的前三个，年份物质站点，均是以字符串存的，所以不用转换成数组）  
lats\_data = np.array(lats)  
lons\_data = np.array(lons)  
concs\_data = np.array(concs)  
print(lats\_data)  
print(lons\_data)  
print(concs\_data)  
'''print(concs\_data.shape)'''  
  
# 缺测值处理，true则为nan，false则为concs\_data本来的值  
arr = np.where(concs\_data == -999, np.nan, concs\_data)  
print(arr)  
  
# 分割三维数组成9个二维数组  
# 先切成9个(1,366,1674)  
sub\_arrays = np.split(arr, 9, axis=0)  
for i, sub\_array in enumerate(sub\_arrays):  
 print(f"Subarray {i}:")  
 print(sub\_array.shape) # 这会显示(1, 366, 1674)  
 print(sub\_array)  
''' for i in sub\_arrays:  
 print(i) '''  
all\_data = []  
# 将9个(1,366,1674)转换成9个(366,1674)  
reshaped\_subarrays = [sub\_array.reshape(366, 1674) for sub\_array in sub\_arrays]  
for i, sub\_array in enumerate(reshaped\_subarrays):  
 print(f"Subarray {i}:")  
 print(sub\_array.shape) # 这会显示(366, 1674)  
 print(sub\_array) # 这会显示二维数组的内容  
  
 # 在每一个（366,1674）跑的过程中，计算每个站点的年平均，即二维按列求和  
 last\_list = [np.nanmean(x) for x in zip(\*sub\_array)]  
 print(last\_list)  
 print(len(last\_list))  
 all\_data.append(last\_list)  
  
all\_data = np.array(all\_data)  
print(all\_data.shape)  
print(all\_data)

import numpy as np  
import pandas as pd  
import netCDF4 as nc  
  
import matplotlib  
import matplotlib as mpl  
import matplotlib.pyplot as plt  
import matplotlib.ticker as mticker  
from matplotlib.colors import LinearSegmentedColormap, ListedColormap  
  
import cartopy  
import cartopy.crs as ccrs  
import cartopy.feature as cfeature  
import cartopy.mpl.ticker as cticker  
from cartopy.mpl.ticker import LongitudeFormatter, LatitudeFormatter  
from cartopy.mpl.gridliner import LONGITUDE\_FORMATTER, LATITUDE\_FORMATTER  
from cartopy.feature import NaturalEarthFeature  
from cartopy.io.shapereader import Reader  
  
import shapefile  
import geopandas as gpd  
  
from shapely import geometry  
import os  
# C:/Users/28166/Desktop/huanke peixun/China\_shapefile/CN-sheng-A.shp  
  
  
  
file = r'C:\Users\28166\Desktop\huanke peixun\mission 2\mission 2 data-DailyConc\_China\_CNEMC\_EachStation\_2013to2021\_nc\Daily\_OBS\_Concs4EachStation\_Y2013toY2021.nc'  
# 以nc文件对象格式打开  
dataset = nc.Dataset(file)  
# 所有变量名  
all\_vars = dataset.variables.keys()  
print(all\_vars)  
# dict\_keys(['char\_year', 'char\_species', 'char\_std\_index', 'std\_lat', 'std\_lon', 'daily\_conc'])  
# 或者可以直接print(dataset.variables.keys())  
  
# 获取单独的一个变量的数据  
years = dataset.variables['char\_year'][:]  
species = dataset.variables['char\_species'][:]  
std\_index = dataset.variables['char\_std\_index'][:]  
lats = dataset.variables['std\_lat'][:]  
lons = dataset.variables['std\_lon'][:]  
concs = dataset.variables['daily\_conc'][:,:,0,:]  
# 搞清楚cons中四个维度9，366，8，1674分别指什么，然后按需去切片，是冒号不是分号  
  
# 转换成数组（有数值的量转化成数组，没有数值的前三个，年份物质站点，均是以字符串存的，所以不用转换成数组）  
lats\_data = np.array(lats)  
lons\_data = np.array(lons)  
concs\_data = np.array(concs)  
print(lats\_data)  
print(lons\_data)  
print(concs\_data)  
'''print(concs\_data.shape)'''  
  
# 缺测值处理，true则为nan，false则为concs\_data本来的值  
arr = np.where(concs\_data == -999, np.nan, concs\_data)  
print(arr)  
  
# 分割三维数组成9个二维数组  
# 先切成9个(1,366,1674)  
sub\_arrays = np.split(arr, 9, axis=0)  
for i, sub\_array in enumerate(sub\_arrays):  
 print(f"Subarray {i}:")  
 print(sub\_array.shape) # 这会显示(1, 366, 1674)  
 print(sub\_array)  
''' for i in sub\_arrays:  
 print(i) '''  
all\_data = []  
# 将9个(1,366,1674)转换成9个(366,1674)  
reshaped\_subarrays = [sub\_array.reshape(366, 1674) for sub\_array in sub\_arrays]  
for i, sub\_array in enumerate(reshaped\_subarrays):  
 print(f"Subarray {i}:")  
 print(sub\_array.shape) # 这会显示(366, 1674)  
 print(sub\_array) # 这会显示二维数组的内容  
  
 # 在每一个（366,1674）跑的过程中，计算每个站点的年平均，即二维按列求和  
 last\_list = [np.nanmean(x) for x in zip(\*sub\_array)]  
 print(last\_list)  
 print(len(last\_list))  
 all\_data.append(last\_list)  
  
all\_data = np.array(all\_data)  
print(all\_data.shape)  
print(all\_data)  
  
# 画图  
proj = ccrs.PlateCarree()  
fig = plt.figure(figsize=(6, 4), dpi=500)  
ax1 = fig.subplots(1, 1, subplot\_kw={'projection': proj})  
extent = [70, 140, 0, 55]  
ax1.set\_extent(extent, crs=proj)  
  
# 读入地图  
china\_map = gpd.read\_file("C:/Users/28166/Desktop/huanke peixun/China\_shapefile/CN-sheng-A.shp")  
china\_map = china\_map.to\_crs(epsg=4326) # epsg=4326,4326是WGS84的代码  
china\_map.plot(ax=ax1, color='white', edgecolor='gray', linewidth=0.3)  
  
gl = ax1.gridlines(crs=proj, draw\_labels=True, linestyle=":", linewidth=0.1, x\_inline=False, y\_inline=False, color='k',  
 alpha=0.5, xlines=False, ylines=False)  
  
gl.top\_labels = False  
gl.right\_labels = False  
  
gl.xformatter = LONGITUDE\_FORMATTER # x轴设为经度的格式  
gl.yformatter = LATITUDE\_FORMATTER # y轴设为纬度的格式  
  
gl.xlocator = mticker.FixedLocator(np.arange(70, 140.5, 10)) # extent[0], extent[1]+0.5, 10  
gl.ylocator = mticker.FixedLocator(np.arange(0, 55.5, 10)) # extent[2], extent[3]+0.5, 10  
  
gl.xlines = False  
gl.ylines = False  
  
font = {'size': 5, 'family': 'Times New Roman', 'weight': 'normal'}  
gl.xlabel\_style = font  
gl.ylabel\_style = font  
ax1.set\_title("2013", font)  
ax1.scatter(lons, lats, c=all\_data[0], s=0.1, cmap='jet')  
'''  
position = plt.axes([1.2,0])  
b=plt.colorbar(ax1,shrink=0.88,cax=position,orientation='vertical',pad=0.05,extend='neither',format='%.2f')  
'''  
fig.savefig('C:/Users/28166/Desktop/huanke peixun/2013.png')  
plt.show()  
  
  
'''  
shp\_path = r'C:/Users/28166/Desktop/huanke peixun/China\_shapefile/CN-sheng-A.shp'  
reader = Reader(shp\_path)  
a\_shapes = list(Reader(shp\_path).geometries())  
ax.add\_geometries(a\_shapes[:],crs=proj,edgecolor='k',facecolor='',lw=0.75)  
enshicity = cfeature.ShapelyFeature(reader.geometries(), proj, edgecolor='k', facecolor='none',linewidth=0.5) # 添加市界细节  
ax.add\_feature(enshicity, linewidth=0.7)  
'''  
  
  
'''  
# 读入地图  
sf = shapefile.Reader("C:/Users/28166/Desktop/huanke peixun/China\_shapefile/CN-sheng-A.shp")  
shapes = sf.shapes()  
for i in shapes:  
 pts = shapes[i].points  
 prt = shapes[i].parts  
 x,y = zip(\*pts)#把经纬度分别给到x,y  
 fig = plt.figure(figsize=[12,18])  
 ax = fig.add\_subplot(111)  
 ax.plot(x, y, '-', lw=1, color='k')  
plt.show()  
'''  
  
  
'''  
fig = plt.figure(figsize=(12,8) )  
ax1 = fig.add\_axes([0.1, 0.1, 0.8, 0.4],projection = ccrs.PlateCarree(central\_longitude=90))  
leftlon, rightlon, lowerlat, upperlat =(60,150,20,80)  
img\_extent = [leftlon, rightlon, lowerlat, upperlat]  
ax1.set\_extent(img\_extent, crs=ccrs.PlateCarree())  
ax1.set\_xticks(np.arange(60,150+30,30), crs=ccrs.PlateCarree())  
ax1.set\_yticks(np.arange(0,65+30,30), crs=ccrs.PlateCarree())  
lon\_formatter = cticker.LongitudeFormatter()  
lat\_formatter = cticker.LatitudeFormatter()  
ax1.xaxis.set\_major\_formatter(lon\_formatter)  
ax1.yaxis.set\_major\_formatter(lat\_formatter)  
'''  
  
'''  
fig = plt.figure(1, figsize=[4,3])  
proj = ccrs.PlateCarree()  
ax = plt.subplot(1, 1, 1, projection=proj)  
extent = [70, 140, 30, 55]  
ax.set(xlim=(70,140),ylim=(30,55))  
ax.set\_xticks(np.arange(extent[0], extent[1]+1, 10), crs=proj)  
ax.set\_yticks(np.arange(extent[-2], extent[-1]+1, 5), crs=proj)  
ax.xaxis.set\_major\_formatter(LongitudeFormatter(zero\_direction\_label=False))  
ax.yaxis.set\_major\_formatter(LatitudeFormatter())  
plt.show()  
'''

ax.scatter（）的参数有哪些

ax.scatter() 是 Matplotlib 中用于绘制散点图的方法，它是基于 matplotlib.pyplot 的。以下是 ax.scatter() 的一些常用参数：

**x 和 y**: 这两个参数是必需的，用于指定散点的 x 和 y 坐标。

**s**: 用于指定散点的大小。可以是一个标量（所有点的尺寸相同）或一个数组（每个点的尺寸不同）。默认值为 50。

**c**: 用于指定散点的颜色。可以是一个颜色字符串，也可以是一个数组，数组中的每个值对应一个点的颜色。默认情况下，颜色是随机分配的。

**marker**: 用于指定散点的形状。例如，'.' 表示点，'o' 表示圆圈，'s' 表示方形等。也可以使用其他形状，如 'D'、'^'、'<'、'>'、'8'、's' 等。

**cmap**: 用于指定颜色映射。如果提供了 c 参数，则 cmap 是必需的。

**norm**: 用于指定归一化对象，它可以将颜色映射到特定的范围。

**vmin 和 vmax**: 这两个参数是用于 norm 的，它们定义了颜色的最小和最大值。

**alpha**: 用于指定散点的透明度。取值范围是 0（完全透明）到 1（完全不透明）。

**edgecolors**: 用于指定散点边缘的颜色。

**linewidths**: 用于指定散点边缘的线宽。

**zorder**: 用于指定散点在图上的堆叠顺序。数值越高，表示越在上面。

**label**: 用于给散点设置标签，这在图例中会显示出来。

以上只是一些常用参数，ax.scatter() 还具有许多其他参数可供使用。要获取完整的参数列表和详细信息，可以查看 Matplotlib 的官方文档或使用 Python 的 help() 函数来查看 ax.scatter() 的帮助信息。

创建一个颜色映射时，nbin,norm,cmap之间的逻辑关系是怎样的，他们分别在构建颜色映射中起到什么作用

在创建颜色映射时，nbin、norm 和 cmap 三个参数各自起到了不同的作用，并相互影响。以下是它们之间的逻辑关系和在构建颜色映射中的作用：

nbin:

nbin 通常表示要创建的颜色映射的类别数量。

当 nbin 确定后，它会为颜色映射提供基本的类别数量。

例如，如果 nbin=5，则表示颜色映射将有5个不同的颜色。

norm:

norm 是一个归一化对象，用于将数据值映射到指定的范围。

它定义了如何将数据值（例如，数据集中每个点的值）映射到颜色映射的连续范围。

例如，如果 norm=matplotlib.colors.Normalize(vmin=0, vmax=1)，则数据值会被归一化到0到1的范围，并用于确定颜色映射中的颜色。

cmap:

cmap 是一个颜色映射对象，它定义了如何将归一化后的数据值映射到具体的颜色。

当 norm 将数据值归一化到指定的范围后，cmap 会根据这些归一化后的值选择相应的颜色。

例如，对于 cmap='viridis'，归一化后的值会在0到1之间，对应于viridis颜色映射中的颜色。

这三个参数之间的关系是：首先确定类别数量（通过 nbin），然后定义如何根据数据值将这些类别映射到颜色范围（通过 norm），最后确定这些颜色如何映射到具体的颜色（通过 cmap）。

需要注意的是，在实际应用中，通常不需要显式地设置 nbin。而是通过 levels 参数来设置等高线图的等高线级别，这些级别与颜色映射的类别数量相匹配。

任务二最终代码

import numpy as np  
import pandas as pd  
import netCDF4 as nc  
  
import matplotlib  
import matplotlib.cm as cm  
import matplotlib as mpl  
import matplotlib.pyplot as plt  
import matplotlib.ticker as mticker  
import matplotlib.colors  
from matplotlib.colors import LinearSegmentedColormap, ListedColormap  
  
import cartopy  
import cartopy.crs as ccrs  
import cartopy.feature as cfeature  
import cartopy.mpl.ticker as cticker  
from cartopy.mpl.ticker import LongitudeFormatter, LatitudeFormatter  
from cartopy.mpl.gridliner import LONGITUDE\_FORMATTER, LATITUDE\_FORMATTER  
from cartopy.feature import NaturalEarthFeature  
import cartopy.io.shapereader as shpreader  
  
import shapefile  
import geopandas as gpd  
  
from shapely import geometry  
import os  
# C:/Users/28166/Desktop/huanke peixun/China\_shapefile/CN-sheng-A.shp  
  
  
  
file = r'C:\Users\28166\Desktop\huanke peixun\mission 2\mission 2 data-DailyConc\_China\_CNEMC\_EachStation\_2013to2021\_nc\Daily\_OBS\_Concs4EachStation\_Y2013toY2021.nc'  
# 以nc文件对象格式打开  
dataset = nc.Dataset(file)  
# 所有变量名  
all\_vars = dataset.variables.keys()  
print(all\_vars)  
# dict\_keys(['char\_year', 'char\_species', 'char\_std\_index', 'std\_lat', 'std\_lon', 'daily\_conc'])  
# 或者可以直接print(dataset.variables.keys())  
  
# 获取单独的一个变量的数据  
years = dataset.variables['char\_year'][:]  
species = dataset.variables['char\_species'][:]  
std\_index = dataset.variables['char\_std\_index'][:]  
lats = dataset.variables['std\_lat'][:]  
lons = dataset.variables['std\_lon'][:]  
concs = dataset.variables['daily\_conc'][:,:,0,:]  
# 搞清楚cons中四个维度9，366，8，1674分别指什么，然后按需去切片，是冒号不是分号  
  
# 转换成数组（有数值的量转化成数组，没有数值的前三个，年份物质站点，均是以字符串存的，所以不用转换成数组）  
lats\_data = np.array(lats)  
lons\_data = np.array(lons)  
concs\_data = np.array(concs)  
print(lats\_data)  
print(lons\_data)  
print(concs\_data)  
'''print(concs\_data.shape)'''  
  
# 缺测值处理，true则为nan，false则为concs\_data本来的值  
arr = np.where(concs\_data == -999, np.nan, concs\_data)  
print(arr)  
  
# 分割三维数组成9个二维数组  
# 先切成9个(1,366,1674)  
sub\_arrays = np.split(arr, 9, axis=0)  
for i, sub\_array in enumerate(sub\_arrays):  
 print(f"Subarray {i}:")  
 print(sub\_array.shape) # 这会显示(1, 366, 1674)  
 print(sub\_array)  
''' for i in sub\_arrays:  
 print(i) '''  
all\_data = []  
# 将9个(1,366,1674)转换成9个(366,1674)  
reshaped\_subarrays = [sub\_array.reshape(366, 1674) for sub\_array in sub\_arrays]  
for i, sub\_array in enumerate(reshaped\_subarrays):  
 print(f"Subarray {i}:")  
 print(sub\_array.shape) # 这会显示(366, 1674)  
 print(sub\_array) # 这会显示二维数组的内容  
  
 # 在每一个（366,1674）跑的过程中，计算每个站点的年平均，即二维按列求和  
 last\_list = [np.nanmean(x) for x in zip(\*sub\_array)]  
 print(last\_list)  
 print(len(last\_list))  
 all\_data.append(last\_list)   
  
all\_data = np.array(all\_data)  
print(all\_data.shape)  
print(all\_data)  
  
x = 2013  
for yeardata in all\_data:  
 # 画图  
 proj = ccrs.PlateCarree()  
 fig = plt.figure(figsize=(6, 4), dpi=400)  
 ax = fig.subplots(1, 1, subplot\_kw={'projection': proj})  
 extent = [70, 140, 0, 55]  
 ax.set\_extent(extent, crs=proj)  
  
 # 读入地图  
 china\_map = gpd.read\_file("C:/Users/28166/Desktop/huanke peixun/China\_shapefile/CN-sheng-A.shp")  
 china\_map = china\_map.to\_crs(epsg=4326) # epsg=4326,4326是WGS84的代码  
 china\_map.plot(ax=ax, color='white', edgecolor='gray', linewidth=0.3)  
  
 gl = ax.gridlines(crs=proj, draw\_labels=True, linestyle=":", linewidth=0.1, x\_inline=False, y\_inline=False, color='k',  
 alpha=0.5, xlines=False, ylines=False)  
  
 gl.top\_labels = False  
 gl.right\_labels = False  
 gl.xformatter = LONGITUDE\_FORMATTER # x轴设为经度的格式  
 gl.yformatter = LATITUDE\_FORMATTER # y轴设为纬度的格式  
  
 gl.xlocator = mticker.FixedLocator(np.arange(70, 140.5, 10)) # extent[0], extent[1]+0.5, 10  
 gl.ylocator = mticker.FixedLocator(np.arange(0, 55.5, 10)) # extent[2], extent[3]+0.5, 10  
  
 gl.xlines = False  
 gl.ylines = False  
  
 font1 = {'size': 8, 'family': 'Times New Roman', 'weight': 'normal'}  
 font2 = {'size': 5, 'family': 'Times New Roman', 'weight': 'normal'}  
 gl.xlabel\_style = font2  
 gl.ylabel\_style = font2  
 '''ax.set\_title("{x}", font)'''  
 plt.title(f'{x}',fontdict=font1)  
  
 cmap = matplotlib.cm.get\_cmap('jet') # 使用'viridis'颜色映射  
 norm = plt.Normalize(vmin=0, vmax=200) # 归一化浓度数据到0到100的范围  
 sc = ax.scatter(lons, lats, c=yeardata, s=0.1, cmap=cmap, norm=norm)  
 cbar = fig.colorbar(sc, ax=ax, extend='neither', shrink=0.92, ticks=np.linspace(0, 200, 9), format='%.2f')  
  
 # 修改标签和标题的字体和字号  
 cbar.ax.tick\_params(labelsize=4)  
 cbar.ax.set\_title('concs', fontproperties='Times New Roman', weight='normal', size=8)  
  
 '''cbar.set\_label(label='abbc', fontproperties='Times New Roman', weight='normal', size=5)'''  
 fig.savefig(f'C:/Users/28166/Desktop/huanke peixun/{x}.png')  
 x = x+1

# 画年最大最小平均值变化

max\_values = []  
min\_values = []  
avg\_values = []  
  
for yeardata in all\_data:  
 # 找每个子列表的最值和均值  
  
 max\_value = max(yeardata)  
 min\_value = min(yeardata)  
 avg\_value = np.nanmean(yeardata)  
  
 max\_values.append(max\_value)  
 min\_values.append(min\_value)  
 avg\_values.append(avg\_value)

print("最大值：", max\_values)  
print("最小值：", min\_values)  
print("平均值：", avg\_values)

year = [2013,2014,2015,2016,2017,2018,2019,2020,2021]  
fig, ax = plt.subplots()  
ax.plot(year, max\_values, 'o-',label="max",lw=0.6,c='b',markersize=3)  
ax.plot(year, min\_values, 'o-',label="min",lw=0.6,c='r',markersize=3)  
ax.plot(year, avg\_values, 'o-',label="avg",lw=0.6,c='g',markersize=3)  
  
plt.title(' Trend of Annual Max,Min,Avg Values of PM2.5 ', fontdict={'family': 'Times New Roman', 'size': 15},pad=18)  
  
font2 = {'family': 'Times New Roman', 'weight': 'normal', 'size': 12}  
plt.xlabel('Year', font2,labelpad=8)  
plt.ylabel('Annual Max,Min,Avg Values of PM2.5', font2,labelpad=10)  
  
labels = ax.get\_xticklabels() + ax.get\_yticklabels()  
[label.set\_fontname('Times New Roman') for label in labels]  
  
plt.legend()  
plt.show()  
fig.savefig('C:/Users/28166/Desktop/huanke peixun/trend.png')  
  
# 画每年超标站点个数

# 显示中文字体，宋体  
plt.rcParams['font.sans-serif']=['STSong']  
  
max\_values = []  
min\_values = []  
avg\_values = []  
  
fir\_stand = []  
sec\_stand = []  
tir\_stand = []  
for\_stand = []  
  
for yeardata in all\_data:  
 n1 = 0  
 n2 = 0  
 n3 = 0  
 n4 = 0  
 # 统计年平均值的一二级超标站点数  
 for i in yeardata:  
 if i > 15:  
 n1 = n1 + 1  
 if i > 35:  
 n2 = n2 + 1  
 if i > 50:  
 n3 = n3 + 1  
 if i > 100:  
 n4 = n4 + 1  
  
 fir\_stand.append(n1)  
 sec\_stand.append(n2)  
 tir\_stand.append(n3)  
 for\_stand.append(n4)  
  
 # 找每个子列表的最值和均值  
 max\_value = max(yeardata)  
 min\_value = min(yeardata)  
 avg\_value = np.nanmean(yeardata)  
  
 max\_values.append(max\_value)  
 min\_values.append(min\_value)  
 avg\_values.append(avg\_value)  
  
year = [2013,2014,2015,2016,2017,2018,2019,2020,2021]  
fig, ax = plt.subplots()  
  
ax.plot(year, fir\_stand, 'o-',label=">15",lw=0.6,c='r',markersize=4)  
ax.plot(year, sec\_stand, 'o-',label=">35",lw=0.6,c='b',markersize=4)  
ax.plot(year, tir\_stand, 'o-',label=">50",lw=0.6,c='g',markersize=4)  
ax.plot(year, for\_stand, 'o-',label=">100",lw=0.6,c='y',markersize=4)  
  
'''ax.plot(year, max\_values, 'o-',label="max",lw=0.6,c='b',markersize=3)  
ax.plot(year, min\_values, 'o-',label="min",lw=0.6,c='r',markersize=3)  
ax.plot(year, avg\_values, 'o-',label="avg",lw=0.6,c='g',markersize=3)'''  
  
plt.title(' 各年均值超标站点个数 ', fontdict={'family': 'STSong', 'size': 15,'weight': 'bold'},pad=18)  
  
font = {'family': 'sans-serif', 'size': 15}  
font2 = {'family': 'Times New Roman', 'weight': 'normal', 'size': 12}  
plt.xlabel('Year', font2,labelpad=8)  
plt.ylabel('Numbers', font,labelpad=10)  
  
labels = ax.get\_xticklabels() + ax.get\_yticklabels()  
[label.set\_fontname('Times New Roman') for label in labels]  
  
plt.legend()  
plt.show()  
fig.savefig('C:/Users/28166/Desktop/huanke peixun/mission 2/图片/各年均值超标站点个数.png')

#画频率分布图

x = 2013  
  
for yeardata in all\_data:  
  
 fig, ax = plt.subplots()  
  
 plt.hist(yeardata, bins=21, color='blue', edgecolor='blue')  
 plt.xlim([0, 210])  
 plt.title(f'{x}年浓度频率分布', fontdict={'family': 'STSong', 'size': 15, 'weight': 'bold'}, pad=18)  
 font = {'family': 'sans-serif', 'size': 15}  
 font2 = {'family': 'Times New Roman', 'weight': 'normal', 'size': 12}  
 plt.xlabel('Value', font2, labelpad=8)  
 plt.ylabel('Frequency', font, labelpad=10)  
  
 labels = ax.get\_xticklabels() + ax.get\_yticklabels()  
 [label.set\_fontname('Times New Roman') for label in labels]  
  
 plt.show(). fig.savefig(f'C:/Users/28166/Desktop/huanke peixun/mission 2/图片/{x}年浓度频率分布.png')  
 x = x + 1

# 一次出九张图的组图

# C:/Users/28166/Desktop/huanke peixun/China\_shapefile/CN-sheng-A.shp  
  
# 显示中文字体，宋体  
plt.rcParams['font.sans-serif']=['STSong']  
  
  
file = r'C:\Users\28166\Desktop\huanke peixun\mission 2\mission 2 data-DailyConc\_China\_CNEMC\_EachStation\_2013to2021\_nc\Daily\_OBS\_Concs4EachStation\_Y2013toY2021.nc'  
# 以nc文件对象格式打开  
dataset = nc.Dataset(file)  
# 所有变量名  
all\_vars = dataset.variables.keys()  
print(all\_vars)  
# dict\_keys(['char\_year', 'char\_species', 'char\_std\_index', 'std\_lat', 'std\_lon', 'daily\_conc'])  
# 或者可以直接print(dataset.variables.keys())  
  
# 获取单独的一个变量的数据  
years = dataset.variables['char\_year'][:]  
species = dataset.variables['char\_species'][:]  
std\_index = dataset.variables['char\_std\_index'][:]  
lats = dataset.variables['std\_lat'][:]  
lons = dataset.variables['std\_lon'][:]  
concs = dataset.variables['daily\_conc'][:,:,0,:]  
# 搞清楚cons中四个维度9，366，8，1674分别指什么，然后按需去切片，是冒号不是分号  
  
# 转换成数组（有数值的量转化成数组，没有数值的前三个，年份物质站点，均是以字符串存的，所以不用转换成数组）  
lats\_data = np.array(lats)  
lons\_data = np.array(lons)  
concs\_data = np.array(concs)  
print(lats\_data)  
print(lons\_data)  
print(concs\_data)  
'''print(concs\_data.shape)'''  
  
# 缺测值处理，true则为nan，false则为concs\_data本来的值  
arr = np.where(concs\_data == -999, np.nan, concs\_data)  
print(arr)  
  
# 分割三维数组成9个二维数组  
# 先切成9个(1,366,1674)  
sub\_arrays = np.split(arr, 9, axis=0)  
for i, sub\_array in enumerate(sub\_arrays):  
 print(f"Subarray {i}:")  
 print(sub\_array.shape) # 这会显示(1, 366, 1674)  
 print(sub\_array)  
''' for i in sub\_arrays:  
 print(i) '''  
all\_data = []  
# 将9个(1,366,1674)转换成9个(366,1674)  
reshaped\_subarrays = [sub\_array.reshape(366, 1674) for sub\_array in sub\_arrays]  
for i, sub\_array in enumerate(reshaped\_subarrays):  
 print(f"Subarray {i}:")  
 print(sub\_array.shape) # 这会显示(366, 1674)  
 print(sub\_array) # 这会显示二维数组的内容  
  
 # 在每一个（366,1674）跑的过程中，计算每个站点的年平均，即二维按列求和  
 last\_list = [np.nanmean(x) for x in zip(\*sub\_array)]  
 print(last\_list)  
 print(len(last\_list))  
 all\_data.append(last\_list)  
  
all\_data = np.array(all\_data)  
print(all\_data.shape)  
print(all\_data)  
  
# 算每年的年平均值并取两位小数  
avgs = []  
for yeardata in all\_data:  
 avg = np.nanmean(yeardata)  
 avgs.append(avg)  
round\_avgs = np.round(avgs,2)  
print(round\_avgs)  
# [75.37 63.26 52.26 48.13 46.67 42.03 42.76 38.45 35.52]

# 转换成字符串  
str\_avg = round\_avgs.astype(str)  
  
  
# 创建地图投影  
proj = ccrs.PlateCarree()  
  
# 设置标题和字体  
plt.rcParams['font.family'] = 'Times New Roman'  
plt.rcParams['font.size'] = 14  
  
# 创建画布和子图  
fig, axs = plt.subplots(nrows=3, ncols=3, figsize=(16.7, 10), subplot\_kw=dict(projection=proj))  
  
# 设置大图标题  
# y用来控制大标题的相对位置  
fig.suptitle('Space Distribution of PM2.5', fontsize=20, weight='bold', y=0.94)  
  
# 设置色条  
# 位置  
fig.subplots\_adjust(right=0.82)  
cbar\_ax = fig.add\_axes([0.89, 0.15, 0.02, 0.7])  
# 颜色  
colorlevel = [0,5,10,20,30,40,60,80,100,120,150,200,240]  
colordict = ['#FFFFFF', '#C2E8FA', '#86C5EB', '#5196CF', '#49A383', '#6ABF4A', '#D9DE58', '#F8B246','#F26429', '#DD3528', '#BC1B23', '#921519']  
color\_map = mcolors.ListedColormap(colordict)  
norm = mcolors.BoundaryNorm(colorlevel, color\_map.N)  
  
n = 2013  
k = 0  
# 绘制散点图  
for i in range(3):  
 for j in range(3):  
 extent = [70, 140, 15, 55]  
 axs[i,j].set\_extent(extent, crs=proj)  
  
 # 绘制散点图  
 sc = axs[i, j].scatter(lons, lats, c=all\_data[k], s=0.1, cmap=color\_map, norm=norm, transform=proj)  
 # 添加中国地图边界  
 # axs[i, j].coastlines(linestyle=':', edgecolor='black')  
 # 添加中国地图边界  
 axs[i, j].add\_feature(cfeature.BORDERS, linestyle='-', edgecolor='black')  
 axs[i, j].add\_feature(cfeature.COASTLINE, linestyle='-', edgecolor='black')  
 # 把大陆填上色  
 axs[i, j].add\_feature(cfeature.LAND)  
 # 添加省份线  
 axs[i, j].add\_feature(cfeature.STATES, linestyle=':', edgecolor='gray', visible=False)  
  
 # 设置经纬度  
 gl = axs[i, j].gridlines(crs=proj, draw\_labels=True, linestyle=":", linewidth=0.1, x\_inline=False, y\_inline=False, color='k', alpha=0.5, xlines=False, ylines=False)  
 gl.top\_labels = False  
 gl.right\_labels = False  
 gl.xformatter = LONGITUDE\_FORMATTER # x轴设为经度的格式  
 gl.yformatter = LATITUDE\_FORMATTER # y轴设为纬度的格式  
  
 gl.xlocator = mticker.FixedLocator([70, 80, 90, 100, 110, 120, 130, 140]) # extent[0], extent[1]+0.5, 10  
 gl.ylocator = mticker.FixedLocator([20, 30, 40, 50, 60]) # extent[2], extent[3]+0.5, 10  
  
 gl.xlines = False  
 gl.ylines = False  
  
 font2 = {'size': 8, 'family': 'Times New Roman', 'weight': 'normal'}  
 gl.xlabel\_style = font2  
 gl.ylabel\_style = font2  
  
 # 添加左上角年份的小方块  
 axs[i, j].text(0.03,0.90,f'{n}', bbox={'facecolor': 'white', 'alpha': 1},fontsize=8,transform=axs[i, j].transAxes)  
 n = n + 1  
  
 # 添加平均值标签  
 axs[i, j].text(0.97, 0.05, f'Avg: {str\_avg[k]}', ha='right', va='bottom', fontsize=8, transform=axs[i, j].transAxes,bbox={'facecolor': 'white', 'alpha': 1})  
 k = k + 1  
  
  
# 添加颜色分析色条  
cbar = fig.colorbar(mappable=sc, cax=cbar\_ax,format='%.2f')  
ax0 = cbar.ax  
# 接下来两句用ax0.和cbar.ax.都行  
cbar.ax.yaxis.set\_major\_locator(mticker.FixedLocator([0,5,10,20,30,40,60,80,100,120,150,200,240]))  
ax0.set\_title('Concs', fontproperties='Times New Roman', weight='normal', size=15, pad=20)  
  
'''cbar.ax.tick\_params(direction='in',length=10,width=2,color='k',labelcolor='red',labelsize=20,pad=5)  
cbar.set\_label('Concs', fontproperties='Times New Roman', weight='normal', size=8)'''  
fig.savefig(f'C:/Users/28166/Desktop/huanke peixun/大图.png')  
# 显示图形  
plt.show()

#画频率分布图

#画频率分布图  
# 创建画布和子图  
fig, axs = plt.subplots(nrows=3, ncols=3, figsize=(17,10))  
# 设置大图标题  
# y用来控制大标题的相对位置  
font1 = {'family': 'STSong', 'size': 20, 'weight': 'bold'}  
fig.suptitle('各年站点PM2.5浓度年均值频数分布', fontdict=font1, y=0.94)  
year = 2013  
k = 0  
  
xlabels = np.linspace(0, 210, 8, dtype=str)  
ylabels = np.linspace(0, 400, 6, dtype=str)  
for i in range(3):  
 for j in range(3):  
 axs[i, j].hist(all\_data[k], bins=22, color='blue', edgecolor='black',rwidth=1)  
 axs[i, j].text(0.03, 0.922, f'{year}', bbox={'facecolor': 'white', 'alpha': 1}, fontsize=8, transform=axs[i, j].transAxes)  
  
 axs[i, j].set\_xlim([0, 210])  
 axs[i, j].set\_ylim([0, 400])  
 axs[i,j].xaxis.set\_major\_locator(mticker.FixedLocator([0,30,60,90,120,150,180,210]))  
 axs[i, j].xaxis.set\_minor\_locator(mticker.FixedLocator([15,45,75,105,135,165,195]))  
 axs[i,j].yaxis.set\_major\_locator(mticker.FixedLocator([0,80, 160, 240, 320, 400]))  
 axs[i,j].tick\_params(labelsize=10)  
  
 '''  
 axs[i, j].set\_xticks(np.linspace(0,210,num=8),labels=xlabels)  
 axs[i, j].set\_yticks(np.linspace(0, 400, num=6),labels=ylabels)  
 axs[i, j].set\_xlim(x\_range)  
 axs[i, j].set\_ylim(y\_range)'''  
  
 font = {'family': 'STSong', 'size': 10}  
 font2 = {'family': 'Times New Roman', 'weight': 'normal', 'size': 10}  
 axs[i, j].set\_xlabel('Concentration', font2, labelpad=2)  
 axs[i, j].set\_ylabel('站点个数', font, labelpad=2)  
 '''  
 # 获取直方图的每个柱子的坐标  
 patches = axs[i,j].patches  
 m = 0  
 # 遍历每个柱子，并添加数字标签  
 for m, patch in enumerate(patches):  
 # 获取柱子的中心坐标  
 x = patch.get\_x() + patch.get\_width() / 2  
 y = patch.get\_height()  
 # 添加数字标签  
 axs[i,j].annotate(str(int(patch.get\_height())), (x, y), textcoords='offset points', xytext=(0, 4), ha='center',fontsize=5)  
 '''  
 k = k + 1  
 year = year + 1  
fig.savefig(f'C:/Users/28166/Desktop/huanke peixun/mission 2/图片/浓度年均值频数分布图.png')  
plt.show()

# 每年，全站点超标天数的平均值，变化趋势（1674,9）按列求和

# 显示中文字体，宋体  
plt.rcParams['font.sans-serif']=['STSong']  
  
file = r'C:\Users\28166\Desktop\huanke peixun\mission 2\mission 2 data-DailyConc\_China\_CNEMC\_EachStation\_2013to2021\_nc\Daily\_OBS\_Concs4EachStation\_Y2013toY2021.nc'  
# 以nc文件对象格式打开  
dataset = nc.Dataset(file)  
# 所有变量名  
all\_vars = dataset.variables.keys()  
print(all\_vars)  
# dict\_keys(['char\_year', 'char\_species', 'char\_std\_index', 'std\_lat', 'std\_lon', 'daily\_conc'])  
# 或者可以直接print(dataset.variables.keys())  
  
# 获取单独的一个变量的数据  
concs = dataset.variables['daily\_conc'][:, :, 0, :]  
concs\_data = np.array(concs)  
# 搞清楚cons中四个维度9，366，8，1674分别指什么，然后按需去切片，是冒号不是分号  
  
print(concs\_data)  
print(concs\_data.shape)  
  
# 缺测值处理，true则为nan，false则为concs\_data本来的值  
arr = np.where(concs\_data == -999, np.nan, concs\_data)  
print(arr)  
  
  
  
# 分割三维数组成1674个二维数组  
# 先切成1674个(9,366)  
sub\_arrays = np.split(arr, 1674, axis=2)  
for i, sub\_array in enumerate(sub\_arrays):  
 print(f"Subarray {i}:")  
 print(sub\_array.shape) # 这会显示(9, 366, 1)  
 print(sub\_array)  
''' for i in sub\_arrays:  
 print(i) '''  
  
all\_data = []  
# 将9个(1,366,1674)转换成9个(366,1674)  
reshaped\_subarrays = [sub\_array.reshape(9, 366) for sub\_array in sub\_arrays]  
for i, sub\_array in enumerate(reshaped\_subarrays):  
 '''print(f"Subarray {i}:")  
 print(sub\_array)  
 print(sub\_array.shape) # 这会显示(9,366)'''  
  
all\_sta\_days1 = []  
all\_sta\_days2 = []  
  
for sub\_array in reshaped\_subarrays:  
 # subarray{i}9年的一二级超标天数  
 days1 = []  
 days2 = []  
 for year in sub\_array:  
 day1 = 0  
 day2 = 0  
 for day in year:  
 if day > 35 :  
 day1 = day1 + 1  
 if day > 75:  
 day2 = day2 + 1  
 days1.append(day1)  
 days2.append(day2)  
 '''print(days1)  
 print(days2)'''  
 all\_sta\_days1.append(days1)  
 all\_sta\_days2.append(days2)  
  
trend\_days1 = [np.nanmean(x) for x in zip(\*all\_sta\_days1)]  
trend\_days2 = [np.nanmean(x) for x in zip(\*all\_sta\_days2)]  
print(trend\_days1)  
print(trend\_days2)  
# [70.69772998805257, 130.91397849462365, 181.23835125448028, 163.2706093189964, 151.62724014336916, 142.6547192353644, 139.56332138590204, 121.21804062126643, 109.79510155316606]  
# [33.42652329749104, 52.2389486260454, 59.6857825567503, 49.38649940262844, 43.93966547192353, 35.60454002389486, 37.18458781362007, 29.299880525686977, 25.869175627240143]  
# 向下取整  
new\_list1 = list(map(math.floor, trend\_days1))  
new\_list2 = list(map(math.floor, trend\_days2))  
print(new\_list1)  
print(new\_list2)  
# [70, 130, 181, 163, 151, 142, 139, 121, 109]  
# [33, 52, 59, 49, 43, 35, 37, 29, 25]  
'''all\_sta\_days1 = np.array(all\_sta\_days1)  
all\_sta\_days2 = np.array(all\_sta\_days2)  
print(all\_sta\_days1.shape) # (1674,9)'''  
  
year = [2013,2014,2015,2016,2017,2018,2019,2020,2021]  
fig, ax = plt.subplots()  
plt.title(' 各年,全站点超标天数平均值 ', fontdict={'family': 'STSong', 'size': 15,'weight': 'bold'},pad=18)  
ax.plot(year, new\_list1, 'o-',label="一级超标天数",lw=0.6,c='r',markersize=4)  
ax.plot(year, new\_list2, 'o-',label="二级超标天数",lw=0.6,c='b',markersize=4)  
font = {'family': 'sans-serif', 'size': 15}  
font2 = {'family': 'Times New Roman', 'weight': 'normal', 'size': 12}  
plt.xlabel('Year', font2,labelpad=8)  
plt.ylabel('Number of Days', font,labelpad=10)  
labels = ax.get\_xticklabels() + ax.get\_yticklabels()  
[label.set\_fontname('Times New Roman') for label in labels]  
plt.legend()  
fig.savefig('C:/Users/28166/Desktop/huanke peixun/mission 2/图片/各年,全站点超标天数平均值.png')  
plt.show()

图形用户界面, 文本, 应用程序

描述已自动生成

1.包的下载

pip install -i https://pypi.tuna.tsinghua.edu.cn/simple/ +包的名称

2.nc文件读取

首先明白了nc文件内部长啥样，看看varibles的具体信息，明确数据处理的思路（看到一个陌生的数据文件类型都不要着急，先尝试搞清楚他到底长啥样）

将需要进行计算的变量提取出来，常规的包括经纬度，浓度，时间等等根据目的进行计算，

3.带地图的散点图的绘制

首先读取shp文件（还有一些问题尚未解决），

清华大学 ：https://pypi.tuna.tsinghua.edu.cn/simple/

豆瓣源：http://pypi.douban.com/simple/

阿里云：http://mirrors.aliyun.com/pypi/simple/

中国科学技术大学 ：http://pypi.mirrors.ustc.edu.cn/simple/

华中科技大学：http://pypi.hustunique.com/

腾讯源：http://mirrors.cloud.tencent.com/pypi/simple

华为镜像源：https://repo.huaweicloud.com/repository/pypi/simple/

# 图片最终美化

import numpy as np  
np.set\_printoptions(threshold=np.inf)  
import pandas as pd  
import netCDF4 as nc  
import xarray as xr  
  
import math  
  
import matplotlib  
import matplotlib.cm as cm  
import matplotlib as mpl  
import matplotlib.pyplot as plt  
import matplotlib.ticker as mticker  
import matplotlib.colors as mcolors  
import matplotlib.transforms as mtransforms  
from matplotlib.colors import LinearSegmentedColormap, ListedColormap  
  
import cartopy  
import cartopy.crs as ccrs  
import cartopy.feature as cfeature  
import cartopy.mpl.ticker as cticker  
from cartopy.mpl.ticker import LongitudeFormatter, LatitudeFormatter  
from cartopy.mpl.gridliner import LONGITUDE\_FORMATTER, LATITUDE\_FORMATTER  
from cartopy.feature import NaturalEarthFeature  
import cartopy.io.shapereader as shpreader  
  
import shapefile  
import geopandas as gpd  
from shapely import geometry  
import os  
import cnmaps  
from cnmaps import get\_adm\_maps, draw\_map  
from cnmaps.sample import load\_dem  
# print(dir(cnmaps))  
# 'MapPolygon', 'ShapelyDeprecationWarning', 'Transformer', 'Union', '\_\_builtins\_\_', '\_\_cached\_\_', '\_\_doc\_\_','\_\_file\_\_',  
# '\_\_loader\_\_', '\_\_name\_\_', '\_\_package\_\_', '\_\_path\_\_', '\_\_spec\_\_', '\_\_version\_\_', 'cartopy',  
# 'ccrs', 'clip\_clabels\_by\_map', 'clip\_contours\_by\_map','clip\_pcolormesh\_by\_map', 'clip\_quiver\_by\_map','clip\_scatter\_by\_map', 'contains','copy' ,  
# 'draw\_map', 'draw\_maps', 'drawing', 'fiona', 'gcj02\_to\_wgs84', '  
# geo', 'geojson', 'geos\_to\_path', 'get\_adm\_maps', 'get\_adm\_names', 'mapping', 'maps', 'matplotlib',  
  
def adjust\_sub\_axes(ax\_main, ax\_sub, shrink):  
 *'''  
 将ax\_sub调整到ax\_main的右下角. shrink指定缩小倍数.  
 当ax\_sub是GeoAxes时, 需要在其设定好范围后再使用此函数.  
 '''* bbox\_main = ax\_main.get\_position()  
 bbox\_sub = ax\_sub.get\_position()  
 # 使shrink=1时ax\_main与ax\_sub等宽或等高.  
 if bbox\_sub.width > bbox\_sub.height:  
 ratio = bbox\_main.width / bbox\_sub.width \* shrink  
 else:  
 ratio = bbox\_main.height / bbox\_sub.height \* shrink  
 wnew = bbox\_sub.width \* ratio  
 hnew = bbox\_sub.height \* ratio  
 bbox\_new = mtransforms.Bbox.from\_extents(  
 bbox\_main.x1 - wnew, bbox\_main.y0,  
 bbox\_main.x1, bbox\_main.y0 + hnew  
 )  
 ax\_sub.set\_position(bbox\_new)  
  
# 设置标题和字体  
plt.rcParams['font.family'] = 'Times New Roman'  
plt.rcParams['font.size'] = 14  
  
# scatter数据处理  
file2 = r'D:\huankepeixun\mission 2\mission 2 data-DailyConc\_China\_CNEMC\_EachStation\_2013to2021\_nc\Daily\_OBS\_Concs4EachStation\_Y2013toY2021.nc'  
dataset2 = nc.Dataset(file2)  
# dict\_keys(['char\_year', 'char\_species', 'char\_std\_index', 'std\_lat', 'std\_lon', 'daily\_conc'])9，366，8，1674  
lats2 = dataset2.variables['std\_lat'][:]  
lons2 = dataset2.variables['std\_lon'][:]  
concs2 = dataset2.variables['daily\_conc'][:, :, 0, :]  
lats\_data2 = np.array(lats2)  
lons\_data2 = np.array(lons2)  
concs\_data2 = np.array(concs2)  
concs\_data2 = np.where(concs\_data2 == -999, np.nan, concs\_data2)  
sub\_arrays = np.split(concs\_data2, 9, axis=0)  
all\_data = []  
reshaped\_subarrays = [sub\_array.reshape(366, 1674) for sub\_array in sub\_arrays] # 将8个(1,366,1674)转换成9个(366,1674)  
for i, sub\_array in enumerate(reshaped\_subarrays):  
 last\_list = [np.nanmean(x) for x in zip(\*sub\_array)] # 在每一个（366,1674）跑的过程中，计算每个站点的年平均，即二维按列求和  
 all\_data.append(last\_list)  
all\_data = np.array(all\_data)  
print(all\_data.shape)  
  
  
proj = ccrs.PlateCarree() # 创建地图投影  
fig = plt.figure(figsize=(16.7, 10))  
fig.suptitle('Space Distribution of PM2.5', fontsize=20, weight='bold', y=0.94) # 设置大图标题,y用来控制大标题的相对位置  
axes\_main = fig.subplots(3, 3, subplot\_kw=dict(projection=proj))  
axes\_sub = fig.subplots(3, 3, subplot\_kw=dict(projection=proj))  
  
extent\_main = [70, 140, 15, 55]  
extents\_sub = [105, 125, 0, 25]  
  
fig.subplots\_adjust(right=0.85) # 设置色条  
  
colorlevel = [0,5,10,20,30,40,60,80,100,120,150,200,240]  
colordict = ['#FFFFFF', '#C2E8FA', '#86C5EB', '#5196CF', '#49A383', '#6ABF4A', '#D9DE58', '#F8B246','#F26429', '#DD3528', '#BC1B23', '#921519']  
color\_map = mcolors.ListedColormap(colordict)  
norm = mcolors.BoundaryNorm(colorlevel, 12)  
  
china = gpd.read\_file('C:/Users/28166/Desktop/china\_map/china.shp')  
nanhai = gpd.read\_file('C:/Users/28166/Desktop/china\_map/9duanxian/9duanxian.shp')  
shengji = gpd.read\_file("C:/Users/28166/Desktop/china\_map/shengji/shengji.shp")  
n = 2013  
k = 0  
for i in range(3):  
 for j in range(3):  
 axes\_main[i, j].set\_extent(extent\_main, crs=proj)  
 china.plot(ax=axes\_main[i, j], color='white', edgecolor='k', zorder=2)   
 # 绘制散点图  
 sc2 = axes\_main[i, j].scatter(lons2, lats2, c=all\_data[k], s=12, edgecolor='k', linewidths=0.1,  
 cmap=color\_map, norm=norm, transform=proj, zorder=5)  
 shengji.plot(ax=axes\_main[i, j], color='white', edgecolor='gray', linewidths=0.2, zorder=4)  
 # 设置经纬度  
 gl = axes\_main[i, j].gridlines(crs=proj, draw\_labels=True, linestyle=":", linewidth=0.1, x\_inline=False,  
 y\_inline=False, color='k', alpha=0.5, xlines=False, ylines=False)  
 gl.top\_labels = False  
 gl.right\_labels = False  
 gl.xformatter = LONGITUDE\_FORMATTER # x轴设为经度的格式  
 gl.yformatter = LATITUDE\_FORMATTER # y轴设为纬度的格式  
 gl.xlocator = mticker.FixedLocator([80, 90, 100, 110, 120, 130, 140]) # extent[0], extent[1]+0.5, 10  
 gl.ylocator = mticker.FixedLocator([20, 30, 40, 50, 60]) # extent[2], extent[3]+0.5, 10  
 gl.xlines = False  
 gl.ylines = False  
 font2 = {'size': 10, 'family': 'Times New Roman', 'weight': 'normal'}  
 gl.xlabel\_style = font2  
 gl.ylabel\_style = font2  
 # 添加年份文字  
 axes\_main[i, j].text(0.03, 0.91, f'{n}', bbox={'facecolor': 'white', 'alpha': 1}, fontsize=8,  
 transform=axes\_main[i, j].transAxes)  
 # 画南海及九段线  
 axes\_sub[i, j].set\_extent(extents\_sub, crs=proj)  
 china.plot(ax=axes\_sub[i, j], color='white', edgecolor='gray', zorder=0, linewidths=0.35)  
 nanhai.plot(ax=axes\_sub[i, j], color='gray', edgecolor='gray', zorder=1)  
 adjust\_sub\_axes(axes\_main[i, j], axes\_sub[i, j], shrink=0.3)  
 cbar\_ax = fig.add\_axes([0.88, 0.15, 0.01, 0.7]) # 位置  
 cbar = fig.colorbar(mappable=sc2, cax=cbar\_ax, format='%.2f', shrink=0.88,  
 ticks=[0, 5, 10, 20, 30, 40, 60, 80, 100, 120, 150, 200, 240])  
 ax0 = cbar.ax # 将colorbar变成一个新的ax对象，可通过ax对象的各种命令来调整colorbar  
 ax0.set\_title('Concs', fontproperties='Times New Roman', weight='normal', size=15, pad=20)  
 ax0.tick\_params(which='major', direction='in', labelsize=12, length=11.5)  
 n = n + 1  
 k = k + 1  
plt.show()

