# 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'][**1**:**7,** :**, 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**, 6,** 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)  
  
# 统计有效站点个数  
nan\_counts = np.isnan(all\_data).sum(axis=**1**).tolist()  
print(nan\_counts)  
num = []  
for i in nan\_counts:  
 x = **1674** - i  
 num.append(x)  
print(num)  
  
# contourf数据处理  
file3 = r'D:\huankepeixun\mission 3\TAP\_Daily\_PM25\_Y2001toY2020\TAP\_Daily\_PM25\_Y2001toY2020.nc'  
dataset3 = xr.open\_dataset(file3)  
lon3 = dataset3.longitude  
lat3 = dataset3.latitude  
concs3 = dataset3.Daily\_PM25.loc[**13**:**19,** :**,** :**,** :].mean("day")  
print(concs3)  
# dict\_keys(['latitude', 'longitude', 'Daily\_PM25'])  
# latitude(450),Latitude from 15.05 to 59.95 by 0.1  
# longitude(700),Longitude from 70.05 to 139.95 by 0.1  
# float32 Daily\_PM25(year, day, lat, lon)(20, 366, 450, 700)  
  
n = **0** # 1674的累加  
m = **0**y = []  
for data in concs3:  
 yi = []  
 for i in range(**1674**):  
 lat0 = lats2[n]  
 lon0 = lons2[n]  
 abslat = np.abs(lat0 - lat3)  
 abslon = np.abs(lon0 - lon3)  
 c = (abslat \*\* **2** + abslon \*\* **2**).data  
 ([xloc]**,** [yloc]) = np.where(c == np.min(c))  
 r = np.where(c == np.min(c))  
 lon\_new = lon3[r[**1**].data].data  
 lat\_new = lat3[r[**0**].data].data  
  
 ds = dataset3.sel(year=slice(m**,** m)**,**lat=slice(lat\_new**,** lat\_new)**,** lon=slice(lon\_new**,** lon\_new))  
  
 yi.append(ds)  
 print(len(yi))  
 y.append(yi)

# data2使用xarray操作  
file2 = r'D:\huankepeixun\mission 2\mission 2 data-DailyConc\_China\_CNEMC\_EachStation\_2013to2021\_nc\Daily\_OBS\_Concs4EachStation\_Y2013toY2021.nc'  
da\_xr2 = xr.open\_dataset(file2)  
lon2 = da\_xr2.std\_lon  
lat2 = da\_xr2.std\_lat  
conc = da\_xr2.daily\_conc.loc[**1**:**7,** :**, 0,** :].mean("day1year\_full")  
print(conc)  
xnum = []  
for con in conc:  
 x = con.count()  
 xnum.append(x)  
print(xnum)  
'''  
print(conc.shape) # (6, 1674)  
print(da\_xr2)  
Data variables:  
 char\_year (year) object ...  
 char\_species (var) object ...  
 char\_std\_index (std) object ...  
 std\_lat (std) float32 ...  
 std\_lon (std) float32 ...  
 daily\_conc (year, day1year\_full, var, std) float32 Dimensions: (year: 9, var: 8, std: 1674, day1year\_full: 366)  
Attributes:  
 start\_date: 2013-01-01 (daily)  
 end\_date: 2021-12-31 (daily)'''

**739, 194, 210, 248, 182, 196**]  
[**935, 1480, 1464, 1426, 1492, 1478**]  
<xarray.DataArray 'daily\_conc' (year: **6,** std: **1674**)>  
array([[ **88.3683 , 73.93997 , 87.08135 , 86.52198 , 89.115974 ,  
 90.19489 , 89.55772 , 84.842155 , 76.44808 , 78.67427 ,  
 91.61615 , 88.38621 , 88.00729 , 87.674866 , 87.14909 ,  
 89.33332 , 84.68455 , 86.9884 , 95.67373 , 94.2532 ,  
 88.27595 ,** nan**, 76.382454 , 87.727776 , 85.45132 ,  
 80.47759 , 90.32843 , 126.25379 , 125.22191 , 122.28315 ,  
 138.13081 , 129.21513 , 130.10715 , 120.124084 , 92.77669 ,  
 98.023796 , 98.35202 , 96.20503 , 100.23967 , 107.006615 ,  
 106.50253 , 57.665787 , 53.518467 , 64.78442 , 56.317364 ,  
 117.32082 , 115.68448 , 111.1612 , 114.439064 , 111.00395 ,  
 135.37238 , 128.73836 , 139.86148 , 130.13293 , 126.51896 ,  
 35.13207 , 34.17483 , 33.818848 , 38.061863 , 38.73439 ,  
 51.30801 , 49.074703 , 54.177246 , 53.126717 , 62.28782 ,  
 96.2192 , 102.33854 , 96.20662 , 95.75371 , 91.418594 ,  
 85.82876 , 87.190125 , 104.26543 , 106.84929 , 113.43668 ,  
 129.84822 , 130.22481 , 143.68454 , 125.234634 , 69.69318 ,  
 75.19877 , 47.156887 , 74.94001 , 72.17675 , 68.81671 ,  
 78.65049 , 63.964516 , 71.08688 , 43.80869 , 46.82312 ,  
 45.287395 , 53.806614 , 41.806347 , 40.28748 , 45.47861 ,  
 40.14607 , 59.99906 , 76.79538 , 75.57151 , 84.16686 ,**...  
 **29.923882 , 42.267815 , 42.97423 , 65.16059 , 39.771267 ,  
 39.26426 , 51.33816 , 52.821922 , 53.356403 , 72.90296 ,  
 65.933136 , 41.2814 , 46.016956 ,** nan**, 48.568363 ,  
 48.976326 , 71.58076 , 69.981476 ,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan**,** nan]]**,** dtype=float32)  
Dimensions without coordinates: year**,** std  
[<xarray.DataArray 'daily\_conc' ()>  
array(**935**)**,** <xarray.DataArray 'daily\_conc' ()>  
array(**1480**)**,** <xarray.DataArray 'daily\_conc' ()>  
array(**1464**)**,** <xarray.DataArray 'daily\_conc' ()>  
array(**1426**)**,** <xarray.DataArray 'daily\_conc' ()>  
array(**1492**)**,** <xarray.DataArray 'daily\_conc' ()>  
array(**1478**)]

# contourf数据处理  
file3 = r'D:\huankepeixun\mission 3\TAP\_Daily\_PM25\_Y2001toY2020\TAP\_Daily\_PM25\_Y2001toY2020.nc'  
datasetx = nc.Dataset(file3)  
dataset3 = xr.open\_dataset(file3)  
lon3 = dataset3.longitude  
lat3 = dataset3.latitude  
concs3 = dataset3.Daily\_PM25.loc[**13**:**19,** :**,** :**,** :].mean("day")  
print(concs3.shape)  
# dict\_keys(['latitude', 'longitude', 'Daily\_PM25'])  
# latitude(450),Latitude from 15.05 to 59.95 by 0.1  
# longitude(700),Longitude from 70.05 to 139.95 by 0.1  
# float32 Daily\_PM25(year, day, lat, lon)(20, 366, 450, 700)

# 画图部分在代码试验  
#'Arial Unicode MS','Microsoft MHei'  
# 画图  
# 设置标题和字体  
plt.rcParams['axes.unicode\_minus'] = False # 用来正常显示负号  
plt.rcParams['font.family'] = 'Arial Unicode MS' # 控制没有特殊说明的位置的字体，如刻度标签，下同  
plt.rcParams['font.size'] = **13**plt.rcParams['font.weight'] = 'bold'  
fig = plt.figure(figsize=(**33, 20**))  
axes\_main = fig.subplots(**2, 3,** sharex=True**,** sharey=True) # 共享刻度坐标  
'''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  
font2 = {'size': **12,** 'family': 'SimHei'**,** 'weight': 'normal'}  
label\_font = {'size': **12,** 'family': 'Arial Unicode MS'**,** 'weight': 'bold'}  
year = **2014**k = **0** # 图中文字1-6  
zimu = ['a'**,**'b'**,**'c'**,**'d'**,**'e'**,**'f']  
x1 = np.linspace(**0,100,100**)  
x2 = np.linspace(**0,200,100**)  
x3 = np.linspace(**0,200,100**)  
y1 = **2**\*x1  
y2 = **0.5**\*x2  
y3 = x3  
for i in range(**2**):  
 for j in range(**3**):  
 # 绘制散点图  
 '''sc = axes\_main[i, j].scatter(lons2, lats2, c='red', s=8, edgecolor='k', linewidths=0.1, zorder=5)'''  
 # 三条蓝线  
 axes\_main[i**,** j].scatter(x1**,**y1**,**c='blue'**,**s=**0.3**)  
 axes\_main[i**,** j].scatter(x2**,**y2**,**c='blue'**,**s=**0.3**)  
 axes\_main[i**,** j].scatter(x3**,**y3**,**c='blue'**,**s=**0.3**)  
 # 坐标轴及刻度设置  
 axes\_main[i**,** j].tick\_params(top=False**,** bottom=True**,** left=True**,** right=False)  
 axes\_main[i**,** j].tick\_params(axis='both'**,** which='major'**,** direction='out'**,** width=**1,** length=**6,** labelsize=**10**)  
 axes\_main[i**,** j].set\_xticks([**0, 50, 100, 150, 200**]) # 哪些值标刻度  
 axes\_main[i**,** j].set\_yticks([**0, 50, 100, 150, 200**])  
 axes\_main[i**,** j].set\_xlim(**0, 200**) # 在原点处开始标刻度值  
 axes\_main[i**,** j].set\_ylim(**0, 200**)  
 # 坐标轴名称  
 if j == **0**:  
 axes\_main[i**,** j].set\_ylabel('CNEMC MDA8 O$\_3$ (μg m$^{-3}$)'**,** fontdict=label\_font**,** style='italic')  
 if i == **1**:  
 axes\_main[i**,** j].set\_xlabel('TAP MDA8 O$\_3$ (μg m$^{-3}$)'**,** fontdict=label\_font**,** style='italic')  
 # 左上角右下角文字信息  
 axes\_main[i**,** j].text(**0.03, 0.94,** f'year : {year}'**,** fontsize=**12,** style='italic'**,**weight='bold'**,** transform=axes\_main[i**,** j].transAxes)  
 axes\_main[i**,** j].text(**0.03, 0.88,** f'station : {num[k]} '**,** fontsize=**12,** style='italic'**,** weight='bold'**,** transform=axes\_main[i**,** j].transAxes)  
 axes\_main[i**,** j].text(**0.03, 0.82,** 'NMB :'**,** fontsize=**12,** style='italic'**,** weight='bold'**,** transform=axes\_main[i**,** j].transAxes)  
 axes\_main[i**,** j].text(**0.03, 0.76,** 'IOA :'**,** fontsize=**12,** style='italic'**,** weight='bold'**,** transform=axes\_main[i**,** j].transAxes)  
 axes\_main[i**,** j].text(**0.90, 0.05,** f'({zimu[k]})'**,** fontsize=**12,** style='italic'**,** weight='bold'**,** transform=axes\_main[i**,** j].transAxes)  
 k = k + **1** n = n + **1**plt.show()

# 任务四完整代码

import numpy as np  
np.set\_printoptions(threshold=np.inf)  
import pandas as pd  
import netCDF4 as nc  
import xarray as xr  
import unicodeit  
  
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 cnmaps  
from cnmaps import get\_adm\_maps**,** draw\_map  
from cnmaps.sample import load\_dem  
  
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  
from scipy.stats import pearsonr  
  
# 将一维列表按对应数值分割成不等长的子列表  
def split\_list\_by\_sizes(lst**,** sizes):  
 start = **0** sublists = []  
 for size in sizes:  
 end = start + size  
 sublist = lst[start:end]  
 sublists.append(sublist)  
 start = end  
 return sublists  
  
# 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'][**1**:**7,** :**, 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**, 6,** 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) (6, 1674)  
# 6\*1674是形状规整的数组，包含nan值  
  
# 统计每年有效站点个数，统计nan个数，判断并统计nan有专门的函数，不需要自己写条件  
nan\_counts = np.isnan(all\_data).sum(axis=**1**)  
print(nan\_counts)  
num = **1674** - nan\_counts  
print(num)  
  
# 将站点数据二维展开成一维，去掉nan值，再按每年的有效站点个数分割成不等长的子列表（二维np数组只能整行整列删除，无法删除单个值;并且np数组形状是规则的，列表形状才可以不规则）  
data\_2 = all\_data.ravel()  
data\_2 = data\_2[~np.isnan(data\_2)] # 展开成一维并删去nan值需要以np数组形式  
data\_2 = data\_2.tolist() # 这两行可以合并data\_2 = data\_2[~np.isnan(data\_2)].tolist()  
# 8275  
sublists = split\_list\_by\_sizes(data\_2**,** num) # 不等长分割需要以List形式，因此需要上步tolist  
for sublist in sublists:  
 print(sublist)  
 print(len(sublist))  
  
# contourf数据处理  
file3 = r'D:\huankepeixun\mission 3\TAP\_Daily\_PM25\_Y2001toY2020\TAP\_Daily\_PM25\_Y2001toY2020.nc'  
dataset3 = nc.Dataset(file3)  
# dict\_keys(['latitude', 'longitude', 'Daily\_PM25'])  
# latitude(450),Latitude from 15.05 to 59.95 by 0.1  
# longitude(700),Longitude from 70.05 to 139.95 by 0.1  
# float32 Daily\_PM25(year, day, lat, lon)(20, 366, 450, 700)  
lats3 = dataset3.variables['latitude'][:]  
lons3 = dataset3.variables['longitude'][:]  
lats\_data3 = np.array(lats3)  
lons\_data3 = np.array(lons3)  
concs3 = dataset3.variables['Daily\_PM25'][**13**:**19,** :**,** :**,** :]  
concs3 = np.array(concs3)  
concs3 = np.where(concs3 == -**999,** np.nan**,** concs3)  
data3 = np.nanmean(concs3**,** axis=**1**)  
  
# 提取对应最近格点的数据  
m = **0** # 年份的累加  
y = []  
for data in data3:  
 yi = []  
 for i in range(**1674**):  
 if np.isnan(all\_data[m**,** i]):  
 continue  
 else:  
 lat0 = lats2[i]  
 lon0 = lons2[i]  
  
 delta\_lat = []  
 delta\_lon = []  
  
 # 使用掩码来获取最小值的索引，不使用掩码的话，nan值会跳过，索引值并不是真正的值，而是删去nan后新序列的索引。  
 for lat in lats\_data3:  
 deltalat = np.fabs(lat - lat0)  
 delta\_lat.append(deltalat)  
 delta\_lat = np.array(delta\_lat)  
 deltalat\_mask = ~np.isnan(delta\_lat)  
 filtered\_lat = delta\_lat[deltalat\_mask]  
 minlat\_value\_filtered = np.min(filtered\_lat)  
 lat\_loc = np.where(delta\_lat == minlat\_value\_filtered)  
  
 for lon in lons\_data3:  
 deltalon = np.fabs(lon - lon0)  
 delta\_lon.append(deltalon)  
 delta\_lon = np.array(delta\_lon)  
 deltalon\_mask = ~np.isnan(delta\_lon)  
 filtered\_lon = delta\_lon[deltalon\_mask]  
 minlon\_value\_filtered = np.min(filtered\_lon)  
 lon\_loc = np.where(delta\_lon == minlon\_value\_filtered)  
  
 ds = data3[m**,** lat\_loc**,** lon\_loc]  
 yi.append(ds)  
 # print(len(yi)) 935, 1480, 1464, 1426, 1492, 1478  
 yi = [item[**0**][**0**] for item in yi] # 列表的元素输出后显示[array([[80.92473]], dtype=float32)]转化为纯数值  
 m = m + **1** y.append(yi)  
for x in y:  
 print(x)  
 print(len(x))  
  
# 计算nmb和ioa  
nmb\_list = []  
ioa\_list = []  
for i in range(**6**):  
 # list没有求和函数，基本上不存在可以直接进行数学计算的工具，常用的功能基本上都在np数组上  
 y\_np = np.array(y[i])  
 sublist\_np = np.array(sublists[i])  
 num\_np = np.array(num)  
 nmb = (np.nansum(y\_np) - np.nansum(sublist\_np)) / num\_np[i] / np.nanmean(sublist\_np)  
 nmb\_list.append(nmb)  
 # nmb = (y\_np.sum()-sublist\_np.sum())/num\_np[i]/sublist\_np.mean() 显示nan，虽然不知道为啥，所以都改用nansum,nanmean函数了  
  
 y\_no\_nan = np.nan\_to\_num(y\_np) # np.nan\_to\_num()函数，将nan转换成0  
 sub\_no\_nan = np.nan\_to\_num(sublist\_np) # 这两步换成0是为了解决，这个错误：array must not contain infs or NaNs  
 r**,** p = pearsonr(y\_no\_nan**,** sub\_no\_nan)  
 ioa\_list.append(r)  
print(nmb\_list)  
print(ioa\_list)  
  
# 站点数据是sublists，，格点数据是y  
# 画图部分在代码试验  
#'Arial Unicode MS','Microsoft MHei'  
# 画图  
# 设置标题和字体  
plt.rcParams['axes.unicode\_minus'] = False # 用来正常显示负号  
plt.rcParams['font.family'] = 'Arial Unicode MS' # 控制没有特殊说明的位置的字体，如刻度标签，下同  
plt.rcParams['font.size'] = **13**plt.rcParams['font.weight'] = 'bold'  
fig = plt.figure(figsize=(**33, 20**))  
axes\_main = fig.subplots(**2, 3,** sharex=True**,** sharey=True) # 共享刻度坐标  
'''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  
font2 = {'size': **12,** 'family': 'SimHei'**,** 'weight': 'normal'}  
label\_font = {'size': **12,** 'family': 'Arial Unicode MS'**,** 'weight': 'bold'}  
year = **2014**k = **0** # 图中文字1-6  
zimu = ['a'**,**'b'**,**'c'**,**'d'**,**'e'**,**'f']  
x1 = np.linspace(**0,100,100**)  
x2 = np.linspace(**0,200,100**)  
x3 = np.linspace(**0,200,100**)  
y1 = **2**\*x1  
y2 = **0.5**\*x2  
y3 = x3  
for i in range(**2**):  
 for j in range(**3**):  
 # 绘制散点图  
 sc = axes\_main[i**,** j].scatter(sublists[k]**,** y[k]**,** c='red'**,** s=**8,** edgecolor='k'**,** linewidths=**0.1,** zorder=**5**)  
 # 三条蓝线  
 axes\_main[i**,** j].scatter(x1**,** y1**,** c='blue'**,** s=**0.3**)  
 axes\_main[i**,** j].scatter(x2**,** y2**,** c='blue'**,** s=**0.3**)  
 axes\_main[i**,** j].scatter(x3**,** y3**,** c='blue'**,** s=**0.3**)  
 # 坐标轴及刻度设置  
 axes\_main[i**,** j].tick\_params(top=False**,** bottom=True**,** left=True**,** right=False)  
 axes\_main[i**,** j].tick\_params(axis='both'**,** which='major'**,** direction='out'**,** width=**1,** length=**6,** labelsize=**10**)  
 axes\_main[i**,** j].set\_xticks([**0, 50, 100, 150, 200**]) # 哪些值标刻度  
 axes\_main[i**,** j].set\_yticks([**0, 50, 100, 150, 200**])  
 axes\_main[i**,** j].set\_xlim(**0, 200**) # 在原点处开始标刻度值  
 axes\_main[i**,** j].set\_ylim(**0, 200**)  
 # 坐标轴名称  
 if j == **0**:  
 axes\_main[i**,** j].set\_ylabel('CNEMC MDA8 O$\_3$ (μg m$^{-3}$)'**,** fontdict=label\_font**,** style='italic')  
 if i == **1**:  
 axes\_main[i**,** j].set\_xlabel('TAP MDA8 O$\_3$ (μg m$^{-3}$)'**,** fontdict=label\_font**,** style='italic')  
 # 左上角右下角文字信息  
 axes\_main[i**,** j].text(**0.03, 0.94,** f'year : {year}'**,** fontsize=**12,** style='italic'**,**weight='bold'**,** transform=axes\_main[i**,** j].transAxes)  
 axes\_main[i**,** j].text(**0.03, 0.88,** f'station : {num[k]}'**,** fontsize=**12,** style='italic'**,** weight='bold'**,** transform=axes\_main[i**,** j].transAxes)  
 axes\_main[i**,** j].text(**0.03, 0.82,** f'NMB : {nmb\_list[k]:.2f}'**,** fontsize=**12,** style='italic'**,** weight='bold'**,** transform=axes\_main[i**,** j].transAxes)  
 axes\_main[i**,** j].text(**0.03, 0.76,** f'IOA : {ioa\_list[k]:.2f}'**,** fontsize=**12,** style='italic'**,** weight='bold'**,** transform=axes\_main[i**,** j].transAxes)  
 axes\_main[i**,** j].text(**0.90, 0.05,** f'({zimu[k]})'**,** fontsize=**12,** style='italic'**,** weight='bold'**,** transform=axes\_main[i**,** j].transAxes)  
 k = k + **1** year = year + **1**plt.show()