

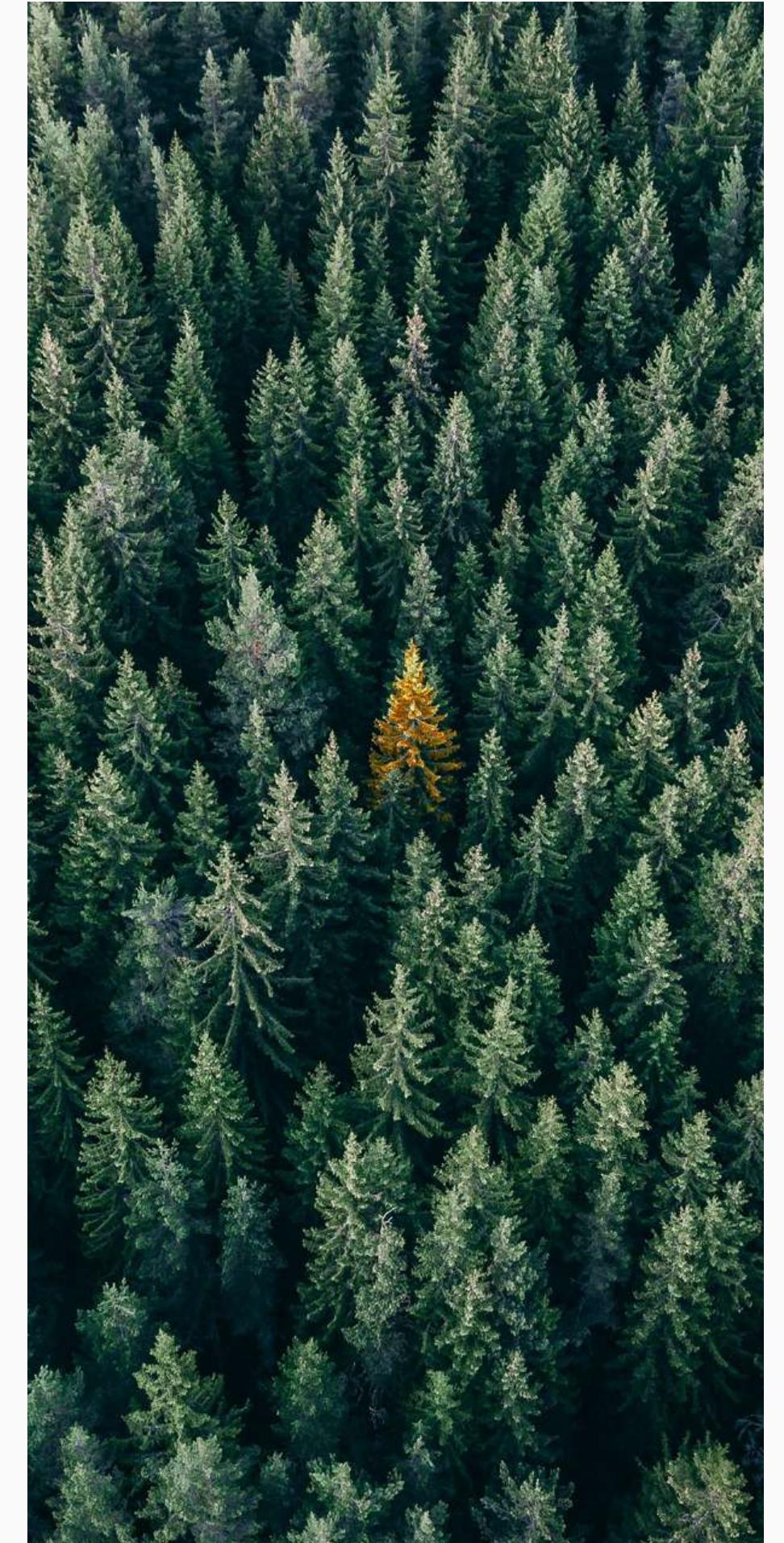


空氣品質分析與預測

國北教大 數資系 陳亭澔

報告大綱

- 分析動機
- 資料來源
- 資料下載及介紹
- 資料處理
- 定義問題
- 資料分析
- 分析結果
- 預測模組
- 結論



分析動機



在工業化發展及健康意識抬頭的社會中，空氣品質逐漸成為非常重要的環境指標，同時也為環境是否安全的評斷之一。

為了瞭解大氣因子對於空氣品質影響及未來空氣品質的預測，我們將透過各地區測站歷年的空氣品質資料及氣候資料進行分析和預測，進而達成目的。

資料來源



環保署 環境資料開放平臺

<https://data.epa.gov.tw>

CWB 觀測資料查詢系統

<https://e-service.cwb.gov.tw/HistoryDataQuery/>

氣象資料開放平臺

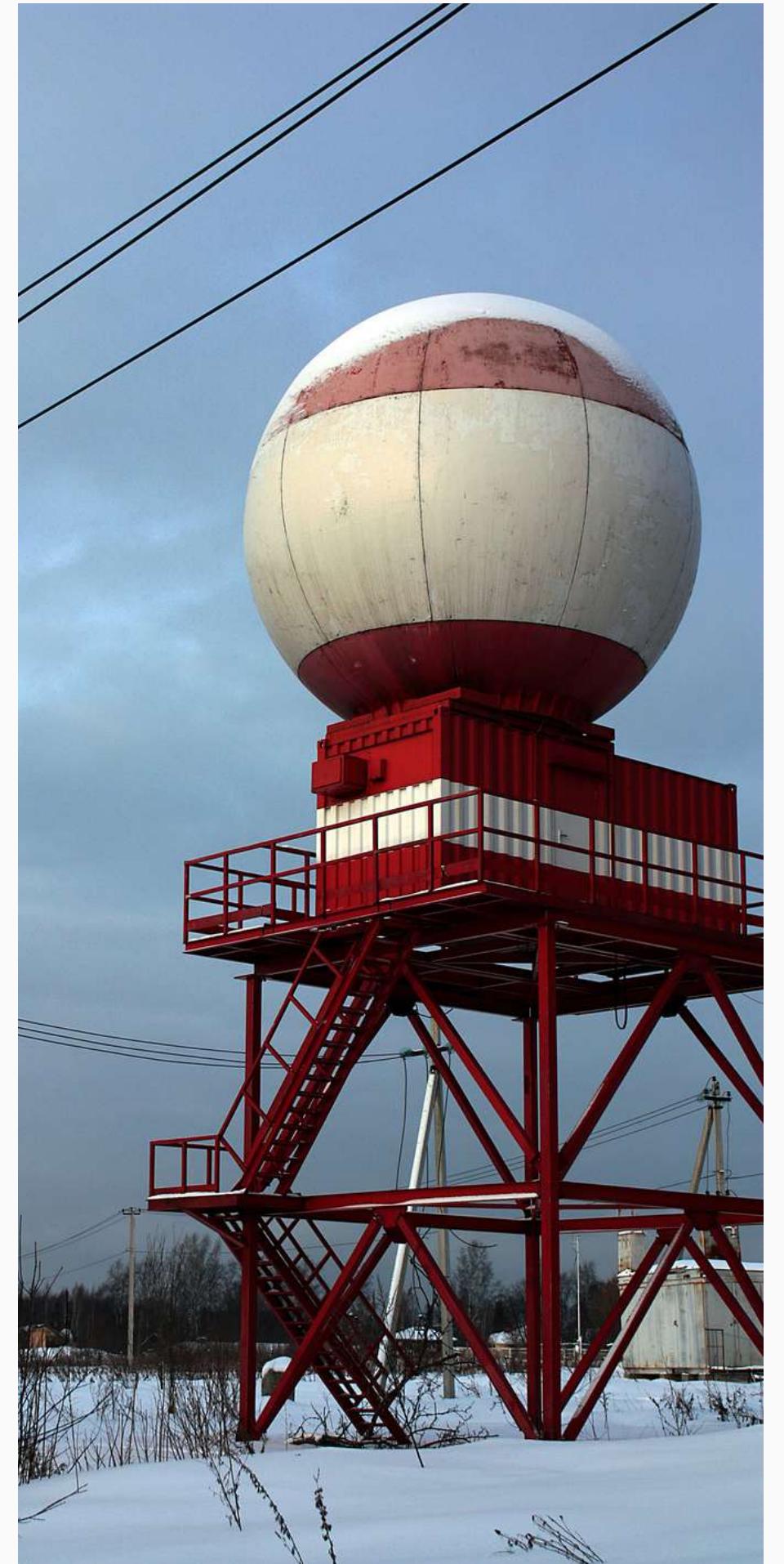
<https://opendata.cwb.gov.tw/index>

資料下載及介紹



氣象測站對應空氣品質測站

- 臺北/萬華
- 板橋/板橋
- 基隆/基隆
- 新屋/觀音
- 湖口/湖口
- 新竹市東區/新竹
- 頭份/頭份
- 臺中/大里
- 田中/彰化
- 埔里/埔里
- 臺西/臺西
- 朴子/朴子
- 嘉義/嘉義
- 臺南/臺南
- 高雄/楠子
- 恆春/恆春
- 宜蘭/宜蘭
- 花蓮/花蓮
- 臺東/臺東
- 澎湖/澎湖
- 金門/金門
- 馬祖/馬祖



資料位置

環保署 環境資料公開平台

資料時間

2016/09~2022/12

測站位置

全台灣

格式

CSV 檔

品質指標(AQI) (2022/09)

品質指標(AQI) (2022/08)

品質指標(AQI) (2022/07)

品質指標(AQI) (2022/06)

說明

下載格式

CSV

JSON

XML

CSV

JSON

XML

CSV

JSON

XML

CSV

JSON

XML

空氣品質指標資料下載

CSV

JSON

XML

CSV

JSON

XML



歷史資料下載

※若開啟csv檔為亂碼，請參考[常見問答之下載CSV格式出現亂碼之操作說明](#)

下載勾選項目 ^

<input type="checkbox"/>	資料集名稱	下載格式	
<input type="checkbox"/>	日空氣品質指標(AQI)	<input type="button" value="CSV"/> <input type="button" value="JSON"/> <input type="button" value="XML"/>	CSV XML JSON
<input checked="" type="checkbox"/>	日空氣品質指標(AQI) (2022/11)	<input type="button" value="CSV"/> <input type="button" value="JSON"/> <input type="button" value="XML"/>	
<input checked="" type="checkbox"/>	日空氣品質指標(AQI) (2022/10)	<input type="button" value="CSV"/> <input type="button" value="JSON"/> <input type="button" value="XML"/>	
<input checked="" type="checkbox"/>	日空氣品質指標(AQI) (2022/09)	<input type="button" value="CSV"/> <input type="button" value="JSON"/> <input type="button" value="XML"/>	
<input checked="" type="checkbox"/>	日空氣品質指標(AQI) (2022/08)	<input type="button" value="CSV"/> <input type="button" value="JSON"/> <input type="button" value="XML"/>	
<input checked="" type="checkbox"/>	日空氣品質指標(AQI) (2022/07)	<input type="button" value="CSV"/> <input type="button" value="JSON"/> <input type="button" value="XML"/>	

資料位置

CWB觀測資料平台

資料時間

2016/09~2022/12

測站位置

指定區域

格式

CSV檔

資料類別 :

時間 :

◎農業氣象觀測網監測系統

◎網頁說明 Readme ◎購買雨量資料說明 ◎站況資訊

◎更新時間為每日12:00 (Updated Time: 12:00)

stem

全臺

Nationwide

北部地區

North Area

中部地區

Central Area

南部地區

South Area



觀測資料查詢 CODiS

CWB Observation Data Inquire System

付款

全臺
Nationwide

北部地區
North Area

中部地區
Central Area

南部地區
South Area

東部地區
East Area

離島
Isl



測站所在縣市：**臺北市 (TaipeiCity)**

測站：**臺北 (TAIPEI)**

資料類型：**資料查詢 (Data Inquiry)**

資料格式：**月報表 (monthly data)**

時間：**2016-09**



[◎農業氣象觀測網監測系統](#)

[◎網頁說明 Readme](#) [◎購買雨量資料說明](#) [◎站況資訊](#)

◎更新時間為每日12:00 (Updated Time: 12:00)



月報表 (monthly data)

測站:466920_臺北

466920_臺北

觀測時間:2016-09

CSV

CSV下載

資料定義請詳見 ◎網頁說明Readme

	press						temperature					Dew Point	RH		
觀測時間 (day)	測站氣壓 (hPa)	海平面氣壓 (hPa)	測站最高氣壓 (hPa)	測站最高氣壓時間 (LST)	測站最低氣壓 (hPa)	測站最低氣壓時間 (LST)	氣溫 (°C)	最高氣溫 (°C)	最高氣溫時間 (LST)	最低氣溫 (°C)	最低氣溫時間 (LST)	露點溫度 (°C)	相對溼度 (%)	最小相對溼度 (%)	最小相對溼度時間 (LST)
ObsTime	StnPres	SeaPres	StnPresMax	StnPresMaxTime	StnPresMin	StnPresMinTime	Temperature	T Max	T Max Time	T Min	T Min Time	Td dew point	RH	RHMin	RHMinTime
01	1000.4	1003.8	1002.2	2016-09-01 00:06	998.2	2016-09-01 16:47	29.6	35.3	2016-09-01 14:07	25.1	2016-09-01 05:47	20.0	57	44	2016-09-01 18:09
02	999.7	1003.1	1000.7	2016-09-02 06:22	998.3	2016-09-02 14:53	28.5	34.3	2016-09-02 12:08	26.0	2016-09-02 18:19	22.2	70	50	2016-09-02 10:29
03	1001.0	1004.5	1002.6	2016-09-03 09:27	999.4	2016-09-03 03:01	27.9	31.8	2016-09-03 13:49	26.1	2016-09-03 04:09	24.4	82	67	2016-09-03 15:11
04	1002.2	1005.6	1003.6	2016-09-04 20:34	1000.5	2016-09-04 14:43	29.7	34.4	2016-09-04 13:09	27.1	2016-09-04 05:58	24.6	75	52	2016-09-04 12:52
05	1003.1	1006.5	1004.4	2016-09-05 20:41	1001.1	2016-09-05 02:21	28.8	31.5	2016-09-05 15:12	27.5	2016-09-05 01:30	24.8	79	73	2016-09-05 16:26
06	1004.1	1007.6	1005.7	2016-09-06 08:41	1002.9	2016-09-06 15:37	28.0	30.3	2016-09-06 15:59	25.4	2016-09-06 23:51	23.8	78	72	2016-09-06 15:49
07	1003.3	1006.8	1004.8	2016-09-07 00:04	1001.6	2016-09-07 14:16	29.4	34.7	2016-09-07 12:52	25.3	2016-09-07 02:48	23.7	73	51	2016-09-07 15:47
08	1004.3	1007.8	1006.3	2016-09-08 21:15	1002.8	2016-09-08 04:09	29.4	34.2	2016-09-08 13:44	26.9	2016-09-08 20:12	23.3	70	53	2016-09-08 13:43
09	1005.6	1009.1	1006.8	2016-09-09 20:22	1004.1	2016-09-09 14:54	27.2	30.6	2016-09-09 11:46	25.3	2016-09-09 18:51	24.0	83	70	2016-09-09 11:08
10	1005.1	1008.6	1006.4	2016-09-10 21:13	1003.6	2016-09-10 14:22	26.5	32.3	2016-09-10 11:38	24.8	2016-09-10 03:57	24.0	87	67	2016-09-10 11:37
									2016-09-11		2016-09-11				

資料處理

```
> .sf-sub-indicator {  
    color: #000 !important;  
}  
nav .cart-menu .cart-icon-wr  
outer.transparent header#top  
nav .sf-menu > li.current_page  
nav .sf-menu > li.current-menu  
nav > ul > li > a:hover > .sf-sub  
nav ul #search-btn a:hover span,  
nav .sf-menu > li.current-menu-li  
a:hover .icon-salient-cart,.ascend  
:1!important;color:#ffffff!imp  
rent header#top nav>ul>li.but  
t-widget-area-toggle a i.l  
outer.transparent
```

```
1 #調用操作系統命令(查詢文件)
2 import os
3
4 #機器學習模組
5 from sklearn.linear_model import LogisticRegression
6 from sklearn.model_selection import train_test_split
7 #標準化
8 from sklearn.preprocessing import StandardScaler
9
10 # 視覺化函式庫
11 import matplotlib.pyplot as plt
12 from matplotlib.pyplot import MultipleLocator
13 import matplotlib as mpl
14 import seaborn as sns
15 %matplotlib inline
16
17 # 資料加工、處理、分析函式庫
18 import numpy as np
19 import numpy.random as random
20 import scipy as sp
21 from pandas import Series, DataFrame
22 import pandas as pd
```

OS 套件

提供與作業系統相關操作的捷徑

Sklearn 套件

train_test_split：將資料切割成訓練資料及測試資料

StandardScaler：將資料標準化

資料視覺化

資料加工、處理、分析

np、random、sp、pandas等資料分析
套件

空氣品質資料匯入

```
1 path = 'airquality'  
2 files = os.listdir(path)  
3 #定義一個空列表  
4 data_list = []  
5 #aqi轉換分級  
6 def aqi_to_level(aqi):  
7     if aqi <= 50:  
8         return 1  
9     elif aqi <= 100:  
10        return 2  
11    elif aqi <= 150:  
12        return 3  
13    elif aqi <= 200:  
14        return 4  
15    elif aqi <= 300:  
16        return 5  
17    else:  
18        return 6
```

`os.listdir(path)` 返回一個包含path的路徑的所有檔案名稱

`def aqi_to_level (aqi):` 定義aqi分級方法，判斷資料中 aqi數值大小並進行分級

aqi值	~50	51~100	101~150	151~200	201~300	301~
品質級數	1	2	3	4	5	6



空氣品質資料匯入



```
20 # 迴圈讀取檔案  
21 for file in files:  
22     tmp = pd.read_csv(path + '/' + file)[['\\"sitename\\"', '\\"monitordate\\"', '\\"aqi\\"']]  
23     data_list.append(tmp)  
24     tmp['aqiLevel'] = tmp['\\"aqi\\"'].map(aqi_to_level)  
25 airquality = pd.concat(data_list)  
26 airquality = airquality.rename(columns={'\\"sitename\\"': 'sitename', '\\"monitordate\\"' : '觀測時間(day)', '\\"aqi\\"' : 'aqi'})  
27 airquality
```

Out[109]:

	sitename	觀測時間(day)	aqi	aqiLevel
0	松山	2016-09-01	156.0	4
1	富貴角	2016-09-01	-1.0	1
2	麥寮	2016-09-01	105.0	3
3	關山	2016-09-01	33.0	1
4	馬公	2016-09-01	73.0	2
...
2335	馬公	2022-11-30	34.0	1
2336	關山	2022-11-30	31.0	1
2337	麥寮	2022-11-30	56.0	2
2338	富貴角	2022-11-30	38.0	1
2339	大城	2022-11-30	34.0	1

for file in files:

1. 抓取airquality資料夾中的一個.csv檔，並取檔案中的\siteName\、\\monitordate\\、\\aqi\\這三欄的值並存入tmp

2. 將tmp元素加在data_list串列後面

3. 將tmp中的\\aqi\\進行aqi_to_level方法，並將值放在tmp新的aqiLevel欄中

pd.concat(data_list)

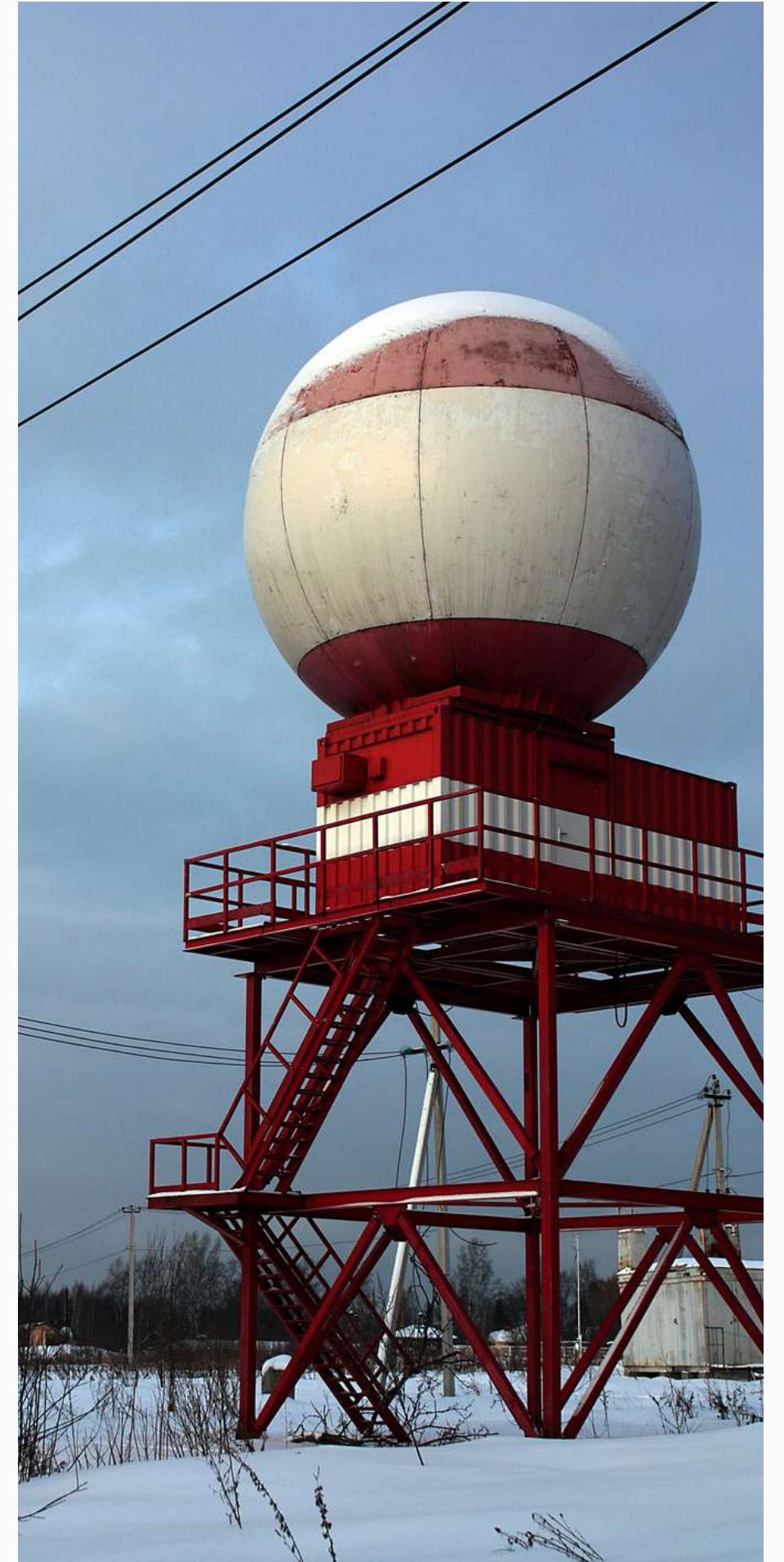
合併資料

airquality.rename()

行欄改名

氣象測站對應空氣品質測站

- 臺北/萬華
- 板橋/板橋
- 基隆/基隆
- 新屋/觀音
- 湖口/湖口
- 新竹市東區/新竹
- 頭份/頭份
- 臺中/大里
- 田中/彰化
- 埔里/埔里
- 臺西/臺西
- 朴子/朴子
- 嘉義/嘉義
- 臺南/臺南
- 高雄/楠子
- 恒春/恒春
- 宜蘭/宜蘭
- 花蓮/花蓮
- 臺東/臺東
- 澎湖/澎湖
- 金門/金門
- 馬祖/馬祖



氣象測站觀測資料匯入



```
1 path = 'weather'
2 files = os.listdir(path)
3 #定義一個空列表
4 data_list = []
5 #宣告字典，對應檔名測站代號
6 station_dict = {'萬華': '466920', '板橋': '466880', '基隆': '466940', '觀音': '467050', '湖口': 'C0D650', '新竹': 'C0D660',
7 '頭份': 'C0E730', '大里': '467490', '彰化': '467270', '埔里': 'C0H890', '臺西': 'C0K530', '朴子': 'C0M650',
8 '嘉義': '467480', '臺南': '467410', '楠梓': '467440', '恆春': '467590', '宜蘭': '467080', '花蓮': '466990',
9 '臺東': '467660', '馬公': '467350', '金門': '467110', '馬祖': '467990'}
10 #反轉key-value
11 station_dict = {v : k for k, v in station_dict.items()}
```

宣告字典

氣象測站資料檔案名稱是英文代碼作為測站的代表，所以宣告一個station_dict 測站字典將測站名稱及代碼做對應

key-value反轉

將字典中的key及value作反轉，以利後續讀檔

氣象測站觀測資料匯入



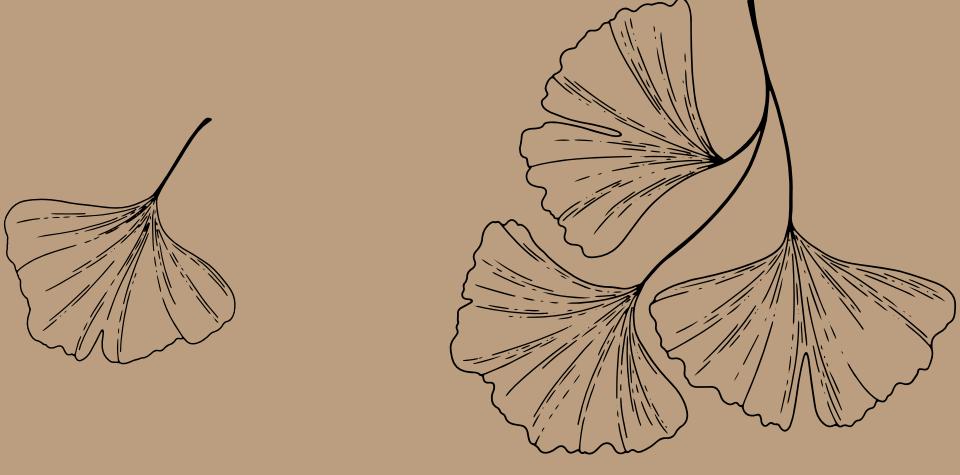
```
14 #區域分類
15 def region(sitename):
16     if sitename == '萬華' or sitename == '板橋' or sitename == '基隆' or sitename == '觀音' or sitename == '湖口' or sitename ==
17         return 'n'
18     if sitename == '大里' or sitename == '彰化' or sitename == '埔里' or sitename == '臺西':
19         return 'm'
20     if sitename == '朴子' or sitename == '嘉義' or sitename == '臺南' or sitename == '楠梓' or sitename == '恆春':
21         return 's'
22     if sitename == '宜蘭' or sitename == '花蓮' or sitename == '臺東':
23         return 'e'
24     if sitename == '馬公' or sitename == '金門' or sitename == '馬祖':
25         return 'o'
```

def region(sitename):

定義區域分級方法，判斷資料中sitename是在哪個區域並進行分類編號回傳

sitename	分類編號回傳
萬華、板橋、基隆、觀音、湖口、新竹、頭份	n
大里、彰化、埔里、臺西	m
朴子、嘉義、臺南、楠梓、恆春	s
宜蘭、花蓮、臺東	e
馬公、金門、馬祖	o

氣象測站觀測資料匯入



```
27 # 迴圈讀取檔案
28 for file in files:
29     tmp = pd.read_csv(path + '/' + file)[['觀測時間(day)', '氣溫(°C)', '相對溼度(%)', '風速(m/s)', '風向(360degree)', '降水量('
30     tmp = tmp.drop(0, axis = 0)
31     file = file.split('-')
32     tmp.insert(0, 'sitename', station_dict[file[0]])
33     tmp['觀測時間(day)'] = [file[1] + '-' + file[2][0:2] + '-% s' % i for i in tmp['觀測時間(day)']]
34     tmp['region'] = tmp['sitename'].map(region)
35     tmp.insert(1, 'month', int(file[2][0:2]))
36     tmp['P'] = tmp['降水量(mm)'].map(lambda x: 0 if x == '0.0' else 1)
37     tmp = tmp.drop('降水量(mm)', axis = 1)
38     data_list.append(tmp)
39 weather_obs = pd.concat(data_list)
40 weather_obs = weather_obs.rename(columns={'氣溫(°C)': 'T', '相對溼度(%)': 'RH', '風速(m/s)': 'WS', '風向(360degree)': 'WD',
41 weather_obs
```

for file in files:

1.抓取weather資料夾中的一個.csv檔，並取檔案中的"觀測時間(day)"、"氣溫(°C)"、"相對溼度(%)"、"風速(m/s)"、"風向(360degree)"、"降水量(mm)"、"日最高紫外線指數"這七欄的值並存入tmp

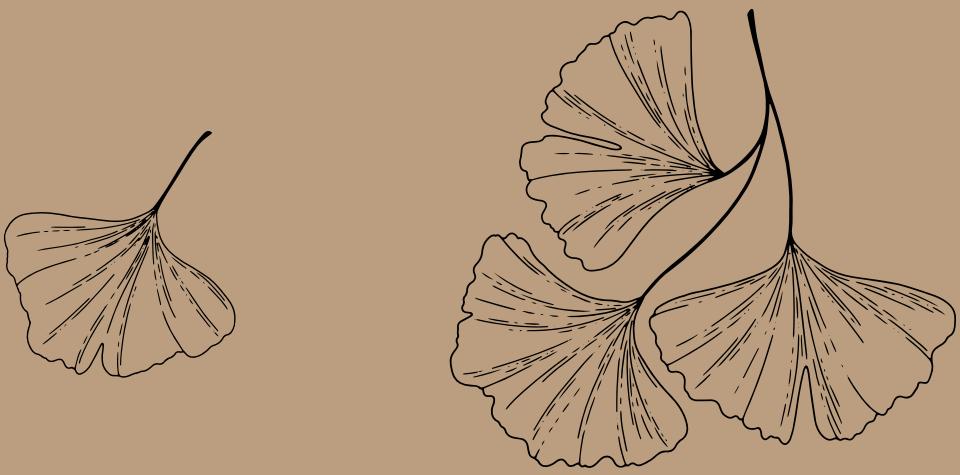
2.file.split('-')：將檔名依照'-'進行分割

3.根據測站代碼(file[0])對應字典，將測站名稱加入sitename欄位

file[0] | file[1] | file[2]

466880 | 2016-09.csv

氣象測站觀測資料匯入



```
27 # 迴圈讀取檔案
28 for file in files:
29     tmp = pd.read_csv(path + '/' + file)[['觀測時間(day)', '氣溫(°C)', '相對溼度(%)', '風速(m/s)', '風向(360degree)', '降水量(mm)']]
30     tmp = tmp.drop(0, axis = 0)
31     file = file.split('-')
32     tmp.insert(0, 'sitename', station_dict[file[0]])
33     tmp['觀測時間(day)'] = [file[1] + '-' + file[2][0:2] + '-% s' % i for i in tmp['觀測時間(day)']]
34     tmp['region'] = tmp['sitename'].map(region)
35     tmp.insert(1, 'month', int(file[2][0:2]))
36     tmp['P'] = tmp['降水量(mm)'].map(lambda x: 0 if x == '0.0' else 1)
37     tmp = tmp.drop('降水量(mm)', axis = 1)
38     data_list.append(tmp)
39 weather_obs = pd.concat(data_list)
40 weather_obs = weather_obs.rename(columns={'氣溫(°C)': 'T', '相對溼度(%)': 'RH', '風速(m/s)': 'WS', '風向(360degree)': 'WD',
41 weather_obs
```

4.根據file[1](年分)之值及file[2][0:2](取前兩位，月份)之值，將年份、月份及欄位原有日期加入**觀測時間(day)**欄位中

5.利用map()函式將**sitename**欄位中之值透過定義好的**region**函式轉換成所在區域代碼

6.**降水量(mm)**欄位中之值若為**0**則回傳**0**，否則回傳**1**至**P**欄位，代表有無降雨

7.垂直合併所有測站資料

8.將欄位改名

file[0] | file[1] | file[2]

466880 | 2016-09.csv

氣象及空氣品質資料結合



#抓sitename、觀測時間相同的值做合併

```
all_data = pd.merge(airquality, weather_obs, on = ['sitename', '觀測時間(day)'])  
all_data
```

	sitename	觀測時間(day)	aqi	aqiLevel	month	T	RH	WS	WD	UVI	region	P
0	馬公	2016-09-01	73.0	2	9	28.9	79	3.6	190	9	o	0
1	金門	2016-09-01	77.0	2	9	29.0	83	2.4	190	8	o	0
2	馬祖	2016-09-01	117.0	3	9	28.1	79	3.8	200	6	o	0
3	埔里	2016-09-01	102.0	3	9	26.3	72	0.8	211	...	m	0
4	宜蘭	2016-09-01	63.0	2	9	27.8	66	1.9	280	8	e	0
...
50131	宜蘭	2022-11-30	29.0	1	11	19.9	93	3.0	50	4	e	1
50132	埔里	2022-11-30	40.0	1	11	24.1	70	0.7	0	...	m	0
50133	馬祖	2022-11-30	27.0	1	11	12.8	94	7.8	30	1	o	1
50134	金門	2022-11-30	26.0	1	11	17.0	83	7.9	50	2	o	0
50135	馬公	2022-11-30	34.0	1	11	22.0	83	6.9	20	5	o	0

pd.merge()

1.利用pd.merge()函式將空氣品質資料與氣象觀測資料根據sitename 及 觀測時間(day)欄位進行**同值合併**

空值或觀測異常值檢查



```
1 all_data = all_data.drop(['sitename', '觀測時間(day)'], axis = 1)
```

```
1 print(all_data.isnull().any())
2 print(all_data.isin(['X']).sum())
3 print(all_data.isin(['V']).sum())
4 print(all_data.isin(['/']).sum())
5 print(all_data.isin(['...']).sum())
6 all_data = all_data.dropna()
7 all_data = all_data.replace('/', np.nan).dropna()
8 all_data = all_data.replace('X', np.nan).dropna()
9 all_data = all_data.replace('...', np.nan).dropna()
```

isnull().any()

計算欄位中有無**null**欄位

isin(['X']).sum()

計算個欄位中**指定特徵**的數量

replace('X',np.nan).dropna()

將欄位中的**指定特徵**換成**空值**，並將空值的資料**刪除**

(指定特徵)

X 故障

V 風向不定

/ 不明

... 無觀測

空值或觀測異常值檢查



aqi	True	aqi	0	aqi	0	aqi	0	aqi	0	aqi	0
aqiLevel	False	aqiLevel	0	aqiLevel	0	aqiLevel	0	aqiLevel	0	aqiLevel	0
month	False	month	0	month	0	month	0	month	0	month	0
T	False	T	31	T	0	T	8	T	1643		
RH	False	RH	51	RH	0	RH	8	RH	1643		
WS	False	WS	17	WS	0	WS	6	WS	1643		
WD	False	WD	17	WD	0	WD	6	WD	1643		
UVI	False	UVI	92	UVI	0	UVI	0	UVI	0	UVI	17673
region	False	region	0	region	0	region	0	region	0	region	0
P	False	P	0	P	0	P	0	P	0	P	0
		dtype: int64		dtype: int64		dtype: int64		dtype: int64		dtype: int64	

各欄中是否有
NULL(空值)

各欄中是否有
X (故障)

各欄中是否有
V (風向不定)

各欄中是否有
/ (不明)

各欄中是否有
... (無觀測)

資料型別轉換

```
1 print('資料型別的確認（型別轉換前）')
2 print(all_data.dtypes)
3 all_data = all_data.assign(T = pd.to_numeric(all_data['T']))
4 all_data = all_data.assign(RH = pd.to_numeric(all_data['RH']))
5 all_data = all_data.assign(WS = pd.to_numeric(all_data['WS']))
6 all_data = all_data.assign(WD = pd.to_numeric(all_data['WD']))
7 all_data = all_data.assign(UVI = pd.to_numeric(all_data['UVI']))
8 print('資料型別的確認（型別轉換後）')
9 print(all_data.dtypes)
```

dtypes

顯示各欄之值的資料型別

assign(T = pd.to_numeric(all_data['T']))

指派將特定欄位的型別**轉換成數字型別**

資料型別的確認（型別轉換前）

aqi	float64
aqiLevel	int64
month	int64
T	object
RH	object
WS	object
WD	object
UVI	object
region	object
P	int64
dtype:	object

資料型別的確認（型別轉換後）

aqi	float64
aqiLevel	int64
month	int64
T	float64
RH	int64
WS	float64
WD	int64
UVI	int64
region	object
P	int64
dtype:	object

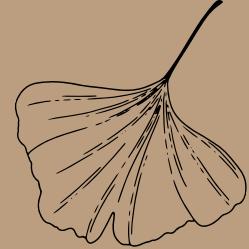
類別變數1/0化

```
1 #類別變數1/0化  
2 all_data = pd.get_dummies(all_data)  
3 all_data
```

	aqi	aqiLevel	month	T	RH	WS	WD	UVI	P	region_e	region_m	region_n	region_o	region_s
0	73.0	2	9	28.9	79	3.6	190	9	0	0	0	0	1	0
1	77.0	2	9	29.0	83	2.4	190	8	0	0	0	0	1	0
2	117.0	3	9	28.1	79	3.8	200	6	0	0	0	0	1	0
4	63.0	2	9	27.8	66	1.9	280	8	0	1	0	0	0	0
5	56.0	2	9	27.9	77	2.3	240	9	0	1	0	0	0	0
...
50130	31.0	1	11	22.5	81	6.5	40	6	1	1	0	0	0	0
50131	29.0	1	11	19.9	93	3.0	50	4	1	1	0	0	0	0
50133	27.0	1	11	12.8	94	7.8	30	1	1	0	0	0	1	0
50134	26.0	1	11	17.0	83	7.9	50	2	0	0	0	0	1	0
50135	34.0	1	11	22.0	83	6.9	20	5	0	0	0	0	1	0

32335 rows × 14 columns

region

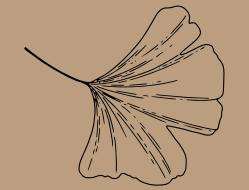
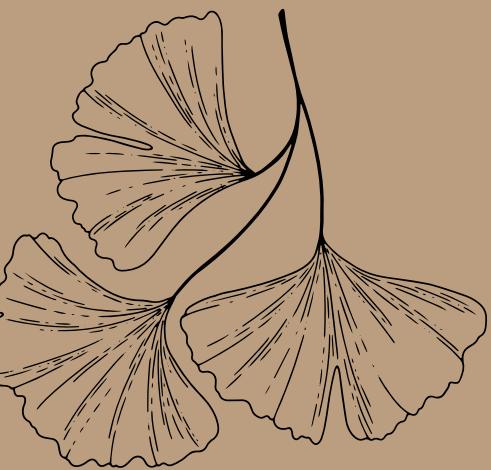


pd.get_dummies()

將類別變數轉成標籤變數

101001100010111
000111010111001
011010101001010
110110110101010
101010101001010

資料切割



```
1 X = all_data.drop(['aqi', 'aqiLevel'], axis = 1)
2 y = all_data["aqi"]
3 X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0, test_size = 0.2)
```

X = all_data.drop(['aqi', 'aqiLevel'], axis = 1)

X = 所有資料除了 aqi 及 aqiLevel 兩欄之外

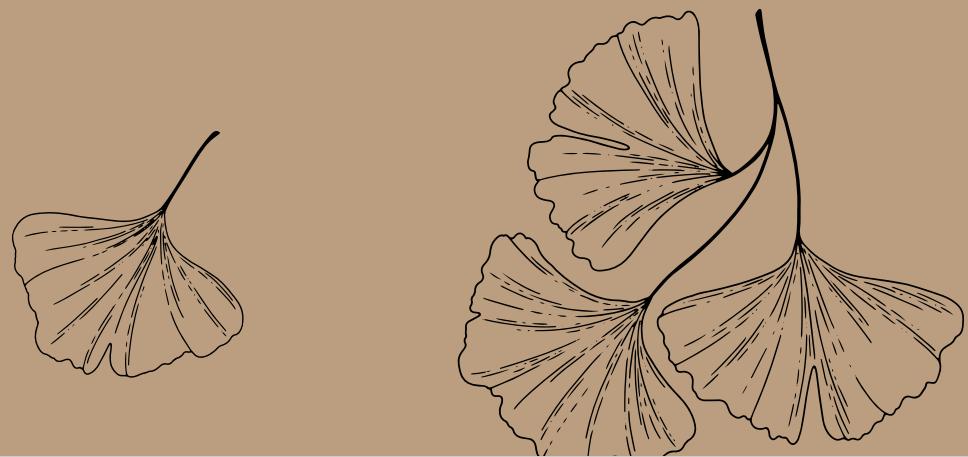
y = all_data['aqi']

y = aqi 該欄之值

X_train, X_test, y_train, y_test = train_test_split(X, y ,random_state = 0 ,test_size = 0.2)

將 X, y 分割成 X_train, X_test, y_train, y_test 這四種，而資料分成 train = 0.8 , test = 0.2

資料標準化



```
1 sc = StandardScaler()  
2 #將數據減去平均值，再除以變異數  
3 sc.fit(X_train)  
4 X_train = sc.transform(X_train)  
5 X_test = sc.transform(X_test)
```

1 X_train

```
array([[ -0.4887812 ,  0.5047178 ,  1.03365483, ..., -0.51297916,  
       1.94916851, -0.61076681],  
      [-0.77958348, -0.3910626 ,  0.47636417, ..., -0.51297916,  
       1.94916851, -0.61076681],  
      [ 0.96523018,  1.01368394,  0.2534479 , ...,  1.94939695,  
      -0.51303928, -0.61076681],  
      ...,  
      [ 0.38362563,  1.56336737, -1.52988223, ..., -0.51297916,  
      -0.51303928, -0.61076681],  
      [ 0.96523018, -0.1671175 , -0.86113343, ...,  1.94939695,  
      -0.51303928, -0.61076681],  
      [-1.07038576, -0.8389528 , -1.19550783, ..., -0.51297916,  
      -0.51303928,  1.63728609]])
```

StandardScaler()

將所有特徵標準化，使數據平均值為0，變異數為1

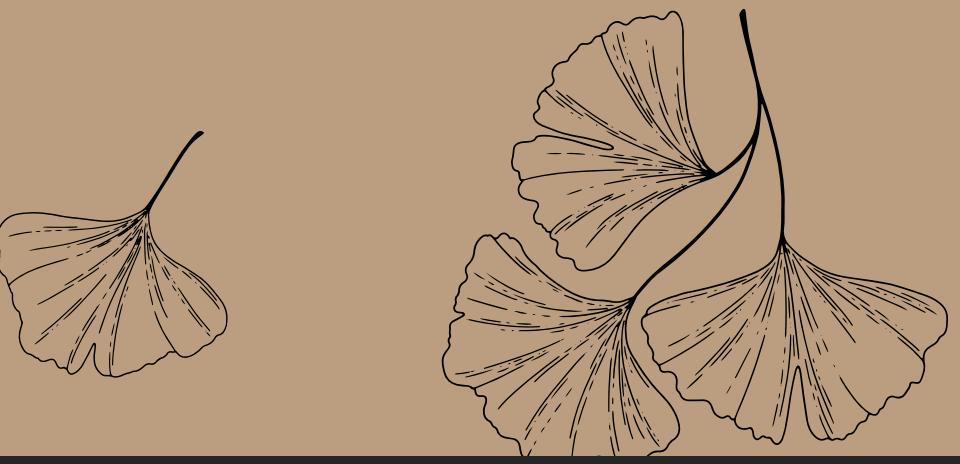
fit()

通過fit函數可以先對需要歸一化的數據集進行的計算

transform()

利用fit()算出的屬性進行標準化

Linear Regression



```
1 from sklearn.linear_model import LinearRegression
2 # 建立線性迴歸模型
3 LR_model = LinearRegression()
4 LR_model.fit(X_train, y_train)
5 # 顯示迴歸係數
6 print('\n迴歸係數\n{}'.format(pd.Series(LR_model.coef_, index = X.columns)))
7 print('截距: {:.3f}'.format(LR_model.intercept_))
8 # 顯示判定係數
9 print('判定係數(train): {:.3f}'.format(LR_model.score(X_train,y_train)))
10 print('判定係數(test): {:.3f}'.format(LR_model.score(X_test,y_test)))
```

迴歸係數

```
month      -1.065266
T          -6.474940
RH         -6.971041
WS         -5.439702
WD          2.177731
UVI        -4.507642
P          -5.800842
region_e   -6.958738
region_m   -0.803657
region_n   -0.721025
region_o    4.653828
region_s    3.349204
dtype: float64
截距: 56.469
判定係數(train):0.316
判定係數(test):0.313
```

LR_model = LinearRegression()

套用LinearRegression進行**線性回歸模型**

**pd.Series(LR_model.coef_,
index = X.columns)**

算出X欄位的**回歸係數**

.intercept_

算出線性回歸模型之**截距**

.score()

算出 **train** 資料及 **test** 資料 線性回歸模型的**判定係數**

Linear Regression



```
1 from sklearn.linear_model import LinearRegression
2 # 建立線性迴歸模型
3 LR_model = LinearRegression()
4 LR_model.fit(X_train, y_train)
5 # 顯示迴歸係數
6 print('\n迴歸係數\n{}'.format(pd.Series(LR_model.coef_, index = X.columns)))
7 print('截距: {:.3f}'.format(LR_model.intercept_))
8 # 顯示判定係數
9 print('判定係數(train): {:.3f}'.format(LR_model.score(X_train,y_train)))
10 print('判定係數(test): {:.3f}'.format(LR_model.score(X_test,y_test)))
```

迴歸係數

```
month      -1.065266
T          -6.474940
RH         -6.971041
WS         -5.439702
WD          2.177731
UVI        -4.507642
P          -5.800842
region_e   -6.958738
region_m   -0.803657
region_n   -0.721025
region_o    4.653828
region_s    3.349204
dtype: float64
截距: 56.469
判定係數(train):0.316
判定係數(test):0.313
```

LR_model = LinearRegression()

套用LinearRegression進行**線性回歸模型**

**pd.Series(LR_model.coef_,
index = X.columns)**

算出X欄位的**回歸係數**

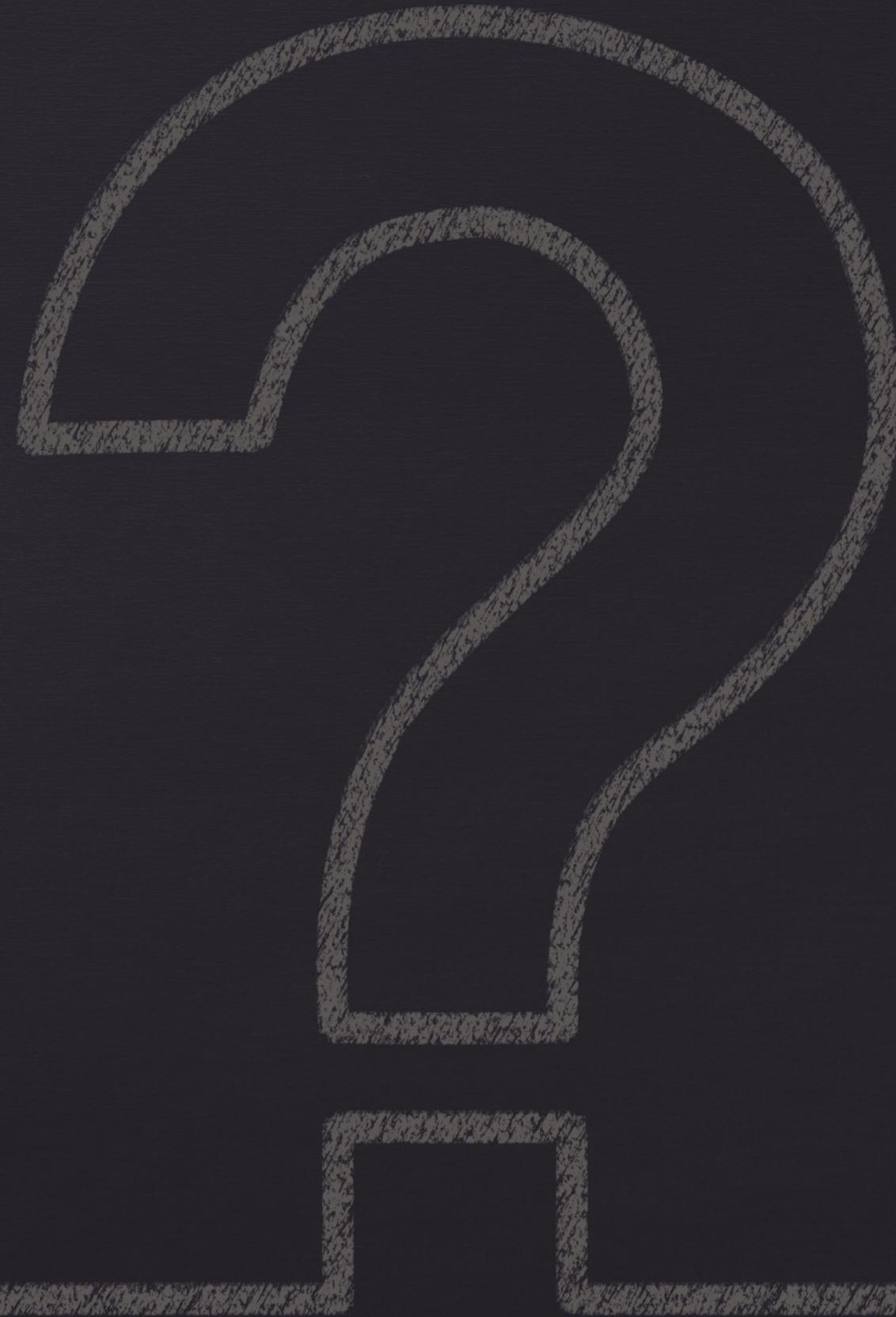
.intercept_

算出線性回歸模型之**截距**

.score()

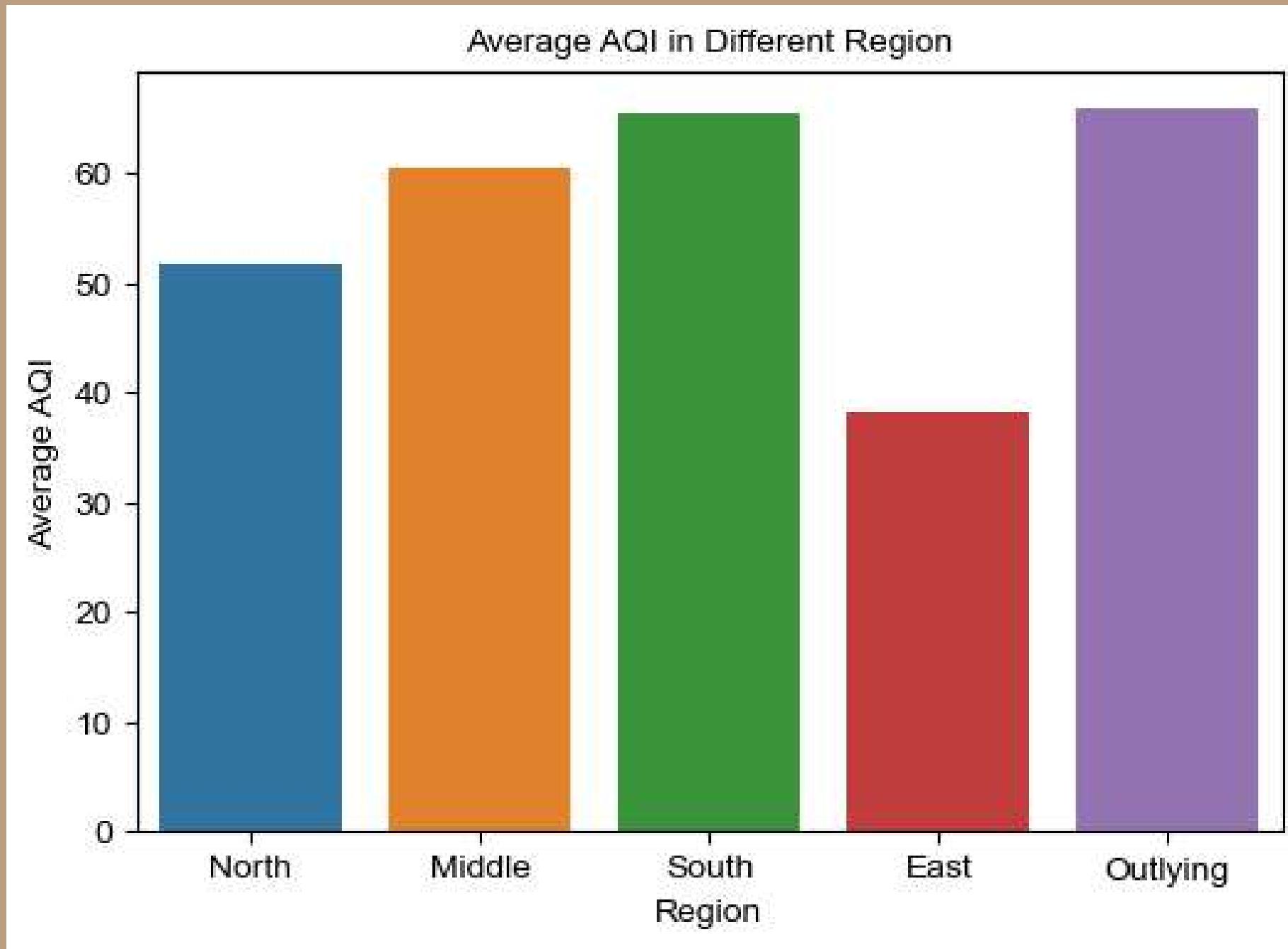
算出 **train** 資料及 **test** 資料 線性回歸模型的**判定係數**

主義問題



- 了解臺灣各區域的空氣品質狀況
- 探究空氣品質與其他大氣因素的關係
- 臺灣各地歷年空氣品質變化是劇增或是下降

資料分析及結果



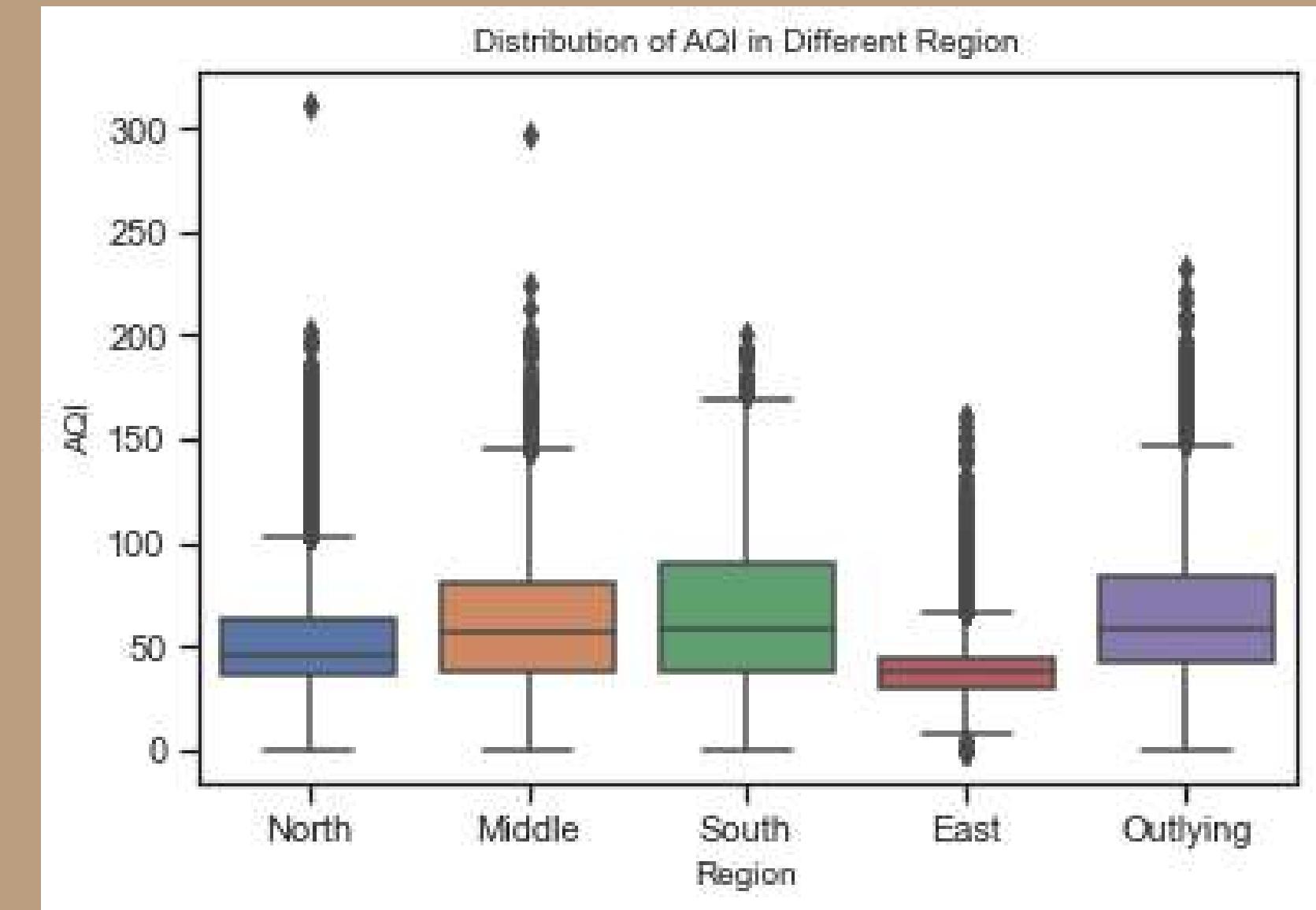
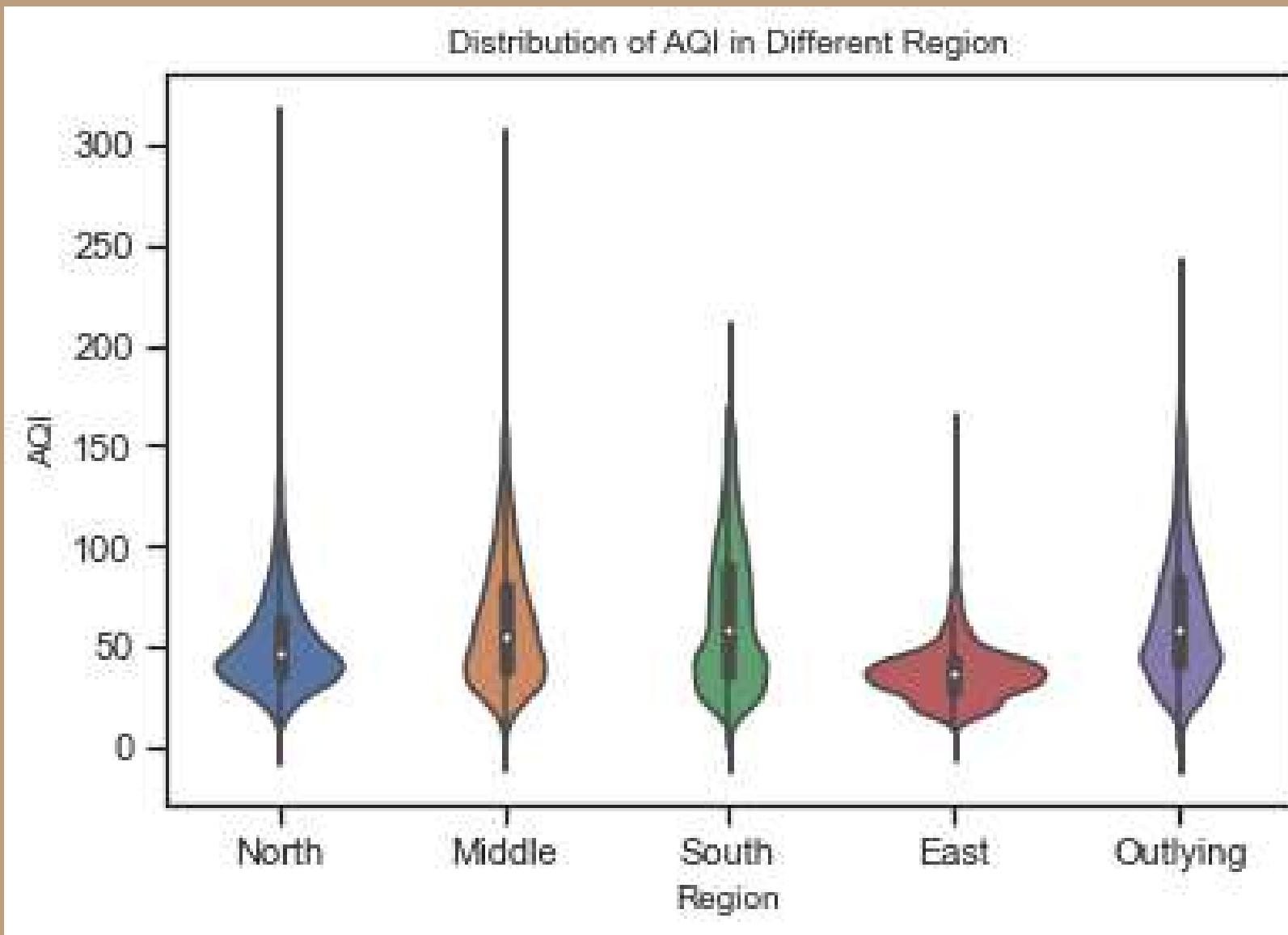
各地區平均AQI值 長條圖

- **南部與離島**地區的平均空氣品質指標高過其他三個區域
- **東部**地區空氣品質較好



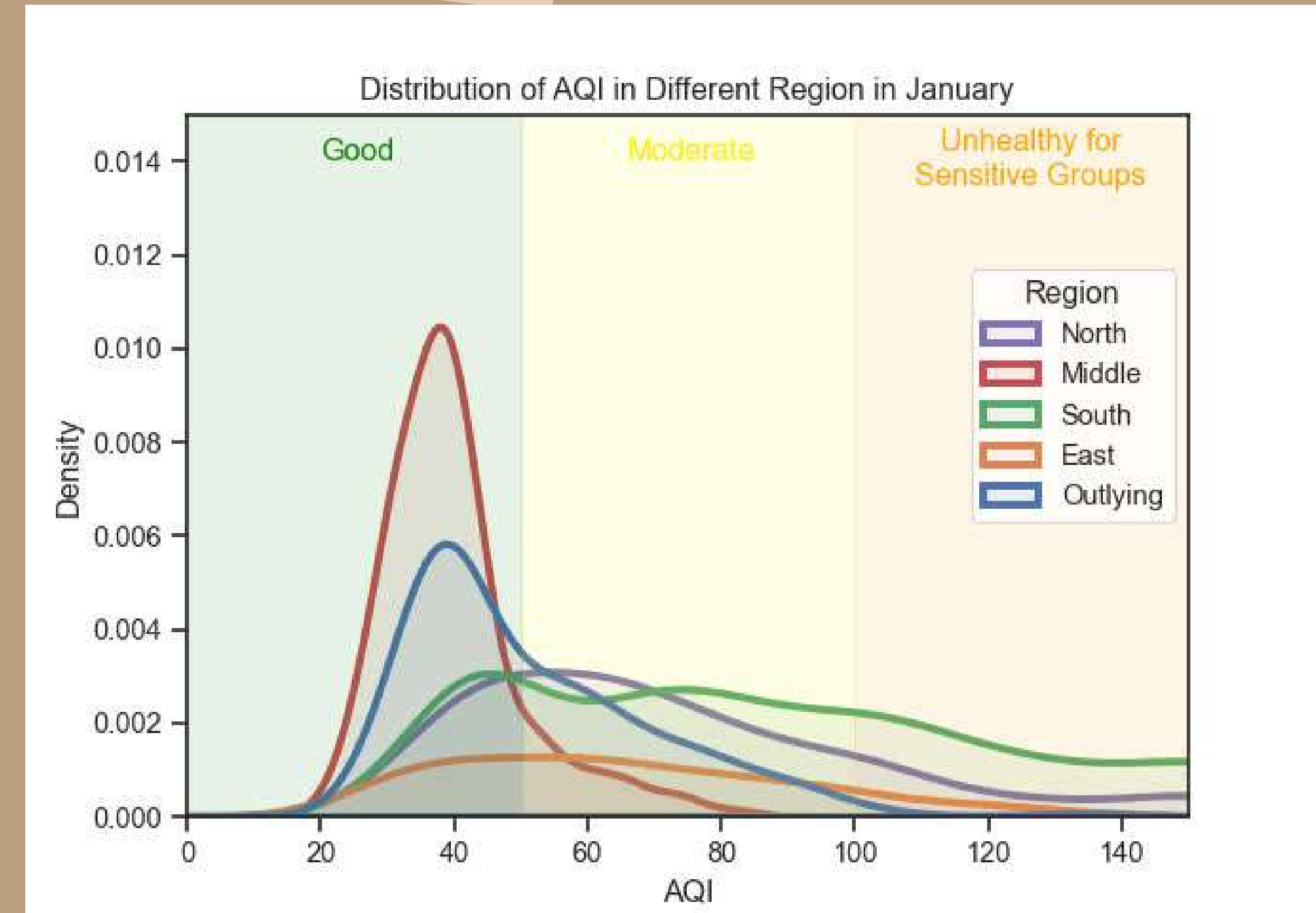
資料分析及結果

各地區AQI值 小提琴圖 箱型圖

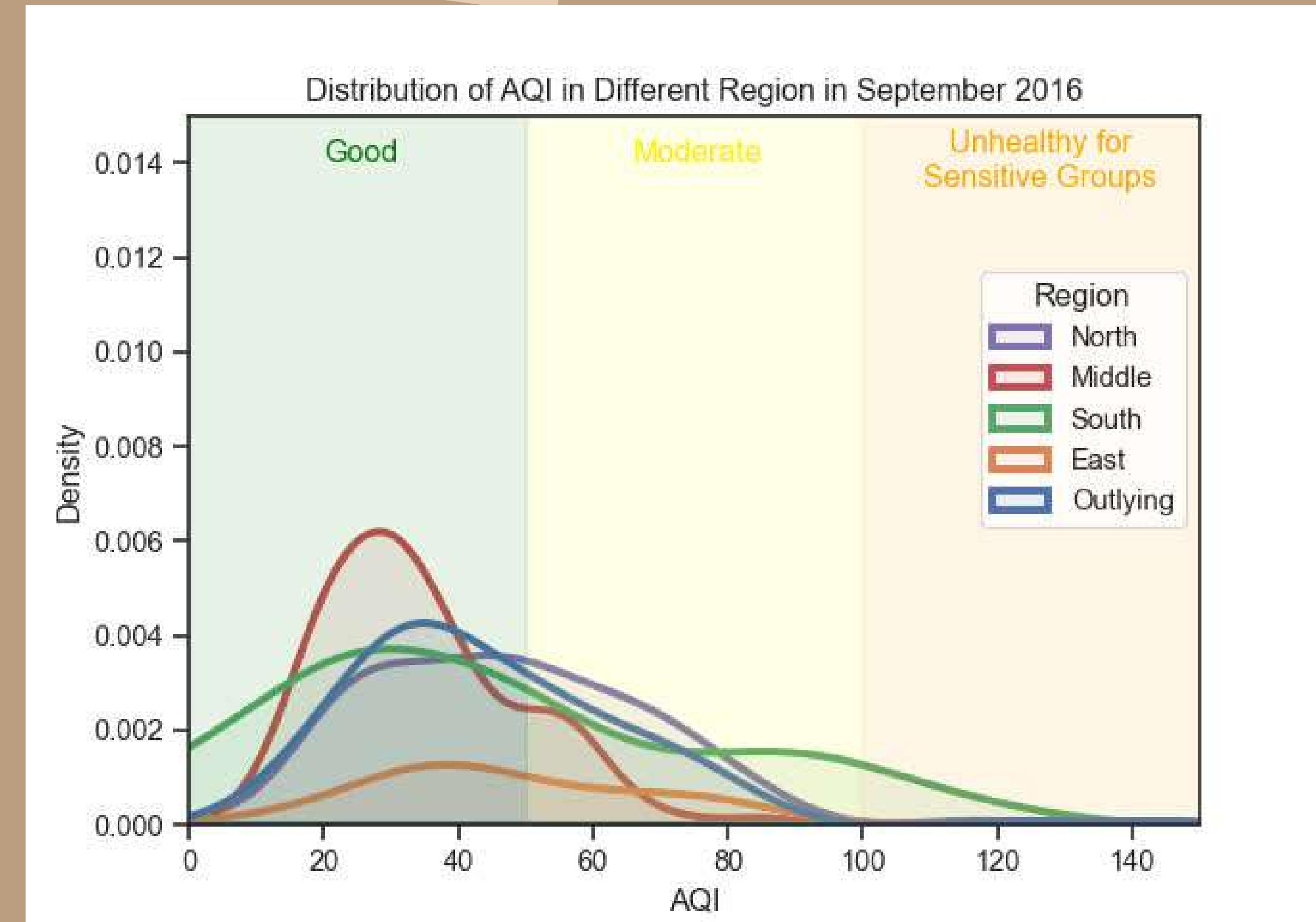


- 南部及離島的AQI指數**範圍較廣**
- 東部AQI**中位數**較其他區域**低**

資料分析及結果

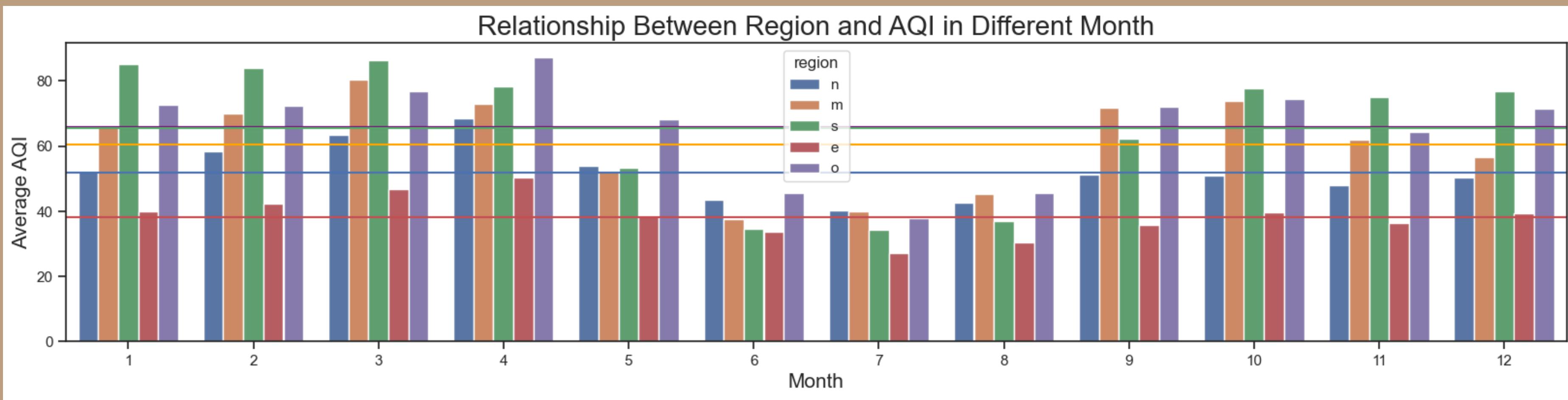


資料分析及結果



資料分析及結果

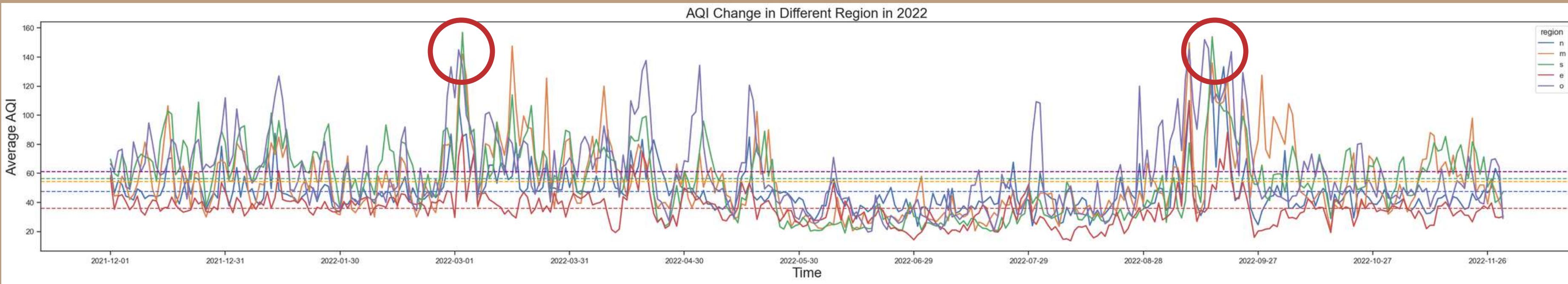
各地區月平均AQI值 長條圖



- 各區域夏季AQI普遍低於冬季
- 中南部在冬季及春季時，AQI指數飆升並超越北部及東部
- 北部四季的AQI指數變化較為平緩，冬季略高於夏季
- 東部四季的AQI指數皆低於其他區域

資料分析及結果

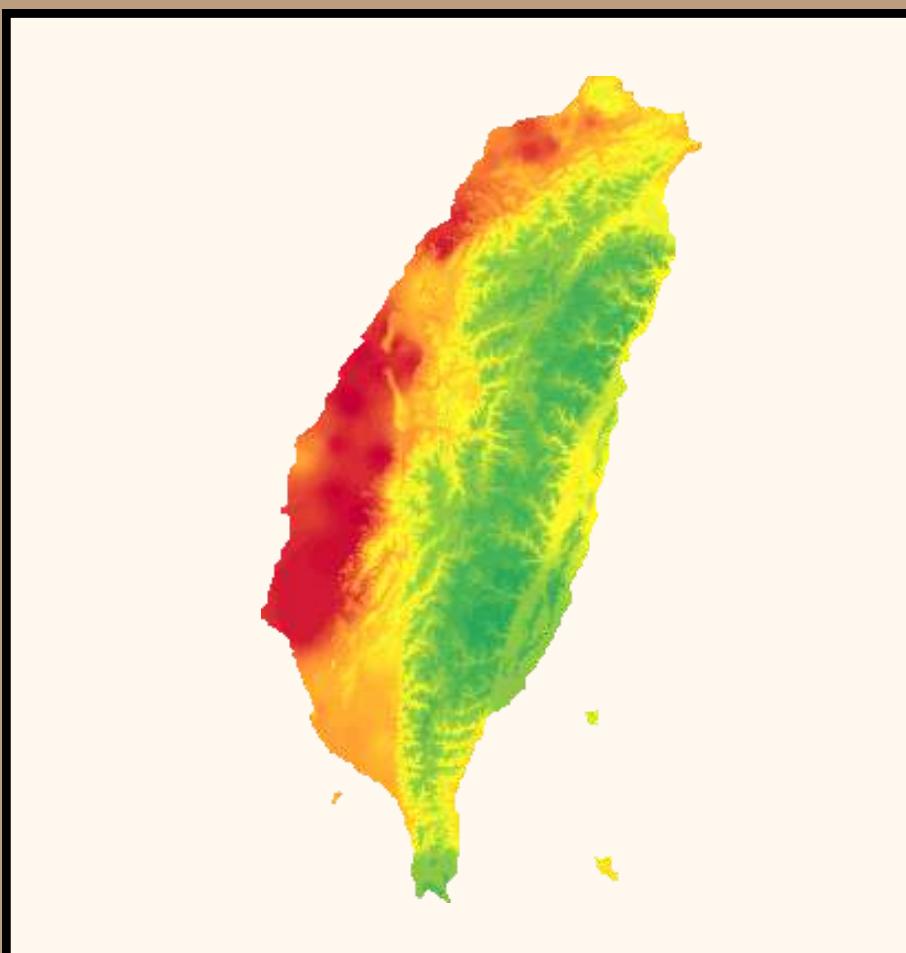
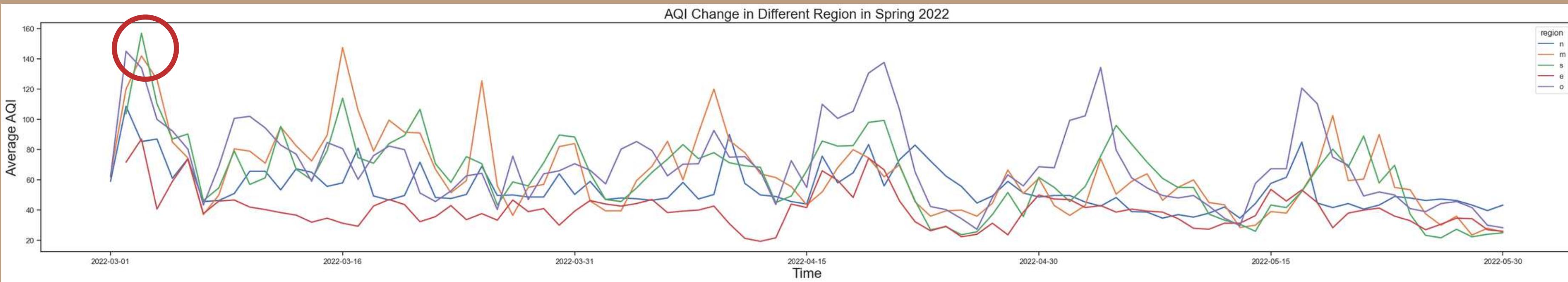
各地區2022 每日AQI值 折線圖



- 各區域**夏季**AQI普遍低於**冬季**
- 南部在**冬季**及**春季**時，AQI指數飆升並超越**北部**及**中部**

資料分析及結果

各地區2022春季 每日AQI值 折線圖

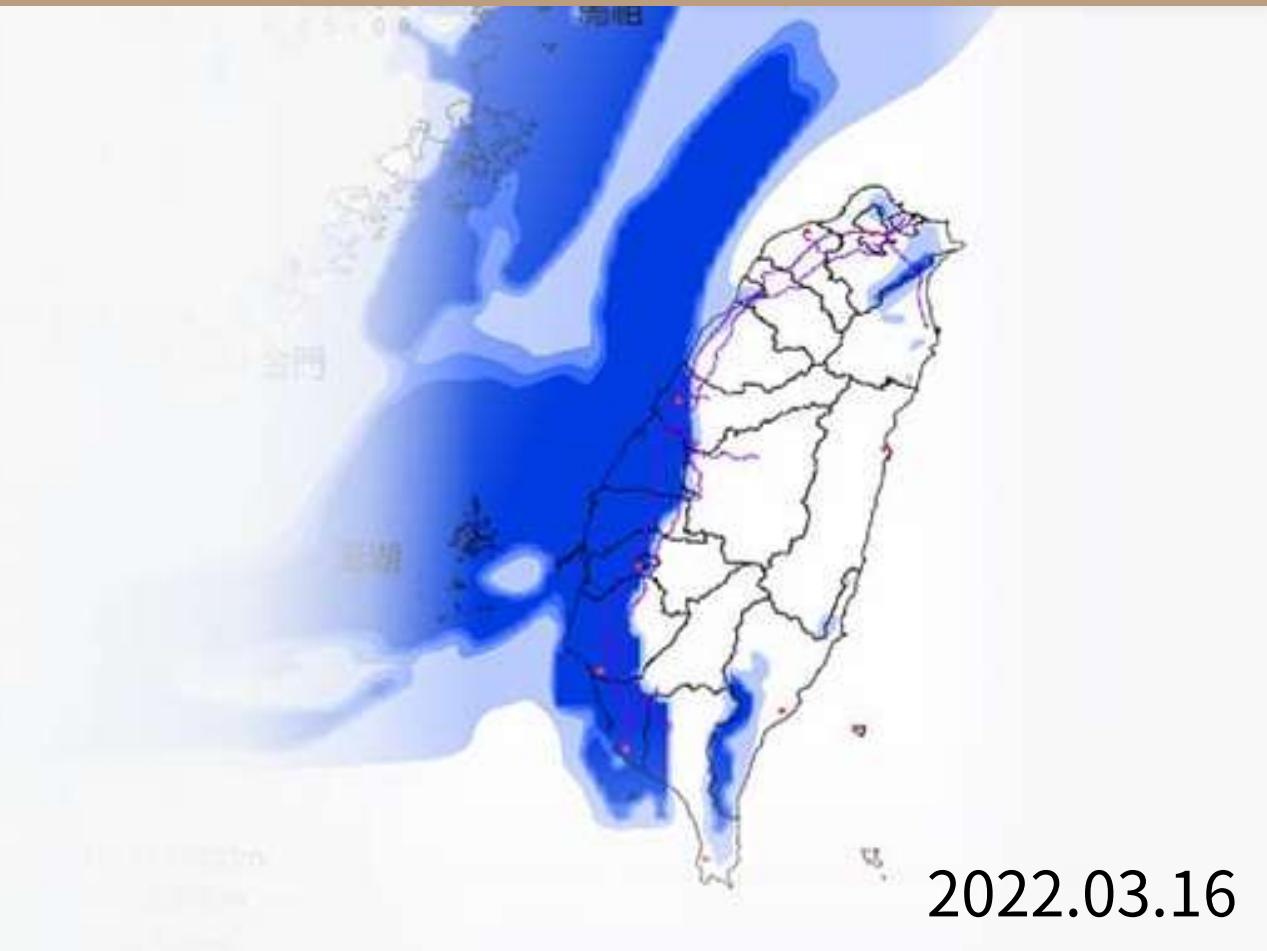
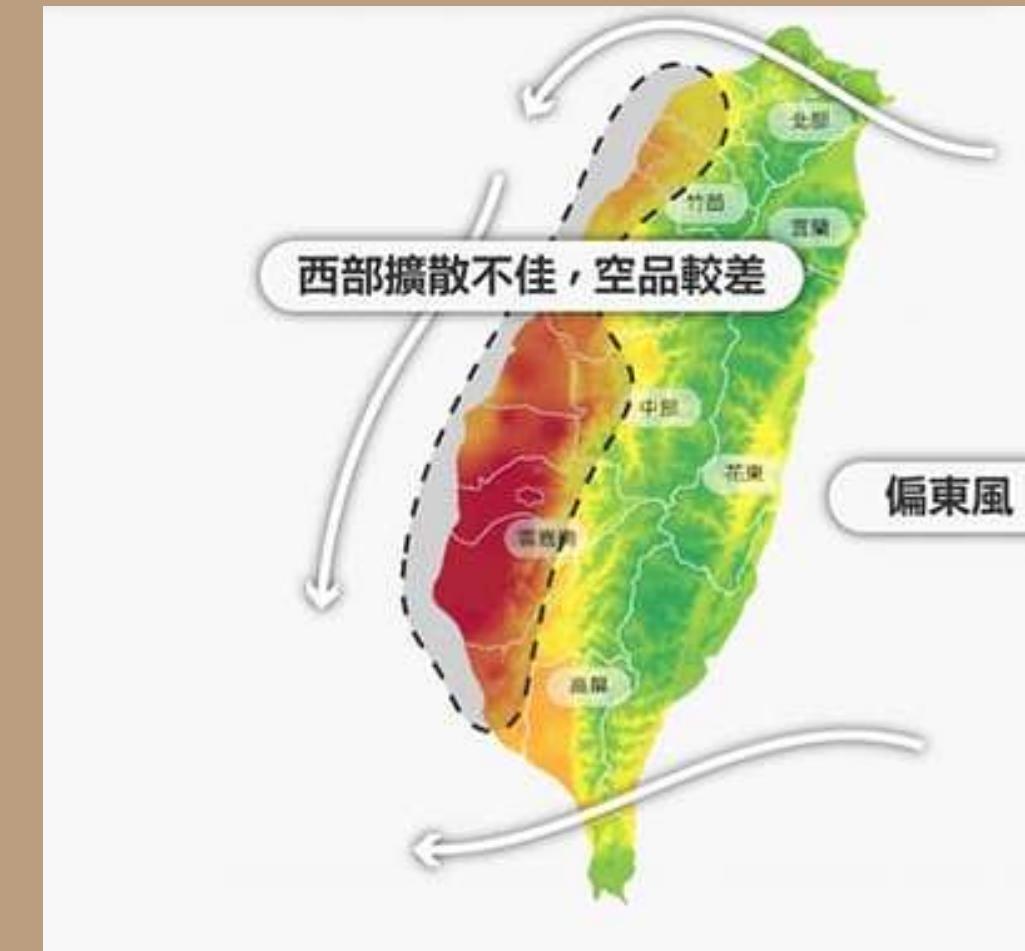
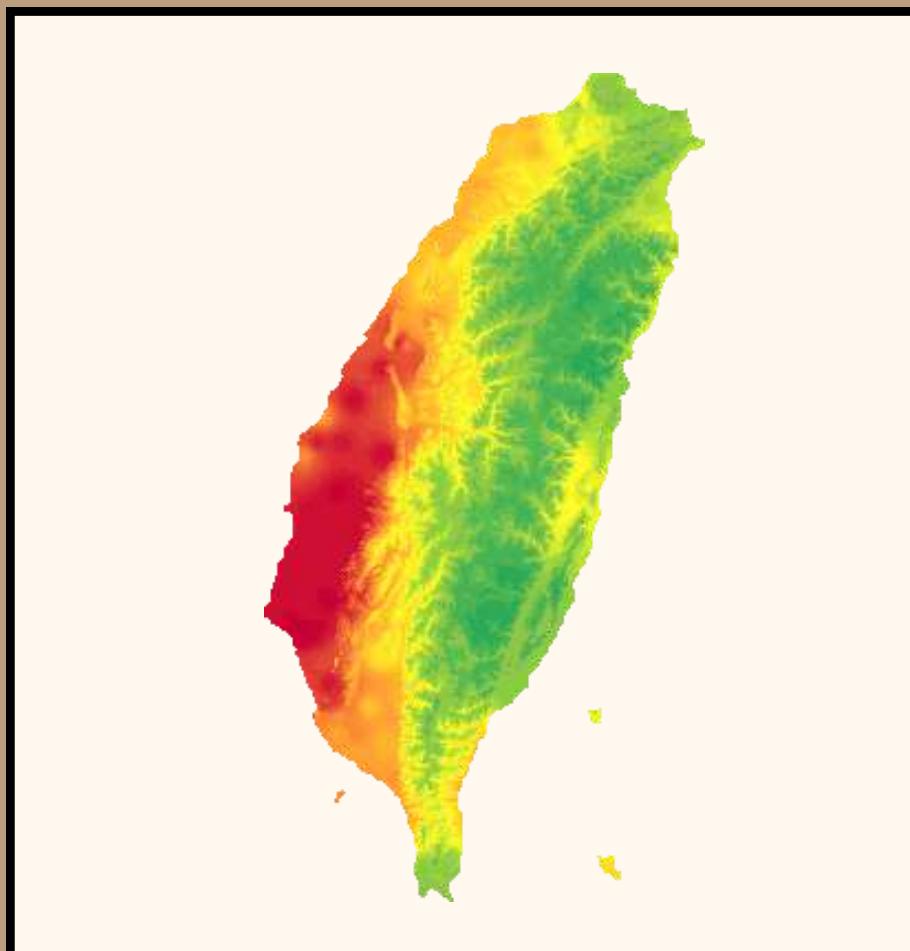
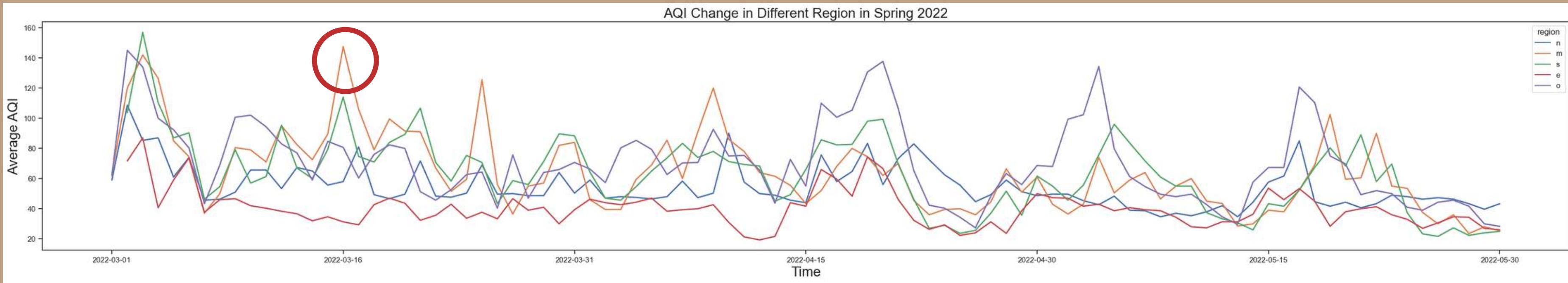


2022/03/04 22:40

〔記者楊綿傑／台北報導〕台灣自本週三起受東北季風挾帶一波的境外污染移入影響，從北部及宜蘭地區開始，至今天已大幅影響中南部地區，加上今日大環境風場轉為偏東風，天氣穩定，西半部擴散條件不佳，致使境外與本地產生污染物難以消散，預計要到明日稍晚才會逐漸改善。而環保署提早從本週二起就透過空氣品質預報資訊，提前協調台電中火、興達、協和及六輕麥電等電廠進行降載減排，各地方政府也迅速進行各種減排應變及宣導健康防護，以減輕空品惡化所造成影響。

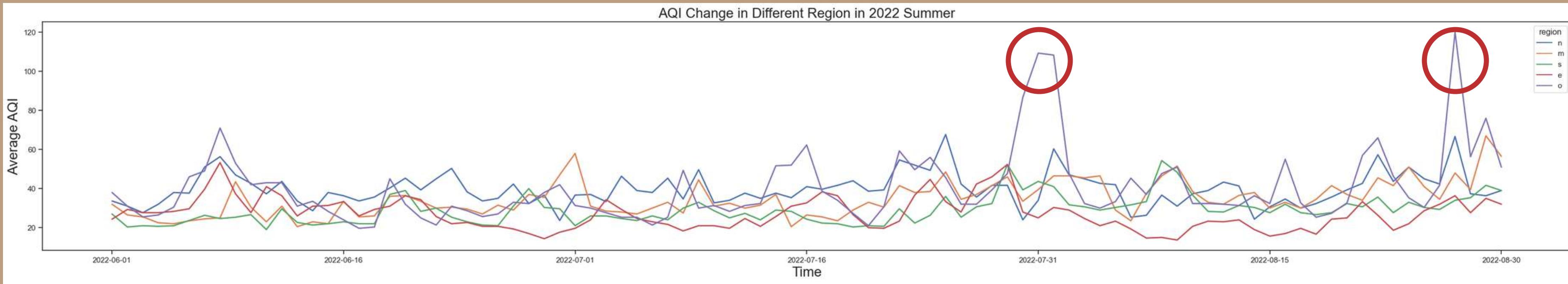
資料分析及結果

各地區2022春季 每日AQI值 折線圖



資料分析及結果

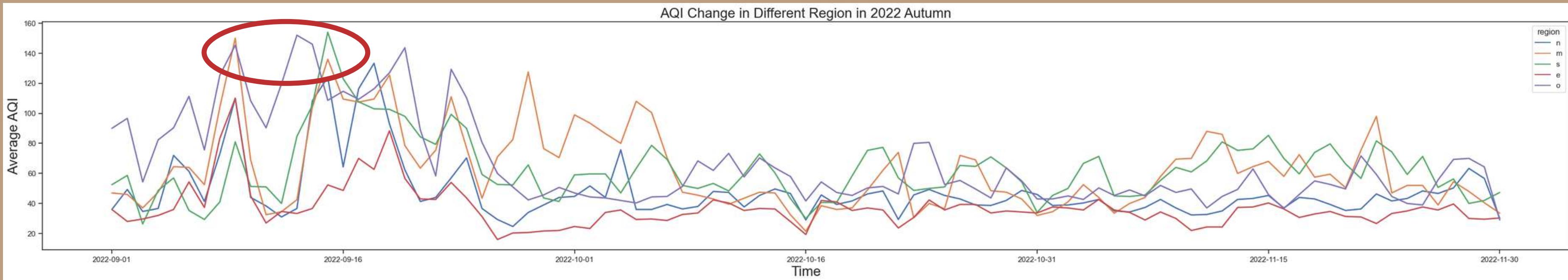
各地區2022夏季 每日AQI值 折線圖



離島地區易受大陸地區**境外汙染**影響

資料分析及結果

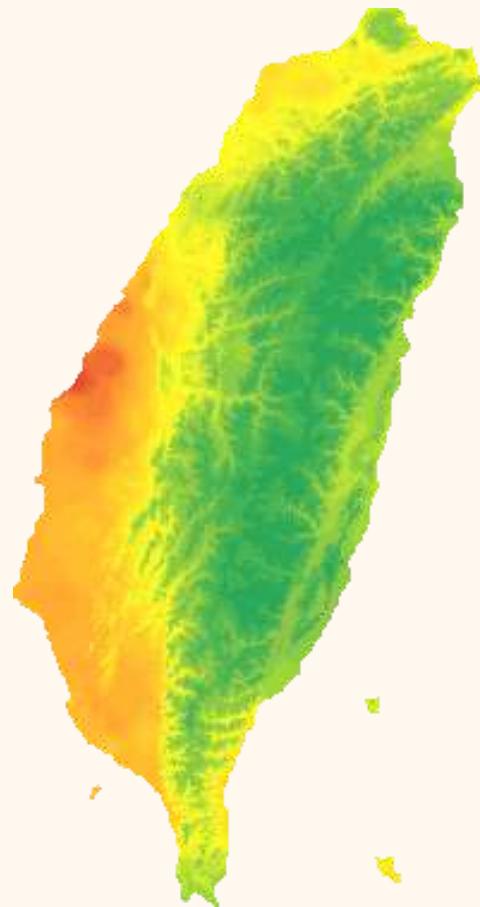
各地區2022秋季 每日AQI值 折線圖



梅花颱風雖然已逐漸遠離臺灣，但其外圍環流卻使得大環境轉為偏西風，同時也帶來一波來自中國的境外污染。金門等離島在12日就開始受到影響，台灣本島西半部地區15日開始受到明顯影響。加上近日各地天氣晴朗溫暖，風速微弱，在光化作用明顯，擴散條件又不佳的情況下，境外與本地的污染物難以消散。

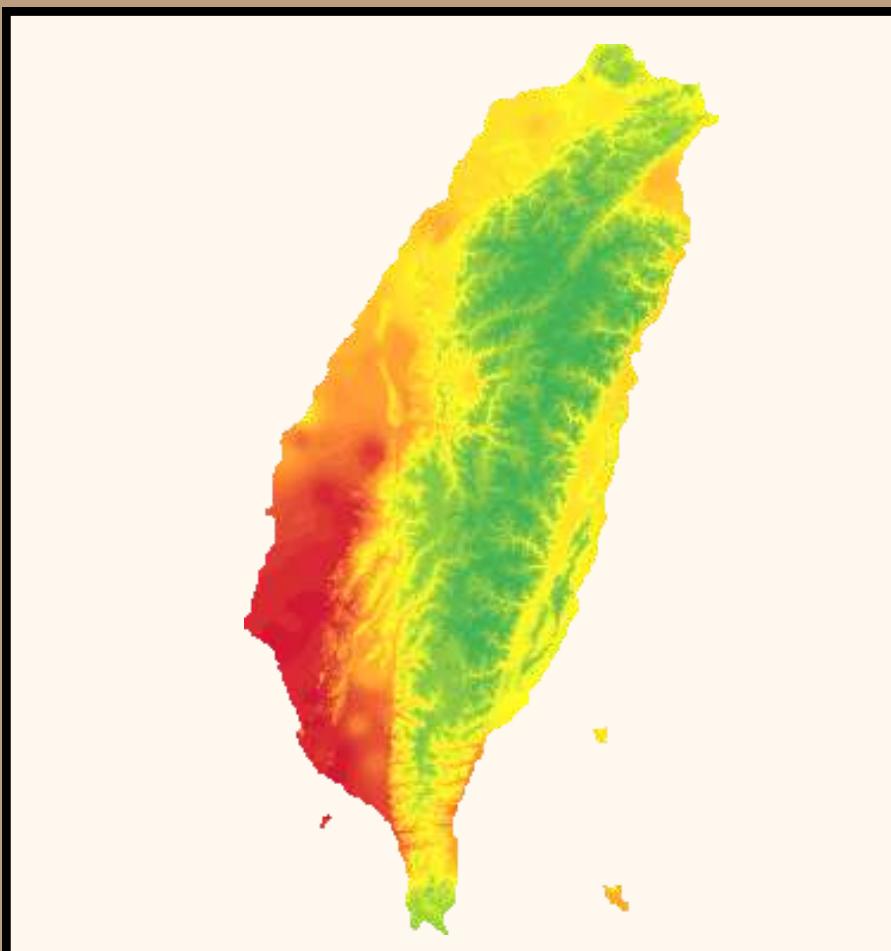
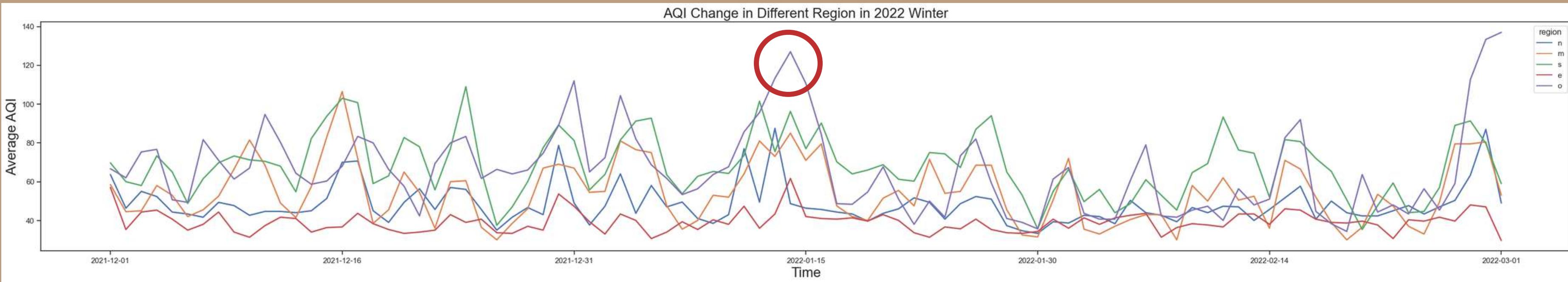
另外週末北風增強後可能又有一波境外污染到來，預估要到下週二以後空氣品質才會有明顯改善。環保署已透過空氣品質預報分析從9月14日起提前協調台電中火、興達、協和及六輕麥電等電廠進行降載減排，各地方政府也迅速進行各種減排因應及宣導健康防護，以減輕空品惡化所造成影響。

2022.09.13



資料分析及結果

各地區2022冬季 每日AQI值 折線圖

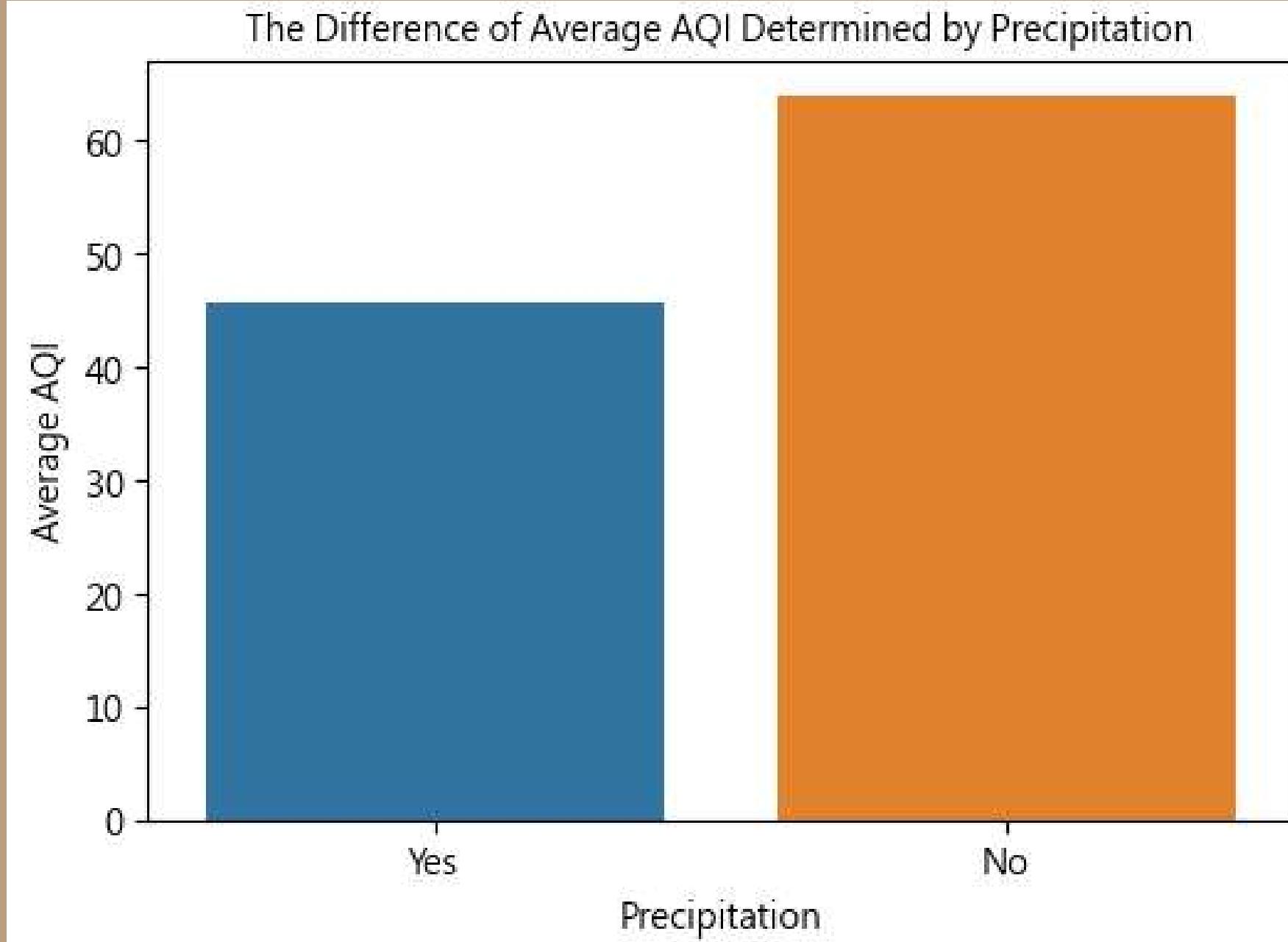


受強烈大陸冷氣團影響，環保署今（13日）表示，環境風場為東北風，中部近山區及雲嘉南以南位於下風處，擴散條件較差，污染物有累積情形，中部近山區及雲嘉南零星地區達橘色警示等級，高屏空品區及馬祖地區為「橘色提醒」等級；金門地區受大陸東南沿海污染物影響達「紅色警示」等級。

環保署指出，依監測資料及氣象局資料顯示，今受強烈大陸冷氣團影響，竹苗以北位於迎風面，擴散條件良好；中部近山區及雲嘉南以南地區位於下風處，易有污染物累積；傍晚起受境外污染物移入影響，北部地區空氣品質稍轉差，影響程度視上游累積程度和降雨情形有所變化。

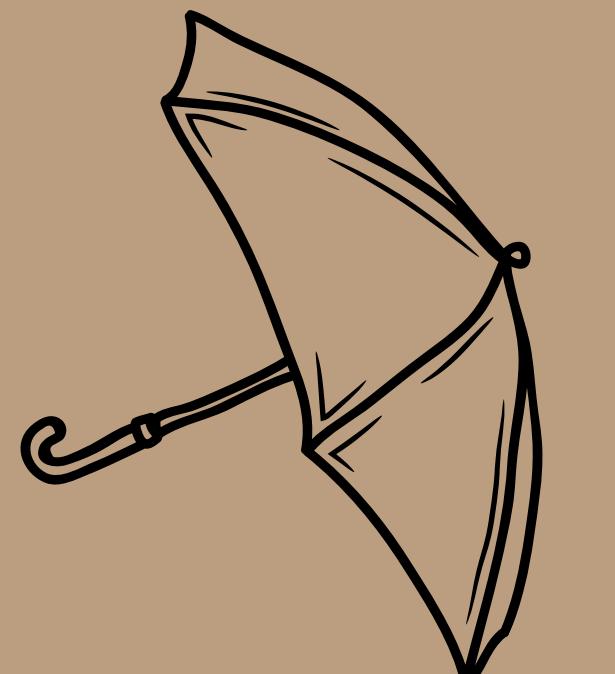
2022.01.15

資料分析及結果



- 有降雨的時候，AQI值較沒降雨時**低**

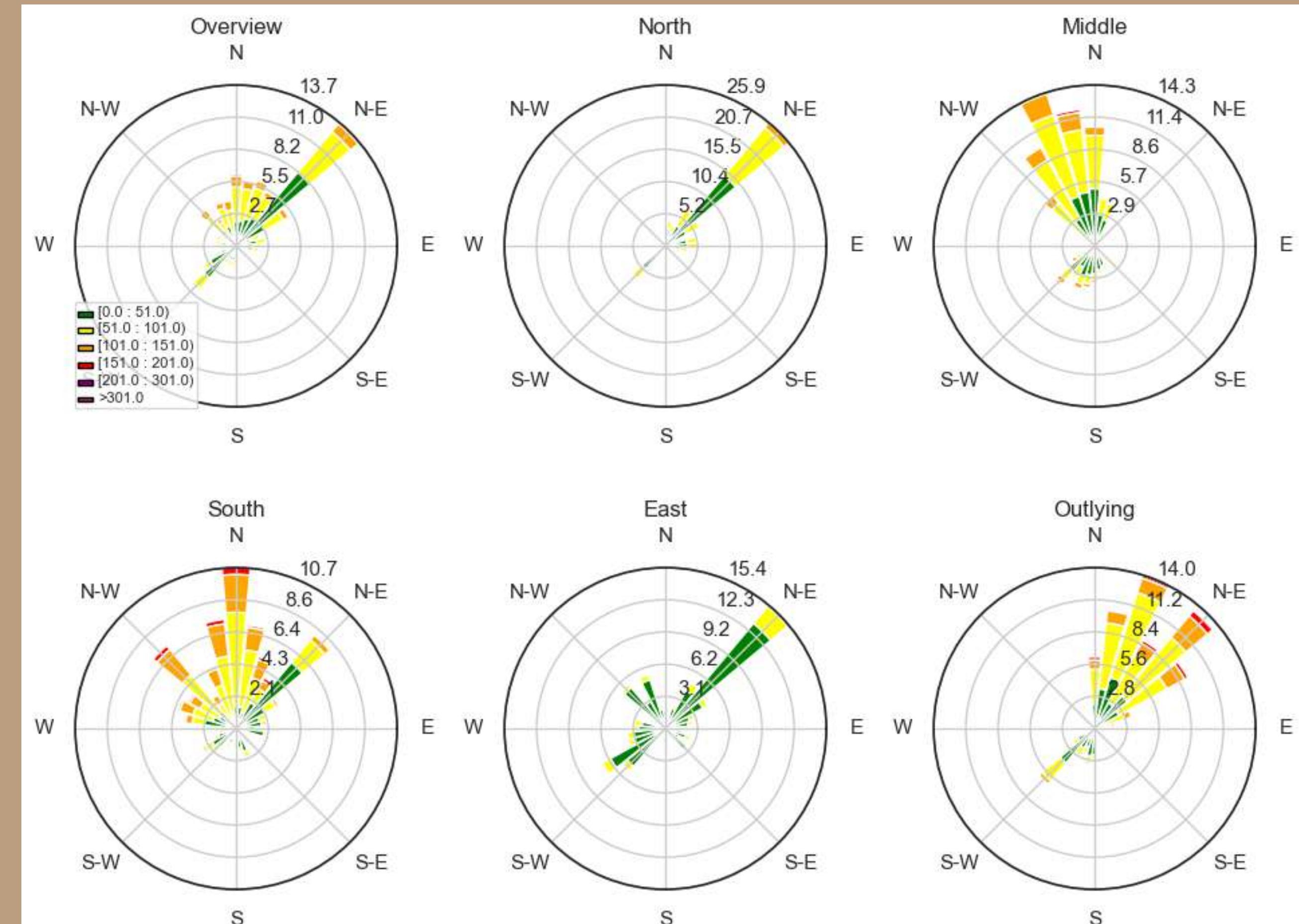
降雨對於AQI的影響 長條圖



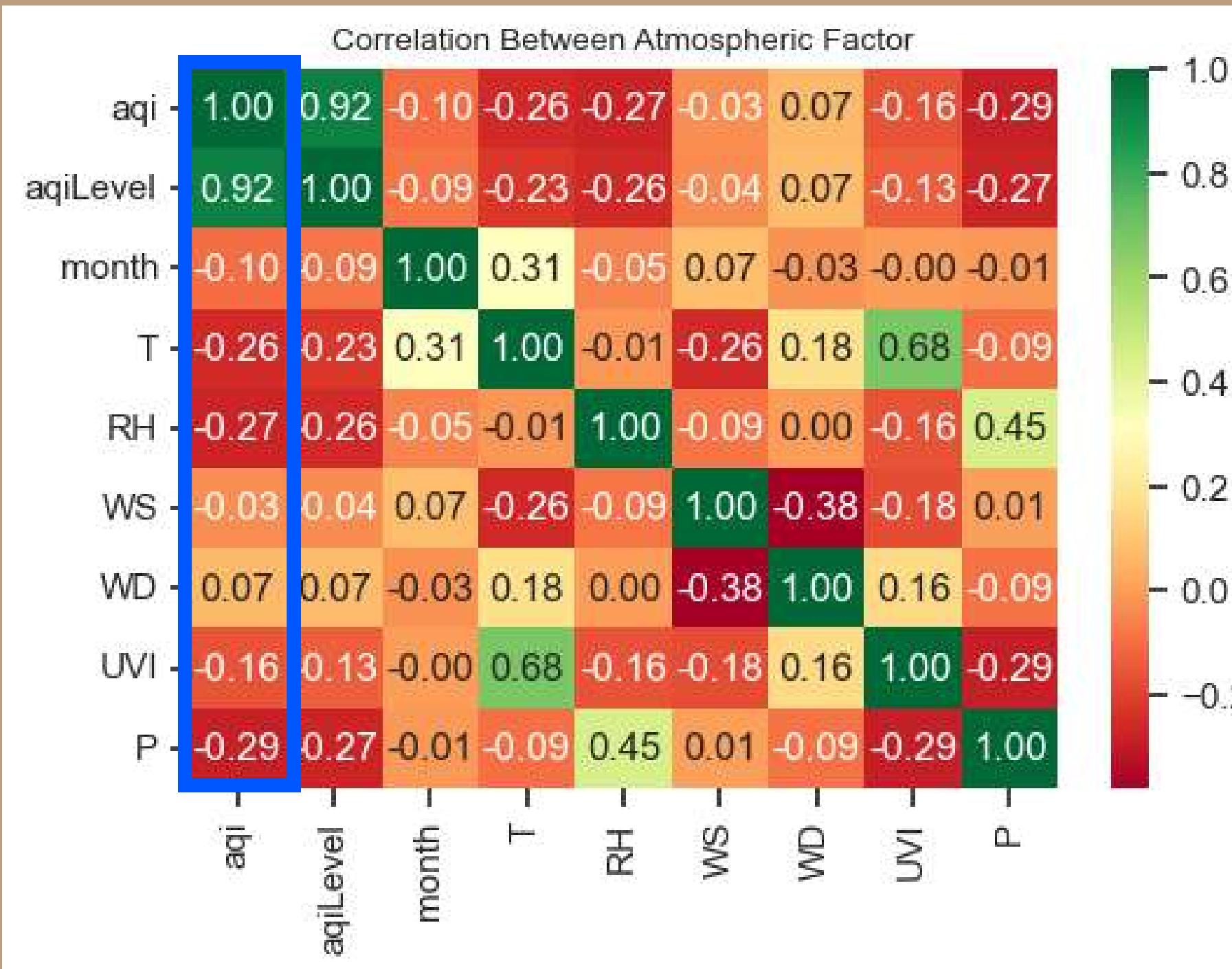
資料分析 及結果

- 全體而言，**東北風**為主要風向，AQI分布平均
- 北部主要以**東北風**為主，空氣品質較**穩定**
- 中部及南部以**北風**為主，空氣品質**黃色及橘色**等級頻率較高
- 離島地區以**東北風**為主，空氣品質**黃色及橘色**等級頻率較高

各地區風向及AQI分布 風花圖

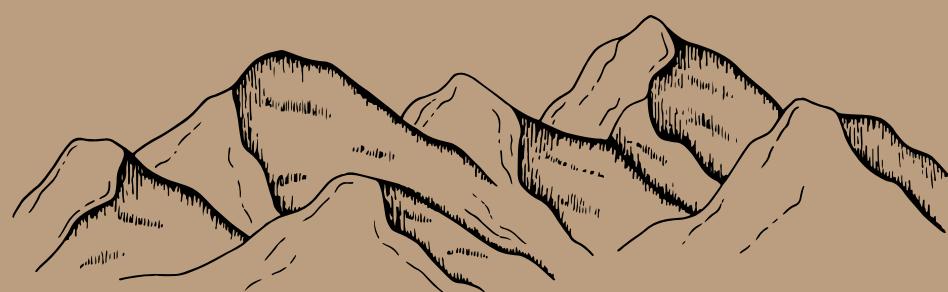


資料分析及結果



- 各項因子對於AQI值的**相關性較低**，
代表**無絕對相關性**

各大氣因子對於AQI值相關性 矩形熱力圖

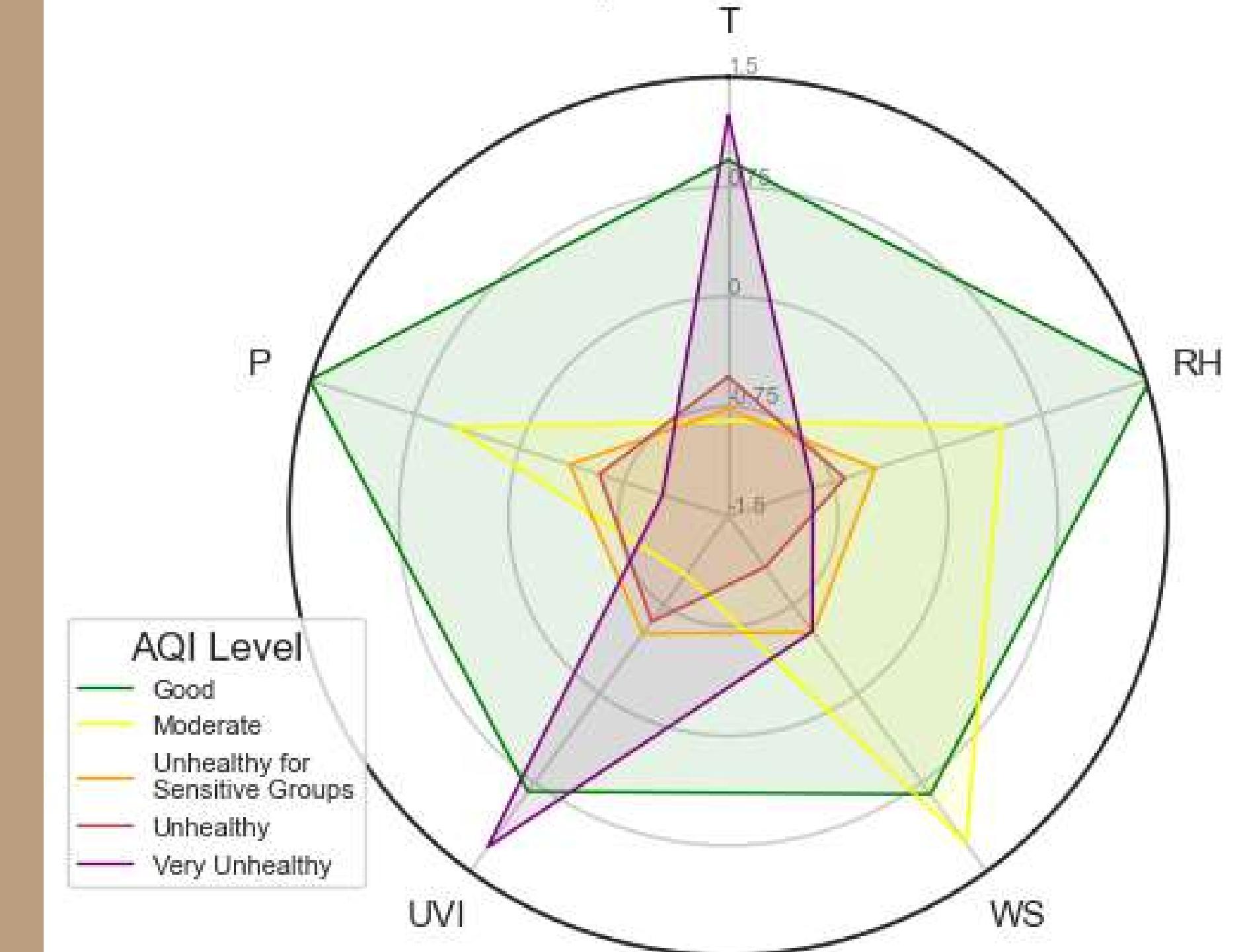


資料分析及結果

各大氣因子對不同AQI等級的影響 雷達圖

- 是否降雨及相對溼度對於空氣品質指標有一定關係的影響
- 降雨會使空氣品質較**佳**，而沒下雨則空氣品質較**差**
- 相對溼度越**高**空氣品質較**佳**，相對溼度越**低**空氣品質較**差**

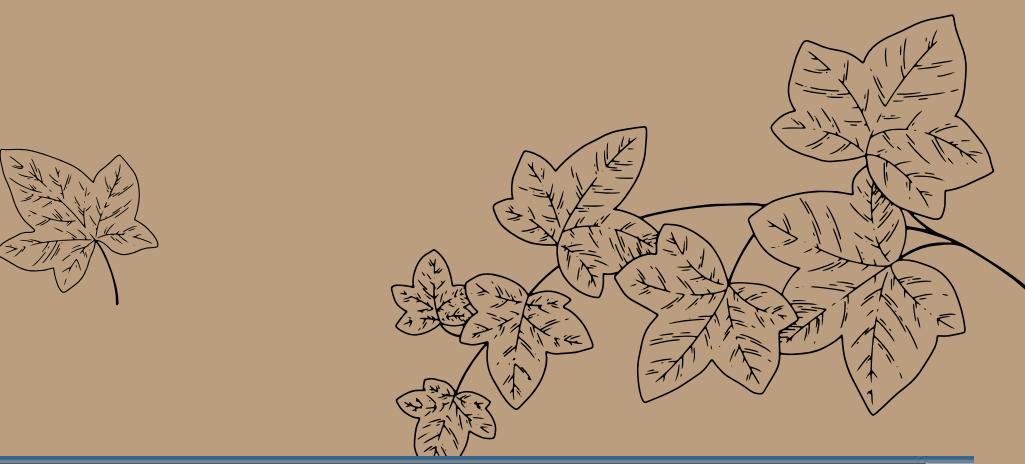
The Influence of Atmospheric Factors on Different AQI Level



預測模組



連接氣象局API取得資料



Swagger
Supported by SMARTBEAR

Select a definition

openAPI

中央氣象局開放資料平臺之資料擷取API 1.0.0

[Base URL: opendata.cwb.gov.tw/api]
[/apidoc/v1](#)

提供目前開放資料之擷取API

Schemes

HTTPS

預報

GET /v1/rest/datastore/F-C0032-001 一般天氣預報-今明 36 小時天氣預報

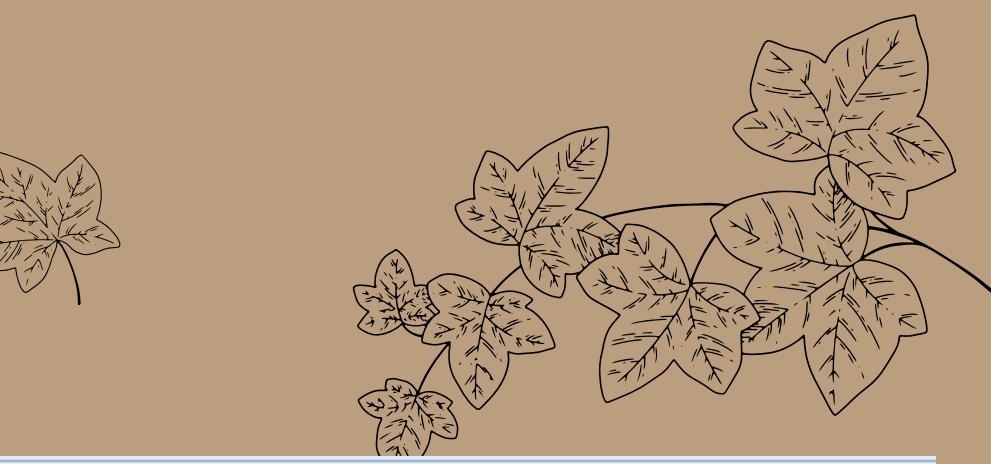
GET /v1/rest/datastore/F-D0047-001 鄉鎮天氣預報-宜蘭縣未來2天天氣預報

GET /v1/rest/datastore/F-D0047-003 鄉鎮天氣預報-宜蘭縣未來1週天氣預報

GET /v1/rest/datastore/F-D0047-005 鄉鎮天氣預報-桃園市未來2天天氣預報

GET /v1/rest/datastore/F-D0047-007 鄉鎮天氣預報-桃園市未來1週天氣預報

連接氣象局API取得資料



GET

/v1/rest/datastore/F-D0047-091 鄉鎮天氣預報-臺灣未來1週天氣預報

^

臺灣各鄉鎮市區預報資料-臺灣各鄉鎮市區未來1週天氣預報

Parameters

Try it out

Name	Description
Authorization * required	氣象開放資料平台會員授權碼
string (query)	<input type="text" value="Authorization"/>
limit	限制最多回傳的資料，預設為回傳全部筆數
number(\$int) (query)	<input type="text" value="limit"/>
offset	指定從第幾筆後開始回傳，預設為第 0 筆開始回傳
number(\$int) (query)	<input type="text" value="offset"/>
format	回傳次級JSON或二進位JSON

連接氣象局API取得資料



format

string

(query)

回傳資料格式，預設為 json 格式

Available values : JSON, XML

locationName

array[string]

(query)

臺灣各縣市

Available values : 宜蘭縣, 花蓮縣, 臺東縣, 澎湖縣, 金門縣, 連江縣, 臺北市, 新北市, 桃園市, 臺中市, 臺南市, 高雄市, 基隆市, 新竹縣, 新竹市, 苗栗縣, 彰化縣, 南投縣, 雲林縣, 嘉義縣, 嘉義市, 屏東縣

- 宜蘭縣
- 花蓮縣
- 臺東縣

elementName

array[string]

(query)

天氣預報因子，預設為全部回傳

Available values : MinCI, MaxAT, MaxCI, MinT, UVI, MinAT, MaxT, WS, WD, Td, PoP12h, T, RH, Wx, WeatherDescription

- MinCI
- MaxAT
- MaxCI

sort

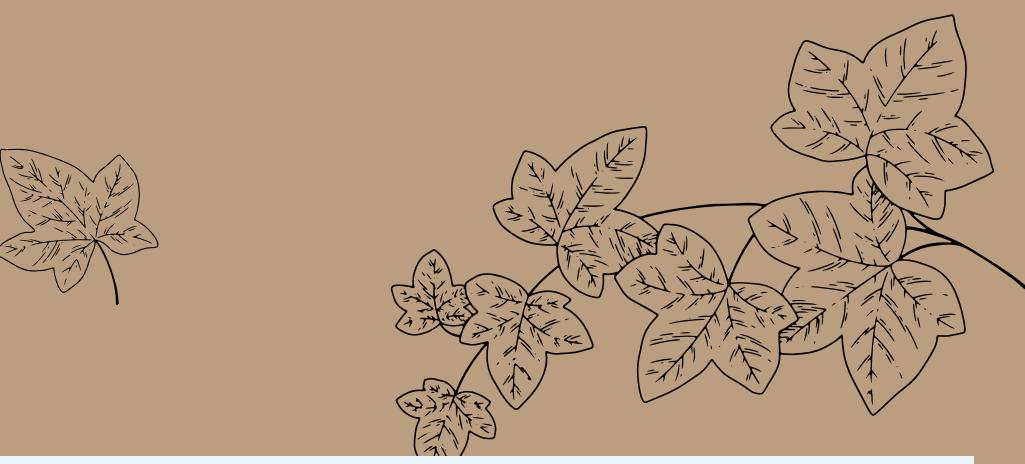
string

(query)

同時對 「startTime」 , 「endTime」 , 「dateTime」 做升冪排序，預設不排序

Available values : time

連接氣象局API取得資料



startTime

array[string]
(query)

時間因子，格式為「yyyy-MM-ddThh:mm:ss」，預設為全部回傳

dataTime

array[string]
(query)

時間因子，格式為「yyyy-MM-ddThh:mm:ss」，預設為全部回傳

timeFrom

string(\$date-time)
(query)

時間區段，篩選需要之時間區段，時間從「timeFrom」開始篩選，直到內容之最後時間，並可與參數「timeTo」合併使用，格式為「yyyy-MM-ddThh:mm:ss」，若使用參數「startTime」或「dataTime」，則參數「timeFrom」的篩選資料則會失效，預設為全部回傳

timeFrom

timeTo

string(\$date-time)
(query)

時間區段，篩選需要之時間區段，時間從內容之最初時間開始篩選，直到 timeTo，並可與參數「timeFrom」合併使用，格式為「yyyy-MM-ddThh:mm:ss」，若使用「startTime」或「dataTime」，則參數「timeTo」的篩選資料則會失效，預設為全部回傳

timeTo

Responses

Response content type ▼

Code

Description

200

OK

連接氣象局API取得資料



```
1 #HTTP請求方法
2 import requests
3 def fetch(location, element):
4     url = 'https://opendata.cwb.gov.tw/api/v1/rest/datastore/F-D0047-091'
5     api_key = ' 馬賽克 '
6     params = {
7         'Authorization': api_key,
8         'locationName': location,
9         'elementName': element
10    }
11    response = requests.get(url, params = params)
12    return get_element_values(response.json())
```

def fetch(location, element) :

定義一個抓取API的方法，將氣象局API中的資料進行抓取，依據氣象局API網站，將需要欄位進行設定：

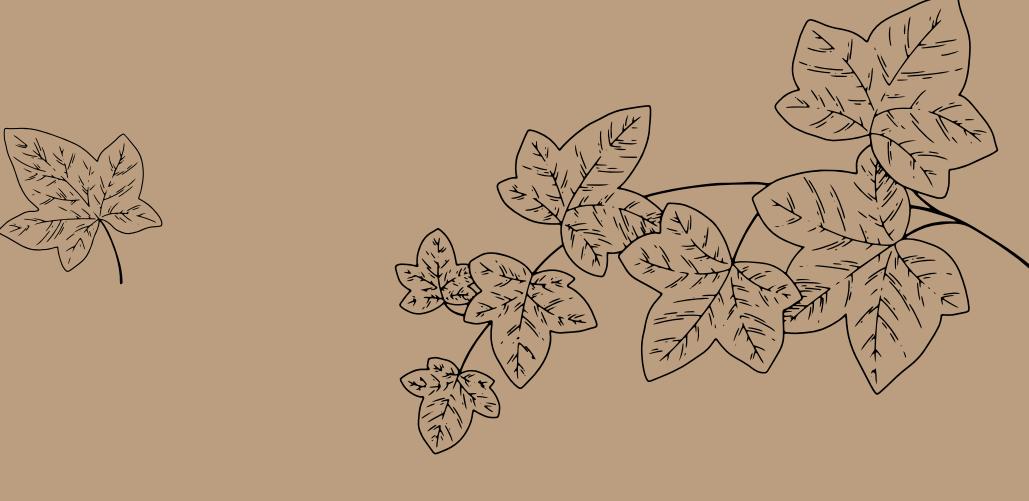
Authorization(必要) 輸入 api_key

locationName 輸入 location

elementName 輸入 element

得到資料後將response轉成.json再回傳給get_element_values()

連接氣象局API取得資料



```
13 def get_element_values(json):
14     json
15     res = json['records']['locations'][0]['location'][0]['weatherElement'][0]['time'][0:4]
16     values = []
17     for i in range(len(res)):
18         values.append(res[i]['startTime'] + '~' + res[i]['endTime'])
19         values.append(res[i]['elementValue'][0]['value'])
20     values.insert(0, json['records']['locations'][0]['location'][0]['weatherElement'][0]['elementName'])
21     return values
```

連接氣象局API取得資料



```
{'success': 'true',
'result': {'resource_id': 'F-D0047-091',
'fields': [{'id': 'contentDescription', 'type': 'String'},
{'id': 'datasetDescription', 'type': 'String'},
{'id': 'locationsName', 'type': 'String'},
{'id': 'dataid', 'type': 'String'},
{'id': 'locationName', 'type': 'String'},
{'id': 'geocode', 'type': 'Double'},
{'id': 'lat', 'type': 'Double'},
{'id': 'lon', 'type': 'Double'},
{'id': 'elementName', 'type': 'String'},
{'id': 'description', 'type': 'String'},
{'id': 'startTime', 'type': 'Timestamp'},
{'id': 'endTime', 'type': 'Timestamp'},
{'id': 'dateTime', 'type': 'Timestamp'},
{'id': 'value', 'type': 'String'},
{'id': 'measures', 'type': 'String']}],
'records': {'locations': [{"datasetDescription": '臺灣各縣市鄉鎮未來1週逐12小時天氣預報',
'locationsName': '台灣',
'dataid': 'D0047-091',
'location': [{"locationName": '新竹縣',
'geocode': '10004000',
'lat': '24.841245',
'lon': '120.995698',
'weatherElement': [{"elementName": 'PoP12h',
'description': '12小時降雨機率'},
{'time': [{"startTime": '2022-12-25 18:00:00',
'endTime': '2022-12-26 06:00:00',
'elementValue': [{"value": '0', 'measures': '百分比"}]},
{'startTime': '2022-12-26 06:00:00',
'endTime': '2022-12-26 18:00:00',
'elementValue': [{"value": '0', 'measures': '百分比'}]}]}]}]}
```

```
{'startTime': '2022-12-26 06:00:00',
'endTime': '2022-12-26 18:00:00',
'elementValue': [{"value": '0', 'measures': '百分比'}]}},
{'startTime': '2022-12-26 18:00:00',
'endTime': '2022-12-27 06:00:00',
'elementValue': [{"value": '0', 'measures': '百分比'}]}},
{'startTime': '2022-12-27 06:00:00',
'endTime': '2022-12-27 18:00:00',
'elementValue': [{"value": '0', 'measures': '百分比'}]}},
{'startTime': '2022-12-27 18:00:00',
'endTime': '2022-12-28 06:00:00',
'elementValue': [{"value": '0', 'measures': '百分比'}]}},
{'startTime': '2022-12-28 06:00:00',
'endTime': '2022-12-28 18:00:00',
'elementValue': [{"value": '10', 'measures': '百分比'}]}},
{'startTime': '2022-12-28 18:00:00',
'endTime': '2022-12-29 06:00:00',
'elementValue': [{"value": '0', 'measures': '百分比'}]}},
{'startTime': '2022-12-29 06:00:00',
'endTime': '2022-12-29 18:00:00',
'elementValue': [{"value": '0', 'measures': '百分比'}]}},
{'startTime': '2022-12-29 18:00:00',
'endTime': '2022-12-30 06:00:00',
'elementValue': [{"value": '0', 'measures': '百分比'}]}},
{'startTime': '2022-12-30 06:00:00',
'endTime': '2022-12-30 18:00:00',
'elementValue': [{"value": '0', 'measures': '百分比'}]}},
{'startTime': '2022-12-30 18:00:00',
'endTime': '2022-12-31 06:00:00',
'elementValue': [{"value": '0', 'measures': '百分比'}]}},
{'startTime': '2022-12-31 06:00:00',
```

連接氣象局API取得資料



```
1 locations = ['宜蘭縣', '花蓮縣', '臺東縣', '澎湖縣', '金門縣', '連江縣', '臺北市', '新北市', '桃園市', '臺中市',
2               '臺南市', '高雄市', '基隆市', '新竹縣', '新竹市', '苗栗縣', '彰化縣', '南投縣', '雲林縣', '嘉義縣',
3               '嘉義市', '屏東縣']
4 elements = ['MinT', 'MaxT', 'RH', 'WS', 'WD', 'UVI', 'PoP12h']
5 forecast = []
6 for location in locations:
7     res = [location]
8     for element in elements:
9         tmp = fetch(location, element)
10    res.append(tmp)
11    forecast.append(res)
12 forecast[0]
```

for location in locations:

依照locations list中的地區進行資料抓取

for element in elements:

依照elements list進行元素抓取

氣象局API數據內容



```
[ '宜蘭縣',
  ['MinT',
   '2022-12-20 18:00:00~2022-12-21 06:00:00',
   '14',
   '2022-12-21 06:00:00~2022-12-21 18:00:00',
   '14',
   '2022-12-21 18:00:00~2022-12-22 06:00:00',
   '11',
   '2022-12-22 06:00:00~2022-12-22 18:00:00',
   '11'],
  ['MaxT',
   '2022-12-20 18:00:00~2022-12-21 06:00:00',
   '18',
   '2022-12-21 06:00:00~2022-12-21 18:00:00',
   '20',
   '2022-12-21 18:00:00~2022-12-22 06:00:00',
   '15',
   '2022-12-22 06:00:00~2022-12-22 18:00:00',
   '18'],
  ['RH',
   '2022-12-20 18:00:00~2022-12-21 06:00:00',
   '97',
   '2022-12-21 06:00:00~2022-12-21 18:00:00',
   '87',
   '2022-12-21 18:00:00~2022-12-22 06:00:00',
   '95',
   '2022-12-22 06:00:00~2022-12-22 18:00:00',
   '85']]
```

```
[ 'WS',
  '2022-12-20 18:00:00~2022-12-21 06:00:00',
  '2',
  '2022-12-21 06:00:00~2022-12-21 18:00:00',
  '2',
  '2022-12-21 18:00:00~2022-12-22 06:00:00',
  '2',
  '2022-12-22 06:00:00~2022-12-22 18:00:00',
  '2'],
  ['WD',
   '2022-12-20 18:00:00~2022-12-21 06:00:00',
   '偏西風',
   '2022-12-21 06:00:00~2022-12-21 18:00:00',
   '偏北風',
   '2022-12-21 18:00:00~2022-12-22 06:00:00',
   '西北風',
   '2022-12-22 06:00:00~2022-12-22 18:00:00',
   '偏北風'],
  ['UVI',
   '2022-12-21 06:00:00~2022-12-21 18:00:00',
   '2',
   '2022-12-22 06:00:00~2022-12-22 18:00:00',
   '4',
   '2022-12-23 06:00:00~2022-12-23 18:00:00',
   '2',
   '2022-12-24 06:00:00~2022-12-24 18:00:00',
   '3']]
```

```
[ 'PoP12h',
  '2022-12-20 18:00:00~2022-12-21 06:00:00',
  '20',
  '2022-12-21 06:00:00~2022-12-21 18:00:00',
  '30',
  '2022-12-21 18:00:00~2022-12-22 06:00:00',
  '10',
  '2022-12-22 06:00:00~2022-12-22 18:00:00',
  '10']]
```

整理回應數據



```
1 from datetime import datetime  
2  
3 def get_month(f):  
4     res = ["month"]  
5     for i in range(1, 9, 2):  
6         time = f[1][i].split('~')[0]  
7         time = datetime.strptime(time, '%Y-%m-%d %H:%M:%S')  
8         res.append(time.month)  
9     return res
```

def get_month(f) :
取得預報時段的月份

def partition(sitename) :
將測站名稱進行區域分類

```
51 def partition(sitename):  
52     if sitename == '臺北市' or sitename == '新北市' or sitename == '基隆市' or sitename == '桃園'  
53         return 'n'  
54     if sitename == '臺中市' or sitename == '彰化縣' or sitename == '南投縣' or sitename == '雲林'  
55         return 'm'  
56     if sitename == '嘉義縣' or sitename == '嘉義市' or sitename == '臺南市' or sitename == '高雄'  
57         return 's'  
58     if sitename == '宜蘭縣' or sitename == '花蓮縣' or sitename == '臺東縣':  
59         return 'e'  
60     if sitename == '澎湖縣' or sitename == '金門縣' or sitename == '連江縣':  
61         return 'o'
```

整理回應數據

```
11 def WD_to_degree(direction):
12     if '偏北' in direction:
13         return 360
14     if '東北' in direction:
15         return 45
16     if '偏東' in direction:
17         return 90
18     if '東南' in direction:
19         return 135
20     if '偏南' in direction:
21         return 180
22     if '西南' in direction:
23         return 225
24     if '偏西' in direction:
25         return 270
26     if '西北' in direction:
27         return 315
```

```
29 def check_WS(speed):
30     res = ''
31     for s in speed:
32         if(s.isdigit()):
33             res += s
34     return int(res)
```

def WD_to_degree(direction) :

將方位進行定位辨識，並進行度數分級

def check_WS(speed) :

確保speed欄位中無符號之值，並轉成int型態



整理回應數據

```
def correct_UVI(f):
    UVI = ['UVI']
    time = f[5][1].split('~')[1]
    time = datetime.strptime(time, '%Y-%m-%d %H:%M:%S')
    if time.hour == 6:
        for i in range(2, 5, 2):
            UVI.append(0)
            UVI.append(int(f[6][i]))
    else:
        for i in range(2, 5, 2):
            UVI.append(int(f[6][i]))
            UVI.append(0)
    #print(UVI)
    return UVI
```

def correct_UVI(f) :

因為UVI一天只有一筆資料，但我們是取四個時段，就代表只有兩天的UVI資料，所以要判斷第一筆資料是早上還是晚上，若是晚上則UVI就賦值0

```
[ 'WD',
  '2022-12-20 18:00:00~2022-12-21 06:00:00',
  '偏西風',
  '2022-12-21 06:00:00~2022-12-21 18:00:00',
  '偏北風',
  '2022-12-21 18:00:00~2022-12-22 06:00:00',
  '西北風',
  '2022-12-22 06:00:00~2022-12-22 18:00:00',
  '偏北風'],
[ 'UVI',
  '2022-12-21 06:00:00~2022-12-21 18:00:00',
  '2',
  '2022-12-22 06:00:00~2022-12-22 18:00:00',
  '4',
  '2022-12-23 06:00:00~2022-12-23 18:00:00',
  '2',
  '2022-12-24 06:00:00~2022-12-24 18:00:00',
  '3'],
```

整理回應數據



```
1 for f in forecast:
2     T = ["T"]
3     for i in range(2, 10, 2):
4         avg = (int(f[1][i]) + int(f[2][i])) / 2
5         T.append(avg)
6         f[3][i] = int(f[3][i])
7         f[4][i] = check_WS(f[4][i])
8         f[5][i] = WD_to_degree(f[5][i])
9         f[7][i] = int(f[7][i]) / 100
10        f.insert(1, get_month(f))
11        f.insert(4, T)
12        f.insert(8, correct_UVI(f))
13        f.insert(11, partition(f[0]))
14        del f[2:4] #刪除minT MaxT
15        del f[7:8] #刪除舊UVI
16 forecast
```

for f in forecast:

利用剛剛定義好的函式對資料進行整理

整理回應數據

```
[['宜蘭縣',
  ['month', 12, 12, 12, 12],
  ['T', 16.0, 15.5, 16.5, 15.5],
  ['RH',
   '2022-12-27 06:00:00~2022-12-27 18:00:00',
   85,
   '2022-12-27 18:00:00~2022-12-28 06:00:00',
   99,
   '2022-12-28 06:00:00~2022-12-28 18:00:00',
   99,
   '2022-12-28 18:00:00~2022-12-29 06:00:00',
   99],
  ['WS',
   '2022-12-27 06:00:00~2022-12-27 18:00:00',
   2,
   '2022-12-27 18:00:00~2022-12-28 06:00:00',
   1,
   '2022-12-28 06:00:00~2022-12-28 18:00:00',
   2,
   '2022-12-28 18:00:00~2022-12-29 06:00:00',
   2],
```

```
[ 'WD',
  '2022-12-27 06:00:00~2022-12-27 18:00:00',
  90,
  '2022-12-27 18:00:00~2022-12-28 06:00:00',
  180,
  '2022-12-28 06:00:00~2022-12-28 18:00:00',
  90,
  '2022-12-28 18:00:00~2022-12-29 06:00:00',
  315],
  ['UVI', 2, 0, 1, 0],
  ['PoP12h',
   '2022-12-27 06:00:00~2022-12-27 18:00:00',
   0.5,
   '2022-12-27 18:00:00~2022-12-28 06:00:00',
   0.6,
   '2022-12-28 06:00:00~2022-12-28 18:00:00',
   0.7,
   '2022-12-28 18:00:00~2022-12-29 06:00:00',
   0.8],
  ['e']]
```

分別取出各時段預報

```
1 def region_to_dummy(region):  
2     if region == 'e':  
3         return [1, 0, 0, 0, 0]  
4     if region == 'm':  
5         return [0, 1, 0, 0, 0]  
6     if region == 'n':  
7         return [0, 0, 1, 0, 0]  
8     if region == 'o':  
9         return [0, 0, 0, 1, 0]  
10    if region == 's':  
11        return [0, 0, 0, 0, 1]
```

def region_to_dummy() :

按照訓練模型將資料進行**1/0化**

分別取出各時段預報



```
13 period1 = []
14 period2 = []
15 period3 = []
16 period4 = []
17 for i in range(1, 5):
18     for f in forecast:
19         tmp = []
20         tmp.append(f[0])
21         tmp.append(f[1][i]) # month
22         tmp.append(f[2][i]) # T
23         for element in range(3, len(f)):
24             if f[element][0] == 'UVI':
25                 tmp.append(f[6][i]) # UVI
26                 continue
27             elif element == 8:
28                 tmp.extend(region_to_dummy(f[8])) # region
29             else:
30                 tmp.append(f[element][2*i]) # 2*i
31         if i == 1:
32             period1.append(tmp)
33         elif i == 2:
34             period2.append(tmp)
35         elif i == 3:
36             period3.append(tmp)
37         else:
38             period4.append(tmp)
```

for i in forecast :
分別取出各時段的氣象資料

```
[['宜蘭縣', 12, 15.0, 67, 2, 360, 360, 0.1, 1, 0, 0, 0, 0],  
 ['花蓮縣', 12, 17.0, 63, 3, 45, 45, 0.1, 1, 0, 0, 0, 0],  
 ['臺東縣', 12, 18.5, 56, 3, 45, 45, 0.1, 1, 0, 0, 0, 0],  
 ['澎湖縣', 12, 16.0, 71, 9, 45, 45, 0.0, 0, 0, 0, 1, 0],  
 ['金門縣', 12, 13.5, 50, 7, 45, 45, 0.0, 0, 0, 0, 1, 0],  
 ['連江縣', 12, 11.0, 60, 6, 45, 45, 0.0, 0, 0, 0, 1, 0],  
 ['臺北市', 12, 14.0, 59, 2, 45, 45, 0.1, 0, 0, 1, 0, 0],  
 ['新北市', 12, 14.5, 59, 2, 45, 45, 0.1, 0, 0, 1, 0, 0],  
 ['桃園市', 12, 13.5, 57, 3, 45, 45, 0.0, 0, 0, 1, 0, 0],  
 ['臺中市', 12, 17.0, 46, 3, 360, 360, 0.0, 0, 1, 0, 0, 0],  
 ['臺南市', 12, 17.5, 52, 3, 315, 315, 0.0, 0, 0, 0, 0, 1],  
 ['高雄市', 12, 19.0, 57, 2, 270, 270, 0.0, 0, 0, 0, 0, 1],  
 ['基隆市', 12, 14.0, 69, 4, 360, 360, 0.1, 0, 0, 1, 0, 0],  
 ['新竹縣', 12, 14.5, 57, 4, 45, 45, 0.0, 0, 0, 1, 0, 0],  
 ['新竹市', 12, 14.0, 54, 5, 45, 45, 0.0, 0, 0, 1, 0, 0],  
 ['苗栗縣', 12, 14.5, 52, 4, 45, 45, 0.0, 0, 0, 1, 0, 0],  
 ['彰化縣', 12, 16.0, 50, 5, 360, 360, 0.0, 0, 1, 0, 0, 0],  
 ['南投縣', 12, 16.5, 45, 3, 315, 315, 0.0, 0, 1, 0, 0, 0],  
 ['雲林縣', 12, 16.5, 46, 4, 315, 315, 0.0, 0, 1, 0, 0, 0],  
 ['嘉義縣', 12, 16.5, 51, 3, 315, 315, 0.0, 0, 0, 0, 0, 1],  
 ['嘉義市', 12, 17.0, 49, 2, 315, 315, 0.0, 0, 0, 0, 0, 1],  
 ['屏東縣', 12, 20.0, 46, 2, 180, 180, 0.0, 0, 0, 0, 0, 1]]
```

分別取出各時段預報

```
40 period1 = pd.DataFrame(period1)
41 print(period1)
42 county_order = list(period1[0])
43 print(county_order)
44 period1 = period1.drop([0], axis = 1)
45 period1 = period1.values.tolist()
46 period2 = pd.DataFrame(period2)
47 period2 = period2.drop([0], axis = 1)
48 period2 = period2.values.tolist()
49 period3 = pd.DataFrame(period3)
50 period3 = period3.drop([0], axis = 1)
51 period3 = period3.values.tolist()
52 period4 = pd.DataFrame(period4)
53 period4 = period4.drop([0], axis = 1)
54 period4 = period4.values.tolist()
55 period1 = sc.fit_transform(period1)
56 period2 = sc.fit_transform(period2)
57 period3 = sc.fit_transform(period3)
58 period4 = sc.fit_transform(period4)
59 period1
```

list(period1[0])

將縣市順序取出並放到country_order

drop([0], axis = 1)

把縣市欄位刪除

.values.tolist()

再將資料轉回成list型態

sc.fit_transform()

標準化

分別取出各時段預報



	0	1	2	3	4	5	6	7	8	9	10	11	12
0	宜蘭縣	12	16.0	97	2	270	0	0.2	1	0	0	0	0
1	花蓮縣	12	18.0	79	2	270	0	0.2	1	0	0	0	0
2	臺東縣	12	18.5	74	2	315	0	0.2	1	0	0	0	0
3	澎湖縣	12	17.0	85	2	360	0	0.2	0	0	0	1	0
4	金門縣	12	15.0	78	4	315	0	0.0	0	0	0	1	0
5	連江縣	12	12.5	78	8	360	0	0.2	0	0	0	1	0
6	臺北市	12	17.0	81	2	90	0	0.6	0	0	1	0	0
7	新北市	12	17.0	87	2	135	0	0.6	0	0	1	0	0
8	桃園市	12	16.0	86	2	180	0	0.6	0	0	1	0	0
9	臺中市	12	17.0	76	2	135	0	0.2	0	1	0	0	0
10	臺南市	12	17.5	84	2	360	0	0.0	0	0	0	0	1
11	高雄市	12	19.5	74	2	360	0	0.0	0	0	0	0	1
12	基隆市	12	18.0	79	4	135	0	0.5	0	0	1	0	0
13	新竹縣	12	15.5	88	0	180	0	0.5	0	0	1	0	0
14	新竹市	12	16.0	85	0	180	0	0.5	0	0	1	0	0
15	苗栗縣	12	14.0	84	1	180	0	0.6	0	0	1	0	0
16	彰化縣	12	16.0	82	2	135	0	0.2	0	1	0	0	0
17	南投縣	12	16.5	81	1	135	0	0.0	0	1	0	0	0
18	雲林縣	12	16.0	81	1	180	0	0.2	0	1	0	0	0
19	嘉義縣	12	16.0	83	1	135	0	0.2	0	0	0	0	1
20	嘉義市	12	16.5	82	1	135	0	0.0	0	0	0	0	1
21	屏東縣	12	19.5	79	1	360	0	0.1	0	0	0	0	1

DataFrame型態

```
array([[ 0.          , -0.37264737,  2.99562845,  0.          ,  0.49800352,
       0.          , -0.29397237,  2.51661148, -0.47140452, -0.68313005,
      -0.39735971, -0.54232614],
       [ 0.          ,  0.88862066, -0.58826541,  0.          ,  0.49800352,
       0.          , -0.29397237,  2.51661148, -0.47140452, -0.68313005,
      -0.39735971, -0.54232614],
       [ 0.          ,  1.20393766, -1.58379148,  0.          ,  0.9743547 ,
       0.          , -0.29397237,  2.51661148, -0.47140452, -0.68313005,
      -0.39735971, -0.54232614],
       [ 0.          ,  0.25798664,  0.60636588,  0.          ,  1.45070589,
       0.          , -0.29397237, -0.39735971, -0.47140452, -0.68313005,
      2.51661148, -0.54232614],
       [ 0.          , -1.00328139, -0.78737062,  1.23176352,  0.9743547 ,
       0.          , -1.21788552, -0.39735971, -0.47140452, -0.68313005,
      2.51661148, -0.54232614],
       [ 0.          , -2.57986642, -0.78737062,  3.69529057,  1.45070589,
       0.          , -0.29397237, -0.39735971, -0.47140452, -0.68313005,
      2.51661148, -0.54232614],
       [ 0.          ,  0.25798664, -0.19005498,  0.          , -1.40740124,
       0.          ,  1.55385394, -0.39735971, -0.47140452,  1.46385011,
      -0.39735971, -0.54232614],
       [ 0.          ,  0.25798664,  1.00457631,  0.          , -0.93105005,
       0.          ,  1.55385394, -0.39735971, -0.47140452,  1.46385011,
     -0.39735971, -0.54232614]]
```

標準化後

分別取出各時段預報



```
1 predict1 = {}
2 predict2 = {}
3 predict3 = {}
4 predict4 = {}
5 count = 0
6 for p in period1:
7     county = county_order[count]
8     predict1[county] = round(list(LR_model.predict([p])).pop())
9     count += 1
10    count = 0
11    for p in period2:
12        county = county_order[count]
13        predict2[county] = round(list(LR_model.predict([p])).pop())
14        count += 1
15    count = 0
16    for p in period3:
17        county = county_order[count]
18        predict3[county] = round(list(LR_model.predict([p])).pop())
19        count += 1
20    count = 0
21    for p in period4:
22        county = county_order[count]
23        predict4[county] = round(list(LR_model.predict([p])).pop())
24        count += 1
25
26 print(predict1)
27 print(predict2)
28 print(predict3)
29 print(predict4)
```

for p in period1:

- 1.各個時段根據模型做預測
- 2.並依照縣市對應，將資料存入字典

```
{'宜蘭縣': 20, '花蓮縣': 37, '臺東縣': 43, '澎湖縣': 69, '金門縣': 84, '連江縣': 77, '臺北市': 42, '新北市': 35, '桃園市': 42, '臺中市': 61, '臺南市': 68, '高雄市': 74, '基隆市': 38, '新竹縣': 50, '新竹市': 52, '苗栗縣': 56, '彰化縣': 56, '南投縣': 64, '雲林縣': 62, '嘉義縣': 68, '嘉義市': 73, '屏東縣': 68}
{'宜蘭縣': 28, '花蓮縣': 35, '臺東縣': 34, '澎湖縣': 54, '金門縣': 82, '連江縣': 71, '臺北市': 63, '新北市': 52, '桃園市': 55, '臺中市': 67, '臺南市': 57, '高雄市': 63, '基隆市': 34, '新竹縣': 51, '新竹市': 55, '苗栗縣': 43, '彰化縣': 59, '南投縣': 65, '雲林縣': 63, '嘉義縣': 69, '嘉義市': 76, '屏東縣': 67}
{'宜蘭縣': 23, '花蓮縣': 20, '臺東縣': 1, '澎湖縣': 51, '金門縣': 80, '連江縣': 96, '臺北市': 63, '新北市': 54, '桃園市': 57, '臺中市': 65, '臺南市': 66, '高雄市': 62, '基隆市': 48, '新竹縣': 53, '新竹市': 56, '苗栗縣': 65, '彰化縣': 65, '南投縣': 56, '雲林縣': 61, '嘉義縣': 72, '嘉義市': 71, '屏東縣': 57}
{'宜蘭縣': 20, '花蓮縣': 16, '臺東縣': 6, '澎湖縣': 50, '金門縣': 73, '連江縣': 112, '臺北市': 54, '新北市': 46, '桃園市': 63, '臺中市': 63, '臺南市': 68, '高雄市': 70, '基隆市': 48, '新竹縣': 51, '新竹市': 56, '苗栗縣': 75, '彰化縣': 58, '南投縣': 58, '雲林縣': 59, '嘉義縣': 67, '嘉義市': 69, '屏東縣': 59}
```

資料視覺化-資料整理



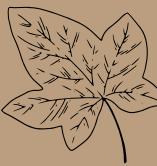
```
1 !pip install folium
1 import folium
1 data = pd.DataFrame()
2 data['countyname'] = ['臺東縣', '宜蘭縣', '臺北市', '雲林縣', '桃園市', '屏東縣', '臺中市', '臺南市', '基隆市', '連江縣', '南投
3 data['countysn'] = ['10014001', '10002001', '63000001', '10009001', '10003001', '10013001', '10006001', '10011001', '100
4 aqi = []
5 for county in data['countyname']:
6     tmp = []
7     tmp.append(predict1[county])
8     tmp.append(predict2[county])
9     tmp.append(predict3[county])
10    tmp.append(predict4[county])
11    aqi.append(tmp)
12 data['aqi'] = aqi
```

資料視覺化-資料整理

```
14 aqi1 = []
15 aqi2 = []
16 aqi3 = []
17 aqi4 = []
18 for county in data['countyname']:
19     aqi1.append(predict1[county])
20 for county in data['countyname']:
21     aqi2.append(predict2[county])
22 for county in data['countyname']:
23     aqi3.append(predict3[county])
24 for county in data['countyname']:
25     aqi4.append(predict4[county])
26 data['aqi1'] = aqi1
27 data['aqi2'] = aqi2
28 data['aqi3'] = aqi3
29 data['aqi4'] = aqi4
30 data
```

index	countyname	countysn	aqi	aqi1 aqi2 aqi3 aqi4			
				aqi1	aqi2	aqi3	aqi4
0	臺東縣	10014001	[43, 34, 1, 6]	43	34	1	6
1	宜蘭縣	10002001	[20, 28, 23, 20]	20	28	23	20
2	臺北市	63000001	[42, 63, 63, 54]	42	63	63	54
3	雲林縣	10009001	[62, 63, 61, 59]	62	63	61	59
4	桃園市	10003001	[42, 55, 57, 63]	42	55	57	63
5	屏東縣	10013001	[68, 67, 57, 59]	68	67	57	59
6	臺中市	10006001	[61, 67, 65, 63]	61	67	65	63
7	臺南市	10011001	[68, 57, 66, 68]	68	57	66	68
8	基隆市	10017001	[38, 34, 48, 48]	38	34	48	48
9	連江縣	09007001	[77, 71, 96, 112]	77	71	96	112
10	南投縣	10008001	[64, 65, 56, 58]	64	65	56	58
11	澎湖縣	10016001	[69, 54, 51, 50]	69	54	51	50
12	苗栗縣	10005001	[56, 43, 65, 75]	56	43	65	75
13	嘉義市	10020001	[73, 76, 71, 69]	73	76	71	69
14	新竹縣	10004001	[50, 51, 53, 51]	50	51	53	51
15	新北市	10001001	[35, 52, 54, 46]	35	52	54	46
16	花蓮縣	10015001	[37, 35, 20, 16]	37	35	20	16
17	高雄市	10012001	[74, 63, 62, 70]	74	63	62	70
18	彰化縣	10007001	[56, 59, 65, 58]	56	59	65	58
19	嘉義縣	10010001	[68, 69, 72, 67]	68	69	72	67
20	金門縣	09020001	[84, 82, 80, 73]	84	82	80	73
21	新竹市	10018001	[52, 55, 56, 56]	52	55	56	56

資料視覺化-資料整理



```
1 tw_geo = 'twgeo.json'
2 period = ['aqi1', 'aqi2', 'aqi3', 'aqi4']
3
4 predictTime = []
5 for i in range(1, 9, 2):
6     startTime = forecast[0][3][i].split('~')[0]
7     endTime = forecast[0][3][i].split('~')[1]
8     startTime = datetime.strptime(startTime, '%Y-%m-%d %H:%M:%S')
9     endTime = datetime.strptime(endTime, '%Y-%m-%d %H:%M:%S')
10    week = startTime.weekday()
11    month = str(startTime.month)
12    day = str(startTime.day)
13    hour = endTime.hour
14    day_or_night = (lambda x: '白天' if x == 18 else '晚上')
15    weekday = (lambda x: '一' if x == 0 else ('二' if x == 1 else ('三' if x == 2 else ('四' if x == 3 else ('五' if
16    time = month + '/' + day + '(' + weekday(week) + ')' + ' ' + day_or_night(hour)
17    print(time)
18    predictTime.append(time)
```

predictTime

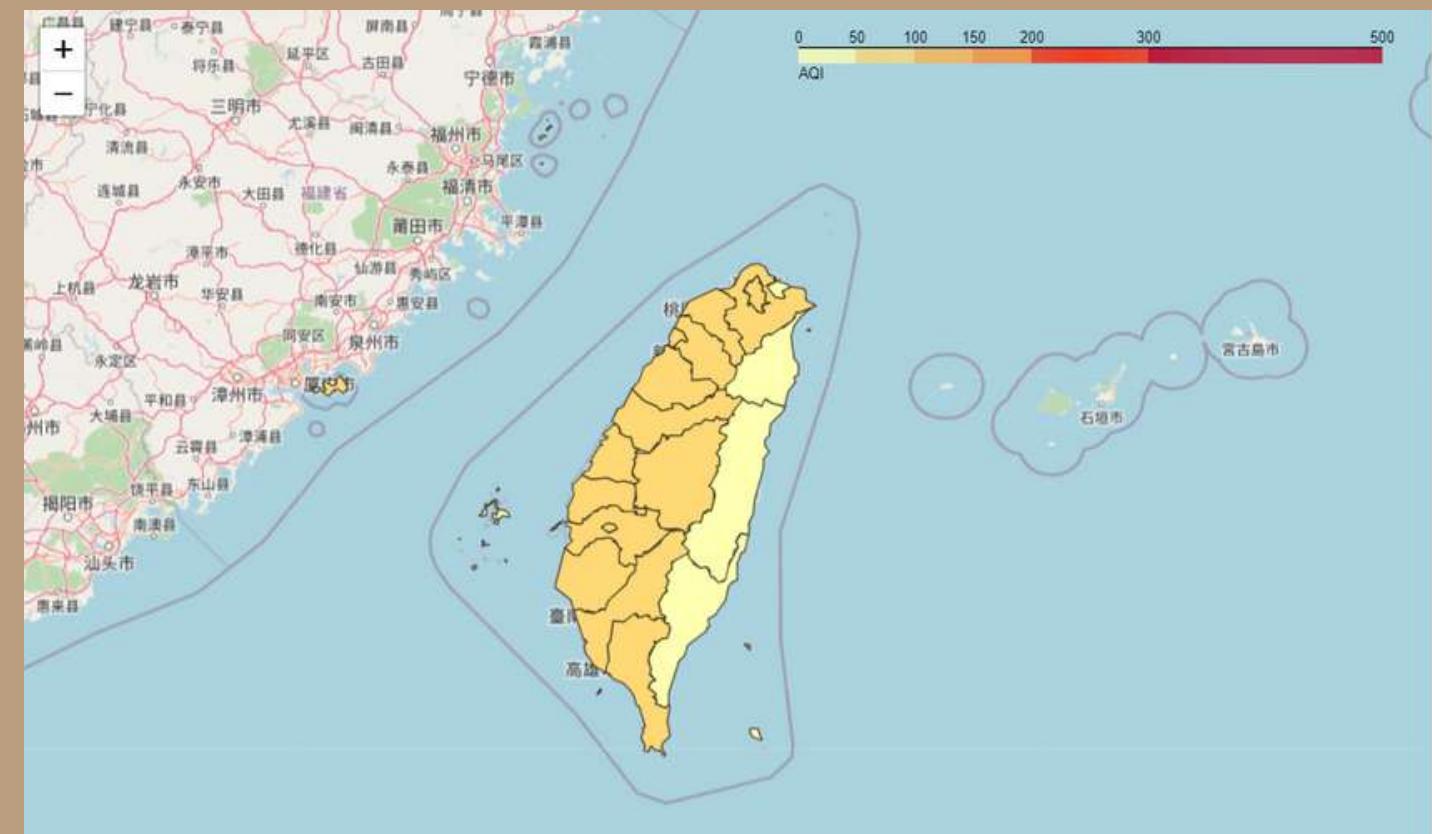
創建出預測時段

資料視覺化-色塊



```
21 fmap1 = folium.Map(location=[25.03, 121.3], zoom_start = 7.4)
22 fmap2 = folium.Map(location=[25.03, 121.3], zoom_start = 7.4)
23 fmap3 = folium.Map(location=[25.03, 121.3], zoom_start = 7.4)
24 fmap4 = folium.Map(location=[25.03, 121.3], zoom_start = 7.4)
25 fmap = [fmap1, fmap2, fmap3, fmap4]
26 res = []
27 for i in range(4):
28     choropleth = folium.Choropleth(
29         geo_data = tw_geo,
30         data = data,
31         columns = ["countysn", period[i]],
32         key_on = 'feature.properties.COUNTYSN',
33         bins = [0,50,100,150,200,300, 500],
34         fill_color = "YlOrRd",
35         fill_opacity = 1,
36         line_opacity = 0.7,
37         legend_name = 'AQI',
38     )
39     choropleth.add_to(fmap[i])
40
41     index = 0
42     for feature in choropleth.geojson.data['features']:
43         countysn = feature['properties']['COUNTYSN']
44         feature['properties'][period[i]] = str(data[data['countysn'] == countysn][period[i]][index])
45         feature['properties']['county'] = str(data[data['countysn'] == countysn]['countyname'][index])
46         index += 1
47
48     choropleth.geojson.add_child(
49         folium.features.GeoJsonTooltip(['county', period[i]], labels = True)
50     )
51     title_html = '''
52         <h3 align="center" style="font-size:20px"><b>{predictTime}</b></h3>
53         '''.format(predictTime = predictTime[i])
54     fmap[i].get_root().html.add_child(folium.Element(title_html))
55
56 fmap[0]
```

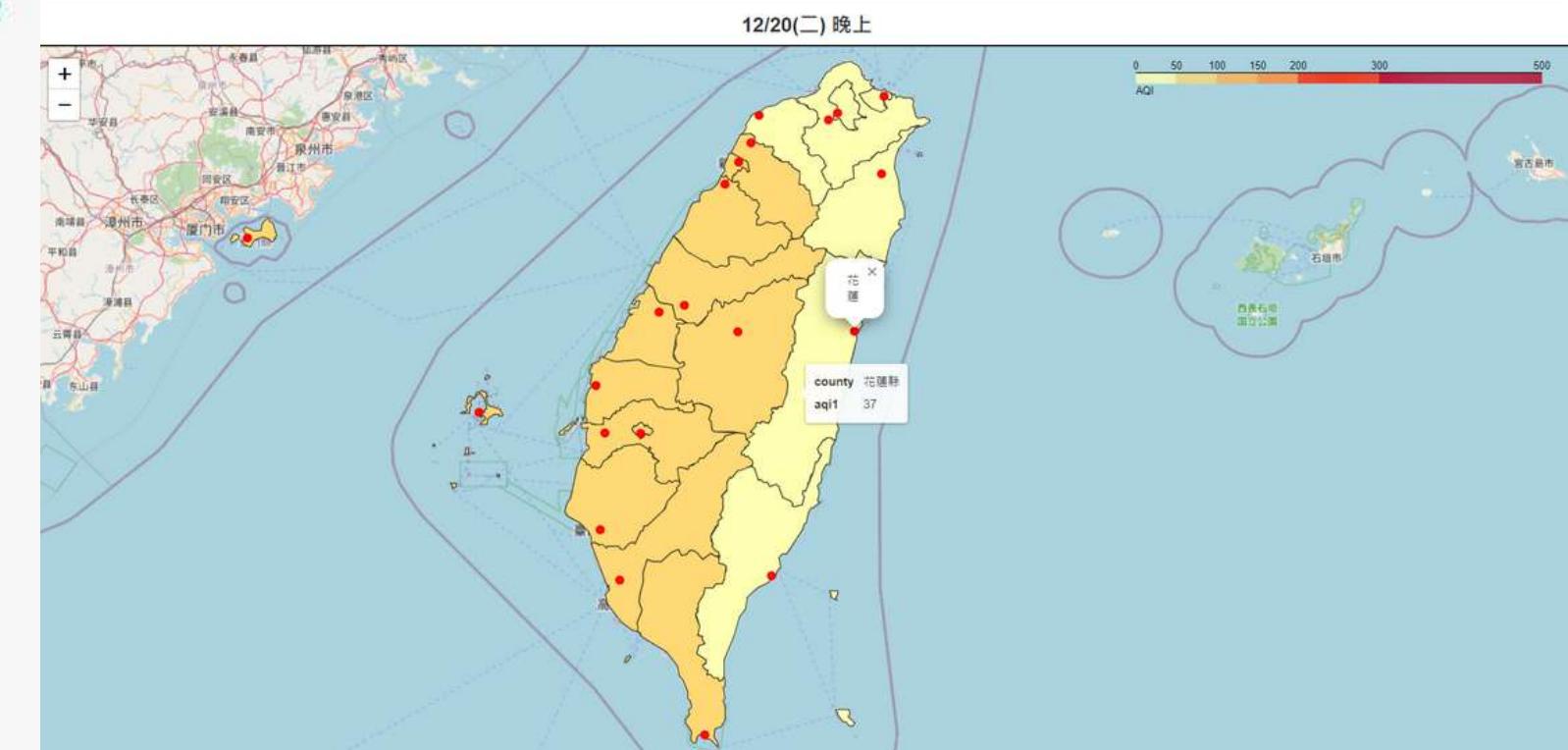
folium.Choropleth()
分別劃出四個時段的圖



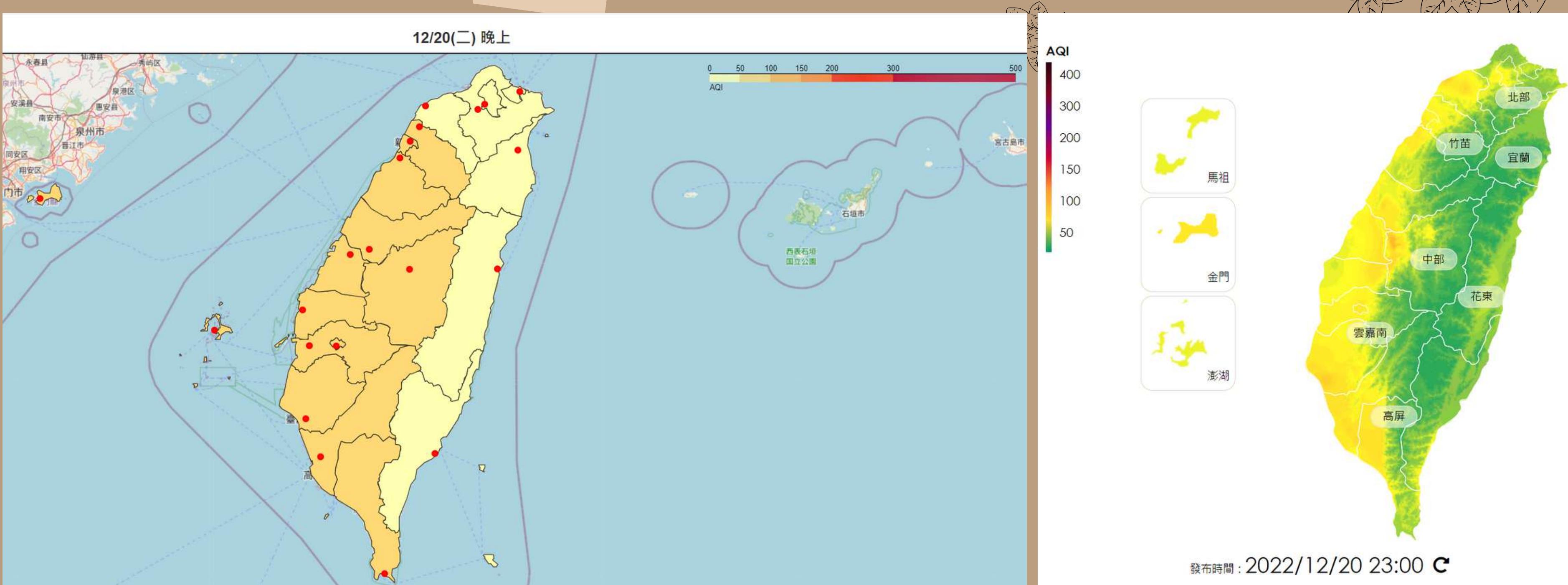
資料視覺化-測站及座標



```
1 location = [['臺東', 22.75535, 121.15045], ['宜蘭', 24.74791, 121.74639], ['萬華', 25.04650, 121.50797], ['臺西', 23.7611, 120.2104], ['觀音', 25.03556, 121.08283], ['恆春', 21.95806, 120.78892], ['大里', 24.09961, 120.67844], ['臺南', 22.9811, 120.2011], ['基隆', 25.12916, 121.76005], ['馬祖', 26.15188, 119.93149], ['埔里', 23.96884, 120.9679], ['馬公', 23.5611, 120.2011], ['頭份', 24.6969, 120.89869], ['嘉義', 23.46477, 120.44125], ['湖口', 24.90009, 121.03886], ['板橋', 25.0111, 121.2104], ['花蓮', 23.9713, 121.59976], ['楠梓', 22.73366, 120.32828], ['彰化', 24.0660, 120.54152], ['朴子', 23.4611, 120.2104], ['金門', 24.43213, 118.31225], ['新竹', 24.80563, 120.97236]]  
7 for i in range(4):  
8     for j in location:  
9         folium.Circle(location = j[1:3],  
10            color = 'red', # Circle 顏色  
11            radius = 1500, # Circle 寬度  
12            fill = True, # 填滿中間區域  
13            fill_opacity = 0.9, # 設定透明度:1是完全不透  
14            popup = j[0]  
15        ).add_to(fmap[i])  
16    if i == 0:  
17        fmap[0].save('airquality1.html')  
18    elif i == 1:  
19        fmap[1].save('airquality2.html')  
20    elif i == 2:  
21        fmap[2].save('airquality3.html')  
22    elif i == 3:  
23        fmap[3].save('airquality4.html')
```



預測結果比對



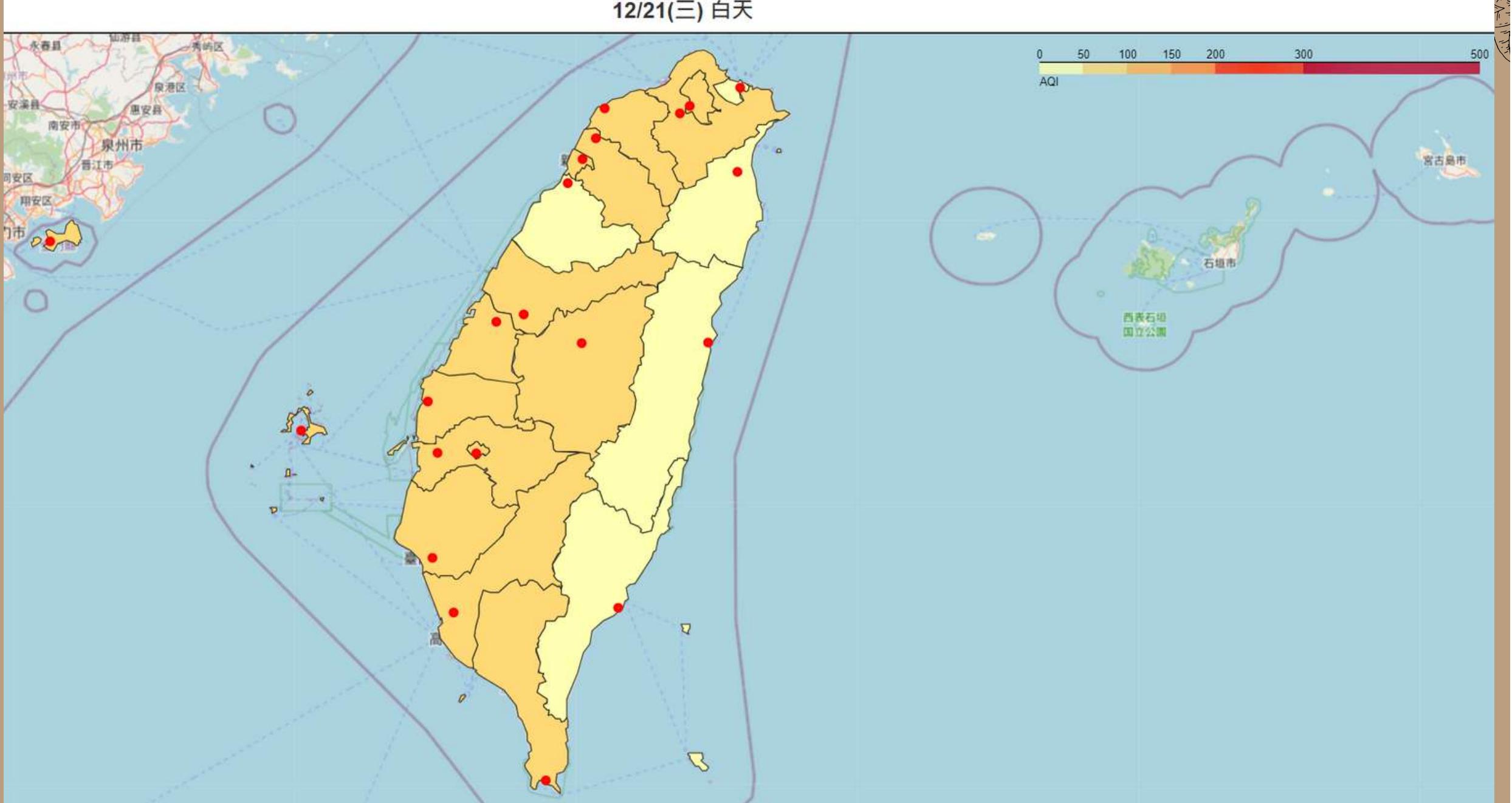
12/20(二)晚上
AQI預測圖

12/20(二)
中央氣象局預測結果

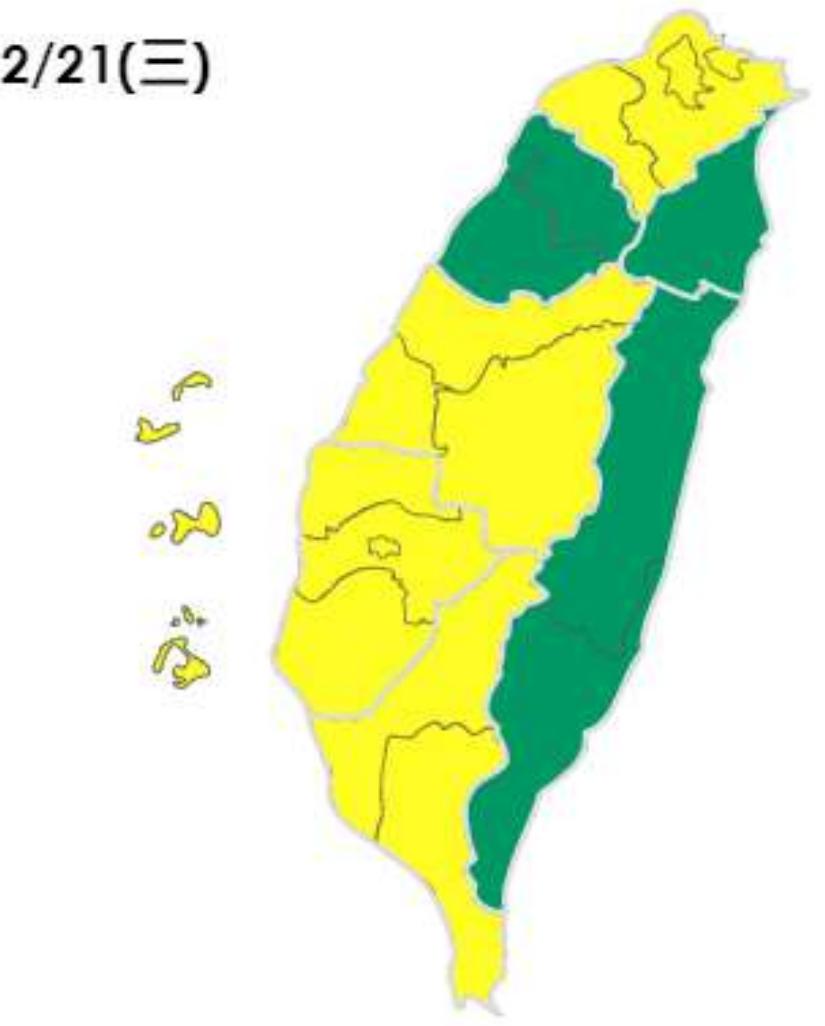
預測結果比對



12/21(三) 白天



12/21(三)



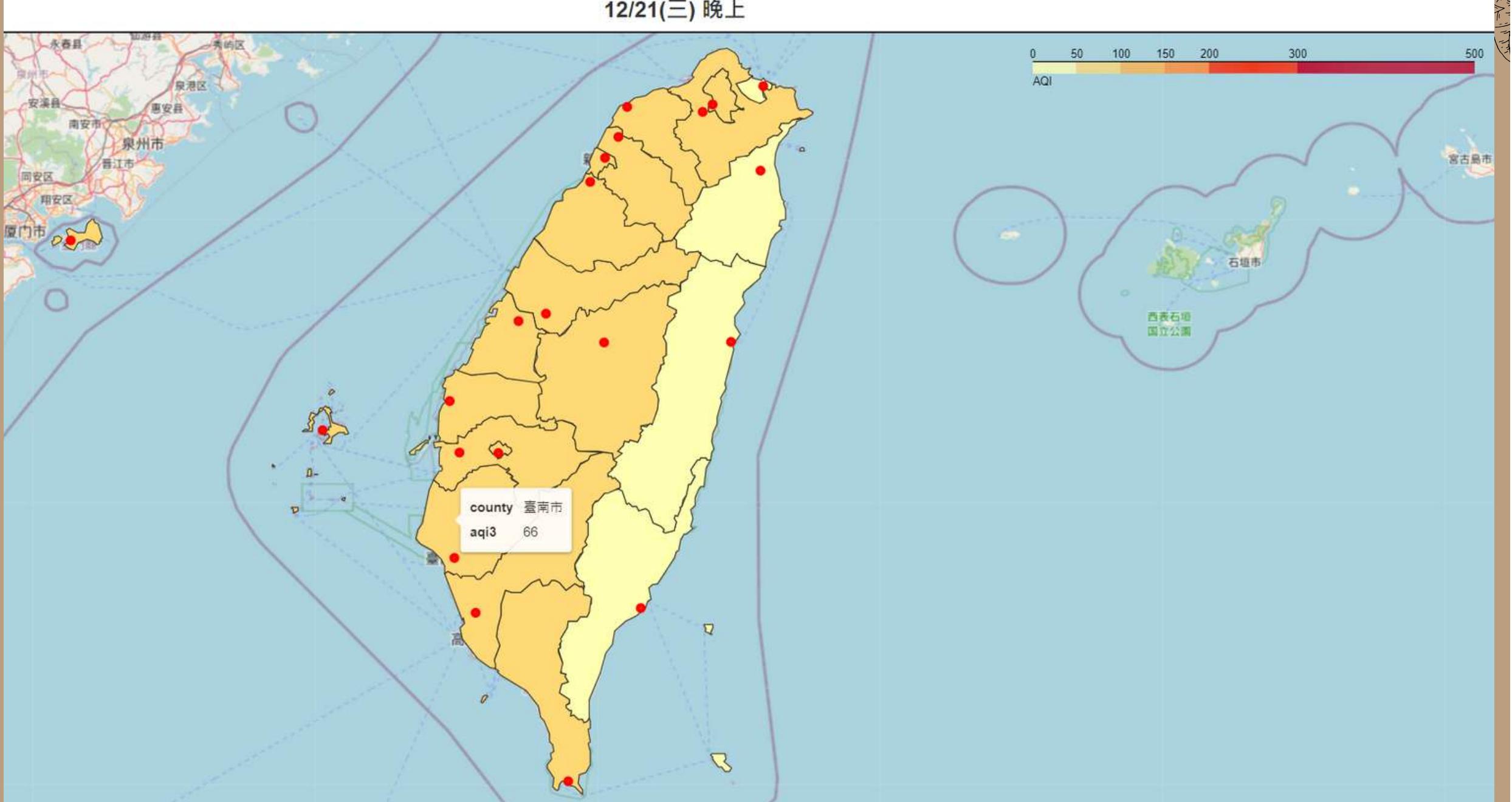
12/21(三)早上
AQI預測圖

12/21(三)
中央氣象局預測結果

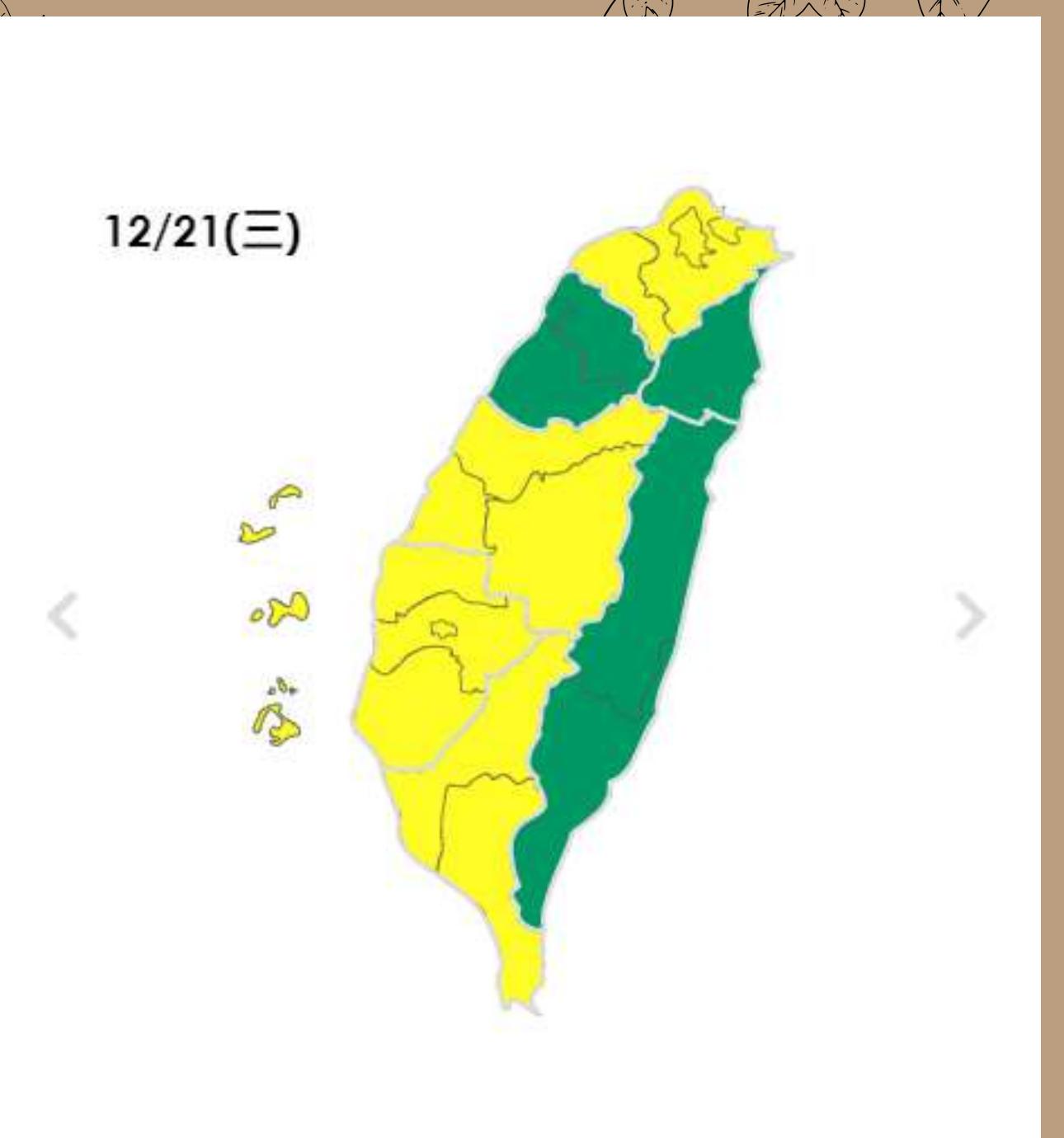
預測結果比對



12/21(三) 晚上



12/21(三)



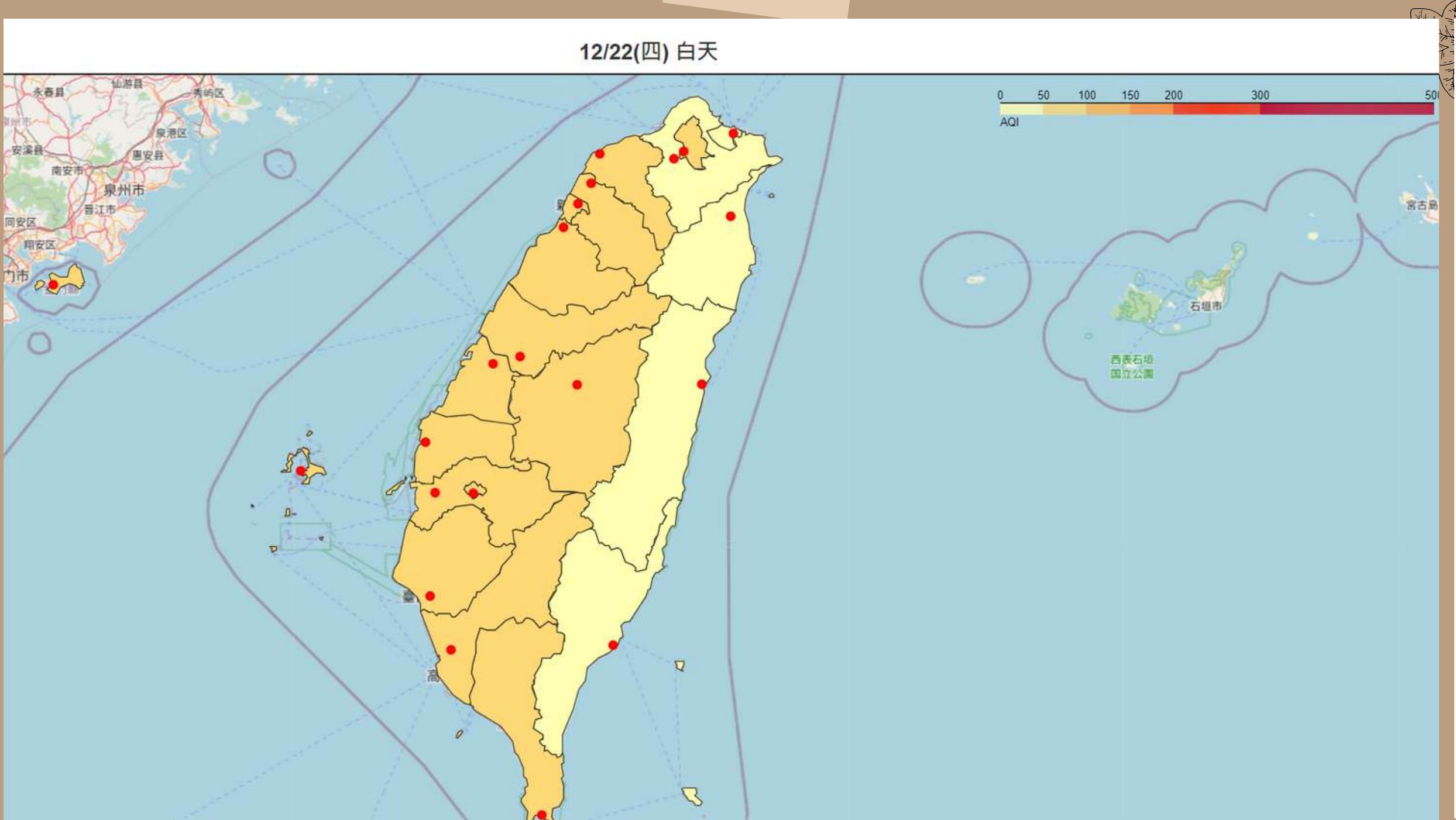
12/21(三)晚上
AQI預測圖

12/21(三)
中央氣象局預測結果

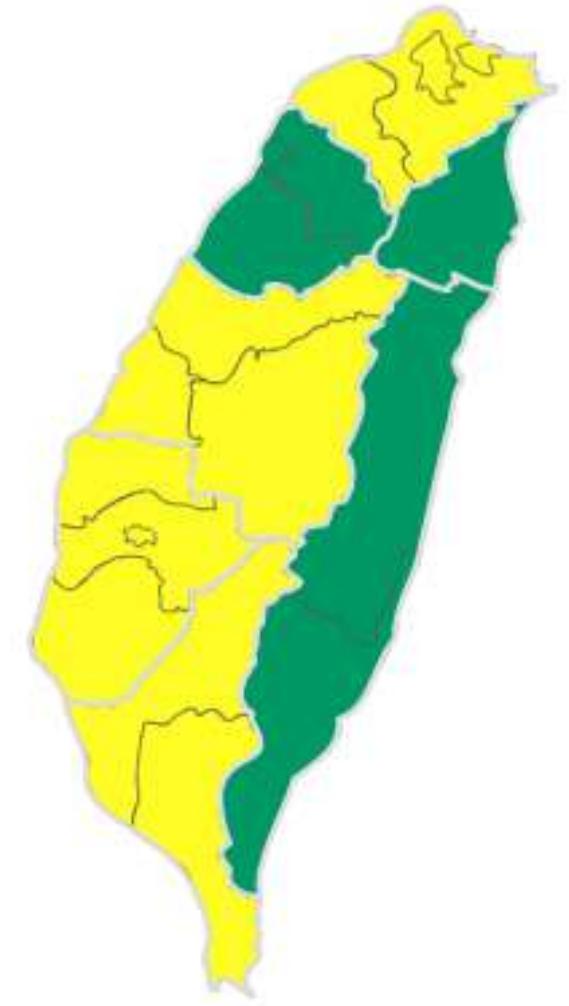
預測結果比對



12/22(四) 白天



12/22(四)



12/22(四)早上
AQI預測圖

12/22(四)
中央氣象局預測結果



CONCLUSION

結論

臺灣各區域的空氣品質狀況：

- 以**區域**分析

東部最好，北部次之，中南部及離島最差

- 以**季節**分析

夏季空氣品質優於其他季節

空氣品質與其他因素的關係：

- **降水** 因素

有降雨的空氣品質較沒下雨的好

- **溫度**、**UVI** 因素

沒有太大關聯

空氣品質與其他因素的關係：

- **風速**因素

風速越強空氣品質較差的機率較低

- **風向**因素

若為東北風，北部空氣品質較佳，中南部空氣品質較差。

若為境外汙染移入，則北部及離島空氣品質會變差。

**臺灣各地歷年空氣品質變化是劇增
或是下降：**

依照統計圖表可以推測出，空氣品質近年來有微量轉好的趨勢。



謝謝聆聽