函數型資料分析期末報告

組員：林祐陞、郭又嘉、張泳樺

題目：利用FPCA做PM2.5的補值並與GLRM比較結果

一. 使用資料：Air Quality in Madrid (2001-2018)，資料集含有不同空氣污染的指標，其中只取2011/1/1-2014/1/1的PM2.5的資料，由於此年份只有六個測站保有PM2.5的觀測值，其餘測站全部遺失，故只取這六個測站作分析。

二. 研究動機：此資料集擁有相當多的遺失值，當面臨不齊全或分佈不相同的資料，FPCA是一個適合使用的分析方法，他依然可以將資料背後的函數重建回來，再利用重建函數補齊遺失值，而GLRM是PCA的統稱，也是常見的補值方法，所以將使用兩者方法來比較補值的成效。

三. 使用工具：python MFPCA套件、H2O GLRM

四. 方法介紹：

1. MFPCA

在PM2.5資料中，假設 Y_{ij} 為空間函數 $X_i(\cdot)$ 在 t_{ij} 測站的經緯度下，帶有測量誤差的PM2.5觀測值

$$Y_{ij} = X_i(t_{ij}) + \epsilon_{ij},\ i = 1,...,n,\ j = 1,...,m$$

利用 Karhunen-Loève 展開，可得

$$Y_{ij} = \mu(t_{ij}) + \sum_{k=1}^{\infty} \xi_{ik}\phi_k(t_{ij}) + \epsilon_{ij}$$

其中，

- $\mu(t) = E(X(t))$ 為平均函數。
- ξ_{ik} 為主成份分數。
- $\phi_k(t_{ij})$ 為特徵函數，以此作為函數估計的基底。

為了將重建空間函數 $X_i(\cdot)$，所以須估計 $\mu(t)$、ξ_{ik} 和 $\phi_k(t)$，下面將說明MFPCA套件中的作法。

(1) 利用局部線性迴歸法估計 $\mu(t)$：

$$\hat{\beta} = \underset{\beta=(\beta_0,\ldots,\beta_d)\in\mathbb{R}^{d+1}}{argmin} \sum_{i=1}^{n}\sum_{j=1}^{m}[Y_{ij} - \beta_0 - \sum_{p=1}^{d}\beta_p(t_{ij,p} - t_p)]^2 K_{h_\mu}(t_{ij} - t)$$

選取合適核函數 $K(\cdot)$ 及帶寬 h_μ，可得 $\hat{\mu}(t) = \hat{\beta}_0$ 。

(2) 由於 $\phi_k(t)$ 需從共變異函數 $C(s,t) = cov(X(s), X(t))$ 得到，所以在第二步中，先假設 $C_i(t_{ij}, t_{il}) = (Y_{ij} - \hat{\mu}(t_{ij}))(Y_{il} - \hat{\mu}(t_{il}))$ 為中心化後的觀測值，則其期望值 $E(C_i(t_{ij}, t_{il})) = E(Y_{ij}Y_{il}) - \hat{\mu}(t_{ij})\hat{\mu}(t_{il})$，因此選擇適當的核函數和帶寬，利用局部線性迴歸法估計未知的 $E(Y_{ij}Y_{il})$：

$$\hat{\beta} = \underset{\beta=(\beta_0,\ldots,\beta_{2d})\in\mathbb{R}^{2d+1}}{argmin} \sum_{i=1}^{n}\sum_{1\le j\ne l\le N_i}[Y_{ij}Y_{il} - \beta_0 - \sum_{p=1}^{d}(\beta_p(s_{ij,p} - s_p) + \beta_{p+d}(t_{il,p} - t_p))]^2 \times K_{h_C}(s_{ij} - s)K_{h_C}(t_{il} - t)$$

最終可得 $\hat{C}(t_{ij}, t_{il}) = \hat{\beta}_0 - \hat{\mu}(t_{ij})\hat{\mu}(t_{il})$。

(3) 將估計的共變異函數 $\hat{C}(s,t)$ 轉換成共變異函數矩陣，對此作特徵值分解，計算出特徵值和特徵向量，經過校正可得到估計的特徵函數 $\hat{\phi}_k(t)$。

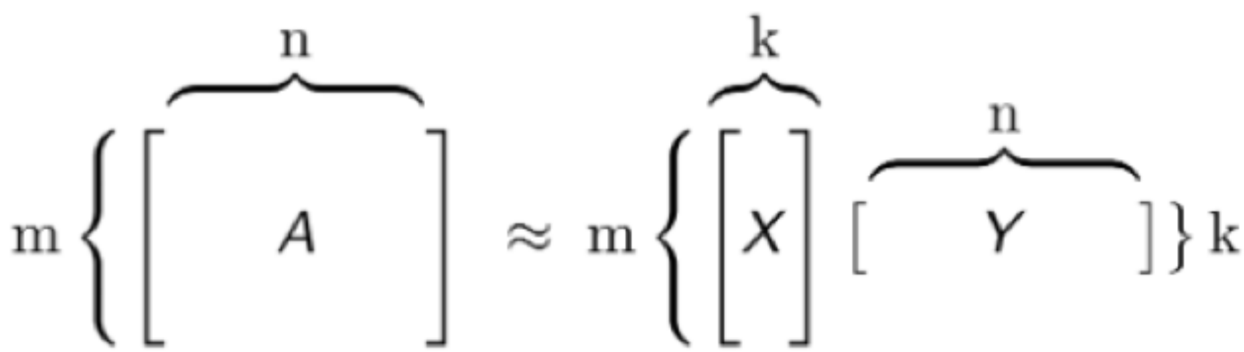
(4) 利用多元常態分配估計主成分分數 ξ_{ik}，可得 $\hat{\xi}_{ik} = \hat{\lambda}_j\hat{\phi}_{ij}\hat{\Sigma}_{Y_i}^{-1}(\tilde{Y}_i - \hat{\mu}_i)$ (詳細推導來自參考資料)。

(5) 利用變異數解數(FVE)選出前 K 個特徵函數並將上述估計值代入原式，可將 $X_i(t)$ 空間函數重建，亦即

$$\hat{X}_i(t) = \hat{\mu}(t) + \sum_{k=1}^{K}\hat{\xi}_{ik}\hat{\phi}_k(t)$$

2. GLRM

在此次PM2.5的資料補值中，資料矩陣 A 可拆解為 X 矩陣和 Y 矩陣，矩陣 X 為原資料降維的資料矩陣，矩陣 Y 則是特徵轉換矩陣。



Loss 上 minimize $\sum_{i,j\in\Omega} L_{ij}(x_i y_j, A_{ij}) + \gamma_1 \sum_{i=1}^{m} r_i(x_i) + \gamma_2 \sum_{j=1}^{n} \tilde{r}_j(y_j)$ ，其中 Ω 表示為有值的範圍。

1. 利用Alternating minimization的方式更新：

   Minimize $\sum_{i,j\in\Omega} L_{ij}(x_i y_j, A_{ij})$

   交替固定矩陣，進行 X 矩陣和 Y 矩陣的逼近。

2. 依需求正規化變體：

   Loss後面兩項正規化依照當傾向任何性質需求時使用，可變體出PCA、NNMF、sparse coding、k-means等等。

variations on GLRMs recover many known models:

Model	$\mathbf{L_{ij}(u,a)}$	$\mathbf{r(x)}$	$\mathbf{\tilde{r}(y)}$	**reference**
PCA	$(u-a)^2$	0	0	[Pearson 1901]
NNMF	$(u-a)^2$	$I_+(x)$	$I_+(y)$	[Lee 1999]
sparse PCA	$(u-a)^2$	$\|x\|_1$	$\|y\|_1$	[D'Aspremont 2004]
sparse coding	$(u-a)^2$	$\|x\|_1$	$\|y\|_2^2$	[Olshausen 1997]
k-means	$(u-a)^2$	$I_1(x)$	0	[Tropp 2004]
matrix completion	$(u-a)^2$	$\|x\|_2^2$	$\|y\|_2^2$	[Keshavan 2010]
robust PCA	$\lvert u-a\rvert$	$\|x\|_2^2$	$\|y\|_2^2$	[Candes 2011]
logistic PCA	$\log(1+\exp(-au))$	$\|x\|_2^2$	$\|y\|_2^2$	[Collins 2001]
boolean PCA	$(1-au)_+$	$\|x\|_2^2$	$\|y\|_2^2$	[Srebro 2004]

而GLRM在資料上可達到以下幾點優勢，

1. Memory：僅保存 X 和 Y 矩陣時，可大幅降低資料儲存所需的內存空間，10GB的資料可壓縮至100MB，在需要使用到原始資料時，也能夠很快的重建資料，並且失真率低。
2. Speed：GLRM可將高維的數據壓縮至小矩陣，導致模型建構和預測上的加速，特別是機器學習領域中，對特徵空間大小上的影響特別顯著。
3. Feature Engineering：Y 矩陣表示訓練數據中特徵轉換的重要組合，可對這些壓縮特徵進行分析。

4. Missing Data Imputation：由 X 和 Y 矩陣重建數據集將會自動估算缺失值，以此方式進行補值。

五. 資料分析

1. 目的：重建每一小時經緯度之下的PM2.5觀測值的函數圖形
2. 資料處理：從原始資料中將每一測站抽取10%資料當開頭，隨機遺失1~10天，當作訓練資料以模擬稀疏資料，以利最後擁有真實值做對比。下圖以前100個資料為例。

	station	28079008	28079024	28079038	28079047	28079048	28079050
0	2011-01-01 01:00:00	18.0	12.0	24.0	NaN	9.0	22.0
1	2011-01-01 02:00:00	28.0	25.0	27.0	28.0	21.0	20.0
2	2011-01-01 03:00:00	22.0	24.0	16.0	23.0	12.0	13.0
3	2011-01-01 04:00:00	13.0	12.0	10.0	15.0	3.0	9.0
4	2011-01-01 05:00:00	9.0	11.0	7.0	10.0	1.0	7.0
...	...	...	...	...	...	...	...
95	2011-01-05 00:00:00	11.0	6.0	8.0	9.0	4.0	11.0
96	2011-01-05 01:00:00	10.0	6.0	7.0	10.0	6.0	8.0
97	2011-01-05 02:00:00	9.0	7.0	7.0	9.0	4.0	10.0
98	2011-01-05 03:00:00	7.0	7.0	6.0	8.0	2.0	10.0
99	2011-01-05 04:00:00	9.0	6.0	6.0	9.0	4.0	8.0

100 rows × 7 columns

原始資料

	station	28079008	28079024	28079038	28079047	28079048	28079050
0	2011-01-01 01:00:00	18.0	12.0	24.0	NaN	9.0	22.0
1	2011-01-01 02:00:00	28.0	25.0	27.0	28.0	21.0	20.0
2	2011-01-01 03:00:00	22.0	24.0	16.0	23.0	12.0	13.0
3	2011-01-01 04:00:00	13.0	NaN	10.0	15.0	3.0	9.0
4	2011-01-01 05:00:00	9.0	11.0	NaN	10.0	1.0	7.0
...	...	...	...	...	...	...	...
95	2011-01-05 00:00:00	11.0	6.0	8.0	9.0	4.0	11.0
96	2011-01-05 01:00:00	10.0	6.0	7.0	10.0	6.0	NaN
97	2011-01-05 02:00:00	9.0	7.0	7.0	9.0	4.0	NaN
98	2011-01-05 03:00:00	7.0	7.0	6.0	8.0	NaN	NaN
99	2011-01-05 04:00:00	NaN	6.0	NaN	9.0	NaN	NaN

100 rows × 7 columns

訓練資料，自行填入NaN

3. 分析流程：

(1) 使用MFPCA套件建立經緯度的空間函數

```
Fpca(x, y, x0, h_mean, h_cov, h_cov_dia, fve = 0.85, binning = True, bin_weight = True,
     ker_fun = 'Epan', bw_select = 'Partition', dtype = 'f4')
```

- 資料：以每一小時為單位，共有26304組
- x：每一小時的稀疏觀測點, shape = (26304, -1)
- y：每一小時的稀疏觀測值, shape = (26304, -1)
- x0：每個維度上的格點數量, shape = (6,6,2)

例子：以前三小時為例

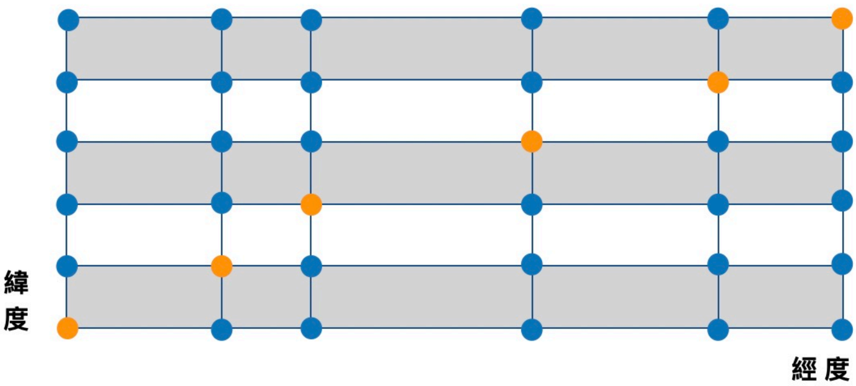
time	28079008	28079024	28079038	28079047	28079048	28079050
2011-01-01 01:00:00	18.0	12.0	24.0	NaN	9.0	22.0
2011-01-01 02:00:00	28.0	25.0	27.0	28.0	21.0	20.0
2011-01-01 03:00:00	22.0	24.0	16.0	23.0	12.0	13.0

```
[array([[40.42156389, -3.68231944],
        [40.41935556, -3.74734722],
        [40.44554444, -3.70712778],
        [40.43989722, -3.69036667],
        [40.46557222, -3.68876944]]),
 array([[40.42156389, -3.68231944],
        [40.41935556, -3.74734722],
        [40.44554444, -3.70712778],
        [40.39811389, -3.686825  ],
        [40.43989722, -3.69036667],
        [40.46557222, -3.68876944]]),
 array([[40.42156389, -3.68231944],
        [40.41935556, -3.74734722],
        [40.44554444, -3.70712778],
        [40.39811389, -3.686825  ],
        [40.43989722, -3.69036667],
        [40.46557222, -3.68876944]]),
```

輸入資料（x）

```
[array([18., 12., 24.,  9., 22.]),
 array([28., 25., 27., 28., 21., 20.]),
 array([22., 24., 16., 23., 12., 13.]),
```

輸入資料（y）



輸入資料（x0）示意圖
x0：棋盤上的每一格點的座標

```
([[[40.42156389, -3.68231944],
   [40.42156389, -3.68231944],
   [40.42156389, -3.68231944],
   [40.42156389, -3.68231944],
   [40.42156389, -3.68231944],
   [40.42156389, -3.68231944]],

  [[40.41935556, -3.74734722],
   [40.41935556, -3.74734722],
   [40.41935556, -3.74734722],
   [40.41935556, -3.74734722],
   [40.41935556, -3.74734722],
   [40.41935556, -3.74734722]],
```

輸入資料（x0）

(2) 預測函數上的值：共預測26304組

```
fit_funx_fpc_scores, fit_funx = result.Restruct_Fun(x = no_missing_cor , y = no_missing_pm25)
```

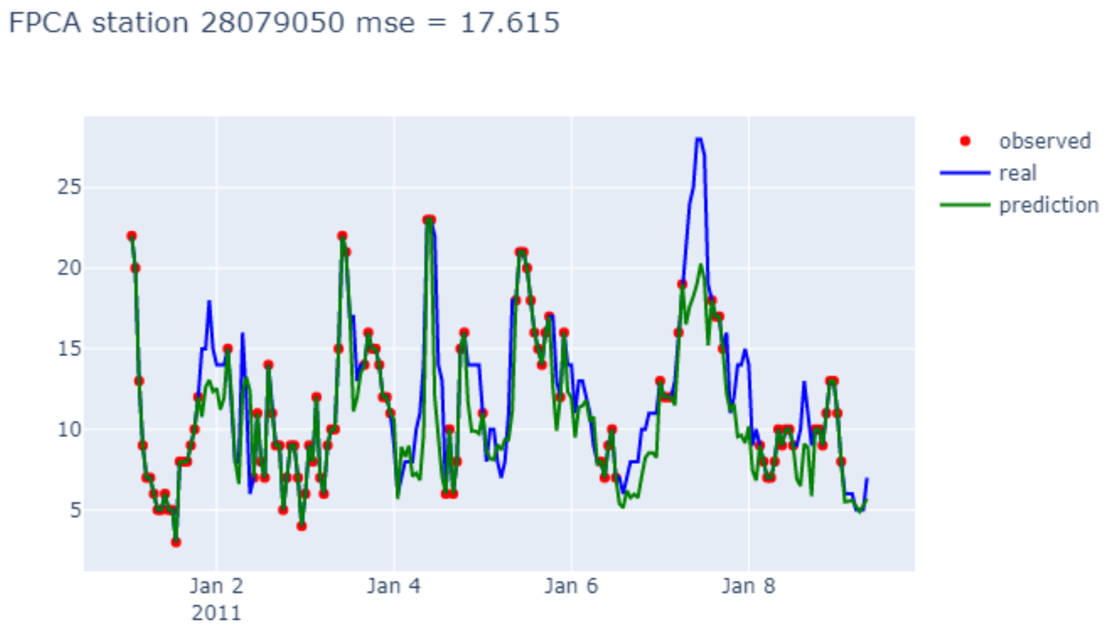
- x：每一小時的稀疏觀測點, shape = (26304, -1)
- y：每一小時的稀疏觀測值, shape = (26304, -1)

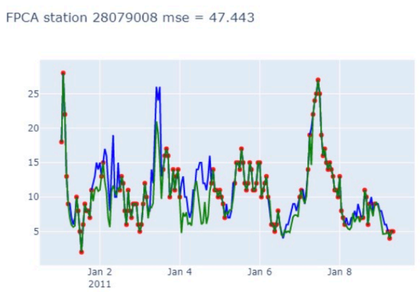
Return：

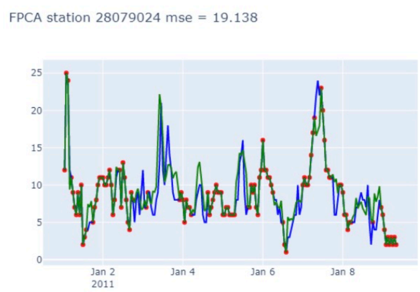
- fpc_scores ：將 $X(s,t)$ 中心化後，投影在特徵函數上的主成份分數
- restruct_fun ：重現在格點上的 $X(s,t)$ 函數，shape = (26304, 6, 6)

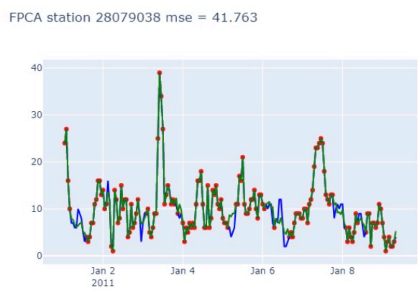
4. 結果

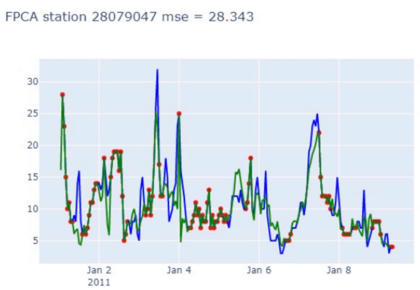
- MFPCA結果：以前200小時為例，畫出紅點為觀察值、藍線為真實曲線、綠線為預測曲線。

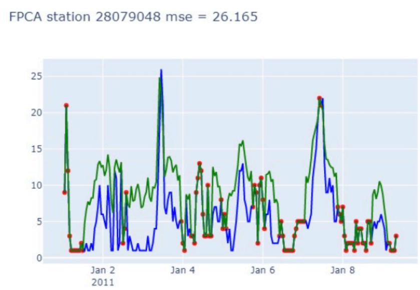


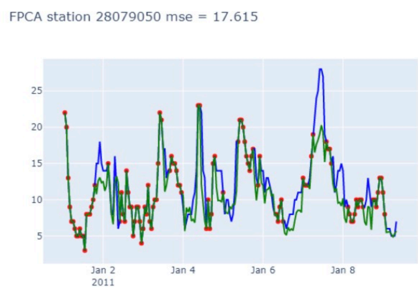






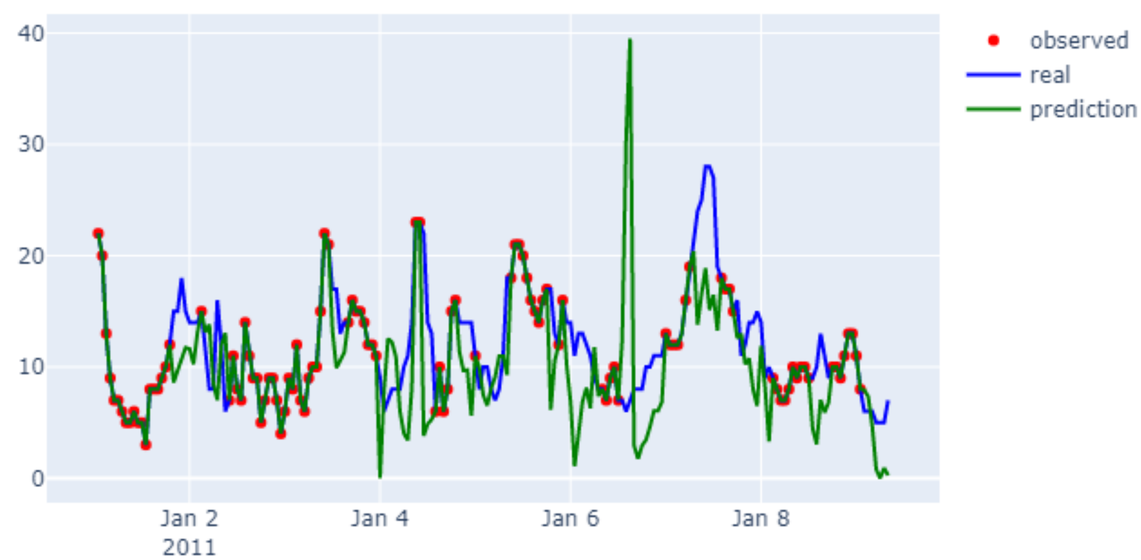




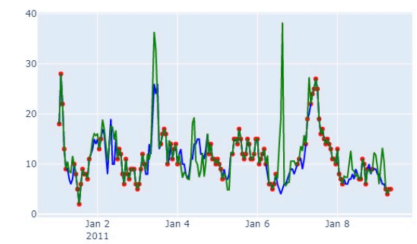


• GLRM結果

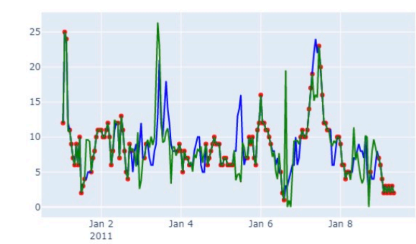
GLRM station 28079050 mse = 68.735



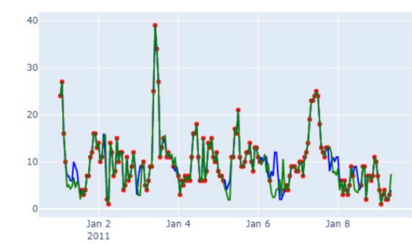
GLRM station 28079008 mse = 40.355



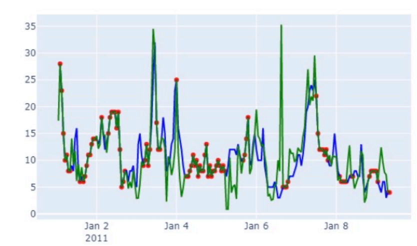
GLRM station 28079024 mse = 37.065



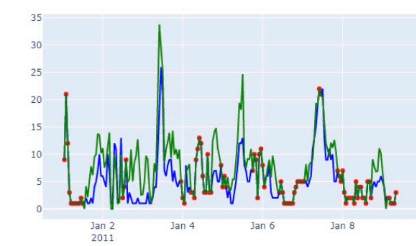
GLRM station 28079038 mse = 51.274



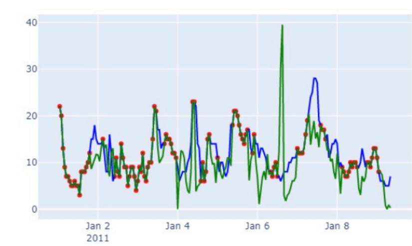
GLRM station 28079047 mse = 54.181



GLRM station 28079048 mse = 47.162



GLRM station 28079050 mse = 68.735



- 比較：只計算自行挖空的缺值的Mean Square Error

MSE	MFPCA	GLRM
Station1	47.443	40.355
Station2	19.138	37.065
Station3	41.763	51.274
Station4	28.343	54.181
Station5	26.165	47.162
Station6	17.615	68.735

六. 參考資料

- 許嘉揚 (2017), A Python Package for Fast Algorithm of Multi-dimensional Functional Principal Component Analysis, 中興大學碩士論文
- 黃敏嘉 (2017), Multivariate Function-on-Function Linear Regression, 中興大學碩士論文
- Madeleine Udell，Corinne Horn，Reza Zadeh，Stephen Boyd Generalized Low Rank Models 2015
- H2O教程