國立政治大學 應用數學系

碩士 學位論文

以 1998 年至 2012 年健保資料庫分析流感疫苗施打成效

The performance of influenza vaccine policy from 1998 to 2012 by National Health Insurance Database

碩士班學生：吳宥柔　撰
指導教授：曾正男　博士

中華民國 105 年 6 月 24 日

國 立 政 治 大 學 應 用 數 學 系

吳宥柔君所撰之碩士學位論文

以 1998 年至 2012 年健保資料庫分析流感疫苗施打成效

The performance of influenza vaccine policy from 1998 to 2012 by National Health Insurance Database

業經本委員會審議通過
論文考試委員會委員：

指導教授：

系主任：

中華民國 105 年 6 月 24 日

國立政治大學 National Chengchi University

中文摘要

流行性感冒是一種容易快速傳染且造成地區性大流行的疾病，雖然政府宣導接種流感疫苗可以有效的防治流行性感冒的疫情，然而接種流感疫苗的人並不多，因此本研究將探討疫苗的有效性，以解決一般民眾對施打疫苗的疑慮。

我們以 1998 年至 2012 年的健保資料庫分析疫苗的有效性，分析方法以 Python 程式將資料分類並做運算，其中一方法以有打疫苗者和未打疫苗者的平均流感就醫次數作為判別疫苗的有效性，另一方法則是用 bootstrapping 來判別疫苗之有效性。

透過本研究的結果可以知道，6 歲以下兒童和 6 歲以上 65 歲以下成人類別疫苗都是呈現有效的，但在 65 歲以上老人則是呈現疫苗無效，雖然如此，在死亡率的比較上，有打疫苗的老人死亡率則是低於沒打疫苗的老人，故仍建議民眾施打疫苗。

關鍵字：流行性感冒、疫苗、健保資料

國立政治大學 National Chengchi University

Abstract

Influenza is a contagious and easily and quickly create a regional pandemic disease, although the government propaganda influenza vaccination can effectively control the influenza, but there is not many people inoculated, so this study will explore vaccine effectiveness.

We use National Health Insurance Database from 1998 to 2012 analysis the performance of influenza vaccination ,and use Python program to classify data and do arithmetic,in which there is a method to vaccination and vaccination is not a doctor's average number of influenza vaccine effectiveness as discrimination, the other rule of thumb is to use bootstrapping to determine the effectiveness of vaccines.

Through the results of this study may know, children under 6 years of age and 65 years of age 6 years and older adult category vaccines are rendered effective, but over the age of 65 is rendered ineffective vaccine, while in mortality, there is vaccination elderly mortality rate is lower than those not vaccinated.

Keywords: Influenza,flu, vaccine,Health Insurance Database

國立政治大學 National Chengchi University

目錄

口試委員會審定書 i
中文摘要 ii
Abstract iii
目錄 iv
表目錄 vi
圖目錄 vii
第一章 前言 1
第一節 研究背景與目的 1
第二節 研究架構 2
第二章 文獻探討 3
第一節 簡介健保資料庫 3
第二節 流行性感冒 5
一、名詞定義 5
二、流行性感冒簡介 6
三、流感預防 7
第三節 流感疫苗 7
第四節 2002 年論文回顧 8
第五節 公共政策 9
第三章 研究方法 10
第一節 資料整理 10
第二節 分析方法 11
一、方法一 11
二、方法二 12

國立政治大學 National Chengchi University

第四章 研究經驗與結果 13
第一節 實驗步驟 15
一、整合所有 ID 15
二、工具程式 16
第二節 研究方法一之經驗與結果 18
一、研究過程 18
二、研究結果 21
第三節 研究方法二之經驗與結果 28
一、研究過程 28
二、研究結果 29
第五章 結論與建議 31
參考文獻 32
附錄 A: 常用程式碼註解 34
附錄 B: 整理資料類之程式碼 35
附錄 C: 方法一使用之程式碼 37
附錄 D: 方法二使用之程式碼 55
國立政治大學 National Chengchi University

表目錄

B.1 流行性感冒與一般感冒比較表 6
B.2 8
B.3 從 87 年開始每年增加流感疫苗公費接種之對象 9

D.1 不同年度國際疾病分類號之對應位置及長度 17
D.2 不分年齡抽樣統計結果 29
D.3 6 歲以下抽樣統計結果 30
D.4 6 歲以上 65 歲以下抽樣統計結果 30
D.5 65 歲以上抽樣統計結果 30

E.1 程式相關指令 34

國立政治大學 National Chengchi University

圖目錄

D.1 流程圖一 14
D.2 流程圖二 14
D.3 逐年打疫苗人數 21
D.4 打疫苗分層人數比較: 綠色是 6 歲以下，藍色是 6 歲以上 65 歲以下，紫色是 65 歲以上 22
D.5 逐年流感就醫人數 22
D.6 感冒分層人數比較: 綠色是 6 歲以下，藍色是 6 歲以上 65 歲以下，紫色是 65 歲以上 23
D.7 6 歲以下施打疫苗後 3 個月內比較 23
D.8 6 歲以下施打疫苗後 3 個月內校正 23
D.9 6 歲以下施打疫苗後 6 個月內比較 24
D.10 6 歲以下施打疫苗後 6 個月內校正 24
D.11 6 歲以下施打疫苗後 9 個月內比較 24
D.12 6 歲以下施打疫苗後 9 個月內校正 24
D.13 6 歲以下施打疫苗後 12 個月內比較 24
D.14 6 歲以下施打疫苗後 12 個月內校正 24
D.15 6 歲以上 65 歲以下施打疫苗後 3 個月內比較 25
D.16 6 歲以上 65 歲以下施打疫苗後 3 個月內校正 25
D.17 6 歲以上 65 歲以下施打疫苗後 6 個月內比較 25
D.18 6 歲以上 65 歲以下施打疫苗後 6 個月內校正 25
D.19 6 歲以上 65 歲以下施打疫苗後 9 個月內比較 25

國立政治大學 National Chengchi University

D.20 6 歲以上 65 歲以下施打疫苗後 9 個月內校正 25
D.21 6 歲以上 65 歲以下施打疫苗後 12 個月內比較 26
D.22 6 歲以上 65 歲以下施打疫苗後 12 個月內校正 26
D.23 65 歲以上施打疫苗後 3 個月內比較 26
D.24 65 歲以上施打疫苗後 3 個月內校正 26
D.25 65 歲以上施打疫苗後 6 個月內比較 27
D.26 65 歲以上施打疫苗後 6 個月內校正 27
D.27 65 歲以上施打疫苗後 9 個月內比較 27
D.28 65 歲以上施打疫苗後 9 個月內校正 27
D.29 65 歲以上施打疫苗後 12 個月內比較 27
D.30 65 歲以上施打疫苗後 12 個月內校正 27
D.31 老人死亡率比較圖 28

國立政治大學
National Chengchi University

第一章

前言

本章以研究背景與目的和論文架構說明整篇論文的開始

第一節　研究背景與目的

流行性感冒 (以下簡稱流感) 在世界各地傳播快速，雖然大部分病症在一至二周內可以自行痊癒，但因為容易引起嚴重併發症，因此在流感的防疫上較為國人所關心。

根據世界衛生組織 (World Health Organization, WHO) 提供的資料，溫帶地區的流感季節為冬季，而熱帶地區則是一年四季皆會引發流感，且疫情不規則。而世界衛生組織每年都會預估當年的病毒株，以供世界各國參考及製作疫苗。但預估也可能出錯，如 2015 年初，世界衛生組織原本預估 A 型流感的病毒株是「德國株」，沒想到卻是「瑞士株」，造成香港近百人死亡。

在 20 世紀其間，發生過三次極為嚴重且有記錄的全球流感大流行：1918 年的西班牙、1958 年的亞洲流感、1968 年的香港流感，三起流行的死亡人數皆超過百萬人以上。而 21 世紀初，2009 年 6 月在墨西哥爆發的 A 型 H1N1 流感，經研究後發現為 A 型流感病毒之突變種，該病毒之遺傳因子結合了人類、鳥禽以及豬隻的流感病毒，世界衛生組織將該次疫情的全球流感警告級別提高到第六級的最高等級，死亡的病例超過一萬人；我國亦有數千名病例受到此波疫情傳染。

在預防流感的方面，我國還是以施打疫苗為主，本研究想探討的是，在政府大力宣導施打疫苗預防流感的情況下，流感疫苗是否真的能有效地控制流感疫情。

第二節　研究架構

本文分為六章，第一章為前言，說明本研究之背景目的以及本篇論文之架構。第二章為文獻探討，先簡介健保資料庫、流行性感冒及流感疫苗和我國施打疫苗的公共政策。第三章為研究方法，詳述資料處理流程及統計方法等。第四章為研究經驗與結果，分別闡述整理資料之方法及研究過程和結果。第五章為研究之結論及建議，說明本研究依據研究結果和討論，歸納出結論，並對政策後續之推展及未來之研究方向提出建議。

國立政治大學
National Chengchi University

第二章

文獻探討

本章是要告訴讀者什麼是健保資料庫、流行性感冒、流感疫苗、2002 年論文回顧和公共政策。健保資料庫的部分包含了醫事機構病床主檔 (BED)、醫事機構診療科別明細檔 (DETA)、醫事機構基本資料檔 (HOSB)、專科醫師證書主檔 (DOC)、醫事人員基本資料檔 (PER)、重大傷病證明明細檔 (HV)。流行性感冒簡介告訴讀者如何分別流感與一般感冒以及預防方法。論文回顧則是想探討我們這 15 年的資料是否與 2002 年的研究 [14] 有類似的結果。公共政策則是敘述每年開放公費疫苗施打的對象。

第一節　簡介健保資料庫

全民健康保險 (簡稱全民健保或健保) 是強制性保險，是一種社會福利政策。強制要求全民投保，藉由全民皆參與健保之方式，使得民眾就醫權益更有保障。各國醫療體系中，大多為自費，缺乏公營健康保險，民眾必須自行加入私人保險，缺乏像全民健保中這種大型之完整資料檔，因此健保資料庫為世界少有的數據資源。

健保資料庫為全民健保局於民國 87 年所建置，自民國 89 年起每年健保局將前一年的健保資料選取可供研究使用的檔案匯出，將身分欄位加密後，交由國衛院製作成「全民健康保險研究資料庫」及各加值資料檔案。學術界或是研究單位皆可向健保局提出申請，一般申請中，包括：基本資料檔、系統抽樣檔、特定主題分檔、承保抽樣歸人檔。[1]

基本資料檔中，包含：醫事機構病床主檔 (BED)、醫事機構診療科別明細檔

(DETA)、醫事機構基本資料檔 (HOSB)、專科醫師證書主檔 (DOC)、醫事人員基本資料檔 (PER)、重大傷病證明明細檔 (HV) 。

由於門診及住院資料數龐大，因此申請健保資料時以系統抽樣之方式，提供研究單位所需的資料。系統抽樣檔內包含住院醫療費用清單明細檔 (DD) 及住院醫療費用醫令清單明細檔 (DO) 系統抽樣為以月為單位，每 20 筆取 1 筆，抽樣比 1/20；而門診處方及治療明細檔 (CD) 及門診處方醫令明細檔 (OO) 系統抽樣以月為單位，每 500 筆取一筆，抽樣比 1/500。

若依不同主題擷取資料製成之加值資料分檔，則分為：中醫藥 (CM)、癌症 (CN)、糖尿病 (DB)、住院醫療費用清單明細檔 (DD)、牙醫 (DN)、特約藥局 (G)、重大傷病醫療資源使用 (HV)、承保檔 (ID)、外傷 (IN)、醫學中心 (MC)、論病例計酬 (NC)、職業病與職業傷害 (OC)、精神疾病住院病患歸人檔 (PSY)、交通意外事故 (TRA)、復健治療 (RH) 等。

而承保抽樣歸人檔以健保承保資料檔為抽樣母群體，依簡單隨機抽樣，決定樣本（人），依此擷取每人各年所有就醫申報資料 (包括門診、住院及特約藥局)。由中央健保局所提供的 2005 年承保資料檔以「身份證字號加上生日加上性別」歸人，可得 25,678,998 人之資料，作為資料母檔。將抽樣母群體 22,717,053 人賦予流水號，產生至少 100 萬個亂數，取與亂數值相同的流水號，來隨機抽取所需的保險對象樣本，接著剔除身份證字號重複者（共 64 個），再補抽至得到 100 萬人樣本為止。將隨機抽出的 100 萬人樣本，利用身份證字號 (已加密) 每 4 萬人為一組共 25 組，與健保資料庫串聯，擷取 1996-2006 年該 100 萬人在全民健保研究資料庫中所有就醫資料，即得 100 萬人承保抽樣歸人檔，日後每年更新，加入這 100 萬人樣本新一年度的就醫資料。[3]

本研究所關注的資料主要是系統抽樣檔中的 CD 檔。CD 檔中包含 FUNC _DATE(就醫日期)、ID _ BIRTHDAY(出生日期)、ID(身分證統一編號，已轉碼)、ACODE_ ICD_ 9_ 1(國際疾病分類號一)，ACODE_ ICD_ 9_ 2(國際疾病分類號二)、ACODE_ ICD_ 9_ 3(國際疾病分類號三)、ID_ SEX(性別) 等資訊，從這些資訊中我們可以知道病患所患的疾病，與疾病發生的時間。

第二節　流行性感冒

通常大家都容易將流行性感冒和一般的感冒搞混，因此本節先定義流行性感冒，接著簡介流行性感冒和說明流感的預防方式

一、 名詞定義

一般人對於流行性感冒和一般感冒有認知上的迷思，分不清楚流行性感冒和一般感冒的差異，並將症狀搞混，因此我們先為讀者解釋流行性感冒和一般感冒的不同。

一般感冒多為輕微的病毒或是細菌感染所引起的，雖然和流感一樣會出現流鼻涕跟咳嗽等現象，但少有發燒、頭痛和全身痠痛或無力的症狀，而流行性感冒則是由流感病毒所引起的，傳染快速且容易引發地區性的流行，症狀比一般感冒嚴重且發病突然，通常會在 24 小時之內出現發冷及高燒的症狀，並且會有全身嚴重乏力、眼球漲痛、腰背及四肢痠痛等症狀。[13]

由衛生福利部疾病管制署 [10] 網站提供的資料，我們可以整理成以下表格讓讀者清楚的知道一般感冒和流行性感冒的異同。

國立政治大學 National Chengchi University

表 B.1 流行性感冒與一般感冒比較表

項目	流行性感冒 (Influenza)	一般感冒 (Common Cold)
病原體	流感病毒	大約有 200 多種病毒可引起，常見的有鼻病毒、呼吸道融合病毒、腺病毒等
影響範圍	全身性	呼吸道局部症狀
發病速度	突發性	突發/漸進性
臨床症狀	喉嚨痛、倦怠、肌酸痛	喉嚨痛、噴嚏、鼻塞
發燒	高燒 3-4 天	少發燒，僅體溫些微升高
病程	1-2 週	約 2-5 天
併發症	可能併發肺炎、心肌炎、腦炎、神經症狀 (雷氏症候群) 等	少見 (中耳炎或其他)
傳染途徑	飛沫傳染；接觸傳染	飛沫傳染；接觸傳染
傳染性	高傳染性	傳染性不一

二、 流行性感冒簡介

流行性感冒，簡稱流感，是一種由病毒 (Influenza Virus) 引起的呼吸道感染。流感病毒屬於正黏液病毒科 (Orthomyxoviridae)，可依表面蛋白與核蛋白分為 A、B、C 三型。A 型流感可感染多物種，包括人類、禽鳥、候鳥、豬、馬、貓、狗等生物，導致全球或地方流行；B 型通常僅感染人類，臨床表現比 A 型流感輕微，但可造成地方流行；C 型僅偶發性感染人類，造成輕微上呼吸道症狀。[7,8]。

感染人類之流感病毒最早於 1933 年由英國人威爾遜·史密斯（WilsonSmith）自流感病患中分離發現 [15,16,19]，是一種會藉由飛沫或接觸傳播，並造成人類感染後產生急性呼吸道感染症狀的病毒，而當時所分離出的流感病毒係為 A 型流感病毒 (H1N1)。一直到 1945 年才產生了第一支抗流感的疫苗。

流感病毒在世界各地常引起週期性且規模大小不一的流行，自 1977 年開始，A 型 (H1N1 與 H3N2) 及 B 型流感病毒不斷循環出現在人類的季節性流感流行之中。[10]

由於流感是經由飛沫傳播的呼吸道疾病，爆發流行快速、散播範圍廣泛以及併發症嚴重，是公共衛生防疫的隱憂之一。流感可能引起的併發症，包括細菌性及病毒性肺炎、心肌炎、腦炎、雷氏症候群等，甚至導致死亡，常見於嬰幼兒及老年人。[6]

三、 流感預防

目前我國主要預防流感的方式仍然是以施打疫苗為主。若以非疫苗的方式預防流感則是多注意個人衛生、發燒篩檢、自我健康管理等，減少流感傳染機會。在衛生教育上也可以加強呼籲民眾施打疫苗和患者及早就醫和在家多休息，避免成為傳播病毒者等訊息。

第三節　流感疫苗

流感疫苗，是針對流行性感冒的疫苗。因為病毒變化的速度很快，每年會發展新的流感疫苗兩次 (南半球跟北半球)。大部分狀況下，疫苗有很好的保護力，一般而言，流感疫苗的保護效力大約在 60-90% 之間，免疫系統較弱者的效果稍差。[12,22]

目前國內使用的流感疫苗是利用雞胚培養病毒，加以去活化以後製成的疫苗。但是這種疫苗的保護效力大約只能維持一至二年，而且流感病毒常常發生突變，所以每年都必須接種一劑疫苗。每年世界衛生組織與美國疾病管制中心都會根據前幾年在全世界收集的資料，推測次年可能流行的病毒亞型，進而建議次年流感季節所用的疫苗成份。最近幾年，全世界流行的病毒株都包括了 H1N1 與 H3N2 兩種亞型的 A 型流感與 B 型流感病毒。[4]

每年流感疫苗製造廠會根據世界衛生組織所公布的流感病毒製做疫苗，但 WHO 主要預估的是歐美地區流行的流感病毒，所以在亞洲地區所流行的的病毒可能會與 WHO 預測的不同，進而爆發流感大流行，有時也可能因為 WHO 預測錯誤或是病毒發生變異造成大流行。像是 1918 年的「西班牙流感」、1957 年的「亞洲流感」以及 1968 年的「香港流感」都是由於以上原因造成非常多人死亡。[6]

人類使用流感疫苗為已有數十年的歷史，目前每一年所使用的流感疫苗可達好幾億劑。文獻上顯示流感疫苗接種比抗病毒藥劑的使用，更具成本效益。[17,18]

近年來公費疫苗共有 3 家廠牌，疫苗於原製造國取得上市許可證外，且均需符合我國衛生福利部食品藥物管理署查驗登記規定。疫苗配送採「先到貨、先鋪貨、先使用」原則，民眾接種疫苗之廠牌將依對象採「隨機」安排方式，無法指定廠牌，各廠牌適用年齡如下表：[11]

表 B.2

持有許可證廠商 / 品名	劑型 (毫升)	適用年齡
國光生物科技股份有限公司 / AdimFlu-S “安定伏” 裂解型流感疫苗	0.25	滿 6 個月以上，3 歲以下使用
	0.5	3 歲以上使用
台灣諾華股份有限公司 / AGRIPPAL S1 安爾保流感疫苗	0.5	3 歲以上使用
賽諾菲股份有限公司 / VAXIGRIP 巴斯德流感疫苗	0.25	滿 6 個月以上，3 歲以下使用
	0.5	3 歲以上使用

第四節　2002 年論文回顧

在” 流行性感冒疫苗施打之成效評估與將來施打方針” 的論文中 [14]，研究者以台南的四間醫院做隨機抽樣臨床實驗，對象為 2000 年 7 月 1 日至 2001 年 6 月 30 日住院且年齡介於 50 至 64 歲之間的病人，隨機分配分為施打組與對照組，於 2001 年末至 2002 年初，在各合作醫院施打疫苗。

該研究的樣本為 1600 人，直接將人分成一半實驗組一半對照組，平均年齡也相同，在施打完疫苗後的三、六、九個月以電話問卷進行追蹤，而 9 個月後的追蹤結果，死亡的部分施打組死亡率約為 1.5%，對照組則為 3.6%，門診使用情形則是施打組為 2.13 次/人月，對照組為 1.98 次/人月，對照組的死亡率高於施打組許多。

然而該研究的樣本只有一年且只有特定的地方特定醫院的人，因此我們想知道，如果以我們現有的樣本 (15 年，每年 100 萬人) 用一樣的方法研究，並且以 0 到 6 歲、6 到 65 歲及 65 歲以上將現有樣本分成三層研究，是否會與該研究得到一樣的結論以做施打疫苗之建議。

第五節　公共政策

我國在流感疫苗的公共政策是，於民國 87 年試辦「65 歲以上高危險群老人流感疫苗接種先驅計畫」，並依衛生福利部傳染病防治諮詢會預防接種組（ACIP）建議逐年擴大實施對象，90 年起，開放所有 65 歲以上老人公費接種；92 年度，將醫療機構之醫護等工作人員、衛生等單位之防疫相關人員、禽畜（雞、鴨、鵝、豬、火雞、鴕鳥）養殖等相關人員，納入公費接種對象；93 年度，增加 6 個月以上 2 歲以下之幼兒；96 年度，增加國小一、二年級學童及衛生保健志工；97 年度，增加重大傷病患、2-3 歲幼兒及國小三、四年級學童；98 年度，增加 3 歲至國小入學前之幼兒及實施空中救護勤務人員；101 年度，增加國小五、六年級學童；102 年新增 60-64 歲具高風險慢性病患；103 年度新增孕婦及 50-59 歲具高風險慢性病患。[9]

為使疫苗保護效力能持續至隔年 2、3 月之流行期，本計畫對象預定實施期程自 104 年 10 月 1 日起開始辦理接種，另將視疫苗結餘狀況，自 105 年 1 月 1 日起與候補對象共同接種至疫苗用罄止。[9]

表 B.3　從 87 年開始每年增加流感疫苗公費接種之對象

年份	開放公費疫苗施打者
87	試辦「65 歲以上高危險群老人流感疫苗接種先驅計畫」
90	所有 65 歲以上老人
92	醫療機構之醫護等工作人員、衛生等單位之防疫相關人員、禽畜養殖等相關人員
93	6 個月以上 2 歲以下之幼兒
96	國小一、二年級學童和衛生保健志工
97	重大傷病患、2-3 歲幼兒和國小三、四年級學童
98	3 歲至國小入學前之幼兒及實施空中救護勤務人員
101	國小五、六年級學童
102	60-64 歲具高風險慢性病患
103	孕婦及 50-59 歲具高風險慢性病患

第三章

研究方法

本章將告訴讀者在健保資料庫龐大的資料下如何整理出我們要的資料，以及我們分析的方法。分析方法分為兩種，第一種分析方法比較有施打疫苗者和沒施打疫苗者的平均流感就醫紀錄來判定疫苗的有效性，第二種方法使用 Bootstrapping 方法重新抽樣來假設檢定判別疫苗的有效性。

第一節　資料整理

研究者現有的資料為 1998 年至 2012 年的健保資料庫中的所有資訊，從健保資料庫的譯碼簿中，我們可以得到需要的資訊，包含身分證字號 (已轉碼)、出生年月、就醫日期、疾病分類號等等的資訊，因此我們需要整理這些資訊，並且找出需要的。

由於健保資料庫裡的資料量非常龐大，若是將所有的資料都放在同一資料夾裡，對於管理上是非常不便的，因此健保局將每一年的資料分成 25 個資料夾存放，每個資料夾內是 4 萬筆的資料，但我們為了要方便使用資料，因此第一步要做的變是將所有資料放進同一資料夾裡，以便我們的研究。

接著根據健保局提供的 ICD9 疾病分類代碼，我們可以知道流行性感冒疫苗的 ICD9 碼為 V048，以及流行性感冒的 ICD9 碼為 487。

有了以上的資訊之後，我們就可以開始設計方法以進行分析。

第二節　分析方法

以下為研究者設計不同的分析方法，使用的樣本以及考慮的變因都不盡相同，也預想會有不盡相同的結果。

一、 方法一

第一種方法我們使用全部健保資料庫裡的人做為樣本，不刪去所有外在的變因，直接將所有資料做分類，計算出結果後進行分析。

每個人會被分在有施打流感疫苗且有流感就醫紀錄，或有打流感疫苗但沒有流感就醫紀錄，或沒有打流感疫苗且有流感就醫紀錄，或是沒有打流感疫苗沒有流感就醫紀錄這四群人中的一群。

接著，我們可以用分類後的資料，算出有打流感疫苗者的平均流感就醫次數，以及不同年齡層中有打流感疫苗者的平均流感就醫次數，畫出折線圖並做比較。

我們將資料整合並分成四群之後，要依年齡在每一群中再分三層，由於 CD 檔中沒有年齡的資訊，因此我們用就醫日期以及出生年月，來算出就醫當時的年齡，以作為分層的依據。

分層後再去觀察打完疫苗後以 3 個月為一個單位的就醫平均次數變化，也就是說，打完疫苗後的 3 個月內、6 個月內、9 個月內和 12 個月內的平均就醫次數，未施打疫苗者則是以每年的 10 月 1 日為計算起始日 (由於流感是冬季容易開始流行，政府通常以 10 月 1 日為開放公費疫苗施打的起始日，故選擇 10 月 1 日為參考起始日)。

這個方法是用流感發生的機率來判別流感疫苗有效或是無效，如果有打流感疫苗者的平均流感就醫次數較低，則判定為流感疫苗對於流感的預防是有效的，反之則無效。而分層後的資料也以此做為判斷的依據。

最後，根據前一章提到的死亡率問題，我們使用重大傷病檔 (HV 檔) 來找出 65 歲以上老人的死亡率。

從 ICD9 譯碼簿中，我們知道在 HV 檔中的第 293 個位置是”Y” 或”M” 的話，就表示此人為死亡的狀態，再根據第 33 到 37 個位置的診斷代碼，找出因呼吸道疾病 (ICD9 碼

480 至 487) 死亡的人，我們再對照 CD 檔的 ID，就可以知道有打疫苗老人的死亡率和沒打疫苗老人的死亡率。

二、 方法二

方法二的發想是因為在前一個研究方法中沒有加入統計分布的部分在驗證的過程之中，我們覺得會因此而降低了結果的說服力，故想了一個新的方法，告訴讀者如何使用假設檢定來驗證疫苗的有效性，結果也較具有說服力。

首先，方法二的樣本選取方法是在有打流感疫苗的人之中先隨機抽取 70% 的人當作實驗組樣本，接著再在沒有打流感疫苗的人之中也隨機抽取 70% 的人當做對照組，各抽 64 次，分別算出有因流感就醫的比例後用兩者的相關係數來訂定虛無假設和對立假設。

寫成數學式子的樣子如下:

X_i: 第 i 次隨機抽取之實驗組的流感就醫比例，i= 1, $\cdots$,64

Y_i: 第 i 次隨機抽取之對照組的流感就醫比例，i= 1, $\cdots$,64

r_i 是 X_i 和 Y_i 的相關係數 i= 1, $\cdots$,64，因此 r_i 會形成一個分布，而且 $\bar{r} = \dfrac{\sum_{i=1}^{64} r_i}{64}$。

接著我們開始使用假設檢定來驗證流感疫苗是否在流感防疫上有效，首先第一步是設定虛無假設及對立假設，因為我們假設流感疫苗對預防有感是有效的，因此我們的虛無假設定的是疫苗有效，對立假設則是定為無效。

所謂的疫苗有效無效，在此方法中我們是以 $\bar{r}$ 判斷，如果 $\bar{r} < 0$ 表示疫苗有效，$\bar{r} \geq 0$ 則無效，也就是說 $H_0 : \bar{r} < 0$，$H_1 : \bar{r} \geq 0$。

下一步是選擇檢驗統計量 T 來決定哪一個檢測是合適的，並在虛無假設下推導檢驗統計量的分布，再選擇一個顯著水平 α(通常是 5% 或者 1%)，針對檢驗統計量 T，根據樣本計算 p-value。

若 p-value< α，接受虛無假設，也就是疫苗有效。若 p-value> α，拒絕虛無假設，接受對立假設，也就是疫苗無效。

第四章

研究經驗與結果

在本章中我們將展示研究過程中如何以 Python 整理健保資料庫中如此大量的資料，和如何在這些資料中找出研究需要的資訊，找到後要怎麼使用、整理以及跑出結果。依據程式跑出的結果以 Matlab 畫出圖形，呈現研究的結果。

我們整理資料的流程是先將 CD 檔分成有施打流感疫苗的人和沒有施打流感疫苗的人，在分成有流感就醫紀錄的人和沒有由趕就醫紀錄的人，接著將兩群有流感就醫紀錄的人分成 0 到 6 歲、6 到 65 歲和 65 歲以上三層，接著就可以使用我們上一章的分析方法計算出結果。

整理過程如流程圖一

國立政治大學 National Chengchi University

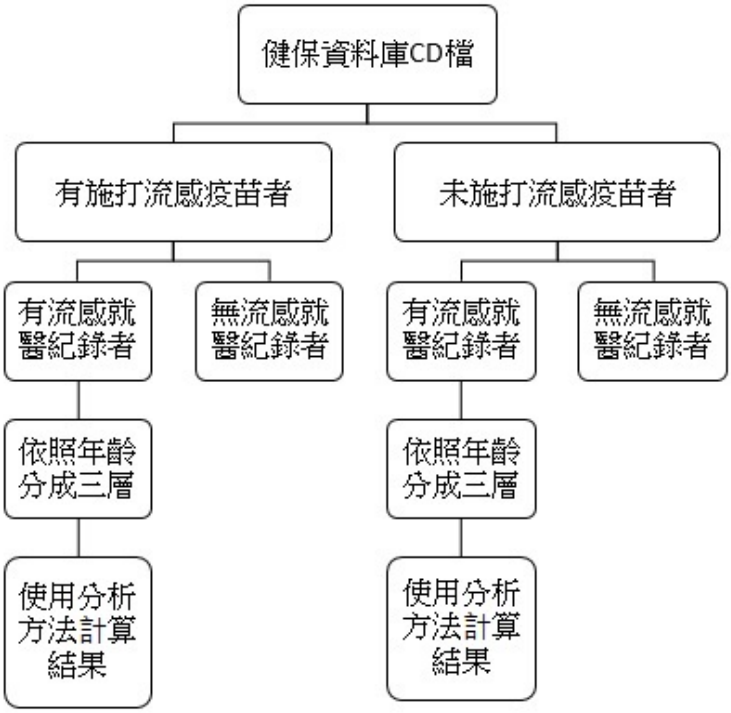


圖 D.1　流程圖一

而死亡率的計算則是把 CD 檔需要的資料與 HV 檔資料整合，接著根據譯碼簿找出死亡紀錄，然後算出死亡率，過程如流程圖二。

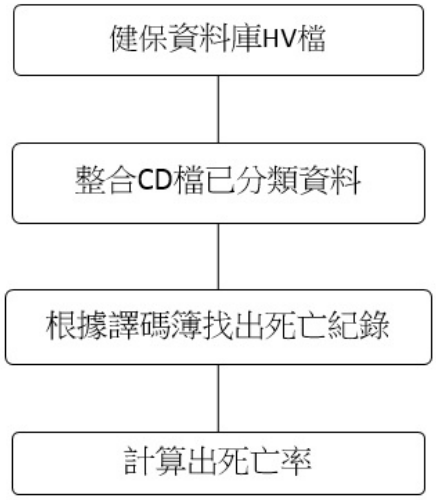


圖 D.2　流程圖二

第一節　實驗步驟

本小節我們將介紹如何利用 Python 程式語言做到整理我們所需要的資料，提供我們設定的方法論來驗證我們的結果。我們將配合程式的呈現，詳細地為讀者說明，方便未來有興趣利用 Python 程式作為健保資料處理工具的人使用。

一、 整合所有 ID

由於 Python 的不同型別在運算上會使用的時間差異會很大，所以，如何慎選資料型別是利用 Python 處理大型數據很重要的藝術。若使用 List 型別來處理，搜尋和取用的速度會比 Set 型別來的緩慢許多。因此，我們先利用 Set 型別把所有 100 萬人資料的 ID 放入集合中，這樣作使得我們將來在整理資料上，會得到較快的速度。

由於健保資料庫的原始檔是將每年 100 萬人的資料分成 25 個資料夾個別存放，因此首先要將所有的 ID 放進同一個資料夾，以利後續使用。還好這些資料夾的命名都有一個以數字為引導的規則，例如，1998 年的 CD 檔的第一個完整路徑如下: /home/ym/NHI/Cohort/R201_ 1998/R201_ CD_ 1998.DAT。故我們仍舊能使用一支程式碼就做到把所有資料整理出來的動作。以下的程式碼是介紹 Python 如何在不同的資料夾下整理資料的方法。

```
import pickle
#引入pickle套件
All = set()
#將All建立為集合型態的物件
for i in range(1998,2013):
#以range方式選擇第n項至第m項的選擇方式為range(n,m+1)
#因此1998~2012年的選擇方式為range(1998,2013)
    for j in range(1,26):
    #同上列原因，1~25個資料夾的選擇為range(1,26)
     if j < 10:
      pathname = '/home/ym/NHI/Cohort/R20'+ str(j) +'_'+str(i)
```

```
        filename = 'R20' + str(j) + '_CD' +str(i) + '.DAT'
       #資料夾因為方便管理，將所有資料名稱的長度設定成一樣，
       #因此1就變為01，一直到9變成09，而程式不會幫我們自動將
       #個位數前面補0，因此要將小於10的資料夾和其他分開寫
      else:
        pathname = '/home/ym/NHI/Cohort/R2' + str(j)+'_'+str(i)
        filename = 'R2'+ str(j)+'_CD'+str(i)+'.DAT'
Filename = pathname+'/'+filename
#將pathname和filename組合成完整檔名並更名為Filename
print 'processing ', i, '-year and ',j,'-th file'
#輸出第幾年的第幾個資料已經完成
for line in open(Filename,'r'):
#讀取Filename裡的資料
All.add((line[91:123],line[83:91]))
#將資料中第92~123個位置的資料及第84~91個位置的資料加進All裡面
#若是以line的方式選取第n個位置到第m個位置的資料是用line[n-1,m]
fid = open('AllID.pkl','w')
#開啟一個叫做AllID的pickle檔並且寫入
pickle.dump(All,fid)
#將All裡面的物件序列化並儲存於fid裡
fid.close()
#在不用fid這個資料夾時用close關閉
```

二、 工具程式

1. 根據前一章方法論的設定，我們需要知道每個人的年齡來做為分層的依據，雖然在健保資料庫中的 CD 檔沒有直接告訴我們每個人年紀的資訊，但 CD 檔中有出生年月以及就醫時間的資訊，因此我們可以利用這兩項資訊寫一個函數算出在就醫時或是打疫苗時的年紀。

2. 在對照 ICD9 碼的譯碼簿時我們可以知道不一樣的年份中國際疾病分類號的位置及長度有所不同 (如表 D.1)，因此我們也需要一個函數可以因應不同的年份對照到該年應有的位置及長度。

因為以上兩點需求在之後的程式中都會同時用到，所以我們將兩個函數寫在同一個程式 (NHI_ toolbox) 裡，在接下來的程式中只要引入此工具 (from NHI_ toolbox import *) 就可以使用此工具裡的函數。

表 D.1 不同年度國際疾病分類號之對應位置及長度

	1997 年至 2003 年			2004 年至 2011 年			2012 年以後		
	長度	起始位置	迄末位置	長度	起始位置	迄末位置	長度	起始位置	迄末位置
國際疾病分類號一	5	131	135	5	132	136	15	132	146
國際疾病分類號二	5	136	140	5	137	141	15	147	161
國際疾病分類號三	5	141	145	5	142	146	15	162	176

以下為 NHI_ toolbox 的程式碼

```
def Age(Birthday, Current):
  year = int(Current[0:4])-int(Birthday[0:4])
  month = int(Current[4:6])-int(Birthday[4:6])
  if month < 0:
    year -= 1
    month += 12
  return (year,month)
#以上為計算年紀之工具
def ICD9_YEAR(yr):
  if (yr>=1997) and (yr<=2003):
    return (130,135), (135,140), (140,145)
  elif (yr>=2004) and (yr<=2011):
```

```
    return (131,136), (136,141), (141,146)
  else:
    return (131,146), (146,161), (161,176)
#以上為對應年分之國際及病分類號ICD9碼位置
```

第二節　研究方法一之經驗與結果

研究方法一是用所有的人當作樣本，找出有施打疫苗者及未施打疫苗者的平均流感就醫次數做為判定疫苗的有效性。本小節將把研究方法一的實際操作程式碼以及結果展示給讀者。

一、 研究過程

前面整合了 ID，也有了工具的程式，接著我們就可以開始將所有的人分類，首先是找出所有有打流感疫苗的人 (find_ flu_ shot.py)，再找出有因為流感而就醫的人 (flu_ seak.py)，然後依年齡分層 (population.py)，最後算出結果 (people_ state.py) 並以 Matlab 畫出圖形。

首先我們要依照年分選擇國際疾病代碼的位置，因此我們使用前一個工具程式的函數來找出 icd9 碼對應位置:

```
for j in range(1998,2013):
  icd1 = ICD9_YEAR(j)[0][0] #輸入年份後輸出的第一個位置
  icd2 = ICD9_YEAR(j)[1][0] #輸入年份後輸出的第二個位置
  icd3 = ICD9_YEAR(j)[2][0] #輸入年份後輸出的第三個位置
```

接著使用流感疫苗的 icd9 碼 V048 找出有打疫苗的人

```
for line in open(fullname,'r'):
      ID = line[91:123] #身分證碼位置
      Date = line[67:75] #就醫日期位置
      Birthday = line[83:91] #出生年月位置
```

```
 if (line[icd1:icd1+4]=='V048') or
    (line[icd2:icd2+4]=='V048') or
    (line[icd3:icd3+4]=='V048'):
 #在對應的第一個或第二個或第三個位置有出現V048
age = Age(Birthday,Date) #算出年齡
if(not(ID in D)):
  D[ID] = list()
D[ID].append([Date,'V048',age])
#將[就醫日期,V048,年齡]加入D[ID]裡
```

用以上程式碼找出有打流感疫苗的人之後，接著就是要將這些資料存檔，讓些下來的程式可以使用我們分類好的資料。

```
filename = 'flu_shot_'+str(j)+'.pkl'
#讓filename是以年為單位的.pkl檔
File = open(filename,'w') #開啟一個寫入模式的檔案
pickle.dump(D,File) #將D寫入File裡
```

以上找出有打疫苗者的詳細程式碼為附錄 C 中的 find_ flu_ shot.py

接著利用流感就醫代碼 487，找出有就醫紀錄的人

```
flu_Set = set(['487'])
    for line in open(fullname,'r'):
      ID = line[91:123]
      Date = line[67:75]
      Birthday = line[83:91]
      seak_set = set() #開一個集合
      seak_set.add(line[icd1:icd1+3]) #將第icd1到icd1+2的位置加到集合裡
      seak_set.add(line[icd2:icd2+3]) #將第icd2到icd2+2的位置加到集合裡
      seak_set.add(line[icd3:icd3+3]) #將第icd3到icd3+2的位置加到集合裡
      if len(seak_set.intersection(flu_Set))>=1:
      #seak_set和flu_Set的交集大於等於一，也就是找出有就醫紀錄的人
```

```
        age = Age(Birthday,Date)
        if(not(ID in D)):
          D[ID] = list()
        D[ID].append((Date,age)) #將(就醫日期,年齡)加入D[ID]裡
```

和前面一樣，我們把之後會用到的資料儲存下來，一樣以年為單位，名稱為 flu_ seak_ 年.pkl。(詳細程式請見附錄 C 的 flu_ seak.py)

再來就是年齡的分層

```
for i in range(1998,2013):
  filename = 'flu_shot_'+str(i)+'.pkl'
  fid = open(filename,'r')
  A = pickle.load(fid) #將fid裡的資料載入A裡
```

以上程式碼是將每年有打流感疫苗的資料載入 A 裡面

```
  ID = A.keys()
  c = set() #開一個名為c的集合
  a = set() #開一個名為a的集合
  o = set() #開一個名為o的集合
  for idx in ID:
    if A[idx][0][2][0]>=65: #A[idx][0][2][0]代表年齡
      o.add(idx) #將年齡大於65的idx加到o集合裡
    elif A[idx][0][2][0]<6:
      c.add(idx) #將年齡小於6的idx加到c集合裡
    else:
      a.add(idx) #將剩下的人的idx加到a集合裡
```

以上程式是將有打流感疫苗的人做年齡上的分類，同樣的方式可以代換成流感就醫紀錄的人，詳細的程式碼請見附錄 C 的 population.py

接著就是算出我們要的數據 (詳見附錄 C 的 people_ state.py)

最後是死亡率的計算，我們先使用 find_ dead_ flu.py 找出因為流感而死亡的人，接著使

用 dead_ rate.py 算出死亡率。

二、 研究結果

我們從 people_ state.py 跑出的結果，以年份做為橫軸，人數做為縱軸畫出圖 D.3、D.4、D.5、D.6，分別為每年打疫苗的總人數、打疫苗總人數分層、每年流感的就醫人數、就醫人數分層。

由圖 D.3 可以看到從 2003 年以後，施打疫苗的人數大致上是下降的，但 2009 年打疫苗的人數快速的上升，推測是因為當年流行的 H1N1 造成過多人死亡，人心惶惶，因此從圖 D.4 中可以看出 6 歲以上 65 歲以下的成人願意自費施打疫苗的人快速的增加。

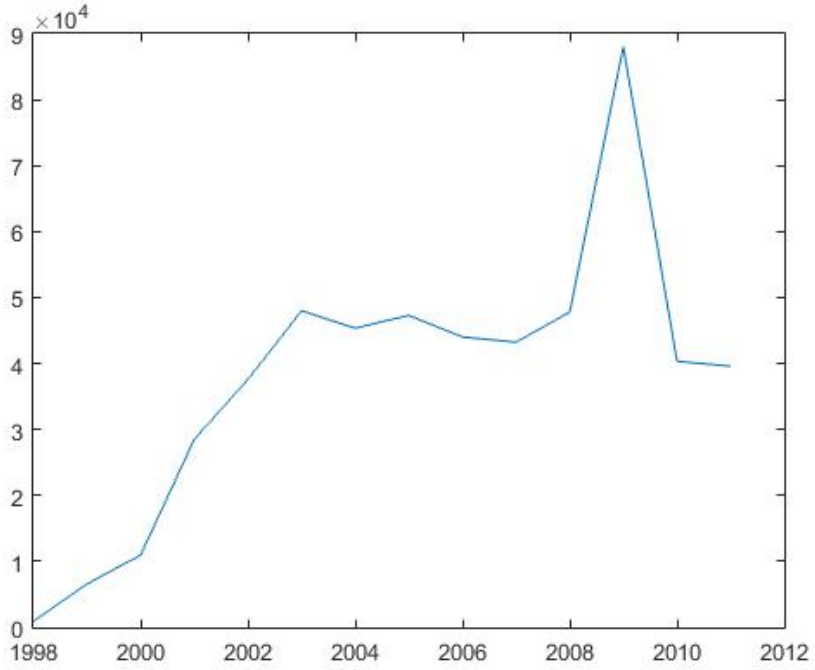


圖 D.3 逐年打疫苗人數

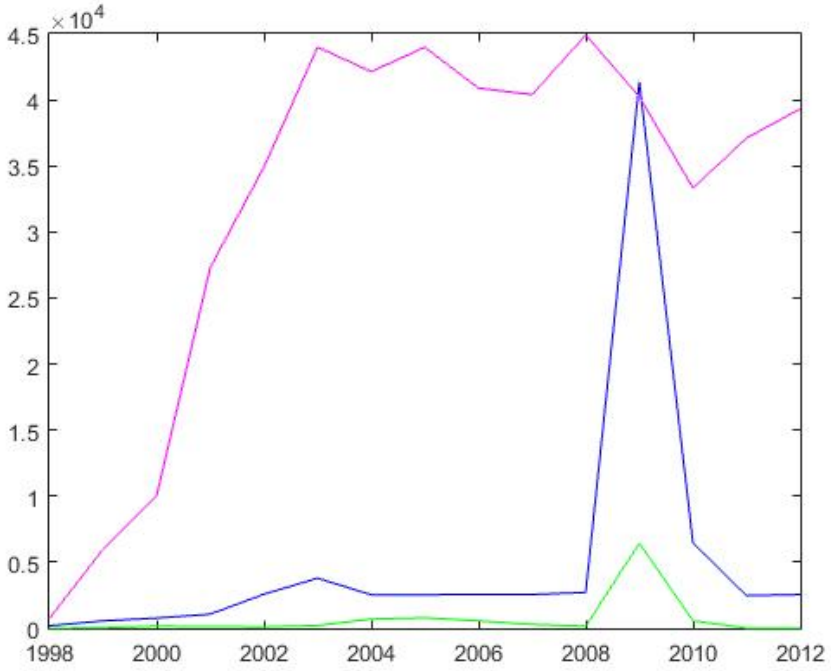


圖 D.4 打疫苗分層人數比較: 綠色是 6 歲以下，藍色是 6 歲以上 65 歲以下，紫色是 65 歲以上

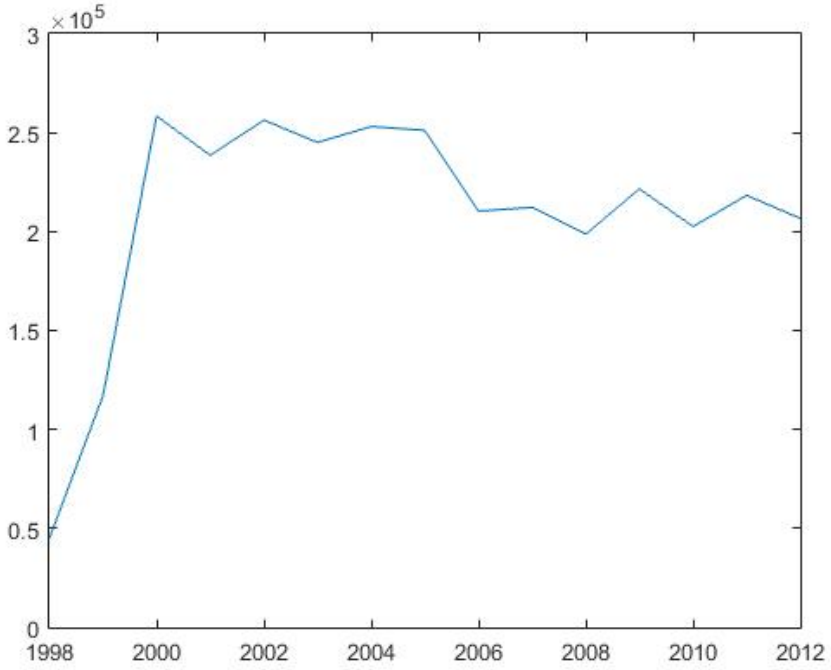


圖 D.5 逐年流感就醫人數

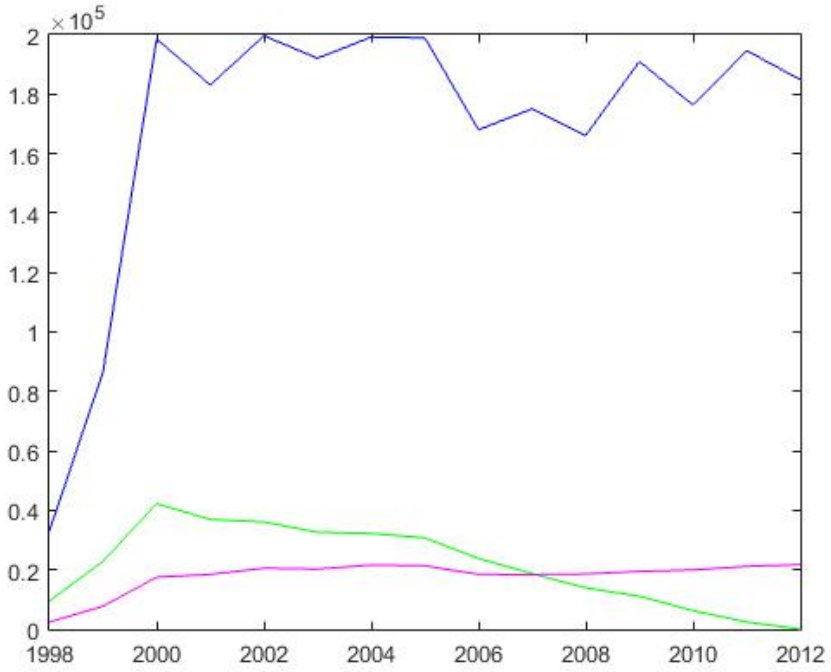
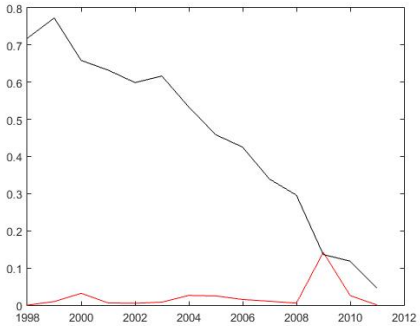


圖 D.6 感冒分層人數比較: 綠色是 6 歲以下，藍色是 6 歲以上 65 歲以下，紫色是 65 歲以上

(一) 6 歲以下結果比較

從 people_ state.py 得到的 6 歲以下結果，我們可以畫成以下圖形 (圖 D.7，圖 D.9，圖 D.11，圖 D.13)，一樣以年份做為橫軸，縱軸則是變成次數，圖中的紅線代表有施打流感疫苗者的流感平均就醫次數，黑線則是代表沒有施打流感疫苗者的流感平均就醫次數，但一般來說，會設定沒有施打疫苗者的流感平均就醫次數為 1，因此我們校正黑線為 $y = 1$，校正的方法是將有施打流感疫苗者的流感平均就醫次數及沒有施打流感疫苗者的流感平均就醫次數分別除以沒有施打流感疫苗者的流感平均就醫次數，結果便是圖 D.8，圖 D.10，圖 D.12，圖 D.14，而校正後的紅線之回歸線斜率為 0.027



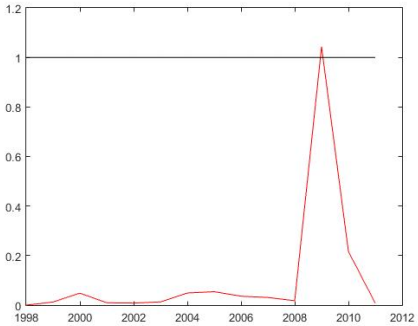
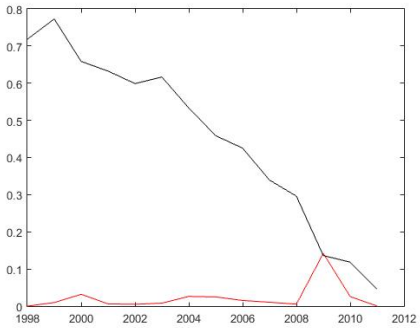


圖 D.7 6 歲以下施打疫苗後 3 個月內比較 **圖 D.8** 6 歲以下施打疫苗後 3 個月內校正



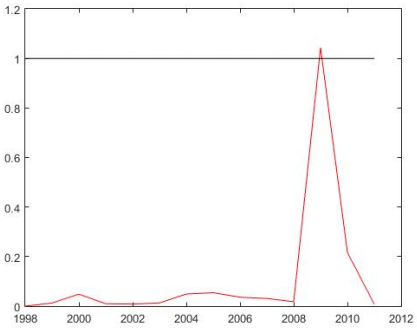
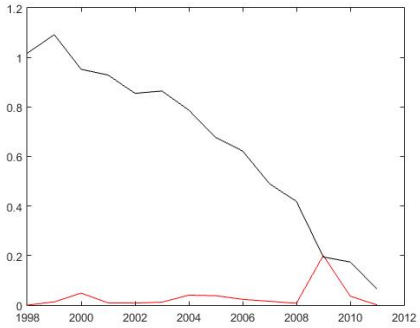


圖 D.9 6 歲以下施打疫苗後 6 個月內比較 **圖 D.10** 6 歲以下施打疫苗後 6 個月內校正



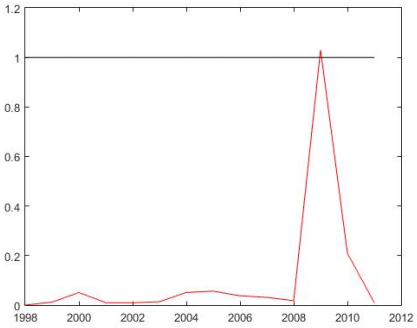
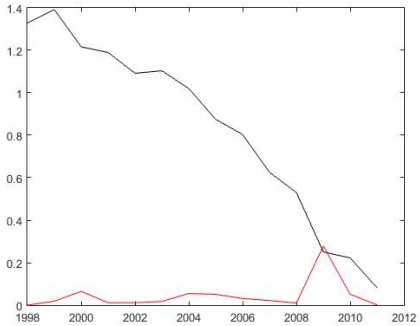


圖 D.11 6 歲以下施打疫苗後 9 個月內比較**圖 D.12** 6 歲以下施打疫苗後 9 個月內校正



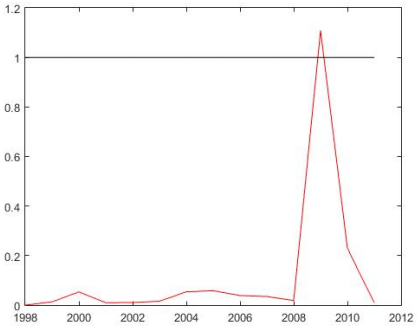


圖 D.13 6 歲以下施打疫苗後 12 個月內比較 **圖 D.14** 6 歲以下施打疫苗後 12 個月內校正

(二) 6 歲以上 65 歲以下結果比較

從 people_ state.py 得到的 6 歲以上 65 歲以下結果，我們可以畫成以下圖形 (圖 D. 15，圖 D.17，圖 D.19，圖 D.21)，一樣以年份做為橫軸，次數作為縱軸，圖中的紅線代表有施打流感疫苗者的流感平均就醫次數，黑線則是代表沒有施打流感疫苗者的流感平均就醫次數，和前面一樣，我們設定沒有施打疫苗者的流感平均就醫次數為 1，因

此我們校正黑線為 $y=1$，校正的方法是將有施打流感疫苗者的流感平均就醫次數及沒有施打流感疫苗者的流感平均就醫次數分別除以沒有施打流感疫苗者的流感平均就醫次數，結果便是圖 D.16，圖 D.18，圖 D.20，圖 D.22，而校正後的紅線之回歸線斜率為 0.038

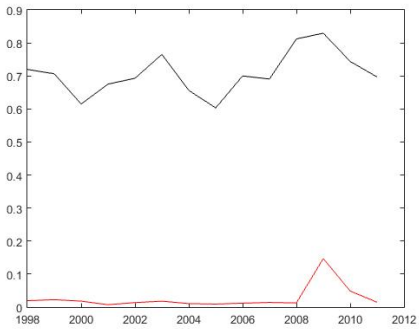


圖 D.15 6 歲以上 65 歲以下施打疫苗後 3 個月內比較

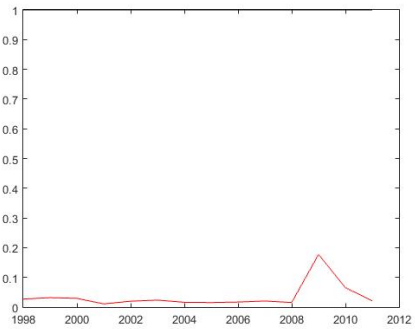


圖 D.16 6 歲以上 65 歲以下施打疫苗後 3 個月內校正

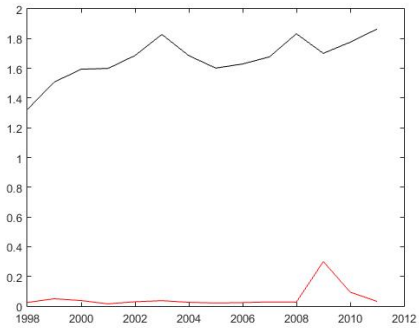


圖 D.17 6 歲以上 65 歲以下施打疫苗後 6 個月內比較

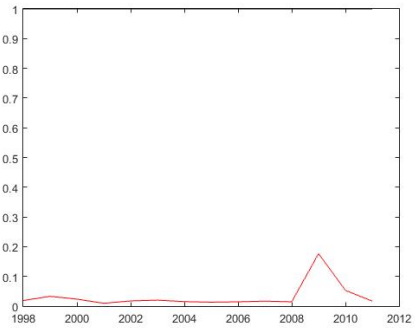


圖 D.18 6 歲以上 65 歲以下施打疫苗後 6 個月內校正

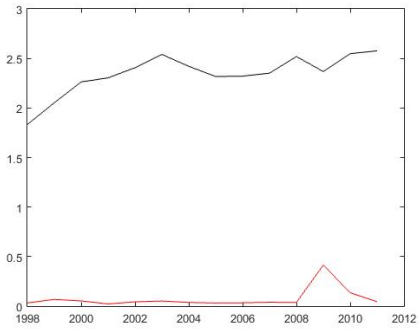


圖 D.19 6 歲以上 65 歲以下施打疫苗後 9 個月內比較

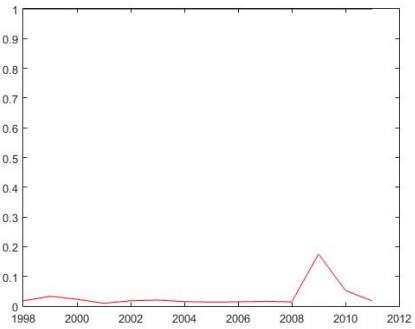


圖 D.20 6 歲以上 65 歲以下施打疫苗後 9 個月內校正

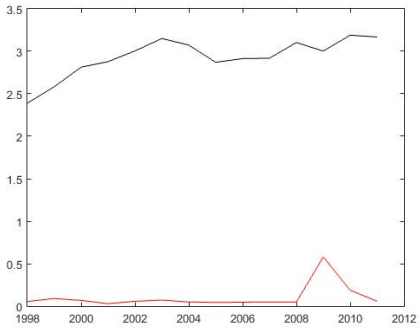


圖 D.21 6 歲以上 65 歲以下施打疫苗後 12 個月內比較

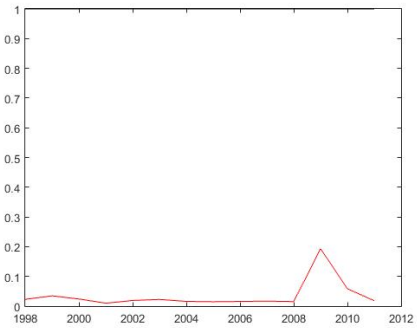


圖 D.22 6 歲以上 65 歲以下施打疫苗後 12 個月內校正

(三) 65 歲以上結果比較

從 people_ state.py 得到的 65 歲以上結果，我們可以畫成以下圖形 (圖 D.23，圖 D.25，圖 D.27，圖 D.29)，一樣以年份做為橫軸，次數作為縱軸，圖中的紅線代表有施打流感疫苗者的流感平均就醫次數，黑線則是代表沒有施打流感疫苗者的流感平均就醫次數，和前面一樣，我們設定沒有施打疫苗者的流感平均就醫次數為 1，因此我們校正黑線為 $y = 1$，校正的方法是將有施打流感疫苗者的流感平均就醫次數及沒有施打流感疫苗者的流感平均就醫次數分別除以沒有施打流感疫苗者的流感平均就醫次數，結果便是圖 D.24，圖 D.26，圖 D.28，圖 D.30，而校正後的紅線之回歸線斜率為-0.1389

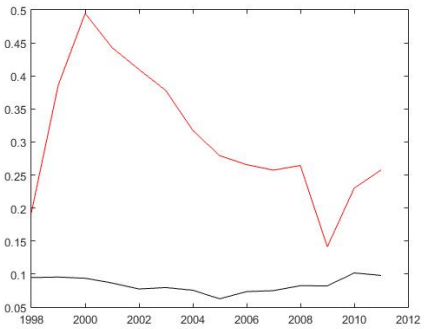


圖 D.23 65 歲以上施打疫苗後 3 個月內比較

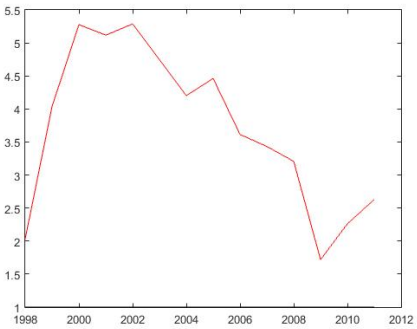


圖 D.24 65 歲以上施打疫苗後 3 個月內校正

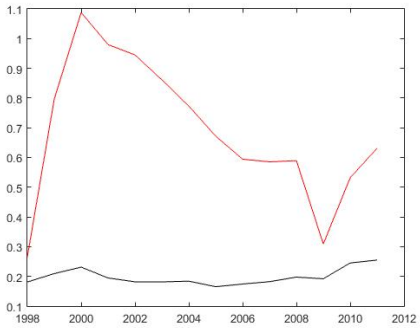


圖 D.25 65 歲以上施打疫苗後 6 個月內比較

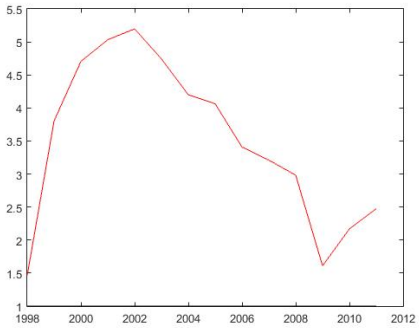


圖 D.26 65 歲以上施打疫苗後 6 個月內校正

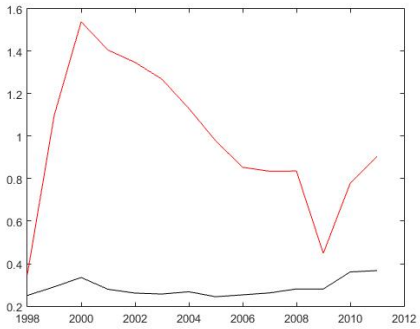


圖 D.27 65 歲以上施打疫苗後 9 個月內比較

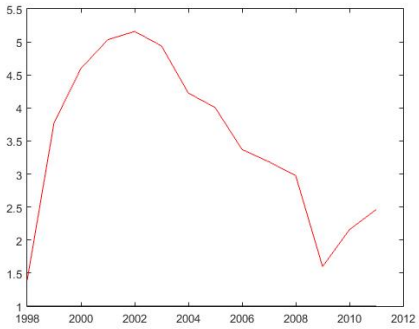


圖 D.28 65 歲以上施打疫苗後 9 個月內校正

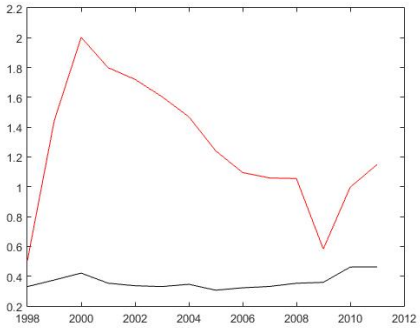


圖 D.29 65 歲以上施打疫苗後 12 個月內比較

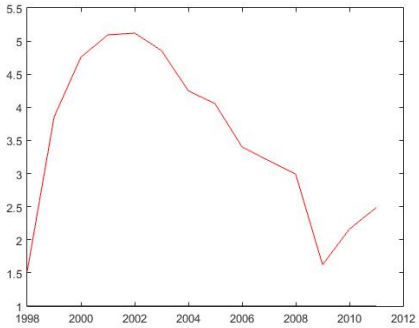


圖 D.30 65 歲以上施打疫苗後 12 個月內校正

(四) 65 歲以上老人死亡率結果

最後我們利用 dead_ rate.py 結果，以年份做為橫軸，以死亡率百分比做為縱軸，畫出圖 D.31，紅線代表 65 歲以上有打疫苗者的死亡率，黑線代表 65 歲以上未打疫苗者

的死亡率，我們可以看到從 2004 年開始黑線就與紅線的差距越來越大，也就是說有施打疫苗的老人死亡率明顯較低。

在文獻 [21] 中是 2002 年的小地區有這樣的狀況，因此我們推測疫苗的效用是有在進步的。

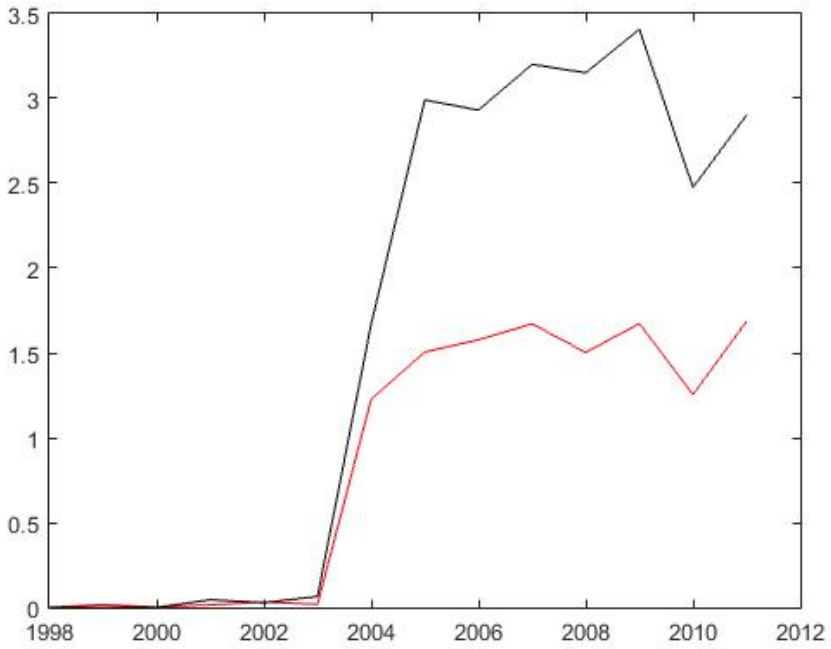


圖 D.31　老人死亡率比較圖

ional Chengchi Univers

第三節　研究方法二之經驗與結果

研究方法二是將樣本數減少，分別隨機抽取有施打疫苗者和未施打疫苗者的 70% 做為樣本，接著使用假設檢定判定疫苗的有效性，本節將告訴讀者如何修改方法一的程式碼得到方法二之結果。

一、 研究過程

本方法中我們沿續前一方法的程式碼，再中間插入以下程式

```
n1 = int(0.7*float(n)) #n1 = 0.7n，n是有打疫苗的人數
Rid = rn.permutation(n) # 將n重新排列
```

然後再加上以下程式

```
for ii1 in range(n1):
 ID = Shot_people[Rid[ii1]] #隨機抽取樣本中的70%
```

以上是有打疫苗的人隨機抽取 70%，同理可以代換成有施打疫苗的人隨機抽取 70%，接著，我們要隨機抽取樣本 64 次，因此在程式前加一行

```
for ttt in range(64):
```

最後將跑出來的結果存成 Result_ sub.pkl 檔，以便下一個程式使用。以上完整程式碼請見附錄 D 的 people_state_sub.py

接著利用 Result_ sub.pkl 裡的資料我們就可以算出 p-value，在 final_ stat.py 中，我們使用

```
stats.ttest_ind(RC[0],WRC[0],equal_var = False)
```

的程式算出 p-value。完整程式碼請見附錄 D 的 final_ stat.py。

二、 研究結果

經由 final_ stat.py 的結果觀察抽樣的平均值是否不同 (以 p-value 判斷，樣本數為 70 萬人次)，並將結果製作表格 D.2、D.3、D.4、D.5。

觀察以下四個表格，表 D.2、D.3、D.4 中未施打疫苗者抽樣的平均值都比有施打疫苗者高，並且 p-value< α =5% 也就是表示疫苗有效，而表 D.5 中則是相反，因此我們推測疫苗對老人無效。

表 D.2 不分年齡抽樣統計結果

打完流感疫苗後 __ 個月	有打疫苗者	未打疫苗者	p-value
3	0.115203082355	0.331528520699	0.0
6	0.25438518888	0.772180587406	0.0
9	0.36386672904	1.09405595206	0.0
12	0.471032509814	1.37821743261	0.0

表 D.3 6 歲以下抽樣統計結果

打完流感疫苗後 __ 個月	有打疫苗者	未打疫苗者	p-value
3	0.0106577170246	0.202780892198	2.8687920563652352e-272
6	0.0222440012756	0.453487952652	7.2018891710436219e-303
9	0.0324975257263	0.652594026212	2.0954198008432469e-305
12	0.0445426389249	0.837814636773	7.0190316231906422e-304

表 D.4 6 歲以上 65 歲以下抽樣統計結果

打完流感疫苗後 __ 個月	有打疫苗者	未打疫苗者	p-value
3	0.0264342502232	0.707551025403	0.0
6	0.0534498380903	1.66461151189	0.0
9	0.0746427856497	2.34426873669	0.0
12	0.10218532053	2.9332325235	0.0

表 D.5 65 歲以上抽樣統計結果

打完流感疫苗後 __ 個月	有打疫苗者	未打疫苗者	p-value
3	0.308517279816	0.0842536444981	0.0
6	0.687461727275	0.198442297676	0.0
9	0.984459875743	0.285305093266	0.0
12	1.26636956999	0.36360513755	0.0

國立政治大學 National Chengchi University

第五章

結論與建議

1. 由方法一及方法二之結果推測，對於全體來說流感疫苗對於控制流感疫情是有效的。
2. 單看 65 歲以上的族群，施打疫苗沒有讓看病的人數減少，反而還增加，但死亡率卻降低了許多，由於因為流感而死亡的病例並不是多數，因此推測是因為流感容易引起許多併發症而導致死亡，有施打疫苗的人較容易因為一些小症狀而就醫，未施打疫苗的人則較容易忽略小症狀而引起併發症，進而延誤就醫造成死亡結果。
3. 對於 65 歲以下的兩個族群，因為較沒有併發症的問題，則不研究死亡率，只看疫苗的有效性，且都是有效的。
4. 對於 65 歲以上的老人族群，衛生教育需要加強；65 歲歲以下的族群則是可以以施打疫苗搭配外出戴口罩，以及個人衛生的宣導加強。
5. 本研究的結果與 2002 年" 流行性感冒疫苗施打之成效評估與將來施打方針"[14] 的結果一致，有施打疫苗者的死亡率較未施打疫苗者的低，但平均就醫次數則是有施打疫苗者較高。

國立政治大學 National Chengchi University

參考文獻

1. 全民健保局, *全民健康保險研究資料庫*, 取自：http://w3.nhri.org.tw/nhird/index.htm
2. 全民健保局, *全民健康保險研究資料庫-資料庫內容說明*, 取自：http://nhird.nhri.org.tw/date_02.html
3. 全民健保局, *全民健康保險研究資料庫-資料庫內容說明*, 取自：http://w3.nhri.org.tw/nhird/date_01.htm
4. 李秉穎, *流感疫苗的誤解與安全議題*, 台灣醫界,51(10)，,17-420,2008。
5. 林倩伃、蕭靜怡、吳宗樹、朱育增、高全良、廖宜真等, *人類流行性感冒的流行病學與偵測系統*, 疫情報導,23(5),254-272,2007。
6. 張秋文、巫坤彬、黃子玫、陳昶勳,*2005 2006 年流感季臺灣地區流感流行情形*, 疫情報導,23(9),489-504,2007。
7. 陳品豪、陳鵬升, *流感的疫苗策略與抗病毒藥物使用*, 家庭醫學與基層醫療,24(9), 304-309 ,2009。
8. 黃曜暉、劉樹泉、戴愛仁、唐雲華、王景順、連德宜、吳采菱,*H1N1 新型流感*, 家庭醫學與基層醫療,24(11),404-410,2009。
9. 衛生福利部疾病管制署,*104 年度流感疫苗接種計畫-20150907*。取自：http://www.cdc.gov.tw/downloadfile.aspx?fid=F779E4D3D5F5FB73
10. 衛生福利部疾病管制署, *季節性流感防治工作手冊 (2015 年 9 月修訂)*。取自：http://www.cdc.gov.tw/uploads/files/4f001cfe-f085-46b3-ba25-e7710710e6b9.pdf
11. 衛生福利部疾病管制署, *季節性流感疫苗Q&A 疫苗篇*。取自：http://www.cdc.gov.tw/qa.aspx?treeid=5784355bfd011a1c&nowtreeid=b75910e5baf64d2d

國立政治大學 National Chengchi University

12. 蕭惠心、孫蕙芝、顏哲傑, *醫療院所工作人員流感疫苗接種後不良反應評估問卷分析*, 疫情報導,21(8),618-631,2005。

13. 簡芝妍, *流感預防 & 治療*, 大利文化,2006

14. 顏哲傑, *流行性感冒疫苗施打之成效評估與將來施打方針*,2002。計畫編號：DOH91-DC-2026

15. Cox N.J. and Subbarao K.,*Influenza.* Lancet; 1999;354: 1277-82.

16. Mandell, GL., Bennett, JE., and Dolin, R. , *'Principles and Practice of InfectiousDiseases' , 6th eds.* Elsevier Inc. 2005: 2060-2078.

17. Nichol, K. L. (2003). ,*The efficacy, effectiveness and cost-effectiveness of inactivated influenza virus vaccines.* Vaccine, 21(16), 1769-1775.

18. Scuffham, P. A., & West, P. A. ,*Economic evaluation of strategies for the control and management of influenza in Europe.* Vaccine, 20(19-20),2562-2578,2002.

19. U.S. Centers for Disease Control and Prevention, National Center for PreventionServices. , *'Prevention and Control of Vaccine-Preventable Diseases: Influenza' , 7theds.* U.S. Dept. of Health and Human Services. 2003; 2006: 235-56.

20. WHO ,*174 H1N1 hospitalized cases in Taiwan: health officials.* 取自：http://www.etaiwannews.com/etn/news_ content.php?id=1053678& lang=eng_ news

21. WHO ,*Influenza.* 取自：http://www.who.int/mediacentre/factsheets/fs211/en/

22. ,*Vaccines against influenza WHO position paper –November 2012. (PDF).* Wkly Epidemiol Rec. Nov 23, 2012, 87 (46): 461–76.

國立政治大學 National Chengchi University

附錄 A: 常用程式碼註解

表 E.1 程式相關指令

程式碼	註解
import A	引入 A 套件
from A import *	引入 A 裡的所有套件
from A import B	引入 A 裡的套件 B
range(m)	從 0 到 m-1 的整數
range(n,m)	從 n 到 m-1 的整數
line[m]	字串中第 m+1 個字
line[m:n]	字串中第 m+1 個字到第 n 個字
int(x)	轉換 x 為整數
len(x)	x 的長度
float(x)	將 x 轉換成浮點數
open(file,'mode')	讀取檔案並回傳檔案串流物件 mode : r 表示讀取 (預設) mode : w 表示寫入
file.close()	檔案不使用後關閉檔案
set()	建立一個集合的物件
set.add(x)	將 x 加入集合裡
A.union(B)	A 聯集 B
A.intersection(B)	A 交集 B
A.difference(B)	A-B
B.difference(A)	B-A
dict()	建一個字典型態 (key:value 配對的複合資料型態) 的物件
dict.append(x)	將 x 加在字典的最後
dict.keys()	回傳 dict _item 物件使 key 依序儲存在 dict _item 物件中
d[key]	從字典 d 中取得 key 的 value
list()	建立一個串列
list.append(x)	將 x 附加到串列的最後
list.sort()	將 list 裡的元素排序

國立政治大學 National Chengchi University

附錄 B: 整理資料類之程式碼

allpeopleset.py

政 治

```
import pickle

#83:91
All = set()
for i in range(1998,2013):
  for j in range(1,26):
    if j <10:
      pathname = '/home/ym/NHI/Cohort/R20' + str(j)+'_'+str(i)
      filename = 'R20'+ str(j)+'_CD'+str(i)+'.DAT'
    else:
      pathname = '/home/ym/NHI/Cohort/R2' + str(j)+'_'+str(i)
      filename = 'R2'+ str(j)+'_CD'+str(i)+'.DAT'

    Filename = pathname+'/'+filename
    print 'processing ', i, '-year and ',j,'-th file'
    for line in open(Filename,'r'):
      All.add((line[91:123],line[83:91]))

fid = open('AllID.pkl','w')
```

```
pickle.dump(All,fid)
fid.close()
```

NHI_ toolbox.py

```
def Age(Birthday, Current):
  year = int(Current[0:4])-int(Birthday[0:4])
  month = int(Current[4:6])-int(Birthday[4:6])
  if month < 0:
    year -= 1
    month += 12
  return (year,month)
#以上為計算年紀之工具
def ICD9_YEAR(yr):
  if (yr>=1997) and (yr<=2003):
    return (130,135), (135,140), (140,145)
  elif (yr>=2004) and (yr<=2011):
    return (131,136), (136,141), (141,146)
  else:
    return (131,146), (146,161), (161,176)
#以上為對應年分之國際及病分類號ICD9碼位置
```

附錄 C: 方法一使用之程式碼

find_ flu_ shot.py

政 治

```
from NHI_toolbox import *
import pickle

Pathname = list()
Filename = list()

for j in range(1998,2013):
  Pathname = list()
  Filename = list()
  for i in range(1,10):
    pathname = '/home/ym/NHI/Cohort/R20' + str(i)+'_' +str(j)
    filename = 'R20'+ str(i)+'_CD'+str(j)+'.DAT'
    Pathname.append(pathname)
    Filename.append(filename)
  for i in range(10,26):
    pathname = '/home/ym/NHI/Cohort/R2' + str(i)+'_'+str(j)
    filename = 'R2'+ str(i)+'_CD'+str(j)+'.DAT'
    Pathname.append(pathname)
    Filename.append(filename)
```

```
icd1 = ICD9_YEAR(j)[0][0]
icd2 = ICD9_YEAR(j)[1][0]
icd3 = ICD9_YEAR(j)[2][0]

D = dict()

for i in range(25):
  fullname = Pathname[i]+'/'+Filename[i]
  print fullname
  for line in open(fullname,'r'):
    ID = line[91:123]
    Date = line[67:75]
    Birthday = line[83:91]
    if (line[icd1:icd1+4]=='V048') or
       (line[icd2:icd2+4]=='V048') or
       (line[icd3:icd3+4]=='V048'):
  age = Age(Birthday,Date)
  if(not(ID in D)):
    D[ID] = list()
  D[ID].append([Date,'V048',age])
print 'year: ', j , 'has ', len(D), 'cases'
filename = 'flu_shot_'+str(j)+'.pkl'
File = open(filename,'w')
pickle.dump(D,File)
File.close()
```

flu_ seak.py

```
from NHI_toolbox import *
import pickle

Pathname = list()
Filename = list()
flu_Set = set(['480','482','487','460'])

for j in range(1998,2013):
  Pathname = list()
  Filename = list()
  for i in range(1,10):
    pathname = '/home/ym/NHI/Cohort/R20' + str(i)+'_' +str(j)
    filename = 'R20'+ str(i)+'_CD'+str(j)+'.DAT'
    Pathname.append(pathname)
    Filename.append(filename)
  for i in range(10,26):
    pathname = '/home/ym/NHI/Cohort/R2' + str(i)+'_'+str(j)
    filename = 'R2'+ str(i)+'_CD'+str(j)+'.DAT'
    Pathname.append(pathname)
    Filename.append(filename)
  icd1 = ICD9_YEAR(j)[0][0]
  icd2 = ICD9_YEAR(j)[1][0]
  icd3 = ICD9_YEAR(j)[2][0]
  D = dict()
  for i in range(25):
    fullname = Pathname[i]+'/'+Filename[i]
```

```
    print fullname
    for line in open(fullname,'r'):
      ID = line[91:123]
      Date = line[67:75]
      Birthday = line[83:91]
      seak_set = set()
      seak_set.add(line[icd1:icd1+3])
      seak_set.add(line[icd2:icd2+3])
      seak_set.add(line[icd3:icd3+3])
      if len(seak_set.intersection(flu_Set))>=1:
        age = Age(Birthday,Date)
        if(not(ID in D)):
          D[ID] = list()
        D[ID].append((Date,age))
  print 'year: ', j , 'has ', len(D), 'cases'
  filename = 'flu_seak_'+str(j)+'.pkl'
  File = open(filename,'w')
  pickle.dump(D,File)
  File.close()
```

population.py

```
import pickle

for i in range(1998,2013):
  filename = 'flu_shot_'+str(i)+'.pkl'
  filename2 = 'flu_seak_'+str(i)+'.pkl'
  fid = open(filename,'r')
```

```
A = pickle.load(fid)
fid.close()
ID = A.keys()
c = set()
a = set()
o = set()
for idx in ID:
  if A[idx][0][2][0]>=65:
    o.add(idx)
  elif A[idx][0][2][0]<6:
    c.add(idx)
  else:
    a.add(idx)
print 'In year ', i
print 'Shot_child = ', len(c)
print 'Shot_adult = ', len(a)
print 'Shot_old = ', len(o)

fid = open(filename2,'r')
A = pickle.load(fid)
fid.close()
c1 = set()
a1 = set()
o1 = set()
ID2 = A.keys()
for idx in ID2:
  if A[idx][0][1][0]>=65:
    o1.add(idx)
```

```
    elif A[idx][0][1][0]<6:
      c1.add(idx)
    else:
      a1.add(idx)
  print 'Seak_child = ', len(c1)
  print 'Seak_adult = ', len(a1)
  print 'Seak_old = ', len(o1)
  c2 = c1.difference(c)
  a2 = a1.difference(a)
  o2 = o1.difference(o)

  print 'Seak_noshot_child = ', len(c2)
  print 'Seak_noshot_adult = ', len(a2)
  print 'Seak_noshot_old = ', len(o2)
  print '======================================'
```

ational Chengchi Universit

people_ state.py

```
import pickle
import numpy as np
import calendar
from scipy import stats
from NHI_toolbox import *

All_stat = list()
All_record = list()
W_All_record = list()
mean_record = list()
```

```
W_mean_record = list()
P_value = list()

for i in range(2011,2012):
  filename1 = '/home/glophy/NHI/YOYO/flu_shot_'+str(i)+'.pkl'
  filename2 = '/home/glophy/NHI/YOYO/flu_seak_'+str(i)+'.pkl'
  filename3 = '/home/glophy/NHI/YOYO/flu_seak_'+str(i+1)+'.pkl'
  fid1 = open(filename1,'r')
  fid2 = open(filename2,'r')
  fid3 = open(filename3,'r')
  A1 = pickle.load(fid1)
  B1 = pickle.load(fid2)
  B2 = pickle.load(fid3)
  fid2.close()
  fid2.close()
  fid3.close()
  Shot_people = A1.keys()
  Seak_people1 = set(B1.keys())
  Seak_people2 = set(B2.keys())
  n = len(Shot_people)
  m = len(set(Seak_people1).difference(set(Shot_people)))
  print 'In year ',i
  print 'There are ', n, ' peoples who have flu shot'
  print 'There are ', m, ' peoples who seak without flu shot'
  print '*********************************'
  All_stat.append((n,m))
  # Rc for child, Rm for middle age, Ro for old man
  Rc = np.zeros([4,n])
```

```
Rm = np.zeros([4,n])
Ro = np.zeros([4,n])
t=0
for ID in Shot_people:
  shot_day = A1[ID][0][0]
  if ID in Seak_people1:
    if A1[ID][0][2][0]<6:
      for seakdata in B1[ID]:
        seak_day = seakdata[0]
        period = Age(shot_day,seak_day)
        if (period[1]>0):
          j = period[1]/3
          Rc[j:,t] += 1
    elif A1[ID][0][2][0]>=60:
      for seakdata in B1[ID]:
        seak_day = seakdata[0]
        period = Age(shot_day,seak_day)
        if (period[1]>0):
          j = period[1]/3
          Ro[j:,t] += 1
    else:
      for seakdata in B1[ID]:
        seak_day = seakdata[0]
        period = Age(shot_day,seak_day)
        if (period[1]>0):
          j = period[1]/3
          Rm[j:,t] += 1
  if ID in Seak_people2:
```

```
    if A1[ID][0][2][0]<6:
      for seakdata in B2[ID]:
        seak_day = seakdata[0]
        period = Age(shot_day,seak_day)
        if (period[1]<12):
          j = period[1]/3
          Rc[j:,t] += 1
    elif A1[ID][0][2][0]>=60:
      for seakdata in B2[ID]:
        seak_day = seakdata[0]
        period = Age(shot_day,seak_day)
        if (period[1]<12):
          j = period[1]/3
          Ro[j:,t] += 1
    else:
      for seakdata in B2[ID]:
        seak_day = seakdata[0]
        period = Age(shot_day,seak_day)
        if (period[1]<12):
          j = period[1]/3
          Rm[j:,t] += 1
  t+=1

rc = [0 for k in range(4)]
rm = rc[:]
ro = rc[:]
ra = rc[:]
for k in range(4):
```

```
    rc[k] = np.mean(Rc[k,:])
    rm[k] = np.mean(Rm[k,:])
    ro[k] = np.mean(Ro[k,:])
    ra[k] = float(rc[k]+rm[k]+ro[k])/3
  mean_record.append([rc,rm,ro,ra])
  All_record.append([Rc,Rm,Ro])

#compute the seak people without flu shot
  Shot_people = set(A1.keys())
  Seak_people1 = set(B1.keys())
  Seak_people1 = list(Seak_people1.difference(Shot_people))
  Seak_people2 = set(B2.keys())
  m = len(Seak_people1)
  WRc = np.zeros([4,m])
  WRm = np.zeros([4,m])
  WRo = np.zeros([4,m])
  shot_day = str(i)+'1001'

  t=0
  for ID in Seak_people1:
    if B1[ID][0][1][0]<6:
      for seakdata in B1[ID]:
        seak_day = seakdata[0]
        period = Age(shot_day,seak_day)
        if (period[1]>0):
          j = period[1]/3
          WRc[j:,t] += 1
    elif B1[ID][0][1][0]>=60:
```

```
    for seakdata in B1[ID]:
      seak_day = seakdata[0]
      period = Age(shot_day,seak_day)
      if (period[1]>0):
        j = period[1]/3
        WRo[j:,t] += 1
  else:
    for seakdata in B1[ID]:
      seak_day = seakdata[0]
      period = Age(shot_day,seak_day)
      if (period[1]>0):
        j = period[1]/3
        WRm[j:,t] += 1

  if ID in Seak_people2:
    if B1[ID][0][1][0]<6:
      for seakdata in B2[ID]:
        seak_day = seakdata[0]
        period = Age(shot_day,seak_day)
        if (period[1]<12):
          j = period[1]/3
          WRc[j:,t] += 1
    elif B1[ID][0][1][0]>=60:
      for seakdata in B2[ID]:
        seak_day = seakdata[0]
        period = Age(shot_day,seak_day)
        if (period[1]<12):
          j = period[1]/3
```

```
            WRo[j:,t] += 1
      else:
        for seakdata in B2[ID]:
          seak_day = seakdata[0]
          period = Age(shot_day,seak_day)
          if (period[1]<12):
            j = period[1]/3
            WRm[j:,t] += 1
    t+=1

wrc = [0 for k in range(4)]
wrm = rc[:]
wro = rc[:]
wra = rc[:]
for k in range(4):
  wrc[k] = np.mean(WRc[k,:])
  wrm[k] = np.mean(WRm[k,:])
  wro[k] = np.mean(WRo[k,:])
  wra[k] = float(wrc[k]+wrm[k]+wro[k])/3
W_mean_record.append([wrc,wrm,wro,wra])
W_All_record.append([WRc,WRm,WRo])



for k in range(4):
  print 'The ', 3*(k+1), ' month'
  print 'The average times for child is ', rc[k], wrc[k]
```

```
    print 'The average times for middle people is ', rm[k], wrm[k]
    print 'The average times for old people is ', ro[k], wro[k]
    print '.........................................'
#stats.ttest_ind(rvs1,rvs2, equal_var = False)
  K = np.zeros([3,4])
  for k in range(4):
    pc = stats.ttest_ind(WRc[k,:],Rc[k,:],equal_var = False)
    pm = stats.ttest_ind(WRm[k,:],Rm[k,:],equal_var = False)
    po = stats.ttest_ind(WRo[k,:],Ro[k,:],equal_var = False)
    K[0,k] = pc[1]
    K[1,k] = pm[1]
    K[2,k] = po[1]
    print 'The ', 3*(k+1), ' month'
    print 'The p-value of Child is ', pc[1]
    print 'The p-value of middle people is ', pm[1]
    print 'The p-value of old people is ', po[1]
    print '==========================================='
  P_value.append(K)

output_file = 'Result.pkl'
fid = open(output_file,'w')
pickle.dump(All_stat,fid)
pickle.dump(All_record,fid)
pickle.dump(W_All_record,fid)
pickle.dump(mean_record,fid)
pickle.dump(W_mean_record,fid)
pickle.dump(P_value,fid)
fid.close()
```

find_ dead_ flu.py

```
from NHI_toolbox import *
import pickle

Pathname = list()
Filename = list()
dead_code = set(range(480,488))

for i in range(1998,2013):
  if (i-1911)>90:
    filename = '/home/ym/NHI/Heavy/NHIRD-104-074/AN'+str(i-1911)+'02/HV'
    +str(i)+'.DAT'
  else:
    filename = '/home/ym/NHI/Heavy/NHIRD-104-074/AN'+str(i-1911)+'01/HV'
    +str(i)+'.DAT'

  icd9 = 32
  D = set()
  for line in open(filename,'r'):
    if len(line)>300:
      ID = line[0:32]
      Birthday = line[39:47]
      if (line[292] == 'Y') or (line[292] == 'M'):
        D.add(ID)

  print 'year: ', i , 'has ', len(D), 'die'
  filename = 'flu_dead_'+str(i)+'.pkl'
```

```
  File = open(filename,'w')
  pickle.dump(D,File)
  File.close()
```

dead_ rate.py

```
import pickle

R = list()
Old = list()
Chi = list()
Adu = list()
for i in range(1998,2012):
  filename = 'flu_shot_'+str(i)+'.pkl'
  filename1 = 'flu_dead_'+str(i)+'.pkl'
  filename2 = 'flu_dead_'+str(i+1)+'.pkl'
  filename3 = 'flu_seak_'+str(i)+'.pkl'
  fid = open(filename,'r')
  A = pickle.load(fid)
  fid.close()
  fid = open(filename1,'r')
  B = pickle.load(fid)
  fid.close()
  fid = open(filename2,'r')
  C = pickle.load(fid)
  fid.close()
  fid = open(filename3,'r')
  D = pickle.load(fid)
```

```
fid.close()

ID_shot = set(A.keys())
ID_noshot = set(D.keys())
ID_dead1 = B
ID_dead2 = C
ID_dead = ID_dead1.union(ID_dead2)
ID_noshot = ID_noshot.difference(ID_shot)
shot_dead_rate =
 float(len(ID_shot.intersection(ID_dead)))/len(ID_shot)
noshot_dead_rate =
 float(len(ID_noshot.intersection(ID_dead)))/len(ID_noshot)
R.append((shot_dead_rate,noshot_dead_rate))

ID_shot = list(ID_shot)
ID_shot_C = set()
ID_shot_A = set()
ID_shot_O = set()
for idx in ID_shot:
  if A[idx][0][2][0] >=65:
    ID_shot_O.add(idx)
  elif A[idx][0][2][0] <6:
    ID_shot_C.add(idx)
  else:
    ID_shot_A.add(idx)
if len(ID_shot_C)==0:
  rc = 0
else:
```

```
  rc = float(len(ID_shot_C.intersection(ID_dead)))/len(ID_shot_C)
ra = float(len(ID_shot_A.intersection(ID_dead)))/len(ID_shot_A)
ro = float(len(ID_shot_O.intersection(ID_dead)))/len(ID_shot_O)

ID_noshot = list(ID_noshot)
ID_noshot_C = set()
ID_noshot_A = set()
ID_noshot_O = set()
for idx in ID_noshot:
  if D[idx][0][1][0]>=65:
    ID_noshot_O.add(idx)
  elif D[idx][0][1][0]<6:
    ID_noshot_C.add(idx)
  else:
    ID_noshot_A.add(idx)
if len(ID_noshot_C)==0:
  rcn = 0
else:
  rcn = float(len(ID_noshot_C.intersection(ID_dead)))/len(ID_noshot_C)
ran = float(len(ID_noshot_A.intersection(ID_dead)))/len(ID_noshot_A)
ron = float(len(ID_noshot_O.intersection(ID_dead)))/len(ID_noshot_O)

Old.append((ro,ron))
Chi.append((rc,rcn))
Adu.append((ra,ran))

print 'In year ', i,
print '.....'
```

```
  print 'the rate of shot_people_dead is',
   (shot_dead_rate)*100,'%'
  print 'the rate of non_shot_people_dead is',
   (noshot_dead_rate)*100,'%'

  print 'rate of shot_child_dead is ', rc*100,'%'
  print 'rate of noshot_child_dead is ',rcn*100,'%'

  print 'rate of shot_adult_dead is ', ra*100,'%'
  print 'rate of noshot_adult_dead is ', ran*100,'%'

  print 'rate of shot_old_dead is ', ro*100,'%'
  print 'rate of noshot_old_dead is ', ron*100,'%'
  print '========================'

fid = open('dead_stat.pkl','w')
pickle.dump(R,fid)
pickle.dump(Chi,fid)
pickle.dump(Adu,fid)
pickle.dump(Old,fid)
fid.close()
```

附錄 D: 方法二使用之程式碼

people_ state_ sub.py

政 治

```
import pickle
import numpy as np
import calendar
from scipy import stats
from NHI_toolbox import *
import numpy.random as rn

All_stat = list()
All_record = list()
W_All_record = list()
mean_record = list()
W_mean_record = list()
P_value = list()

Super_record = list()
for ttt in range(64):
        print ttt, 'times **********'
        print '-----------------------'
    for i in range(1998,2012):
```

```
filename1 = '/home/glophy/NHI/YOYO/flu_shot_'+str(i)+'.pkl'
filename2 = '/home/glophy/NHI/YOYO/flu_seak_'+str(i)+'.pkl'
filename3 = '/home/glophy/NHI/YOYO/flu_seak_'+str(i+1)+'.pkl'
fid1 = open(filename1,'r')
fid2 = open(filename2,'r')
fid3 = open(filename3,'r')
A1 = pickle.load(fid1)
B1 = pickle.load(fid2)
B2 = pickle.load(fid3)
fid2.close()
fid2.close()
fid3.close()
Shot_people = A1.keys()
Seak_people1 = set(B1.keys())
Seak_people2 = set(B2.keys())
n = len(Shot_people)
    n1 = int(0.7*float(n))
    Rid = rn.permutation(n)
m = len(set(Seak_people1).difference(set(Shot_people)))
print 'In year ',i
print 'There are ', n, ' peoples who have flu shot'
print 'There are ', m, ' peoples who seak without flu shot'
print '*********************************'
All_stat.append((n,m))
# Rc for child, Rm for middle age, Ro for old man
Rc = np.zeros([4,n1])
Rm = np.zeros([4,n1])
Ro = np.zeros([4,n1])
```

```
t=0
    for ii1 in range(n1):
      #make the random sampling 70%
      ID = Shot_people[Rid[ii1]]
  shot_day = A1[ID][0][0]
  if ID in Seak_people1:
    if A1[ID][0][2][0]<6:
  for seakdata in B1[ID]:
    seak_day = seakdata[0]
    period = Age(shot_day,seak_day)
    if (period[1]>0):
      j = period[1]/3
      Rc[j:,t] += 1
    elif A1[ID][0][2][0]>=60:
  for seakdata in B1[ID]:
    seak_day = seakdata[0]
    period = Age(shot_day,seak_day)
    if (period[1]>0):
      j = period[1]/3
      Ro[j:,t] += 1
    else:
  for seakdata in B1[ID]:
    seak_day = seakdata[0]
    period = Age(shot_day,seak_day)
    if (period[1]>0):
      j = period[1]/3
      Rm[j:,t] += 1
  if ID in Seak_people2:
```

```
    if A1[ID][0][2][0]<6:
  for seakdata in B2[ID]:
    seak_day = seakdata[0]
    period = Age(shot_day,seak_day)
    if (period[1]<12):
      j = period[1]/3
      Rc[j:,t] += 1
    elif A1[ID][0][2][0]>=60:
  for seakdata in B2[ID]:
    seak_day = seakdata[0]
    period = Age(shot_day,seak_day)
    if (period[1]<12):
      j = period[1]/3
      Ro[j:,t] += 1
    else:
  for seakdata in B2[ID]:
    seak_day = seakdata[0]
    period = Age(shot_day,seak_day)
    if (period[1]<12):
      j = period[1]/3
      Rm[j:,t] += 1
  t+=1

rc = [0 for k in range(4)]
rm = rc[:]
ro = rc[:]
ra = rc[:]
for k in range(4):
```

```
  rc[k] = np.mean(Rc[k,:])
  rm[k] = np.mean(Rm[k,:])
  ro[k] = np.mean(Ro[k,:])
  ra[k] = float(rc[k]+rm[k]+ro[k])/3
mean_record.append([rc,rm,ro,ra])
All_record.append([Rc,Rm,Ro])

Shot_people = set(A1.keys())
Seak_people1 = set(B1.keys())
Seak_people1 = list(Seak_people1.difference(Shot_people))
Seak_people2 = set(B2.keys())
m = len(Seak_people1)
    m1 = int(float(m)*0.7)
    # random sampling 70%
    Rid = rn.permutation(m)
WRc = np.zeros([4,m1])
WRm = np.zeros([4,m1])
WRo = np.zeros([4,m1])
shot_day = str(i)+'1001'

t=0
    for ii2 in range(m1):
      ID = Seak_people1[Rid[ii2]]
  if B1[ID][0][1][0]<6:
    for seakdata in B1[ID]:
  seak_day = seakdata[0]
  period = Age(shot_day,seak_day)
  if (period[1]>0):
```

```
  j = period[1]/3
  WRc[j:,t] += 1
elif B1[ID][0][1][0]>=60:
  for seakdata in B1[ID]:
seak_day = seakdata[0]
period = Age(shot_day,seak_day)
if (period[1]>0):
  j = period[1]/3
  WRo[j:,t] += 1
else:
  for seakdata in B1[ID]:
seak_day = seakdata[0]
period = Age(shot_day,seak_day)
if (period[1]>0):
  j = period[1]/3
  WRm[j:,t] += 1

if ID in Seak_people2:
  if B1[ID][0][1][0]<6:
for seakdata in B2[ID]:
  seak_day = seakdata[0]
  period = Age(shot_day,seak_day)
  if (period[1]<12):
    j = period[1]/3
    WRc[j:,t] += 1
  elif B1[ID][0][1][0]>=60:
for seakdata in B2[ID]:
  seak_day = seakdata[0]
```

```
    period = Age(shot_day,seak_day)
    if (period[1]<12):
      j = period[1]/3
      WRo[j:,t] += 1
    else:
  for seakdata in B2[ID]:
    seak_day = seakdata[0]
    period = Age(shot_day,seak_day)
    if (period[1]<12):
      j = period[1]/3
      WRm[j:,t] += 1
  t+=1

wrc = [0 for k in range(4)]
wrm = rc[:]
wro = rc[:]
wra = rc[:]
for k in range(4):
  wrc[k] = np.mean(WRc[k,:])
  wrm[k] = np.mean(WRm[k,:])
  wro[k] = np.mean(WRo[k,:])
  wra[k] = float(wrc[k]+wrm[k]+wro[k])/3
W_mean_record.append([wrc,wrm,wro,wra])
W_All_record.append([WRc,WRm,WRo])

for k in range(4):
  print 'The ', 3*(k+1), ' month'
  print 'The average times for child is ', rc[k], wrc[k]
```

```
        print 'The average times for middle people is ', rm[k], wrm[k]
        print 'The average times for old people is ', ro[k], wro[k]
        print '.........................................'
    #stats.ttest_ind(rvs1,rvs2, equal_var = False)
      K = np.zeros([3,4])
      for k in range(4):
        pc = stats.ttest_ind(WRc[k,:],Rc[k,:],equal_var = False)
        pm = stats.ttest_ind(WRm[k,:],Rm[k,:],equal_var = False)
        po = stats.ttest_ind(WRo[k,:],Ro[k,:],equal_var = False)
        K[0,k] = pc[1]
        K[1,k] = pm[1]
        K[2,k] = po[1]
        print 'The ', 3*(k+1), ' month'
        print 'The p-value of Child is ', pc[1]
        print 'The p-value of middle people is ', pm[1]
        print 'The p-value of old people is ', po[1]
        print '==========================================='
      P_value.append(K)
        Super_record.append((mean_record,W_mean_record))
output_file = 'Result_sub.pkl'
fid = open(output_file,'w')
pickle.dump(Super_record,fid)
fid.close()
```

final_ stat.py

```
import pickle
from scipy import stats
```

```
import numpy as np

fid = open('Result_sub.pkl','r')
A = pickle.load(fid)
fid.close()

RC = [list(), list(), list(), list()]
WRC = [list(), list(), list(), list()]
RM = [list(), list(), list(), list()]
WRM = [list(), list(), list(), list()]
RO = [list(), list(), list(), list()]
WRO = [list(), list(), list(), list()]
RA = [list(), list(), list(), list()]
WRA = [list(), list(), list(), list()]

for i in range(64):
  for j in range(14):
    #A[0][0][] : no W
    #A[0][1][] : W
    for k in range(4):
      RC[k].append(A[0][0][i*14+j][0][k])
      RM[k].append(A[0][0][i*14+j][1][k])
      RO[k].append(A[0][0][i*14+j][2][k])
      RA[k].append(A[0][0][i*14+j][3][k])
      WRC[k].append(A[0][1][i*14+j][0][k])
      WRM[k].append(A[0][1][i*14+j][1][k])
      WRO[k].append(A[0][1][i*14+j][2][k])
      WRA[k].append(A[0][1][i*14+j][3][k])
```

```
print 'The mean of RC-3 is ', np.mean(RC[0])
print 'The mean of WRC-3 is ', np.mean(WRC[0])
print 'The p-value of RC-WRC-3 is ',
 stats.ttest_ind(RC[0],WRC[0],equal_var = False)
print '==============================='

print 'The mean of RC-6 is ', np.mean(RC[1])
print 'The mean of WRC-6 is ', np.mean(WRC[1])
print 'The p-value of RC-WRC-6 is ',
 stats.ttest_ind(RC[1],WRC[1],equal_var = False)
print '==============================='

print 'The mean of RC-9 is ', np.mean(RC[2])
print 'The mean of WRC-9 is ', np.mean(WRC[2])
print 'The p-value of RC-WRC-9 is ',
 stats.ttest_ind(RC[2],WRC[2],equal_var = False)
print '==============================='

print 'The mean of RC-12 is ', np.mean(RC[3])
print 'The mean of WRC-12 is ', np.mean(WRC[3])
print 'The p-value of RC-WRC-12 is ',
 stats.ttest_ind(RC[3],WRC[3],equal_var = False)
print '*******************************'

print 'The mean of RM-3 is ', np.mean(RM[0])
print 'The mean of WRM-3 is ', np.mean(WRM[0])
```

```
print 'The p-value of RM-WRM-3 is ',
 stats.ttest_ind(RM[0],WRM[0],equal_var = False)
print '==============================='

print 'The mean of RM-6 is ', np.mean(RM[1])
print 'The mean of WRM-6 is ', np.mean(WRM[1])
print 'The p-value of RM-WRM-6 is ',
 stats.ttest_ind(RM[1],WRM[1],equal_var = False)
print '==============================='

print 'The mean of RM-9 is ', np.mean(RM[2])
print 'The mean of WRM-9 is ', np.mean(WRM[2])
print 'The p-value of RM-WRM-9 is ',
 stats.ttest_ind(RM[2],WRM[2],equal_var = False)
print '==============================='

print 'The mean of RM-12 is ', np.mean(RM[3])
print 'The mean of WRM-12 is ', np.mean(WRM[3])
print 'The p-value of RM-WRM-12 is ',
 stats.ttest_ind(RM[3],WRM[3],equal_var = False)
print '*******************************'

print 'The mean of RO-3 is ', np.mean(RO[0])
print 'The mean of WRO-3 is ', np.mean(WRO[0])
print 'The p-value of RO-WRO-3 is ',
 stats.ttest_ind(RO[0],WRO[0],equal_var = False)
print '==============================='
```

```
print 'The mean of RO-6 is ', np.mean(RO[1])
print 'The mean of WRO-6 is ', np.mean(WRO[1])
print 'The p-value of RO-WRO-6 is ',
 stats.ttest_ind(RO[1],WRO[1],equal_var = False)
print '=============================='

print 'The mean of RO-9 is ', np.mean(RO[2])
print 'The mean of WRO-9 is ', np.mean(WRO[2])
print 'The p-value of RO-WRO-9 is ',
 stats.ttest_ind(RO[2],WRO[2],equal_var = False)
print '=============================='

print 'The mean of RO-12 is ', np.mean(RO[3])
print 'The mean of WRO-12 is ', np.mean(WRO[3])
print 'The p-value of RO-WRO-12 is ',
 stats.ttest_ind(RO[3],WRO[3],equal_var = False)
print '******************************'

print 'The mean of RA-3 is ', np.mean(RA[0])
print 'The mean of WRA-3 is ', np.mean(WRA[0])
print 'The p-value of RA-WRA-3 is ',
 stats.ttest_ind(RA[0],WRA[0],equal_var = False)
print '=============================='

print 'The mean of RA-6 is ', np.mean(RA[1])
print 'The mean of WRA-6 is ', np.mean(WRA[1])
print 'The p-value of RA-WRA-6 is ',
 stats.ttest_ind(RA[1],WRA[1],equal_var = False)
```

```
print '==============================='

print 'The mean of RA-9 is ', np.mean(RA[2])
print 'The mean of WRA-9 is ', np.mean(WRA[2])
print 'The p-value of RA-WRA-9 is ',
 stats.ttest_ind(RA[2],WRA[2],equal_var = False)
print '==============================='

print 'The mean of RA-12 is ', np.mean(RA[3])
print 'The mean of WRA-12 is ', np.mean(WRA[3])
print 'The p-value of RA-WRA-12 is ',
 stats.ttest_ind(RA[3],WRA[3],equal_var = False)
print '******************************'
```

National Chengchi University