2025年臺灣國際科學展覽會
優勝作品專輯

作品編號　160029

參展科別　物理與天文學

作品名稱　磁星短x射線爆發特徵分析：以1E2259+586為例

得獎獎項　四等獎

就讀學校　臺北市數位實驗高級中等學校

指導教師　蔡兆陽

胡欽評

作者姓名　陳昫言

駱韋晴

關鍵詞　Magnetar、Short X-ray burst、HDBSCAN

作者簡介

我們是來自臺北市數位實驗高中的陳昀言和駱韋晴。一開始我不懂這些，以為只是單純的觀察星空。然而，當我走進天文的世界，才明白探索宇宙是人類最深邃且充滿無限想像的追求，高一開始進入了物理與天文學的世界。

這一路上要特別謝謝全力支持我們的蔡兆陽老師，以及胡欽評教授，還有家人、老師、朋友們一直以來的陪伴，讓我們勇敢去闖，獲得這次寶貴的經驗。

附件三

研究報告封面

2025 年臺灣國際科學展覽會

研究報告

區別：

科別： 物理與天文科

作品名稱：磁星短 x 射線爆發特徵分析：以 1E2259+586 為例

關鍵詞：Magnetar、Short X-ray burst、HDBSCAN

編號：

研究計畫內容

磁星短 x 射線爆發特徵分析：以 1E2259+586 為例

中文摘要

我們是探討磁星的短 X 射線爆發(Short X-ray burst)。利用 RXTE 太空望遠鏡觀測磁星 1E2259+586 的數據，經由 Bayesian block 方法對光變曲線篩選找出爆發，並配合「波松分佈」與「虛無假設」找出 50 筆爆發事件(爆發的正確性有 5σ 的信心水準)。再利用 HDBSCAN 非監督式學習演算法來對短 X 射線爆發進行分群，找出此磁星有「短暫且高能爆發、中等持續與能量爆發、較長持續且溫和爆發、快速且低能爆發」現象，暗示了磁星爆發的多樣性並有不同的爆發機制。此外我們也發現磁星可能有「週期性」的現象，也許是自轉週期、地殼受的應力或磁場變化經過同樣時間累積(有週期性)而爆發。我們也比對有快速電波爆發 (Fast radio burst, FRB)的磁星 SGR 1935+2154，看是否 1E2259+586 有 FRB 現象，結果暗示 1E2259+586 可能沒有 FRB 現象。

英文摘要

We are investigating the short X-ray bursts of magnetars. Using the data observed by the RXTE space telescope for the magnetar 1E2259+586, we applied the Bayesian block method to filter the light curve to identify bursts. We then used the ‘Poisson distribution’ and ‘null hypothesis’ to identify 50 burst events (the correctness of the bursts has a confidence level of 5σ).

Furthermore, we used the HDBSCAN unsupervised learning algorithm to cluster the short X-ray bursts, revealing that this magnetar has phenomena of ‘short and high-energy bursts, medium-duration and energy bursts, longer-duration and mild bursts, and fast and low-energy bursts’. This suggests the diversity of magnetar bursts and different burst mechanisms.

In addition, we found that magnetars may have a ‘periodic’ phenomenon, which

could be due to the rotation period, the stress on the crust, or changes in the magnetic field accumulating over the same period (periodic) and then bursting. We also compared with the magnetar SGR 1935+2154, which has fast radio bursts (FRB), to see if 1E2259+586 has FRB phenomena. The results suggest that 1E2259+586 may not have FRB phenomena.

壹、研究動機

2020 年，Andersen, B. C.等人對磁星 SGR 1935+2154 的研究提供了重要發現，他們發現此顆磁星有快速電波爆發(FRB)現象，這一發現促使我們想深入研究磁星的短 X 射線爆發特性。因此我們的研究重點是探索磁星 1E2259+586 的短 X 射線爆發特性，以及這些爆發是否與快速電波爆發存在相似性。

貳、研究目的及研究問題

一、研究目的

（一）對磁星 1E2259+586 找出每次爆發時的參數值：爆發時間、爆發時間間隔（T90）、爆發時光子數、軟光子(能量 2~4 kev)數目、硬光子(能量 4~10 kev)數目、硬指數(Hardness ratio)、爆發與前一個爆發所經過時間。

（二）將找出來的爆發參數利用 HDBSCAN 演算法分群，利用分群後的結果來了解磁星的物理機制。

（三）嘗試利用分群結果找此磁星有沒有 FRB 的一些性質。

二、文獻回顧

（一）磁星

磁星屬於中子星的一種，這些星體的磁場比一般中子星磁場(10^{10}~10^{12} 高斯)強度還大，磁星磁場的強度大約是 10^{15}~10^{16} 高斯，遠超過普通中子星或任何其他已知天體。

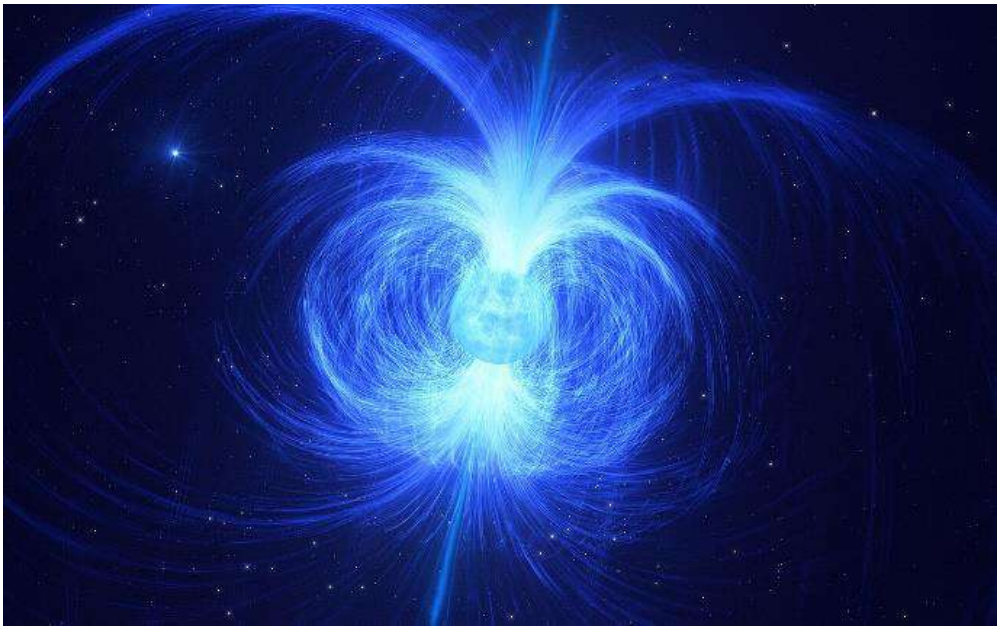
磁星的形成過程涉及超新星爆炸後的中子星中強烈的磁場放大。這

可能是由於中子星的早期階段強烈的對流和快速旋轉產生的動力學作用。這種強磁場可以解釋磁星觀測到的一系列特異現象，包括「短 X 射線爆發」、「快速藍光變暗暫現象（FBOTs）」和與「長伽馬射線暴（lGRBs）」等。

磁星的一個獨特特點是它們的 X 射線和伽馬射線輻射。這些輻射通常是突發性的，展現出來為短暫的亮度增強或能量爆發，這些事件被稱為「星震」。這些星震產生的原因可能是強烈磁場對星體內部結構的重新排列和扭曲所致。

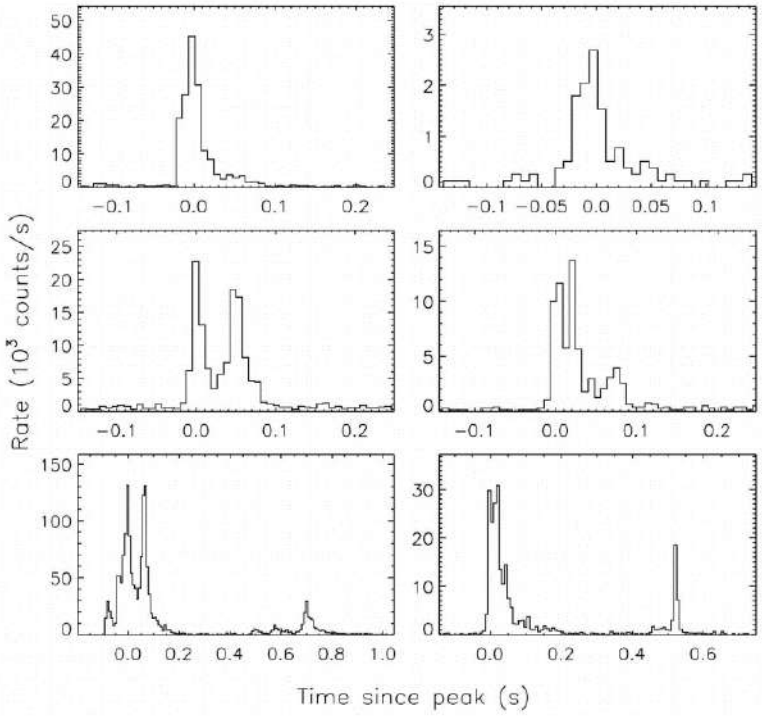
磁星的研究對於理解極端物理環境下的物質行為非常重要。由於它們的磁場強度遠遠超出地球上實驗室能夠產生的範圍，研究磁星提供了一個獨特的機會來探索物理學的未知領域。此外，磁星的觀測還有助於我們更好地理解恆星演化的晚期階段，以及宇宙中極端物理條件下的物質和輻射。

圖一：磁星示意圖



來源：取自 https://reurl.cc/13EAQD

圖二：磁星 SGRs 1806−20 爆發時的光變曲線



來源：Göğüs E, 2001, TEMPORAL AND SPECTRAL CHARACTERISTICS OF SHORT BURSTS FROM THE SOFT GAMMA REPEATERS 1806-20 AND 1900+14 , ApJ 558:228-236ApJ 558:228-236

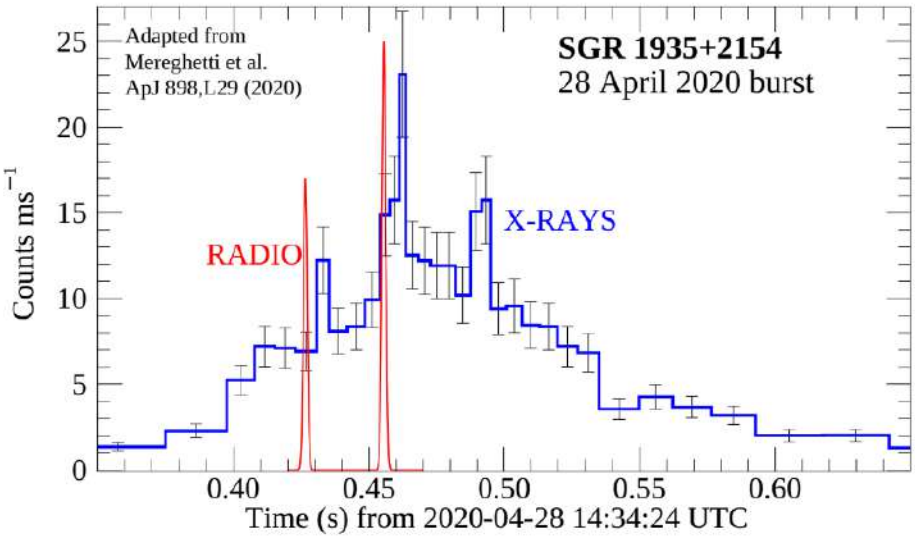
（二）快速電波爆發(Fast radio burst，可簡稱 FRB)

快速電波爆發是一種瞬間的電波脈衝，時間長度從幾分之一毫秒到 3 秒不等，快速電波爆發平均在一毫秒內釋放的能量與太陽在三天內釋放的能量一樣多。第一個快速電波爆發是由 Duncan Lorimer 和他的學生 David Narkevic 在 2007 年發現。隨後快速電波爆發被發現的次數愈來愈多，而且都是在銀河系外被偵測到。

快速電波爆發的來源有許多理論，有人認為和地球人造的、活躍星系核(AGN)、緻密天體(如中子星)、黑洞或中子星碰撞等發出，甚至還有科學家認為是外星生物造成。後來偵測到快速電波爆發有極化的情況，表示來源有極強的磁場，超新星爆炸產生的磁星可能有 FRB 現象。

2020 年 4 月 28 日，Andersen, B. C. et al. 的 CHIME/FRB Collaboration 團隊觀察到了一次非常引人注目的事件：磁星 SGR 1935+2154 發生了一次強烈的爆發，並伴隨著一個 FRB。這一發現對於理解 FRB 的來源具有重要意義，因為它是「第一次將 FRB 與磁星活動直接相關聯的證據」。這也是第一次在銀河系內觀測到的快速電波爆發。而磁星就是磁場極大的中子星，本身會有短 X 射線爆發的現象。

圖三：2020 年 4 月 28 日 CHIME/FRB 望遠鏡偵測到 SGR 1935+2154 爆發的光變曲線圖。



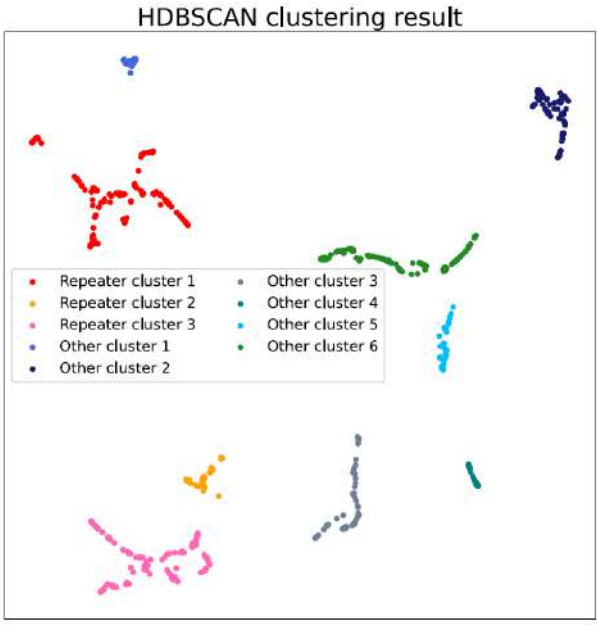
資料來源： Mereghetti S. et al., (2021), ApJ 921(1) :L3

（三）分群研究

在磁究星體的爆發性質中，我們參考這篇論文使用 HDBSCAN 進行分群快速電波爆發（FRB）。首先，他們利用 UMAP（Uniform Manifold Approximation and Projection）將 FRB 樣本投影到低維空間，並將訓練集中的非重複 FRB、重複 FRB 和驗證集中的重複 FRB 分別標記為灰色、綠松色和粉紅色。接著，使用 HDBSCAN 進行分群，將 FRB 樣本分為九個群集。其定義重複群集為包含超過 10%重複 FRB 的群集，結果顯示有三個群集被分類為重複群集，其餘六個群集則不是。這些群集分別標

記為重複群集 1-3 和其他群集 1-6。最後，研究人員使用真實陽性率（True Positive Rate）來評估模型能力，並展示了分群結果和每個群集的總結。

圖四：此篇論文的分群結果圖



資料來源：Bo Han Chen, Tetsuya Hashimoto, et al. (2022), Uncloaking hidden repeating fast radio bursts with unsupervised machine learning, MNRAS 509, 1227-1236

參、研究設備及器材

一、安裝 Ubuntu 作業系統的電腦

二、Python：版本 3.10.12

三、RXTE 太空望遠鏡資料：NASA 網路資料庫

肆、研究過程或方法及進行步驟

一、研究流程

（一）由太空望遠鏡 RXTE 網路資料下載磁星 1E2259+586 的觀測數據。

（二）將原始資料處理得到光變曲線。

（三）使用 Bayesian block 方法對光變曲線進行篩選找出爆發的部分，並用人工再次確認是否為爆發事件。

（四）利用「波松分佈」配合「虛無假設」分析，確認爆發的正確性有 5σ 的信心水準。

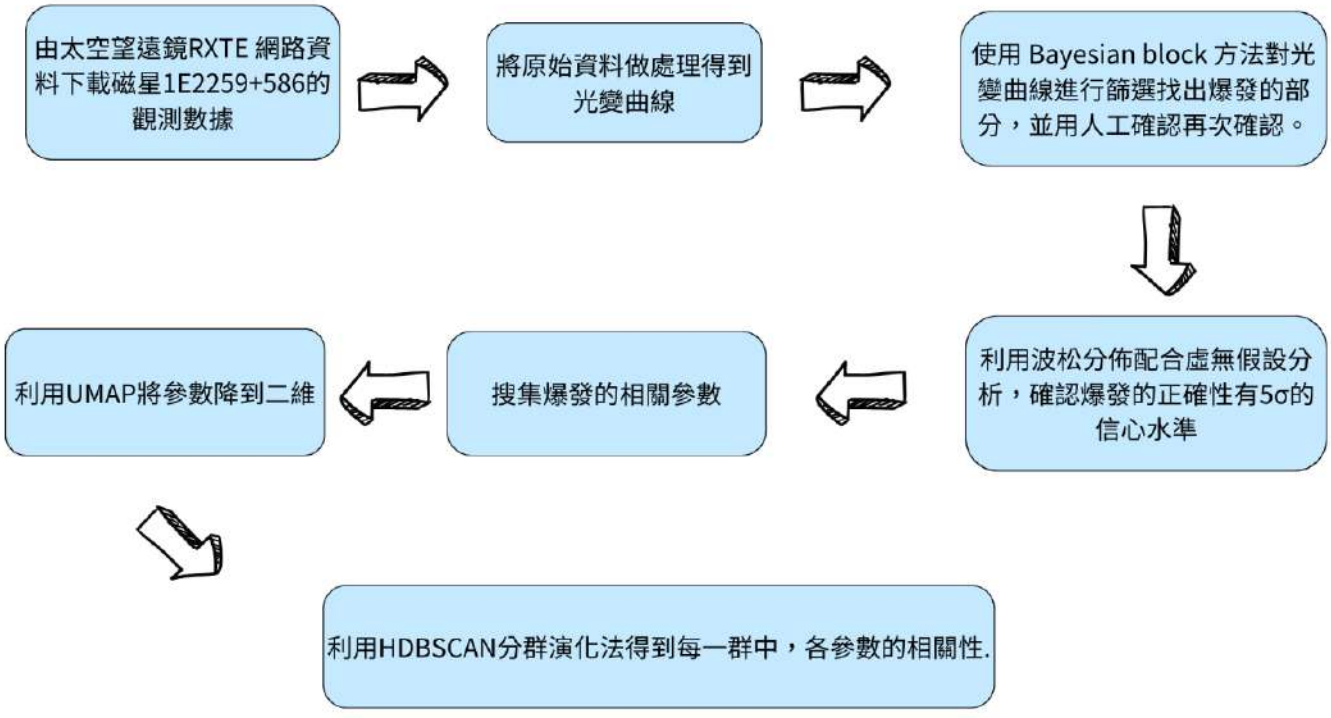
（五）搜集爆發的相關參數：爆發時間、爆發時間間隔（T90）、爆發時光子數、軟光子(能量 2~4 kev)數目、硬光子(能量 4~10 kev)數目、硬指數(Hardness ratio)、爆發與前一個爆發所經過時間。

（六）利用 UMAP 方法，將參數降到二維。

（七）利用 HDBSCAN 分群演化法得到許多群組。

（八）探討每個群組爆發參數對應磁星的物理機制。

圖五：流程圖



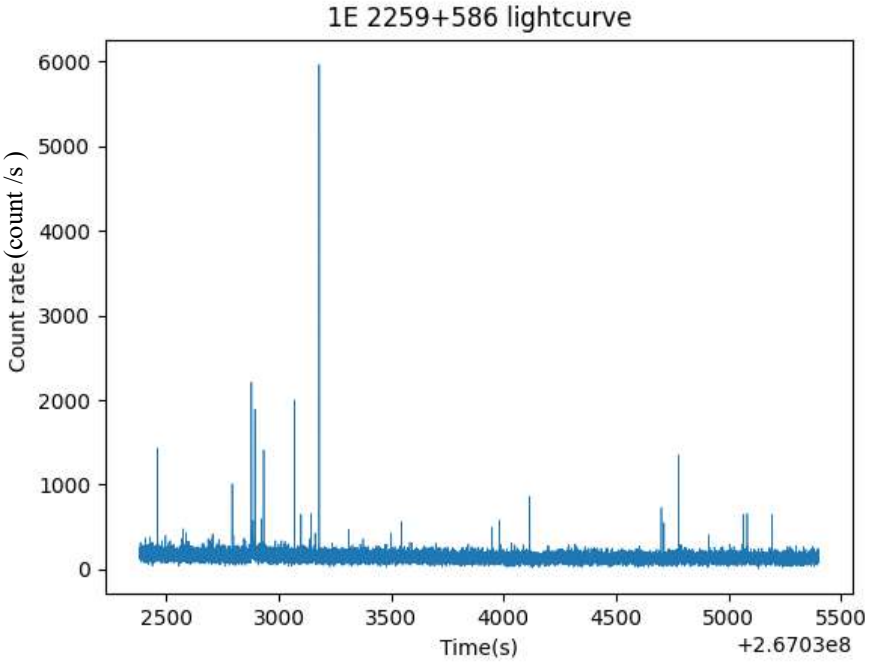
來源：作者自行繪製

二、研究方法

（一）資料處理找出光變曲線

利用 Python 程式語言將 NASA 中磁星 1E2259+586 的資料處理，並得到光畫曲線，如圖六

圖六：磁星 1E2259+586 某段時間的光變

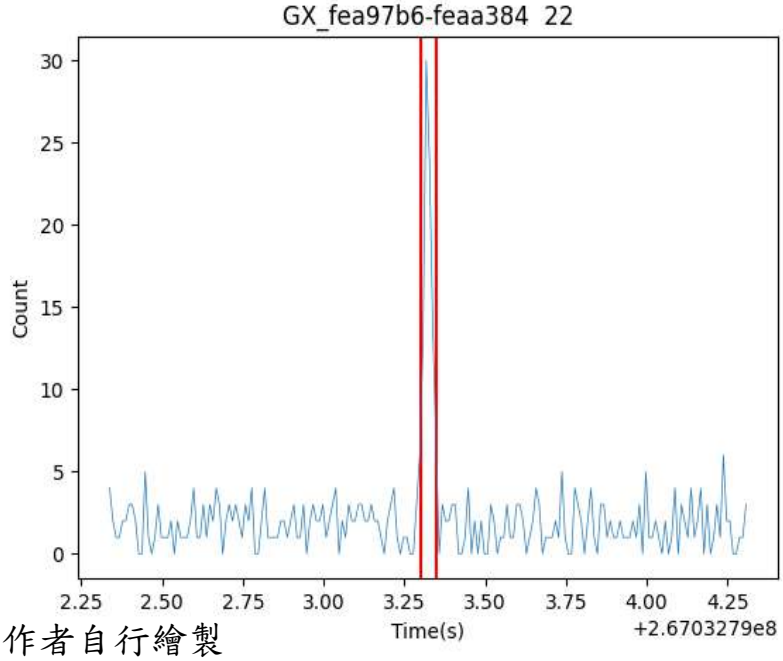


來源：作者自行繪製

（二）找出爆發處

我們使用 Bayesian block 方法，如圖七與圖八，紅色線就是光子數目變化較大處，我們就以兩條紅色線間當成是一個爆發事件；而間隔時間取其 T90 當成是爆發時間間隔。

圖七：180 秒且利用 Bayesian block 找出爆發處的光變曲線



來源：作者自行繪製

（三）「再次」確認是爆發事件：

有的爆發事件光子數比背景光子平均值大不多，為了去除 Bayesian block 方法還是將此情況當成是爆發事件，我們利用「**波松分佈**」配合「**虛無假設**」確定找到的是爆發處（**5σ 的信心水準**：一百七十萬分之一不是爆發現象）。最後得到「**50 筆爆發事件**」

波松分佈：

$$P(x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

其中：

x：爆發時的光子數(大於等於 0 之某整數)

λ：沒爆發光子數平均值

e：尤拉常數 2.71828

（四）將爆發事件的參數分群

為了研究磁星 1E2259+586 的物理機制，我們使用 HDBSCAN 非監督式演算法來進行。將許多爆發事件得到的「爆發時間、爆發時間間隔（T90）、爆發時光子數、軟光子(能量 2~4 kev)數目、硬光子(能量 4~10 kev)數目、硬指數(Hardness ratio)、爆發與前一個爆發所經過時間」分群。

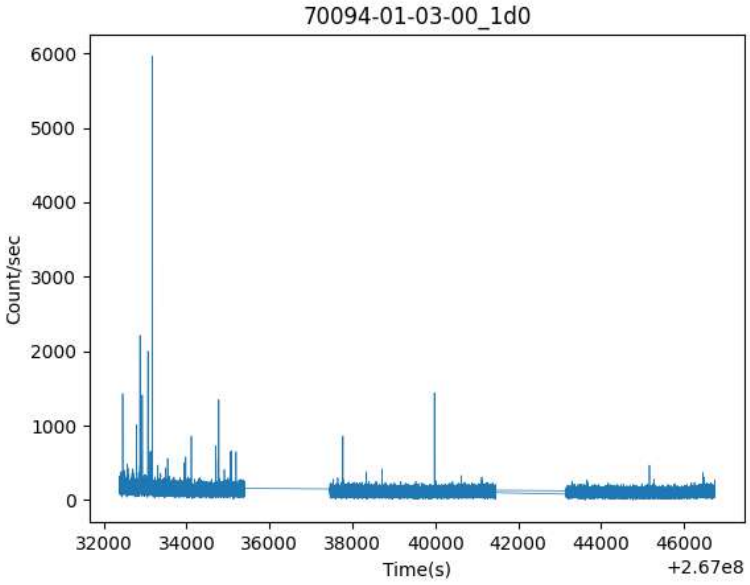
（五）硬指數

「硬指數」(Hardness ration)的定義是「高能量光子數/ 低能量光子數)」，本研究是用 (4~12 keV) / (2~4 keV)。此指數描述爆發能量分佈的一個指標，較高的值表示較高能量的發射。

（六）本次研究爆發現象的時間

在 2002 年 6 月 18 日的觀測期間(RXTE observation identification 70094-01-03-00)，於 UT15:39:18 開始，經過 250 分鐘。所選取的能量範圍為 12keV 以下。

圖八：本次研究爆發現象的時間段的 lightcurve



來源：作者自行繪製

（七）T90

用 Bayesian block 方法找到爆發的邊界值，在發的邊界值中計算第 5%偵測到的光子到第 95%偵測到的光子所經過的時間，單位為毫秒(ms)。

伍、預期結果、已有初步之結果

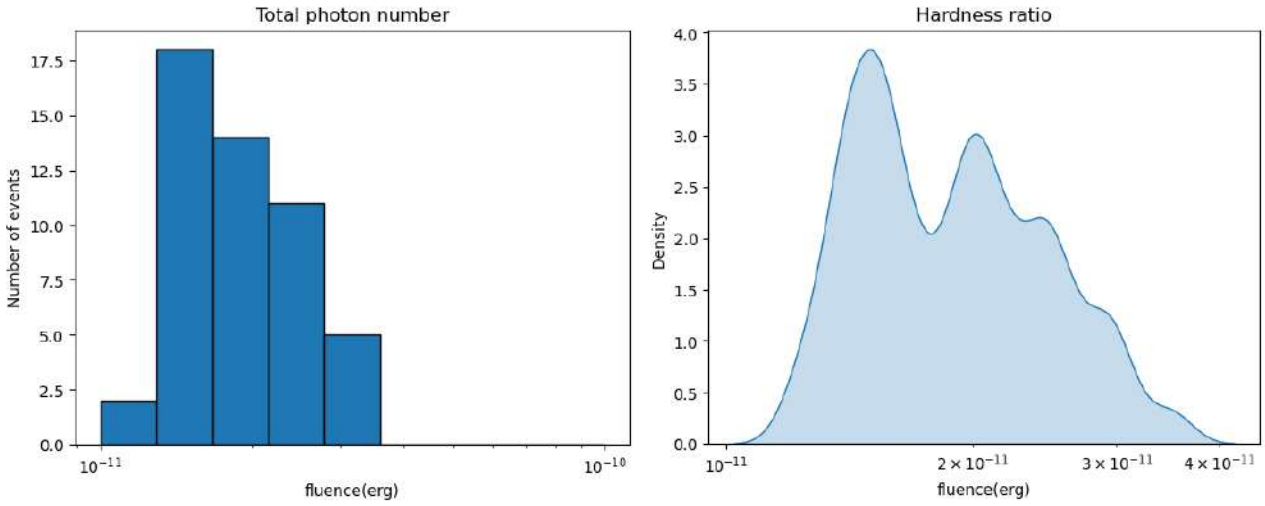
一、磁星爆發時測量到的光子數量（counts）和對應的爆發數量之間關係，由圖九可以得到以下結論

（一）磁星爆發具有多樣性，有時釋放大量光子，有時則較少。

（二）產生「**大量光子的爆發事件較少**」，這可能是由於只有在特定條件下才能觸發這種高能量的爆發。

（三）光子數量的分佈顯示，隨著爆發光子數量的增加，事件數量就呈趨勢減少，這有助於理解不同種類爆發的頻率。

圖九：磁星爆發時測量到的能量(fluence)和對應的爆發數量之關係，左為直方圖、右為 KED 圖



來源：作者自行繪製

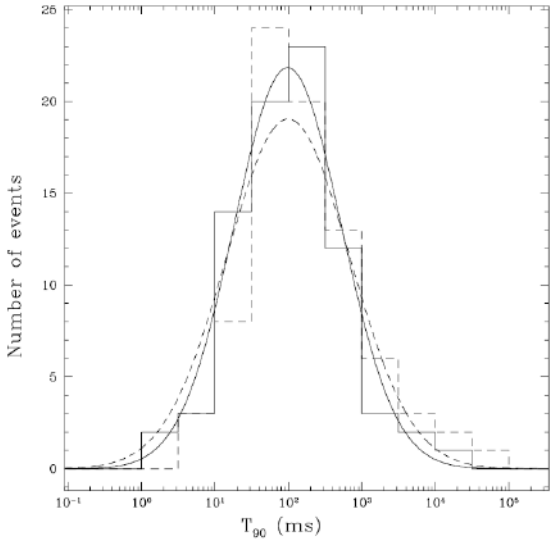
二、在磁星爆發時間間隔中可能會因為尋找爆發時的 Bayesian block 邊界設定不一致而造成差異，因此將我們將換為 T90 較為精準，並將們所找的與參考論文中的進行比對，發現結果不太一樣，我們想可能是以下因素所導致：

（一）光子能量取的不一樣，我們為 12keV 以下，文獻為 2~60keV

（二）我們使用 Bayesian block 方法尋找時會因 p0 值設不同而有所差異

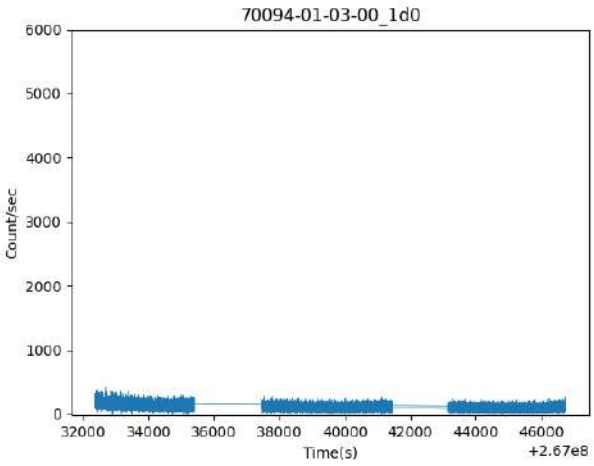
（三）確認是爆發時的信心水準取的不一致

圖十：參考文獻的 T90 與數量之關係



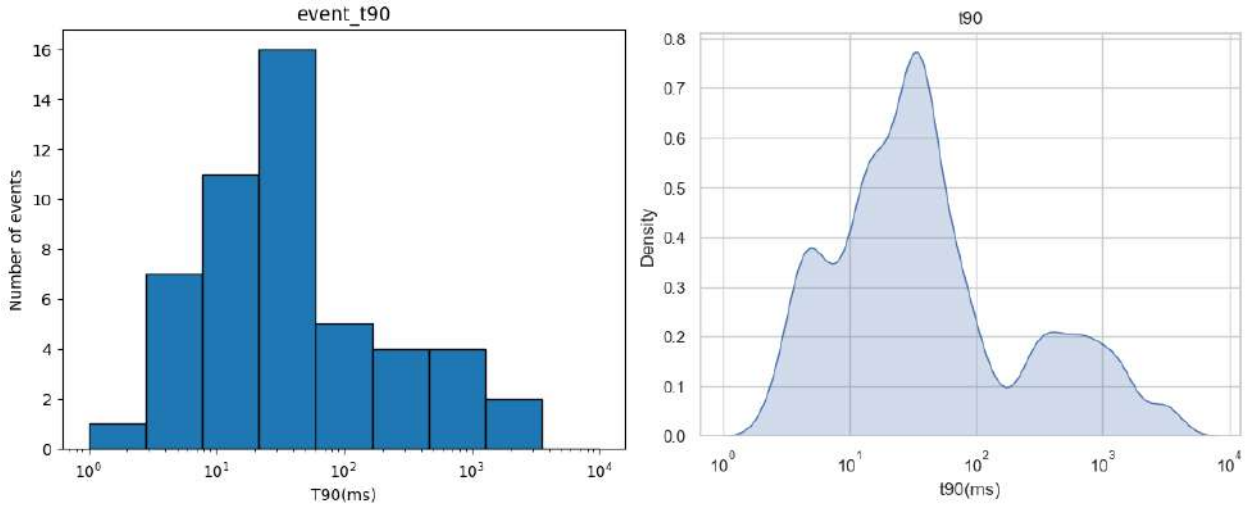
來源：參考文獻十的圖 4

圖十一：移除掉爆發後的 Light



來源：作者自行繪製

圖十二：磁星爆發時間間隔與這些間隔出現次數之關係，
左為直方圖、右為 KED 圖



來源：作者自行繪製

我們將爆發移除後的 Light curve 發現並沒有爆發現象（如圖十一），因此可以先暫時認為所找到的爆發是正確的。爆發時間間隔與這些間隔出現次數之關係，由圖十二可以得到以下結論

（一）爆發時間間隔不是均勻分布的，而是集中在特定的幾個範圍內，這表明可能有多種爆發機制

（二）因為有幾個爆發時間間隔多於其它，可能暗示磁星爆發具有某種「週期性」或重複性的特徵。

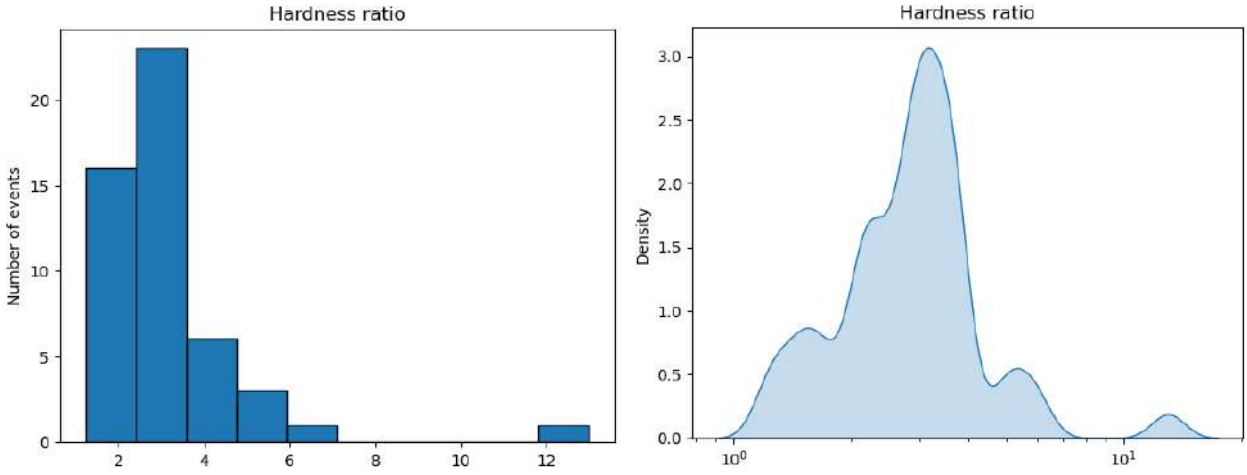
（三）毫秒等級的爆發時間間隔次數相當少。

三、磁星爆發時測量到的硬度比（hardness ratio）與對應的事件次數之間的關係，由圖十三可以得到以下結論

（一）磁星爆發產生的光子能量分布具有多樣性，從中低能光子到高能光子均有。

（二）高硬度比的爆發事件較為常見，這可能是因為磁星的標準爆發模式傾向於產生更多的高能光子。

圖十三：磁星爆發時測量到的硬度比（hardness ratio）與對應的事件次數之間的關係，左為直方圖、右為 KED 圖

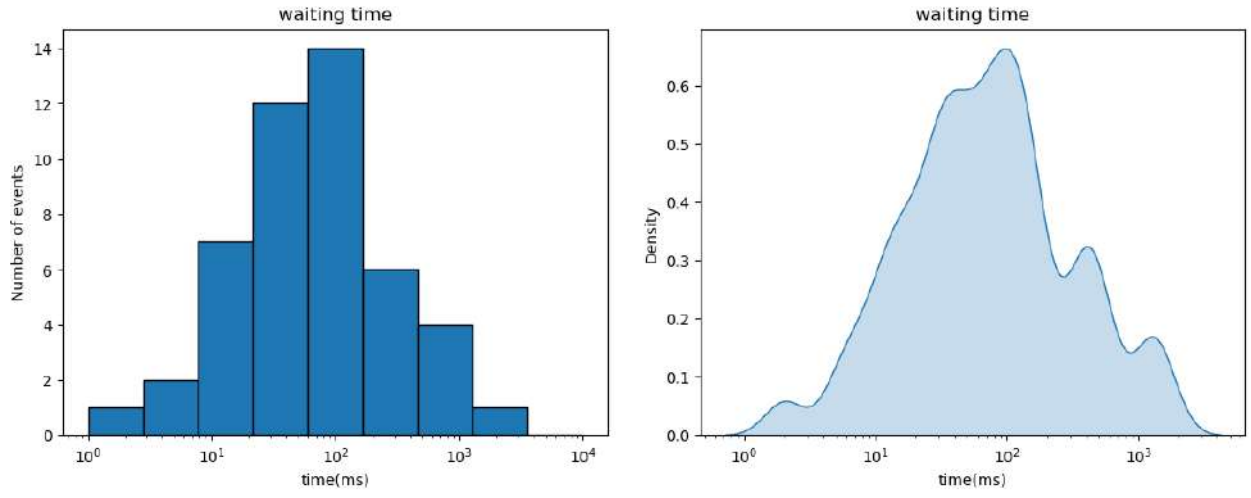


來源：作者自行繪製

四、磁星爆發時測量到的硬度比（hardness ratio）與對應的事件次數之間的關係，由圖十三可以得到以下結論

（一）分佈以 100 毫秒附近數量最高

圖十四：磁星爆發爆發與前一個爆發所經過時間與對應的事件次數之間的關係，左為直方圖、右為 KED 圖

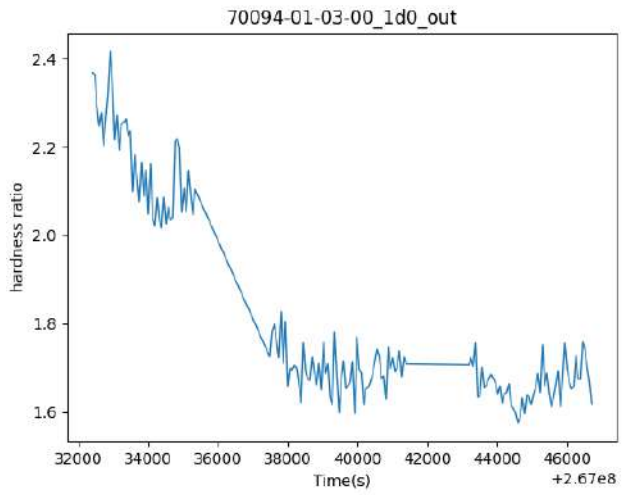
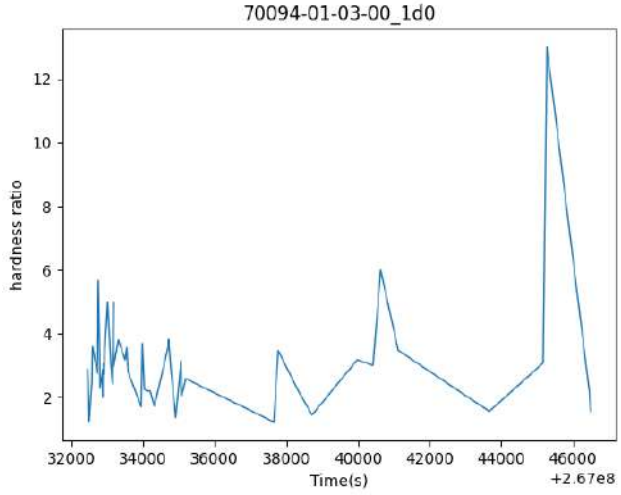


來源：作者自行繪製

五、硬指數在磁星上的演變，結果如下

（一）性質：在這段時間磁星所釋放出的能量逐漸變低，但爆發的 hardness ratio 並未依圖十五的整體趨勢改變。

（二）可能機制：在圖十五中，時間 32000~35000 的爆發數量較多，後面相對較少，可以因為密集的爆發釋放了大量的內部能量，讓其磁場衰減等。

圖十五、硬指數隨時間變化之關係	圖十六、爆發硬指數隨時間變化關係
	
來源：作者自行繪製	來源：作者自行繪製

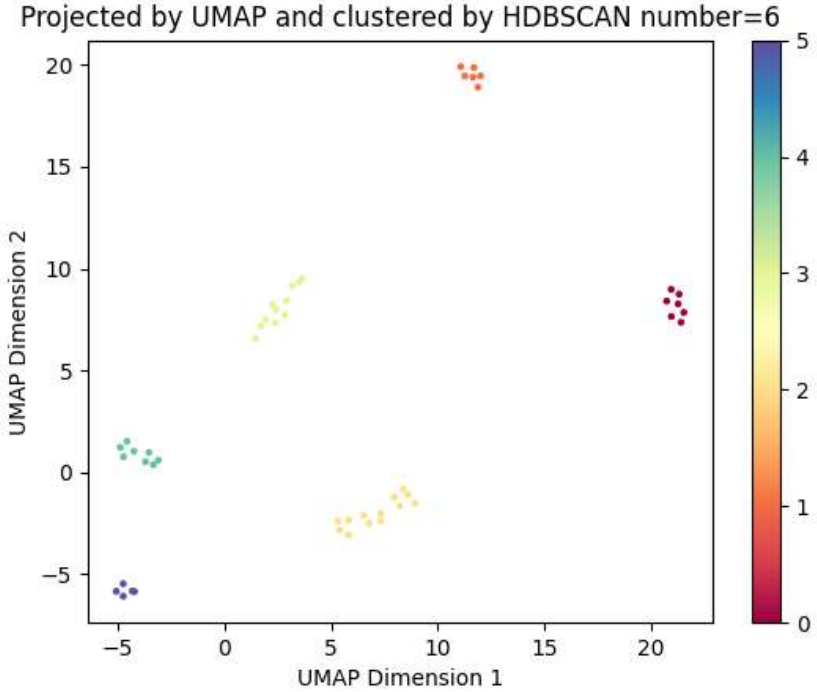
六、在分群前，為了不讓某些參數數字過大影響結果，因此分別對其做了一些處理，以下為處理方式：

（一）爆發時間：每個爆發的時間減去第一顆光子進來的時間

（二）爆發時間間隔、爆發時測到的光子數、硬指數：取 log 值

先用 UMAP 將爆發參數降維之後，再利用 HDBSCAN 分群演算法進行分群得到圖十四與表一如下，可以發現每一群的特性：

圖十七：利用 HDBSCAN 將爆發時的參數分群，可以看到分出 6 群與每群的爆發的數量多寡



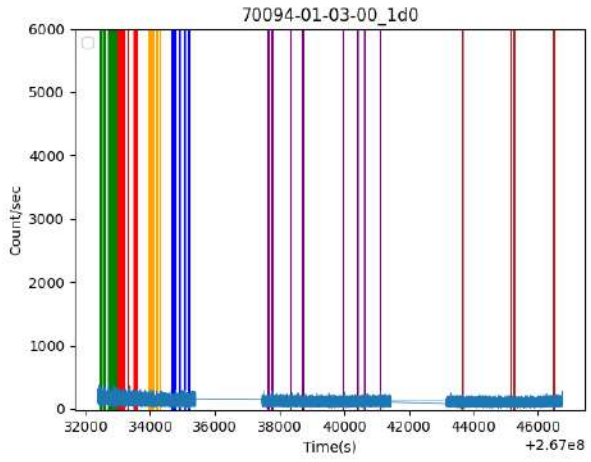
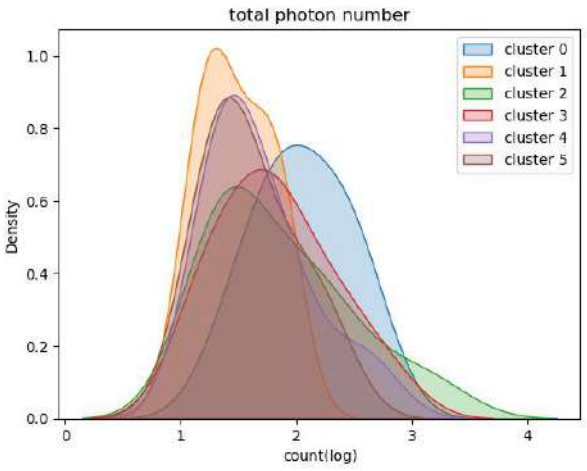
來源：作者自行繪製

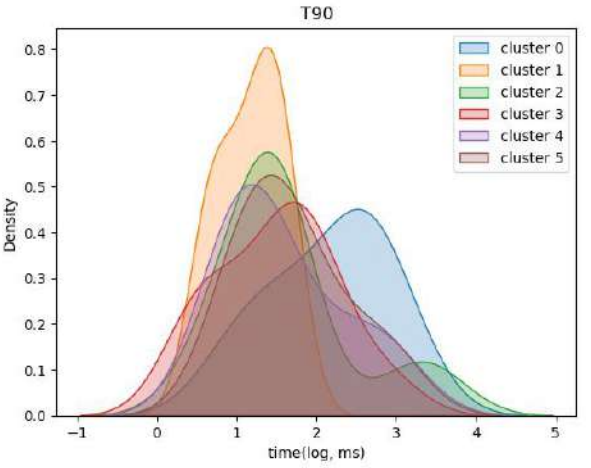
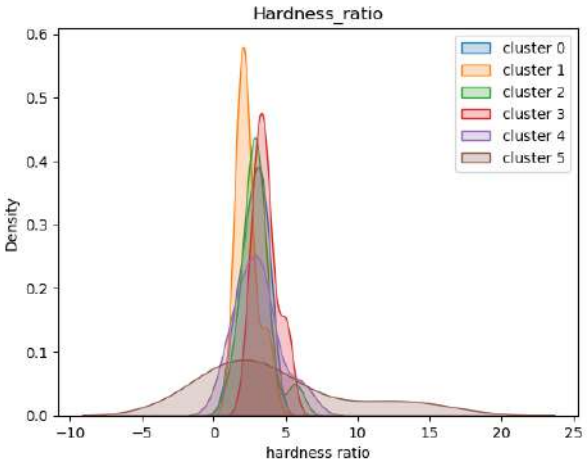
我們將這 6 群用表格整理

表一：利用使用 HDBSCAN 分群演算法找出 6 種爆發群組

cluster label	爆發時間（秒）	爆發時間間隔 T90(log 值毫秒)	爆發時測到的光子數（log 值）	硬指數
0	2521.264	2.15±0.75	2.05±0.42	2.78±0.86
1	1700.28	1.16±0.40	1.49±0.30	2.30±0.86
2	346.3591	1.67±0.82	1.86±0.60	2.94±1.03
3	853.3943	1.50±0.74	1.81±0.50	3.55±0.82
4	6930.385	1.58±0.76	1.67±0.45	3.00 ± 1.50
5	13011.83	1.70±0.69	1.60±0.40	4.27±4.92

來源：作者自行繪製

圖十七、不同群爆發在時間上的 KED 分佈圖（x 軸為秒取 log 值）	圖十八、爆發時測到的光子數分群 KED 分佈圖（x 軸為個數取 log 值）
	
來源：作者自行繪製	來源：作者自行繪製

圖十九、T90 分群 KED 分佈圖（x 軸為毫秒取 log 值）	圖二十、硬指數分群 KED 分佈圖
	
來源：作者自行繪製	來源：作者自行繪製

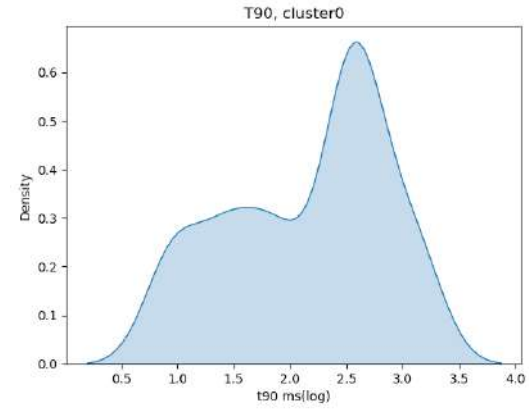
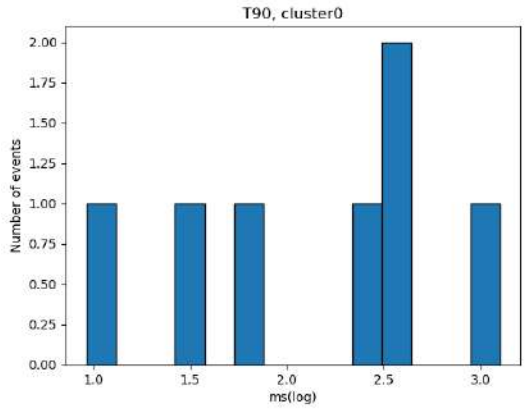
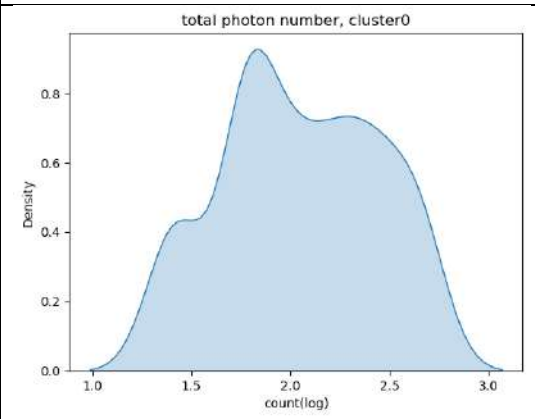
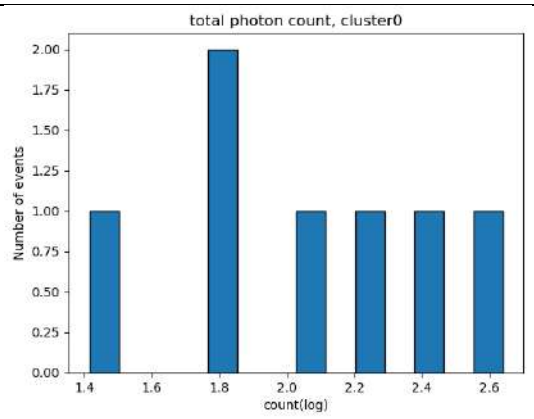
七、這五群可分析如下

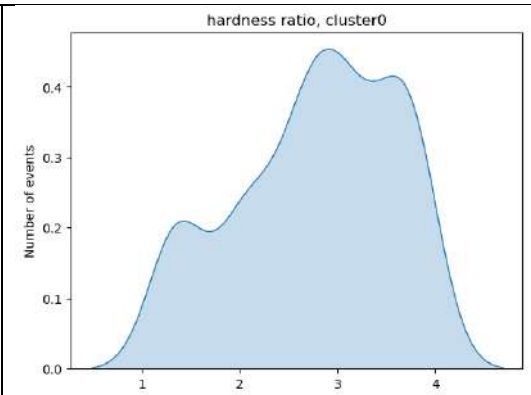
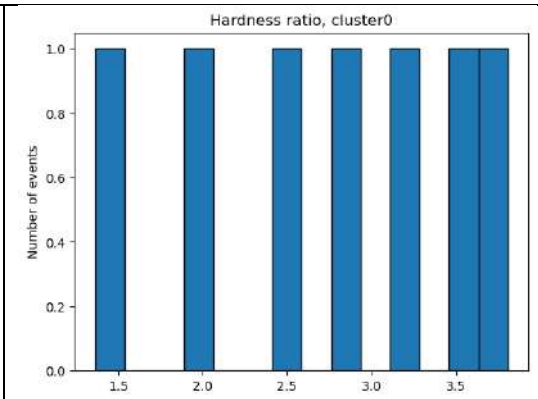
(一)、cluster label=0：

cluster label	爆發時間（秒）	爆發時間間隔（秒）	爆發時測到的光子數（log 值）	硬指數
0	2521.264	2.15±0.75	2.05±0.42	2.78±0.86

1、性質：擁有最長的爆發時間間隔，硬指數都在偏低。這可能表示這類爆發是較為「緩慢且穩定」的事件，也可從分群圖十七可看出此類爆發數量是較少的。

2、可能機制：磁星地殼的長期應力積累可能導致能量在較長的時間內緩慢而穩定地釋放，這可能不會造成劇烈的高能輻射爆發。

	KED 圖	直方圖
爆發時間間隔（秒）		
爆發時測到的光子數（個）		

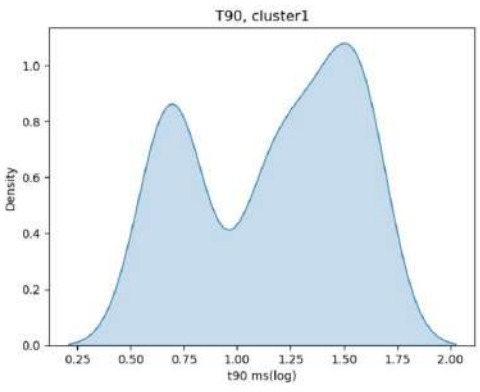
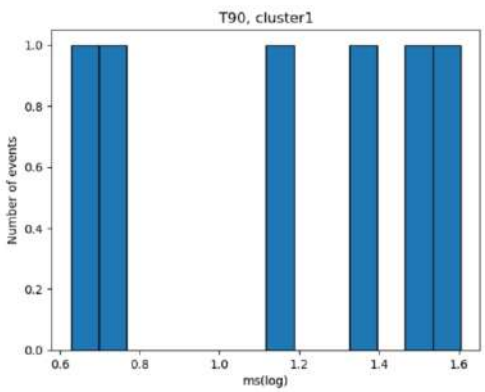
硬指數		

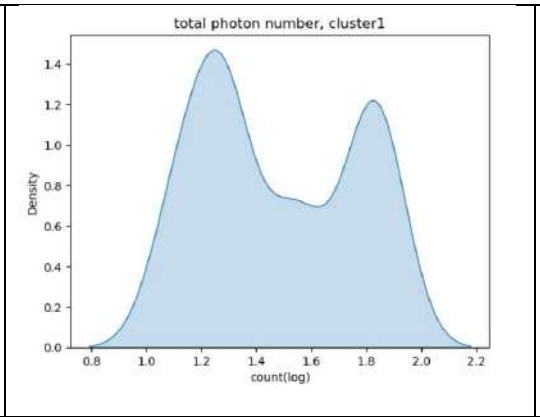
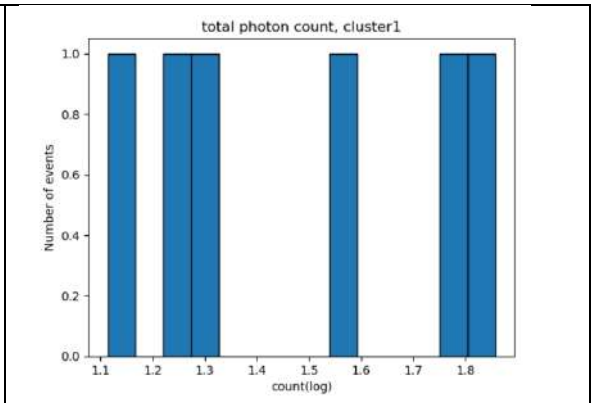
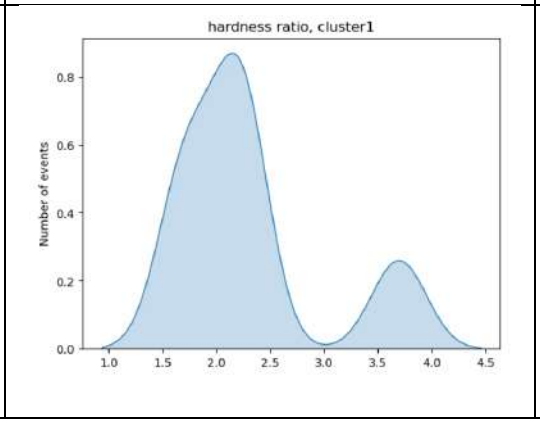
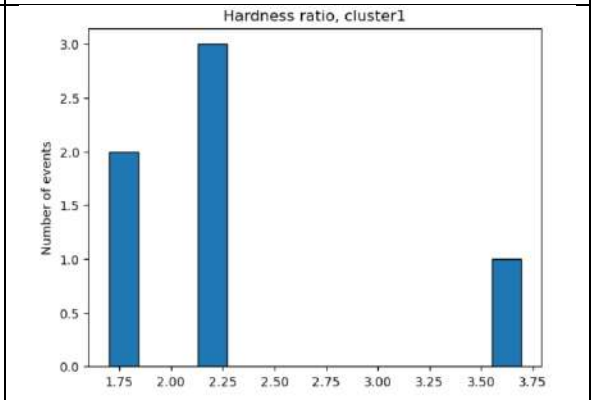
(二)、cluster label=1：

cluster label	爆發時間（秒）	爆發時間間隔（秒）	爆發時測到的光子數（個）	硬指數
1	1700.28	1.16±0.40	1.49±0.30	2.30±0.86

1、性質：爆發時間間隔最少，光子數最少，硬指數最低。可能是「能量低且快速」爆發事件。

2、可能機制：這些爆發涉及較低能量的過程。這可能是由於磁星磁場強度較低的區域發生的活動或是地殼小範圍的應力釋放。

	KED 圖	直方圖
爆發時間間隔（秒）		

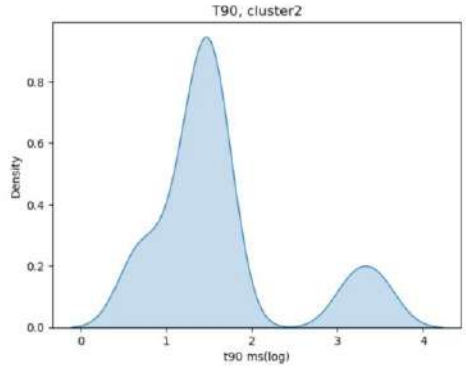
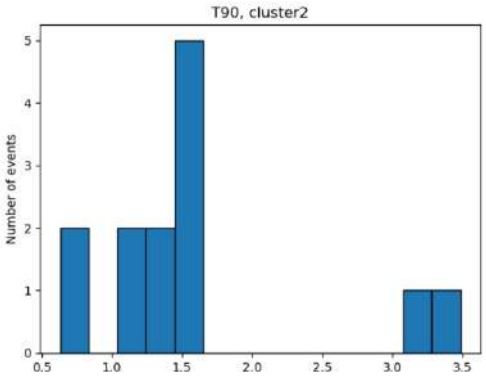
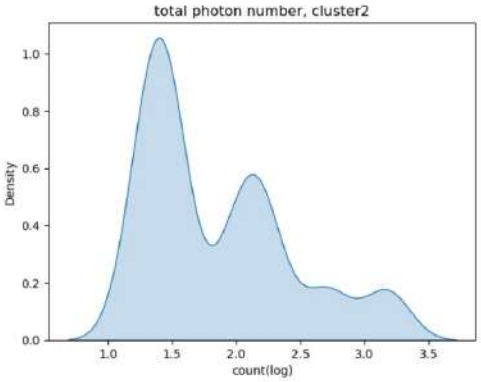
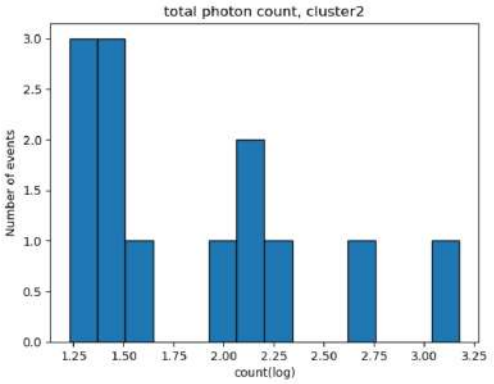
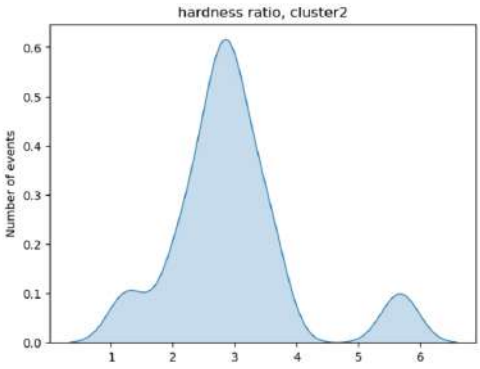
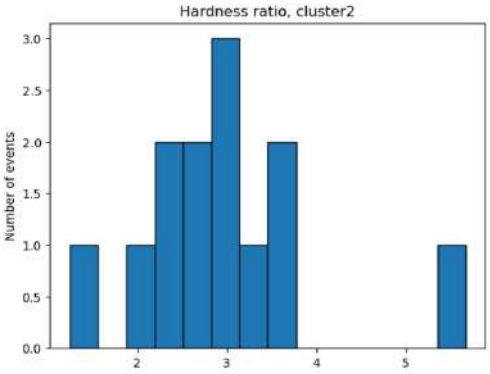
爆發時測到的光子數（個）		
硬指數		

(三)、cluster label=2：

cluster label	爆發時間（秒）	爆發時間間隔（秒）	爆發時測到的光子數（個）	硬指數
2	346.3591	1.67±0.82	1.86±0.60	2.94±1.03

1、性質：爆發時間間隔較長、光子數偏多、硬指數適中。顯示這是一個「**緩慢且穩定**」的事件，也可從分群圖十七可看出此類爆發數量是偏多的。

2、可能機制：磁星地殼的長期應力積累可能導致能量在較長的時間內緩慢而穩定地釋放，這可能不會造成劇烈的高能輻射爆發。

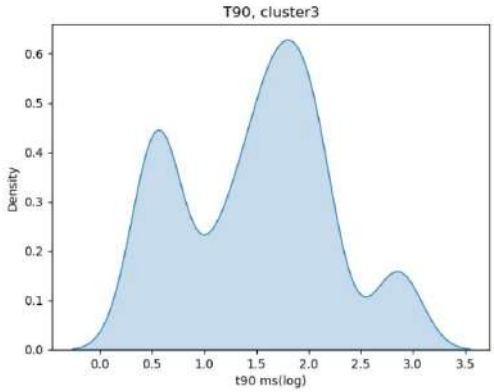
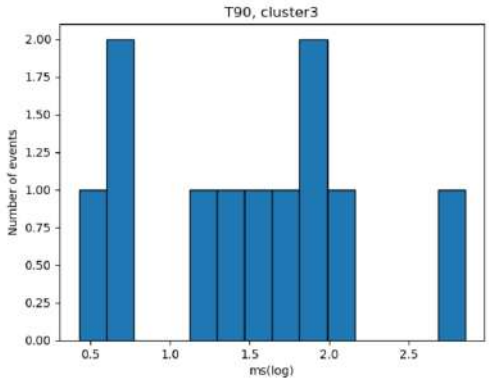
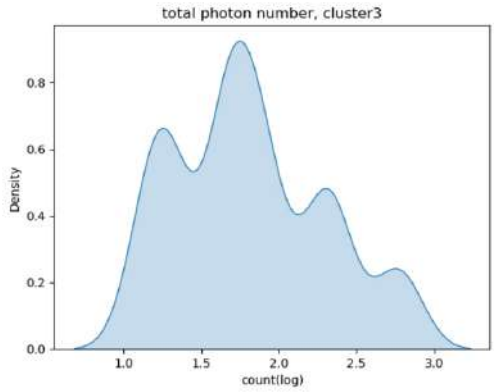
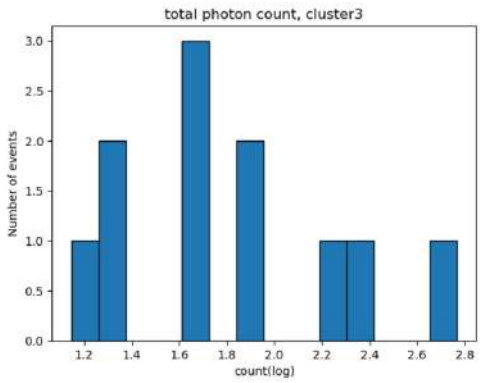
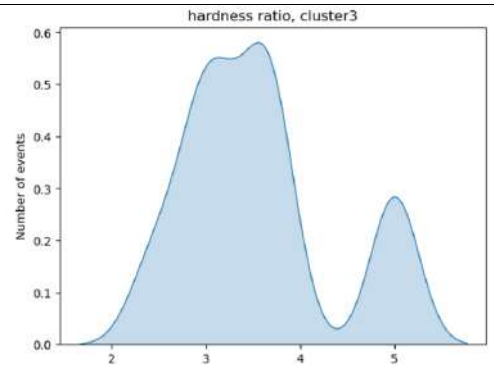
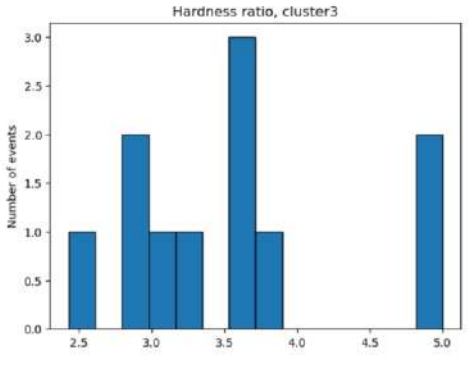
	KED 圖	直方圖
爆發時間間隔（秒）		
爆發時測到的光子數（個）		
硬指數		

（四）、cluster label=3：

cluster label	爆發時間（秒）	爆發時間間隔（秒）	爆發時測到的光子數（個）	硬指數
3	853.3943	1.50±0.74	1.81±0.50	3.55±0.82

1、性質：爆發時間間隔較短，光子數較多，硬指數偏高。顯示這是一個「高能量且快速」的事件，也可從分群圖十七可看出此類爆發數量是偏多的。

2、可能機制：包括磁場重連、星震和磁場能量釋放。

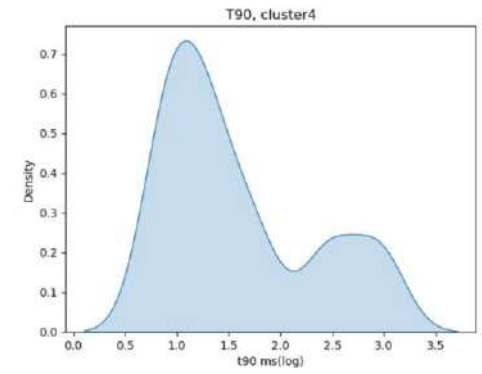
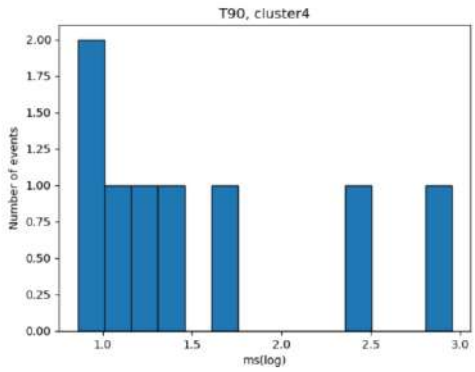
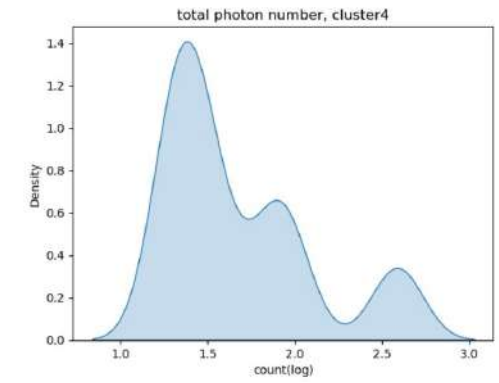
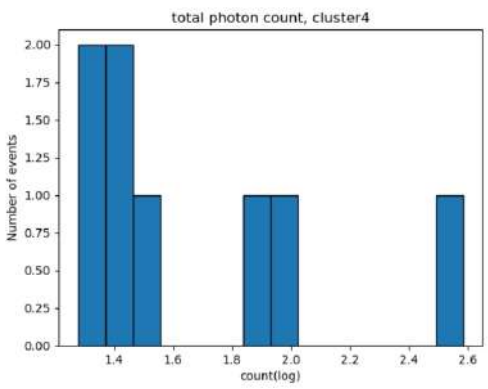
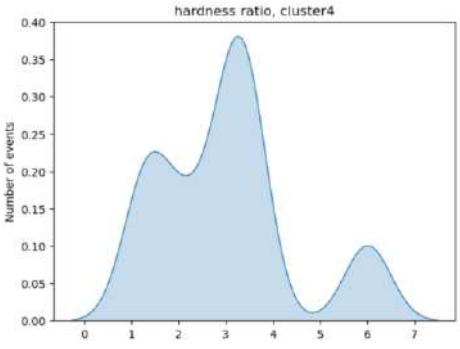
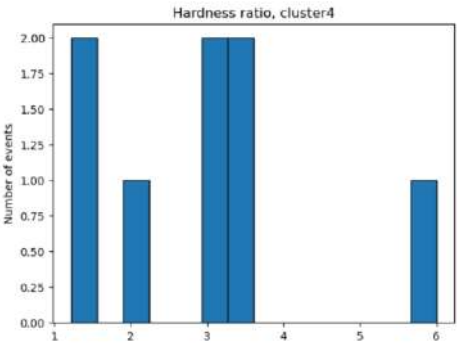
	KED 圖	直方圖
爆發時間間隔(秒)		
爆發時測到的光子數(個)		
硬指數		

（五）、cluster label=4：

cluster label	爆發時間（秒）	爆發時間間隔（秒）	爆發時測到的光子數（個）	硬指數
4	6930.385	1.58±0.76	1.67±0.45	3.00±1.50

1、性質: 爆發性質 Cluster 3 相近

2、可能機制: 與 Cluster 3 相近，週期性的出現。

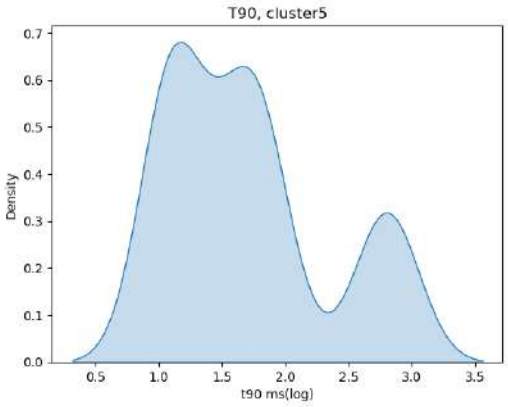
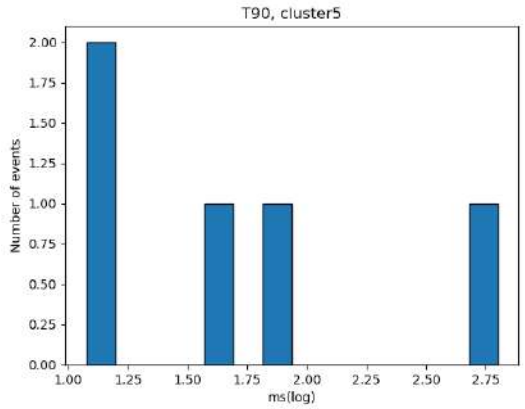
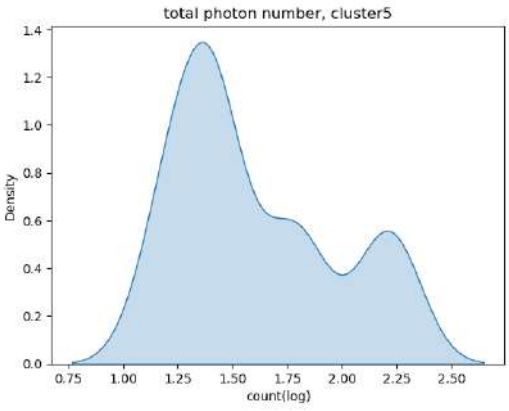
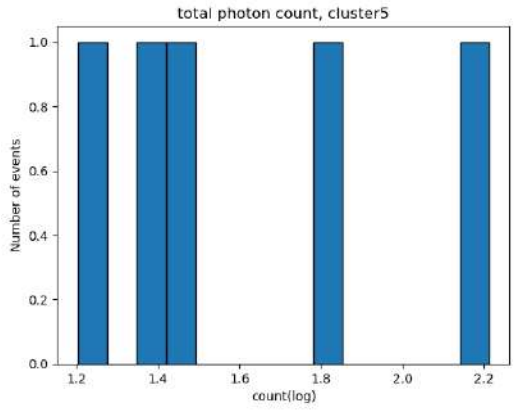
	KED 圖	直方圖
爆發時間間隔(秒)		
爆發時測到的光子數(個)		
硬指數		

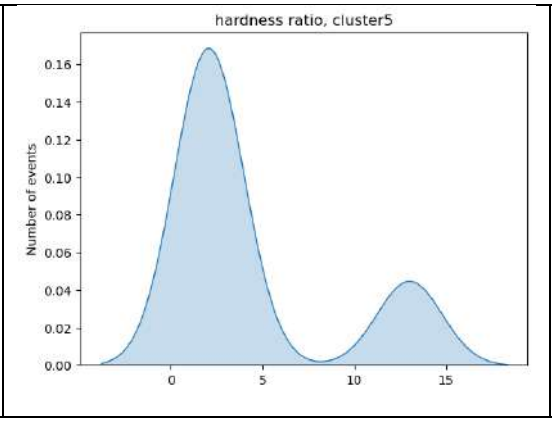
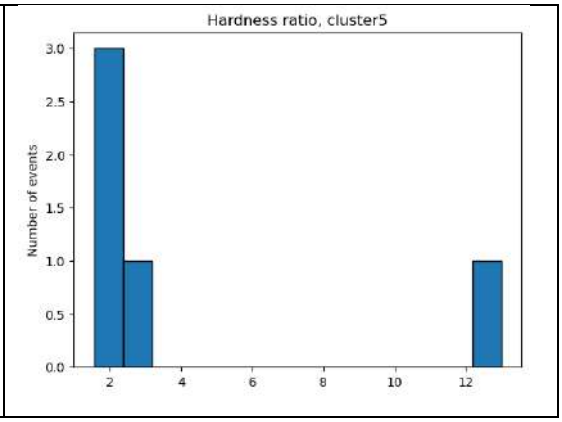
（六）、cluster label=5：

cluster label	爆發時間（秒）	爆發時間間隔（秒）	爆發時測到的光子數（log 值）	硬指數
5	13011.83	1.70±0.69	1.60±0.40	4.27±4.92

3、性質：擁有較長的爆發時間間隔，光子數偏少，硬指數都最高，但有可能是有一個特別高能能量的爆發所造成。從分群圖十二可看出此類爆發數量是最少的。

4、可能機制：這可能對應於磁星的「**快速且劇烈的能量釋放**」，如「地殼的突然斷裂」或「快速的磁場重組」

	KED 圖	直方圖
爆發時間間隔(秒)		
爆發時測到的光子數(個)		

硬指數		

二、將上述整理起來可以知道磁星的爆發有下面的類別，如表得到以下的表格

表二：磁星爆發的類別

磁星爆發的類別	群組(cluster label)
短暫且高能爆發	1、5
中等持續與能量爆發	2、3、4
較長持續且溫和爆發	0

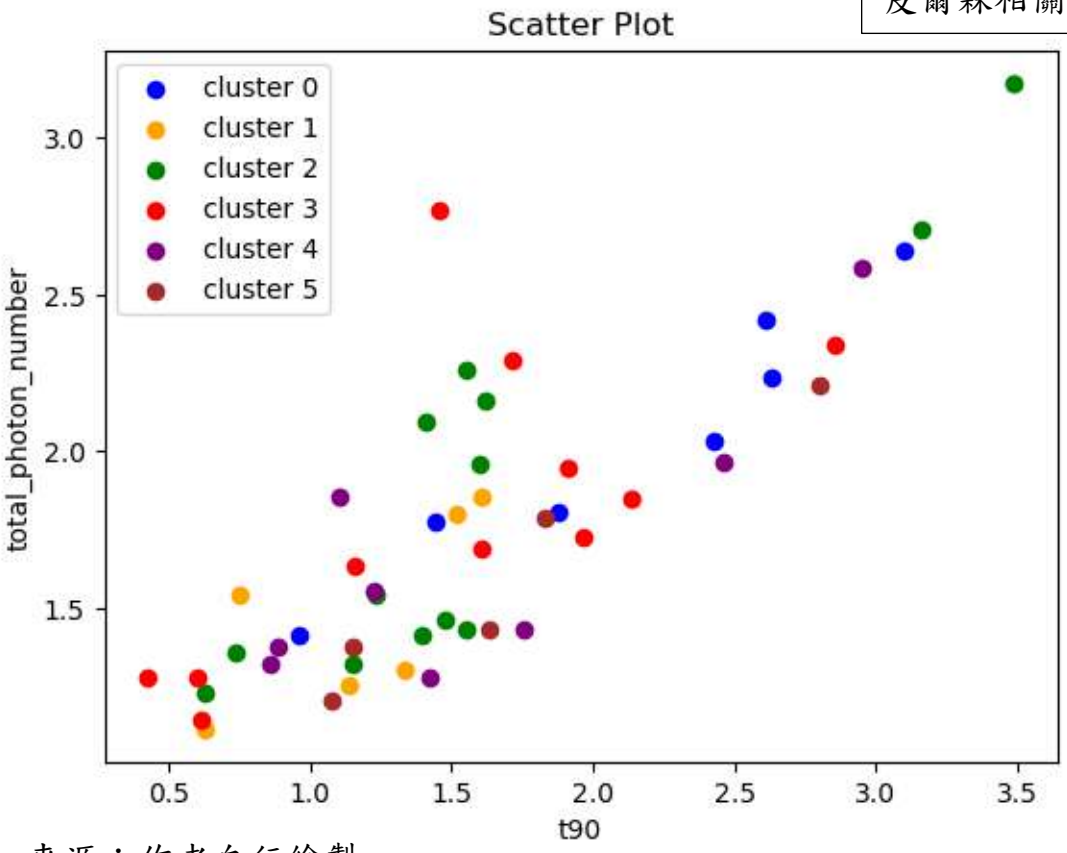
來源：作者自行繪製

三、相關性

在爆發參數之間「爆發時間間隔」與「爆發時測到的光子數」呈現正相關，代表較長的時間間隔會累積更多能量，導致更多光子釋放。

圖二十一：爆發時間間隔與爆發時測到的光子數的相關性圖

皮爾森相關係數：0.9112



來源：作者自行繪製

爆發參數之間的相關係：

表三：爆發參數之間的相關係數

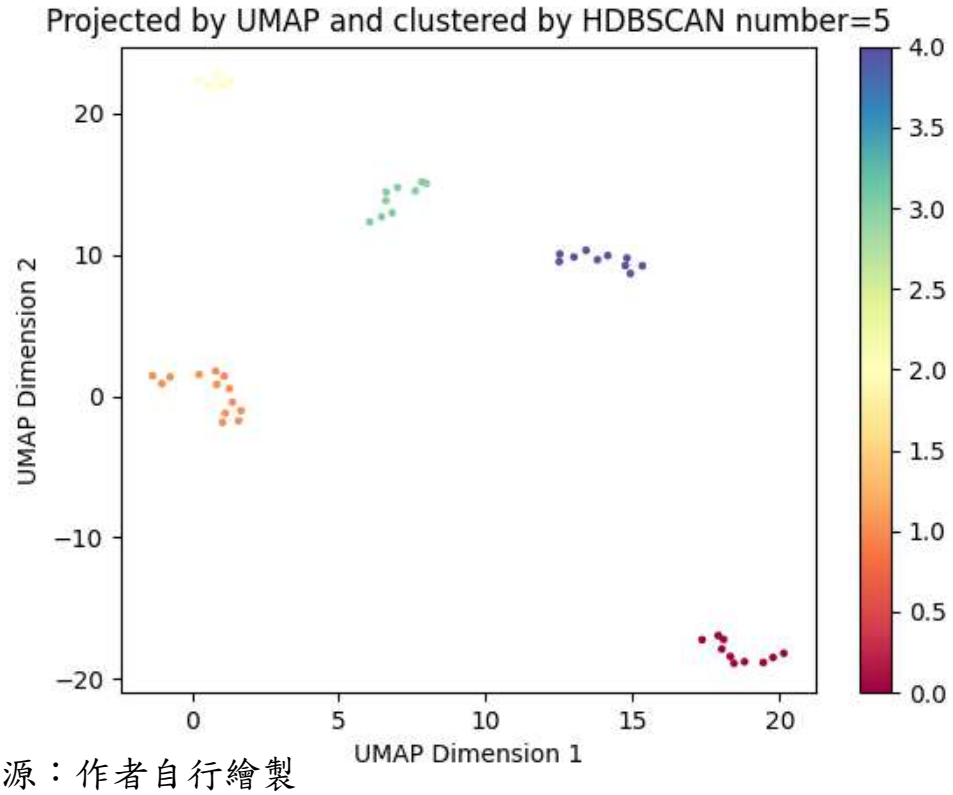
	爆發時間	爆發時間間隔	爆發時測到的光子數	硬指數
爆發時間				
爆發時間間隔	-0.033			
爆發時測到的光子數	-0.133	0.9112		
硬指數	0.1106	-0.072	-0.113	

來源：作者自行繪製

四、其它分群測試

從上述分群結果中可以看出"cluster 3"和"cluster 4"非常相似，因此我們將「爆發時間」移除後增加「爆發與前一個爆發時間」分群了一次，探討「爆發時間」參數影響分群的呈度，結果如下：

圖二十二：用 4 個爆發參數分群，可以看到分出 5 群與每群的爆發的數量多寡



來源：作者自行繪製

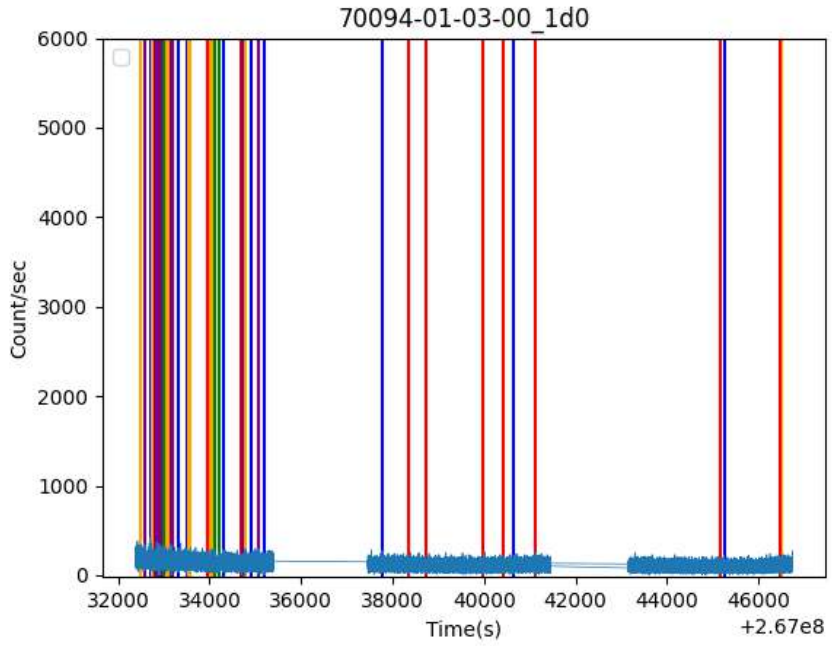
表四：利用使用 HDBSCAN 分群演算法找出 4 種爆發群組

cluster label	爆發與前一個爆發時間間隔	T90(log 值毫秒)	爆發時測到的光子數(log 值)	硬指數
0	141.9352	1.74±0.87	1.74±0.51	4.10±3.37
1	40.53523	1.68±0.80	1.83±0.48	2.98±0.51
2	79.93948	1.51±1.17	1.79±0.81	2.95±0.51
3	730.5508	1.62±0.71	1.70±0.40	2.64±0.51
4	11.23383	1.60±0.50	1.84±0.48	3.18±0.51

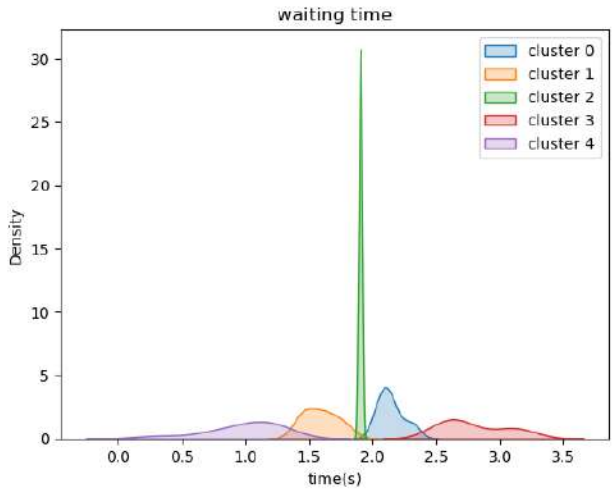
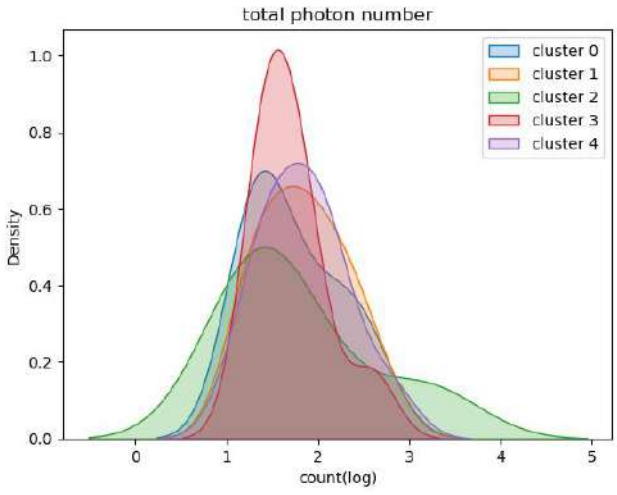
來源：作者自行繪製

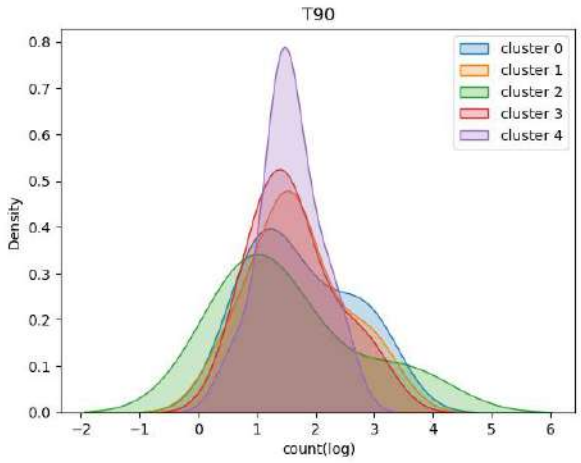
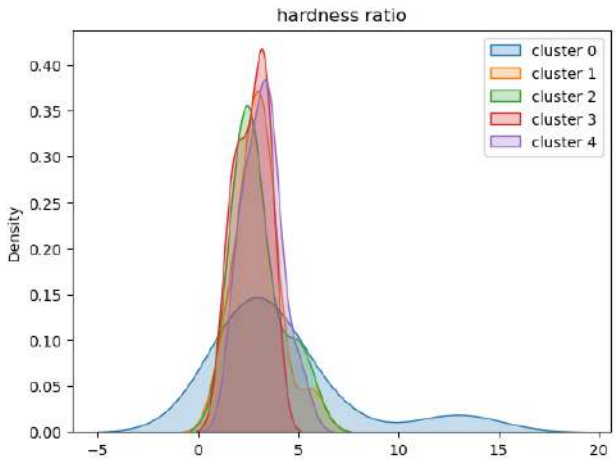
以下各個爆發參數在不同群的分佈，從分佈結果中可以看出每群中的爆發和有「爆發時間」的分群是完全不一樣的：

圖二十三：不同群分爆發在時間上的分佈



來源：作者自行繪製

圖二十四、爆發與前一個爆發時間間隔分群 KED 分佈圖（x 軸為秒取 log 值）	圖二十五、爆發時測到的光子數分群 KED 分佈圖（x 軸為個數取 log 值）
	
來源：作者自行繪製	來源：作者自行繪製

圖二十六、T90 分群 KED 分佈圖 （x 軸為毫秒取 log 值）	圖二十七、硬指數分群 KED 分佈圖
	
來源：作者自行繪製	來源：作者自行繪製

結論：

從上述的分群結果，可以看出「爆發與前一個爆發時間間隔」性質明顯的不同，並且在其它爆發參數性質有所不同，爆發更在時間上與前一次分群有所不同。顯示了爆發很有可能是週期性的出現。

二、與磁星 SGR 1935+2154(有 FRB 現象)的相關性

在這個時間段所分群的爆發以「短暫且高能爆發」這群最有可能和 FRB 有相關性，但以目前的證據難以直接說和 FRB 有相關性，因此我們還需要尋找其它爆發參數再加以確認 FRB 會滿足一些特性，例如它的高頻信號會比低頻信號到達地球、爆發時間間隔是毫秒等。

陸、本計畫之創見性及其未來應用

一、找一些 FRB 的特性相似（例如，釋放能量的規模、爆發的持續時間或者高頻與低頻電磁波到地球時間的偏移性等等）以利找出磁星與 FRB 的更多的相聯性。

二、圖十還是可以看到硬指數高的有一些事件，我們要研究這些事件為何會發出這麼多的「高能量光子」。

三、用不同的分群法分析比較。

四、大爆發與小爆發的關連性，例如有的小爆發是在大爆發後，有的小爆發是自己產生的。

柒、結論

一、由分群可以看出爆發的多樣性，包含各群的性質與發生次數多寡，可以讓我們對磁星爆發的機制更了解。

二、爆發時間間隔與爆發時測到的光子數呈現正相關，相關係數為 0.9112，這意味著較長的時間間隔會累積更多能量，導致更多光子釋放。

三、由統計圖表可以發現爆發可能隱含著「週期性」，也許是自轉週期、地殼受的應力或磁場變化經過同樣時間(有週期性)累積而爆發有關。

四、爆發的從硬指數不會隨磁星時間變化而改變。

五、雖然目前還看不出 1E2259+586 有任何關連性，不過我們可以把此方法複制到其它磁星上，看是否能找出更多顆有 FRB 的磁星。

捌、參考資料（文獻）及其他

一、Andersen, B. C., Bandura, K. M., Bhardwaj, M., Bij, A. ; Boyce, M. M. search by orcid ; Boyle, P. J. et al., (2020); *A bright millisecond-duration radio burst from a Galactic magnetar,* arXiv:2005.10324v2; CHIME/FRB Collaboration

二、Bo Han Chen, Tetsuya Hashimoto , Tomotsugu Goto, et al. (2022), *Uncloaking hidden repeating fast radio bursts with unsupervised machine learning*, MNRAS 509, 1227-1236

三、CHIME/FRB Collaboration, (2020), *A bright millisecond-duration radio burst from a Galactic magnetar*. Nature 587, 54–58

四、E. Petroff · J. W. T. Hessels · D. R. Lorimer, (2022), *Fast radio bursts at the dawn of the 2020s*, arXiv:2107.10113v2

五、Z. G. Dai1. (2020). *A Magnetar-asteroid Impact Model for FRB 200428 Associated with an X-Ray Burst from SGR 1935+2154*. The Astrophysical Journal Letters, 897:L40

六、Lorimer, D. R. ; Bailes, M. ; McLaughlin, M. A. ; Narkevic, D. J. ; Crawford, F. (2007); *A Bright Millisecond Radio Burst of Extragalactic Origin, Science*, Volume 318, Issue 5851, pp. 777

七、圖一：磁星示意圖。取自 https://reurl.cc/13EAQD

八、圖二：磁星爆發的光變曲線。Göğüs E, Kouveliotou C, Woods PM, Thompson C, Duncan RC, Briggs MS, 2001, *TEMPORAL AND SPECTRAL CHARACTERISTICS OF SHORT BURSTS FROM THE SOFT GAMMA REPEATERS 1806-20 AND 1900+14* , ApJ 558:228-236

九、圖三：Mereghetti S. et al., (2021), *INTEGRAL Limits on Past High- energy Activity from FRB 20200120E in M81*. ApJ 921(1) :L3

十、Fotis P. Gavriil, Victoria M. Kaspi, and Peter M. Woods (2004). A Comprehensive study of the X-ray bursts from the magnetar candidate 1E 2259+586.

玖、附件（資料處理與分析程式程）

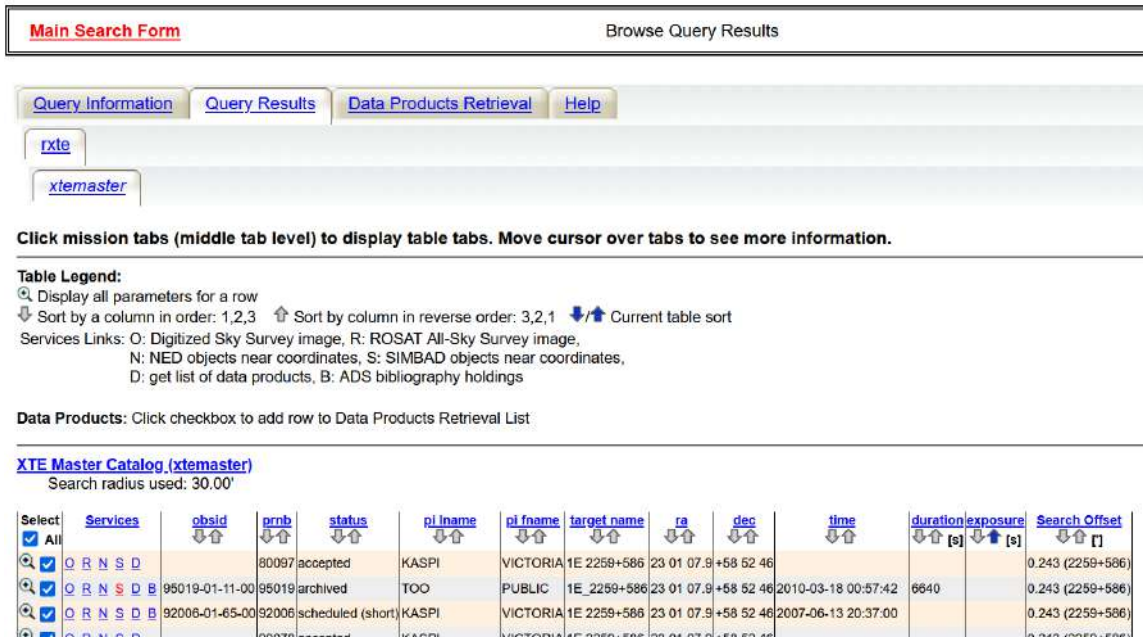
下載資料

從 heasarc 的觀測資料上下載：

https://heasarc.gsfc.nasa.gov/

所要研究的磁星 1E 2259+586

下載全部的觀測資料：



但檔案太大了，個網站一次只能下載 2GB 所以需要分成非常多段來下載

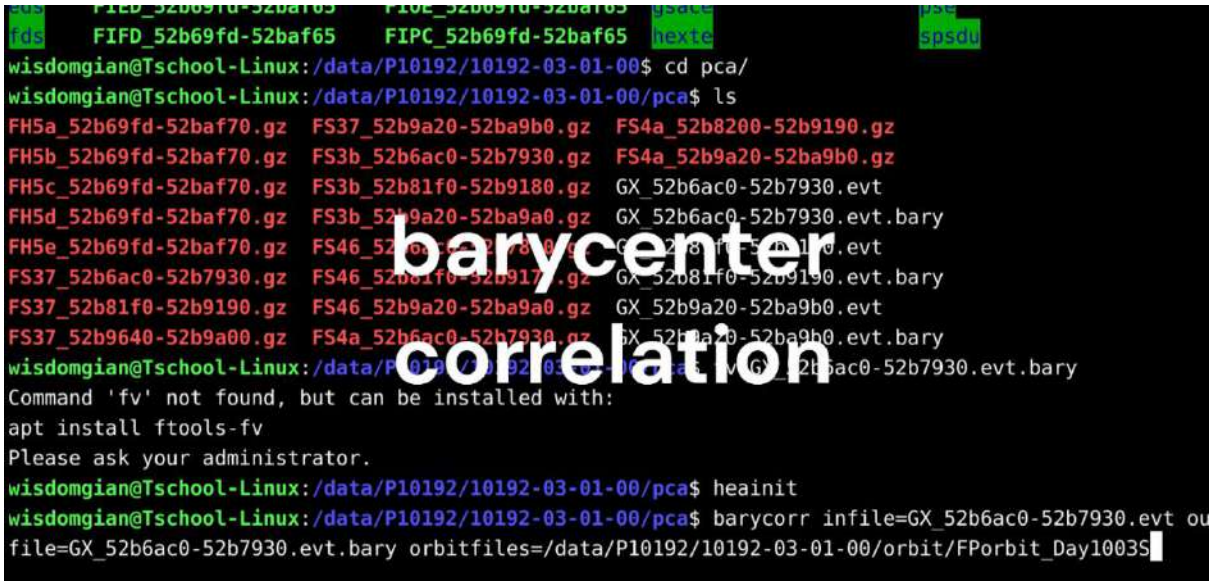
下載時間：約 3 小時

用程式做質心軌道的修正

指令：

barycorr infile=<filename> outfile=<filename> orbitfiles=<filename>

參考網址：https://heasarc.gsfc.nasa.gov/ftools/caldb/help/barycorr.html



但檔案太多了不能一個一個打指令，這時候需要寫一個 shell script

程式碼

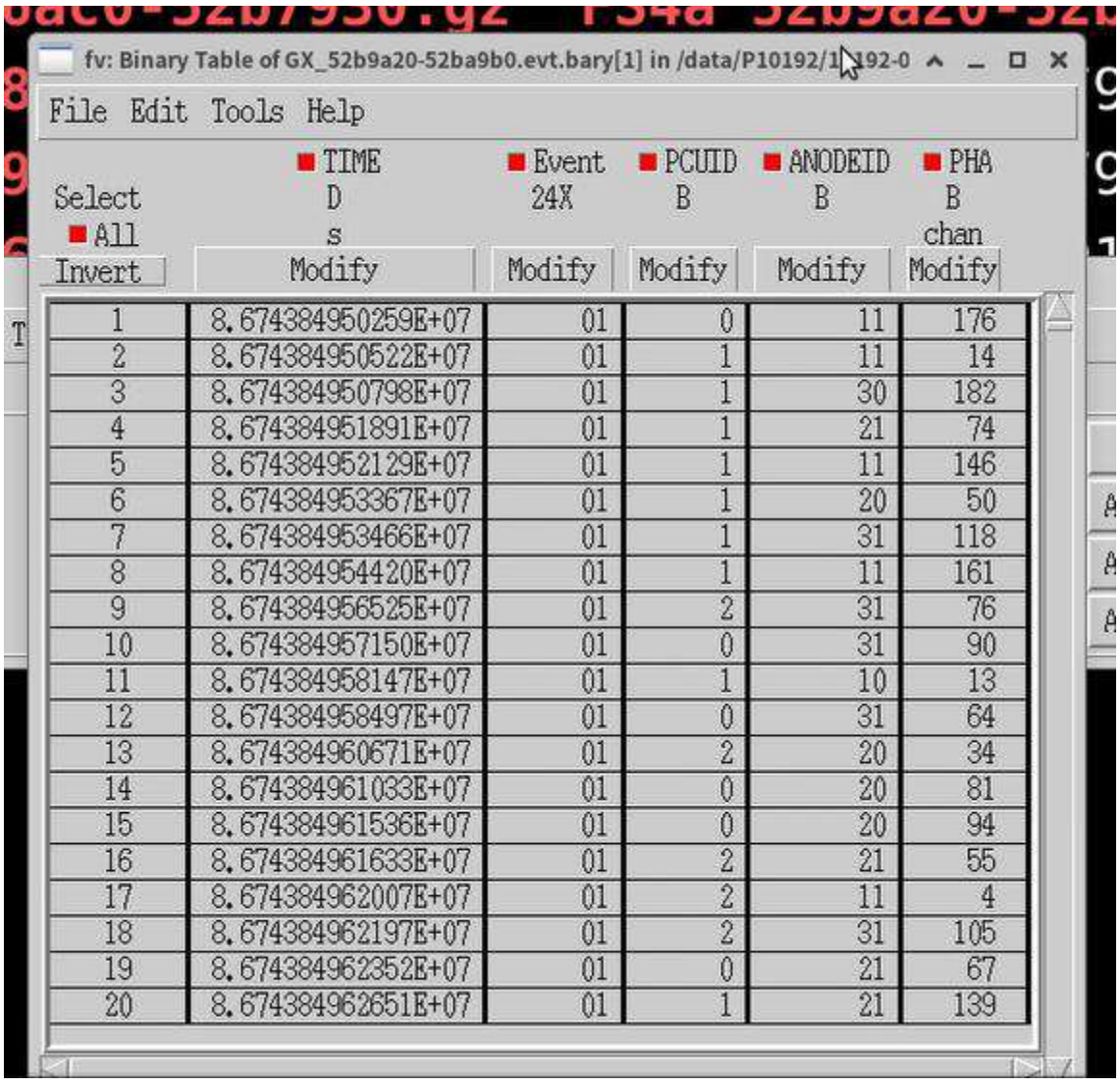
```
================================================================
#!/usr/bin/bash

path=‘find ./P* -name "1*"‘

for i in ‘find $path/pca/*evt‘
    do
        barycorr $i $i.bary $path/orbit/*

    done
================================================================
```

查看裡面的資料

指令：

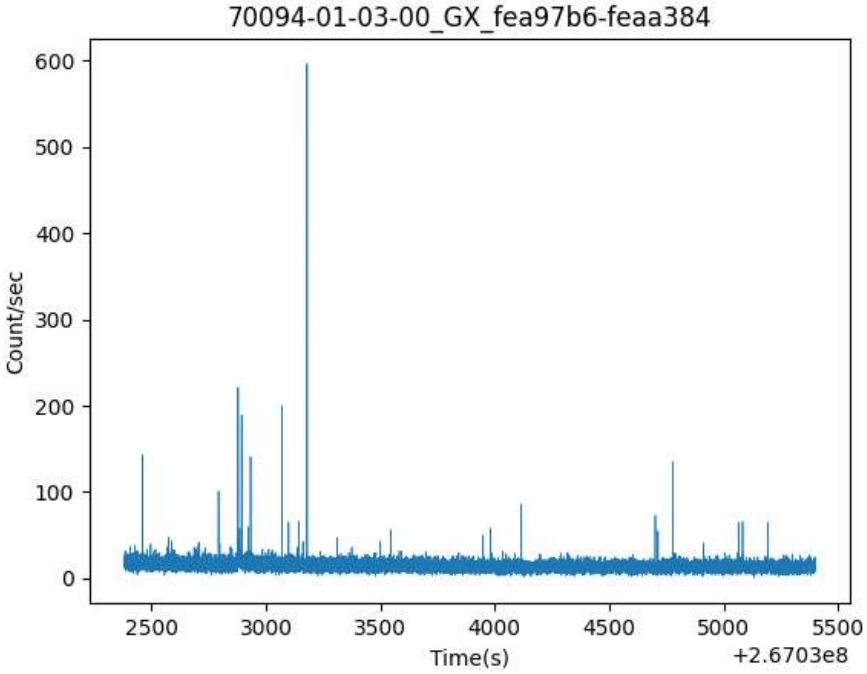
heainit

fv 檔名.bary



	TIME D s	Event 24X	PCUID B	ANODEID B	PHA B chan
	Modify	Modify	Modify	Modify	Modify
1	8.674384950259E+07	01	0	11	176
2	8.674384950522E+07	01	1	11	14
3	8.674384950798E+07	01	1	30	182
4	8.674384951891E+07	01	1	21	74
5	8.674384952129E+07	01	1	11	146
6	8.674384953367E+07	01	1	20	50
7	8.674384953466E+07	01	1	31	118
8	8.674384954420E+07	01	1	11	161
9	8.674384956525E+07	01	2	31	76
10	8.674384957150E+07	01	0	31	90
11	8.674384958147E+07	01	1	10	13
12	8.674384958497E+07	01	0	31	64
13	8.674384960671E+07	01	2	20	34
14	8.674384961033E+07	01	0	20	81
15	8.674384961536E+07	01	0	20	94
16	8.674384961633E+07	01	2	21	55
17	8.674384962007E+07	01	2	11	4
18	8.674384962197E+07	01	2	31	105
19	8.674384962352E+07	01	0	21	67
20	8.674384962651E+07	01	1	21	139

畫出 lightcurve（沒有 bayesian block）



程式碼

```
==============================start==================================
#!/usr/bin/python

import numpy as np
from astropy.io import fits

fits_file_path =  '/data/P70094/70094-01-03-00/pca/GX_fea97b6-feaa384.evt.bary'

hdul = fits.open(fits_file_path)
data = hdul[1].data
hdul.close()
data_array = np.array(data)
#=========================================
threshold = 29
rows_to_keep = []
for row in data_array:
    if row[4] <= threshold:
        rows_to_keep.append(row)
new_data_array = np.array(rows_to_keep)
#=========plot ==================
import matplotlib.pyplot as plt
first = new_data_array['TIME'][0]
last = new_data_array['TIME'][-1]
time_array = new_data_array['TIME']
```

```
interval=0.1

hist_array = np.histogram(time_array, bins = np.arange(first, last, interval))
time_array=hist_array[1]
time_array2 = np.arange(time_array[0]+0.5*interval,time_array[-1],interval)

new_hist_array=[]
for i in hist_array[0]:
    new_hist_array.append(i/interval)

plt.plot(time_array2,new_hist_array,'',linewidth=0.5)
plt.title("70094-01-03-00_fea97")
plt.xlabel("Time(s)")
plt.ylabel("Count/sec")
plt.show()

==========================end=====================================
```

算出所有的 bayesian block

運算時間：一個星期

其中一段時間算出的結果：

```
edges= [2.67043159e+08 2.67043666e+08 2.67043666e+08 2.67044376e+08
 2.67044376e+08 2.67045117e+08 2.67045117e+08 2.67045168e+08
 2.67045168e+08 2.67045168e+08 2.67045274e+08 2.67045274e+08
 2.67045770e+08 2.67046154e+08 2.67046461e+08 2.67046462e+08
 2.67046463e+08 2.67046490e+08 2.67046490e+08 2.67046491e+08
 2.67046742e+08 2.67046744e+08]
```

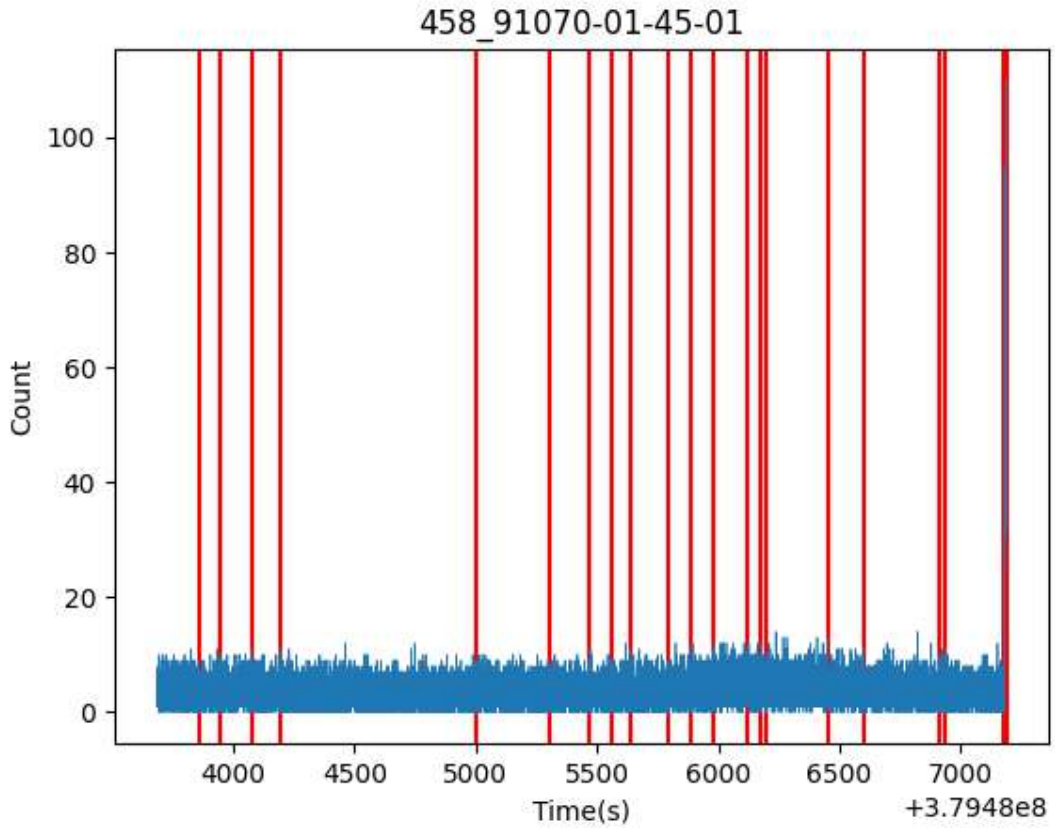
程式碼

```
=========================================================================
#!/usr/bin/python

import numpy as np
from astropy.io import fits

import matplotlib.pyplot as plt
from astropy.stats import bayesian_blocks
import os

f = open('path_test.txt')
text = []
```

```
for line in f:
    text.append(line)
for z in range(len(text)):
    text1 = text[z]

    bu = "\n"
    print("text1=",text1)
    cleaned_path = text1.strip("'\n")
    print(cleaned_path)
    fits_file_path = cleaned_path

    print("fits_file_path=", fits_file_path)

    hdul = fits.open(fits_file_path)
    data = hdul[1].data
    hdul.close()
    data_array = np.array(data)

#=========================================
    threshold = 29
    rows_to_keep = []

    for row in data_array:

        if row[4] <= threshold:

            rows_to_keep.append(row)

    new_data_array = np.array(rows_to_keep)

    time_array = new_data_array['TIME']
#=========================================
#bayesian_block


    fitness = 'events'
    edges=bayesian_blocks(time_array,fitness=fitness)

    indices=[0,-1]
    new_edges=np.delete(edges,indices)

    edges_line = len(new_edges)

#=========================================================

    name_f = cleaned_path
```

```
    file_r = name_f.split('/')[2]
    file_rr = name_f.split('/')[3]
    file_name = name_f.split('/')[5].replace('.evt.bary', '')
    a = '/data/edges/' + file_r
    b = a + '/' + file_rr

#=====================================
    if not os.path.exists(a):
        os.makedirs(a)
    if not os.path.exists(b):
        os.makedirs(b)

    array_str = str(list(new_edges))

    path = f'/data/edges/{file_r}/{file_rr}/{file_name}.txt'
    file = open(path, 'w', encoding='utf-8')
    file.write(array_str)
    file.close()
=========================================================================
```

畫出有 bayesian block 的 lightcurve



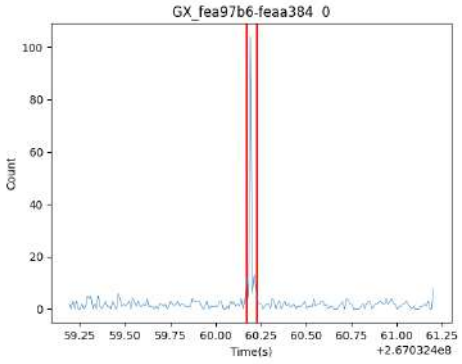
程式碼

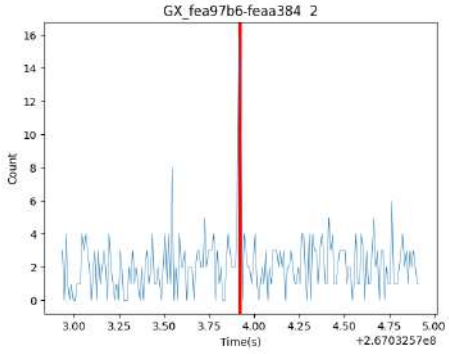
```
=================================================================
!/usr/bin/python

import numpy as np
from astropy.io import fits

fits_file_path =  '/data/P70094/70094-01-03-00/pca/GX_feac1d0-feacfd0.evt.bary'
hdul = fits.open(fits_file_path)
data = hdul[1].data
hdul.close()
data_array = np.array(data)

#=========================================
threshold = 29

rows_to_keep = []

for row in data_array:

    if row[4] <= threshold:

        rows_to_keep.append(row)

new_data_array = np.array(rows_to_keep)
#=========plot ==================
import matplotlib.pyplot as plt
from astropy.stats import bayesian_blocks

#first = new_data_array['TIME'][0]
#last = new_data_array['TIME'][-1]
time_array = new_data_array['TIME']

interval=0.1
#=========================================================
#allow edges array save to file

new_edges= [2.67043666e+08, 2.67043666e+08, 2.67044376e+08, 2.67044376e+08,
 2.67045117e+08, 2.67045117e+08, 2.67045168e+08, 2.67045168e+08,
 2.67045168e+08, 2.67045274e+08, 2.67045274e+08, 2.67045770e+08,
 2.67046154e+08, 2.67046461e+08, 2.67046462e+08, 2.67046463e+08,
 2.67046490e+08, 2.67046490e+08, 2.67046491e+08, 2.67046742e+08,]
#=========================================
first=time_array[0]
last=time_array[-1]
```

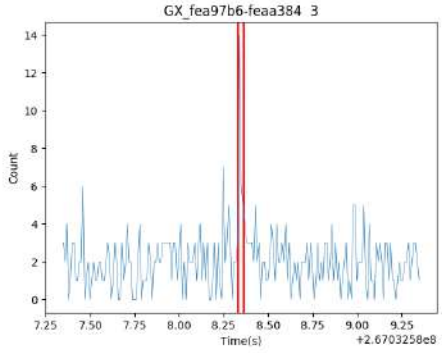
```
#plt.hist(time_array, bins =
np.arange(new_data_array['TIME'][0],new_data_array['TIME'][-1],interval))
hist_array = np.histogram(time_array, bins = np.arange(first, last, interval))
time_array=hist_array[1]
time_array2 = np.arange(time_array[0]+0.5*interval,time_array[-1],interval)

new_hist_array=[]
for i in hist_array[0]:
    new_hist_array.append(i/interval)
#=========================================================
line_edges=len(new_edges)

for i in range(line_edges):
    plt.axvline(
        x=new_edges[i],
        color="red") # Plotting a vertical line
#=========================================================
plt.plot(time_array2,new_hist_array,'',linewidth=0.5)
plt.title("70094-01-03-00_1d0")
plt.xlabel("Time(s)")
plt.ylabel("Count/sec")
plt.savefig("test1.eps", format='eps')
plt.show()
=========================================================================
```

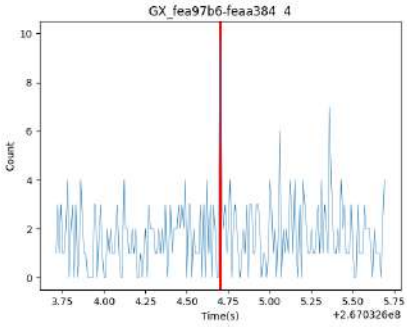
畫出放大後的 burst

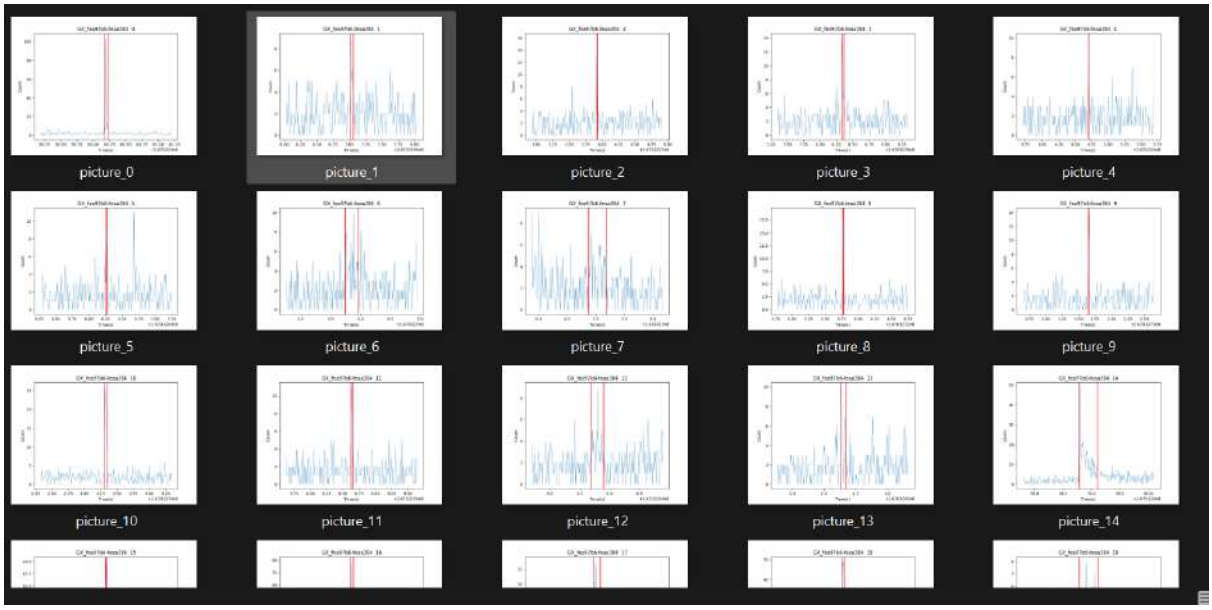
需要人工再進行確認



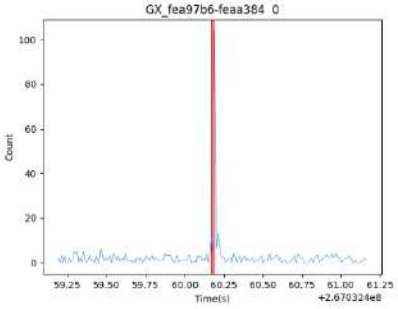


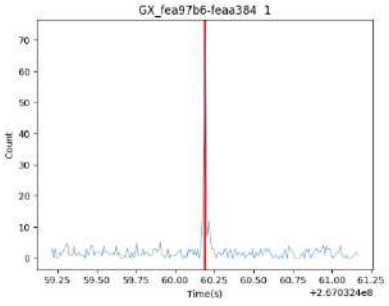

GX_fea97b6-feaa384_3
Count
Time(s)
+2.6703258e8

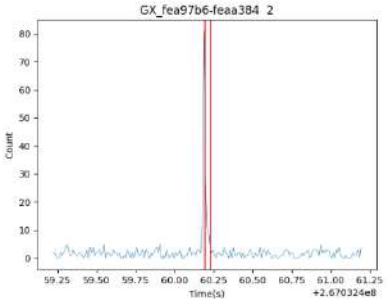

GX_fea97b6-feaa384_4
Count
Time(s)
+2.670326e8


picture_0
picture_1
picture_2
picture_3
picture_4
picture_5
picture_6
picture_7
picture_8
picture_9
picture_10
picture_11
picture_12
picture_13
picture_14

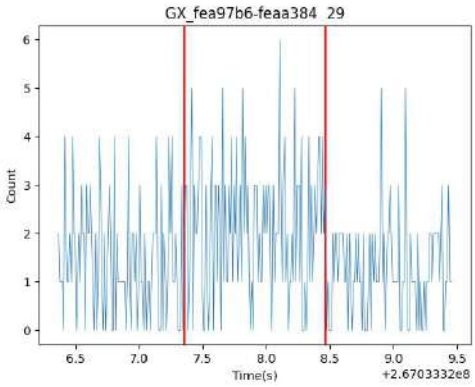
需要合併的 burst

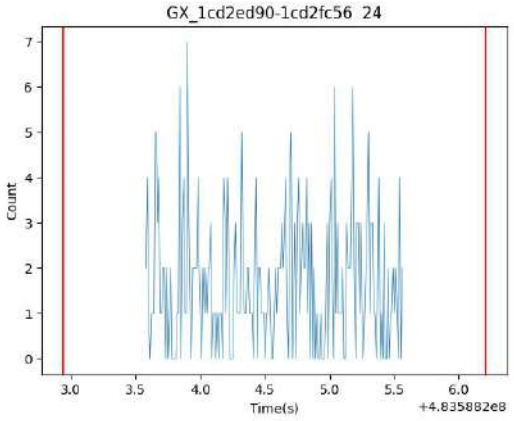

GX_fea97b6-feaa384_0
Count
Time(s)
+2.670324e8


GX_fea97b6-feaa384_1
Count
Time(s)
+2.670324e8


GX_fea97b6-feaa384_2
Count
Time(s)
+2.670324e8

不是 burst 的圖





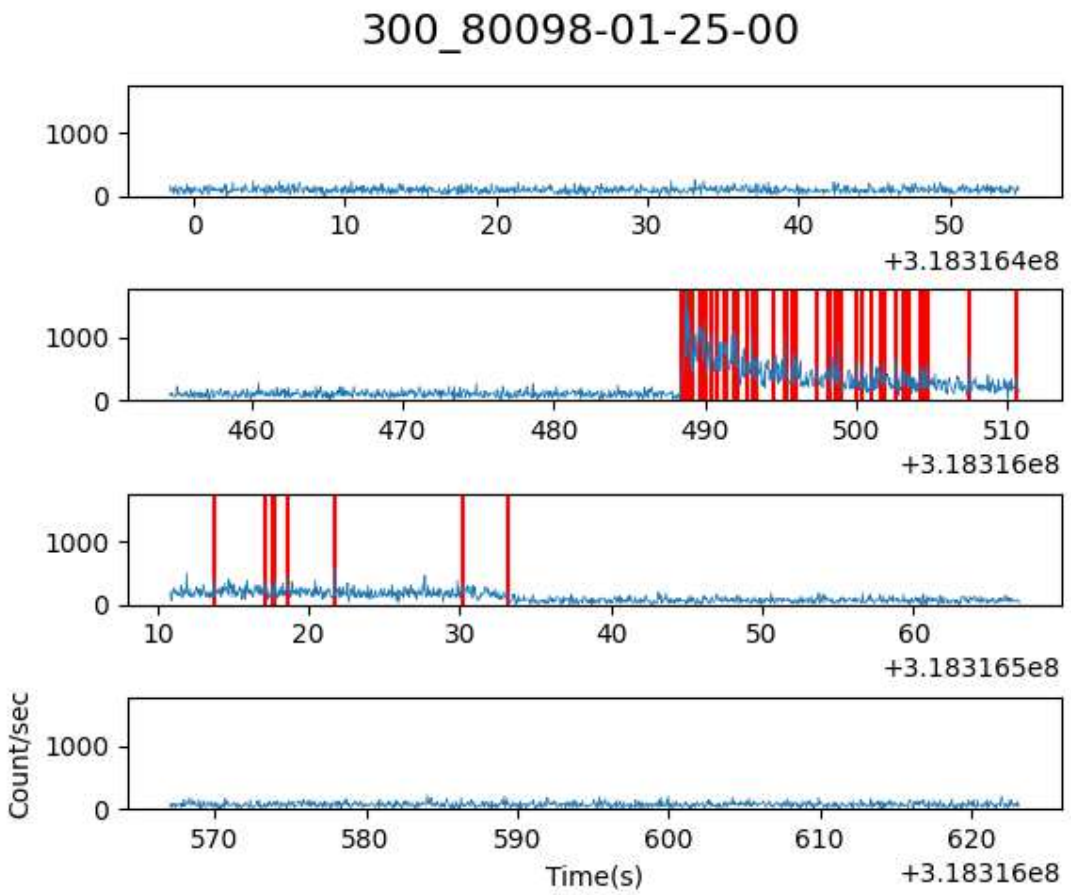
畫出圖的程式碼

```
======================start=========================================
#!/usr/bin/python

import numpy as np
from astropy.io import fits

fits_file_path =  '/data/P70094/70094-01-03-00/pca/GX_feac1d0-feacfd0.evt.bary'
hdul = fits.open(fits_file_path)
data = hdul[1].data
hdul.close()

data_array = np.array(data)
#============================================
threshold = 29
#threshold = 255
rows_to_keep = []

for row in data_array:

    if row[4] <= threshold:

        rows_to_keep.append(row)

new_data_array = np.array(rows_to_keep)
print(new_data_array)
#=========plot ===================
import matplotlib.pyplot as plt
from astropy.stats import bayesian_blocks

time_array = new_data_array['TIME']
```

```
interval=0.1
#============================================================
#allow edges array save to file

file = open("test.txt", "r")
print(file.read())

new_edges= [2.67043666e+08, 2.67043666e+08, 2.67044376e+08, 2.67044376e+08,
 2.67045117e+08, 2.67045117e+08, 2.67045168e+08, 2.67045168e+08,
 2.67045168e+08, 2.67045274e+08, 2.67045274e+08, 2.67045770e+08,
 2.67046154e+08, 2.67046461e+08, 2.67046462e+08, 2.67046463e+08,
 2.67046490e+08, 2.67046490e+08, 2.67046491e+08, 2.67046742e+08,]

#============================================================
#find burst before and after 5 second


def time_array_interval(array, value):

    new_array=[]
    for i in array:
        if abs(i-value) <= 5:
            new_array.append(i)
    return new_array
#============================================
array=time_array
value=new_edges[0]

new_time_array_interval=time_array_interval(array,value)

first=new_time_array_interval[0]
last=new_time_array_interval[-1]

hist_array = np.histogram(new_time_array_interval, bins = np.arange(first, last,
interval))
new_time_array_interval=hist_array[1]
time_array2 =
np.arange(new_time_array_interval[0]+0.5*interval,new_time_array_interval[-
1],interval)

new_hist_array=[]
for i in hist_array[0]:
    new_hist_array.append(i/interval)
#============================================================
array=new_edges
value=new_edges[0]
```

```
edgess=time_array_interval(array,value)

line_edges=len(edgess)

for i in range(line_edges):
    plt.axvline(
        x=edgess[i],
        color="red") # Plotting a vertical line
#=======================================
plt.plot(time_array2,new_hist_array,'',linewidth=0.5)
plt.title("70094-01-03-00_1d0")
plt.xlabel("Time(s)")
plt.ylabel("Count/sec")
plt.savefig("test1.eps", format='eps')
plt.show()
==================end============================================
```

畫出有問題的爆發



收集爆發的相關參數

所收集到參數

[(start_edge, end_edge), subtract_time, count_number(threshold = 29), soft_number, hard_number, hardness ratio, others_time, others_count, poisson, t90, waiting_burst]

hardness ratio = hard_number/soft_number

soft：2~4 kev

hard：4~10 kev

程式碼

```
==========================================================================
#!/usr/bin/python

import numpy as np
from astropy.io import fits
import ast
from scipy.stats import poisson
import os
import matplotlib.pyplot as plt
from scipy.stats import sem
from scipy.interpolate import interpld

f = open('/data/resh/path_test.txt')
text = []
for line in f:
    text.append(line)
f.close()

spec = [211,212,213]

for z in spec:

    text1 = text[z]
    print("text1=",text1)
    cleaned_path = text1.strip("'\n")
    print(cleaned_path)
    fits_file_path = cleaned_path
```

```
hdul = fits.open(fits_file_path)
data = hdul[1].data
hdul.close()
data_array = np.array(data)
#=================================================

name_f = cleaned_path

file_r = name_f.split('/')[2]
file_rr = name_f.split('/')[3]
file_name = name_f.split('/')[5].replace('.evt.bary', '')

path1 = f'/data/resh/edge/p0_0.01/{file_r}/{file_rr}/{z}{file_name}.txt'
path2 = f'/data/resh/edge/p0_0.01/{file_r}/{file_rr}/{file_name}.txt'
#=================================================
with open(path2, "r") as d:
    new_edges = d.read()
new_edges = ast.literal_eval(new_edges)
#=================================================
# Read the burst number
with open(path1, 'r', encoding='UTF-8') as file:
    burst_list = file.read()
burst_list = ast.literal_eval(burst_list)
print("burst_list = ", burst_list, len(burst_list))
file.close()
#====================================
def start_edge_func(burst_list):
    start_edge = new_edges[q]
    return start_edge
#====================================
def end_edge_func(burst_list):
    end_edge = new_edges[p+1]
    return end_edge
#====================================
def sub_time(burst_list):
    sub_num = new_edges[p+1] - new_edges[q]
    return sub_num
#====================================
def count_a():
    threshold = 29
    rows_to_keep = []
    for row in data_array:

        if row[4] <= threshold:
```

```
                rows_to_keep.append(row)

        new_data_array = np.array(rows_to_keep)
        return new_data_array
    #====================================
    def count_soft():
        threshold = 9
        rows_to_keep = []
        for row in data_array:

            if row[4] <= threshold:

                rows_to_keep.append(row)

        new_data_array = np.array(rows_to_keep)
        return new_data_array
    #====================================
    def count_hard():
        threshold1 = 9
        threshold2 = 24
        rows_to_keep = []
        for row in data_array:

            if row[4] > threshold1 and row[4] <= threshold2:

                rows_to_keep.append(row)

        new_data_array = np.array(rows_to_keep)
        return new_data_array
    #====================================
    def time_array_interval(start_edge, end_edge, new_data_array):
        global new_time_array
        new_time_array = []
        for row in new_data_array:
            if row[0] >= start_edge and row[0] <= end_edge:
                new_time_array.append(row[0])
        global time_array
        time_array = []
        for row in new_data_array:
            if row[0] >= (start_edge + end_edge)/2-5 and row[0] <= (start_edge +
end_edge)/2+5:
                time_array.append(row[0])
        num = len(new_time_array)
        return num
    #====================================
    def divide(hard_number, soft_nuumber):
```

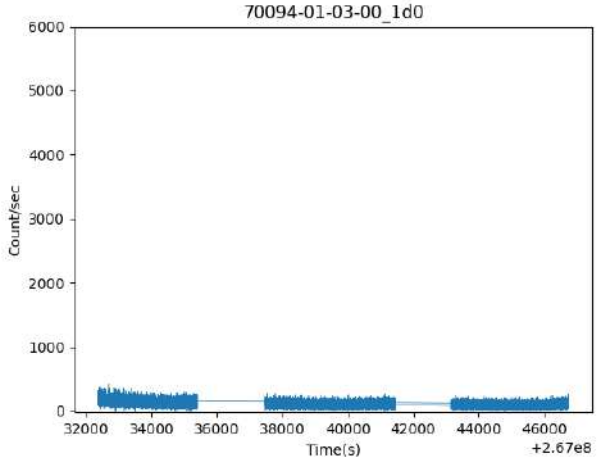
```
    divide_num = hard_number / soft_number
    return divide_num
#==================================
def others_time(before_time_limit, after_time_limit):
    others_time_num = (after_time_limit - before_time_limit) - sub_time_num
    return others_time_num
#==================================
def others_conut(array, before_time_limit, after_time_limit, count_a_num):
    new_array=[]
    for i in array:
        if i >= before_time_limit and i <= after_time_limit:
            new_array.append(i)
    others_count_num = len(new_array) - count_a_num
    return others_count_num
#==================================
def possion_da(others_time, others_count, sub_time_num, count_a_num):
    mu=(others_count/others_time)*sub_time_num
    k=count_a_num
    r=poisson.pmf(k, mu)
    return r
#==================================
def calculate_t90(numbers):

    # 將數據進行排序
    sorted_numbers = sorted(numbers)
    # 計算第 5%的位置
    t5_percentile_index = (5 / 100) * len(sorted_numbers)
    # 由於索引是從 0 開始的，我們需要減去 1 來獲取正確的索引
    # 並且使用 int 來獲取最接近的較小整數索引
    t5_index = int(t5_percentile_index) - 1
    # 確保索引不會小於 0
    t5_index = max(t5_index, 0)
    # 第 5%的數
    t5 = sorted_numbers[t5_index]

    t95_percentile_index = (95 / 100) * len(sorted_numbers)
    t95_index = int(t95_percentile_index) - 1
    t95_index = max(t95_index, 0)
    t95 = sorted_numbers[t95_index]

    # 計算 T90
    t90 = t95 - t5
    return t90
#==================================
def waiting_burst(prior_edge, start_edge):
    waiting_time = start_edge - prior_edge
```

```
        return waiting_time
    #===================================
    time_array2 = 0
    hist_array = []

    def paint_burst(new_time_array_interval):

        first=new_time_array_interval[0]
        last=new_time_array_interval[-1]

        global time_array2
        global hist_array
        hist_array = np.histogram(new_time_array_interval, bins =
np.arange(first, last, interval))
        time_array=hist_array[1]
        time_array2 = np.arange(time_array[0]+0.5*interval,time_array[-
1],interval)
        return 0
    #===================================

    def paint_edges(start_edge, end_edge):
        plt.axvline(x=start_edge, color="red") # Plotting a vertical line
        plt.axvline(x=end_edge, color="red")
    #===================================

    def paint(start_edge, end_edge):

        plt.figure()

        new_time_array_interval=time_array

        paint_burst(time_array)
        paint_edges(start_edge, end_edge)
        plt.plot(time_array2, hist_array[0],'',linewidth=0.5)
        title = (
                f"file_name : {file_name} "
                f"burst time : {sub_time_num} \n"
                f"t90 : {t90} "
                f"number before&after ckecking: {q},{p} to {i}"
                )
        plt.title(title)
        plt.xlabel("Time(s)")
        plt.ylabel("Count")

        filename = f'/data/resh/edge/p0_0.01/{z}{file_name}/picture_{i}.png'
        plt.savefig(filename)
```

```
        plt.clf()
        print("paint success")
    #==================================
    two_dimensional_array = []
    i = 0
    for u in range(int(len(burst_list)/2)):
        interval = 0.01
        num_list = []
        print("u = ",u)
        q = burst_list[2*u]
        p = burst_list[2*u + 1]
        start_edge = start_edge_func(burst_list)
        end_edge = end_edge_func(burst_list)
        num_list.append((start_edge, end_edge))
        sub_time_num = sub_time(burst_list)
        num_list.append(sub_time_num)
        new_data_array_a = count_a()
        count_a_num = time_array_interval(start_edge, end_edge,
new_data_array_a)
        num_list.append(count_a_num)

        i = i + 1
        new_data_array = count_soft()
        soft_number = time_array_interval(start_edge, end_edge, new_data_array)
        num_list.append(soft_number)
        new_data_array = count_hard()
        hard_number = time_array_interval(start_edge, end_edge, new_data_array)
        num_list.append(hard_number)
        count_a_num = time_array_interval(start_edge, end_edge,
new_data_array_a)

        if hard_number == 0.0:
            divide_num = 0.0
        elif soft_number == 0.0:
            divide_num = 'nan'
        else :
            divide_num = (divide(hard_number, soft_number))
        num_list.append(divide_num)

        if sub_time_num >= 10:
            div = round(new_edges[q] - new_edges[p+1])
            before_time_limit = new_edges[q] - div*2
            after_time_limit = new_edges[p+1] + div*2
        else:
            before_time_limit = ((new_edges[q] + new_edges[q+1]) / 2) - 1
            after_time_limit = ((new_edges[p] + new_edges[p+1]) / 2) + 1
```

```
        other_time = others_time(before_time_limit, after_time_limit)
        other_count = others_conut(new_data_array_a['TIME'], before_time_limit,
after_time_limit, count_a_num)
        num_list.append(other_time)
        num_list.append(other_count)
        possion_num = possion_da(other_time, other_count, sub_time_num,
count_a_num)
        if possion_num >= 5.73e-07:
            num_list = []
            continue
        num_list.append(possion_num)
        print(new_time_array)
        if new_time_array == []:
            continue
        print(count_a_num)
        t90 = calculate_t90(new_time_array)
        if t90<=0.001:
            continue
        num_list.append(t90)
        if u==0:
            waiting_time = 'nan'
        else:
            waiting_time = waiting_burst(prior_edge, start_edge)
        num_list.append(waiting_time)
        prior_edge = start_edge
        print("num_list = ", num_list)
        paint(start_edge, end_edge)
        two_dimensional_array.append(num_list)

    print("two_dimensional_array = ", two_dimensional_array)

    array_str = str(list(two_dimensional_array))

    path = f'/data/resh/edge/p0_0.01/{file_name}.txt'
    file = open(path, 'w', encoding='utf-8')

    file.write(array_str)
    file.close()
=========================================================================
```

拿掉 burst 的 lightcurve



程式碼

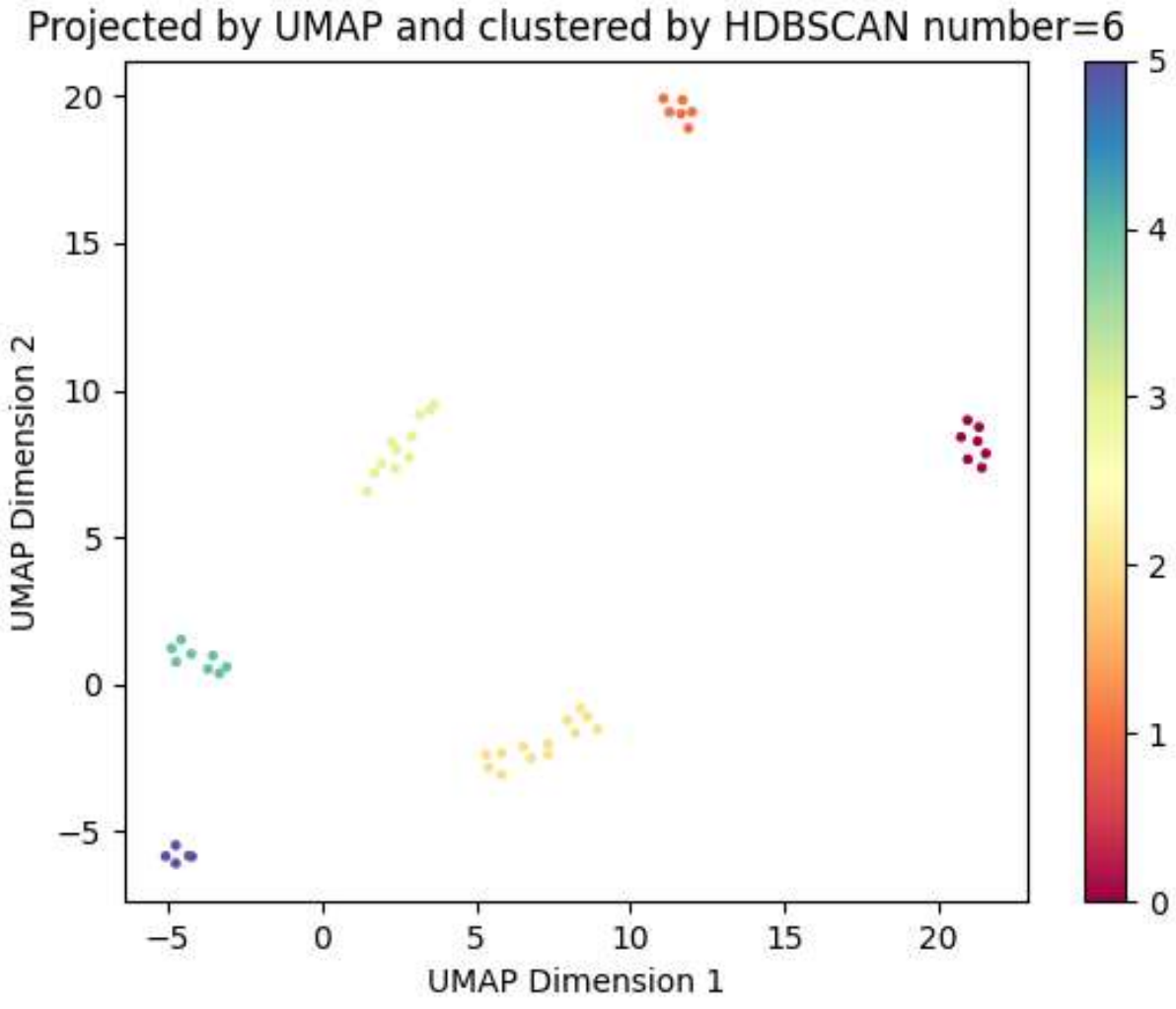
```
================================================================================
import numpy as np
import matplotlib.pyplot as plt
from astropy.io import fits
import ast
#============================
f = open('/data/resh/path_test.txt')
text = []
for line in f:
    text.append(line)
f.close()

v = 'p0_0.01'
spec = [211, 212, 213]
time_array_p, hist_array_p, start_edges, end_edges = [], [], [], []

for z in spec:

    text1 = text[z]
    print("text1=",text1)
    cleaned_path = text1.strip("'\n")
    print(cleaned_path)
    fits_file_path = cleaned_path
    hdul = fits.open(fits_file_path)
    data = hdul[1].data
    hdul.close()

    data_array = np.array(data)
#================================
    name_f = cleaned_path
    file_name =name_f.split('/')[5].replace('.evt.bary', '')
```

```
    path1 = f'/data/resh/edge/{v}/{file_name}.txt'
    with open(path1, "r") as d:
        new_data = d.read()
        d.close()
    array = eval(new_data)

    for i in range(len(array)):
        start_edges.append(array[i][0][0])
        end_edges.append(array[i][0][1])
#================================
    threshold = 29
    rows_to_keep = []
    for row in data_array:

        if row[4] <= threshold:

            rows_to_keep.append(row)

    new_data_array = np.array(rows_to_keep)
#================================
    time_array = new_data_array['TIME']
    for i in range(len(start_edges)):
        time_array = [x for x in time_array if not (start_edges[i] <= x <=
end_edges[i])]
#=========plot ==================

    interval=0.1

    first=time_array[0]
    last=time_array[-1]

    hist_array = np.histogram(time_array, bins = np.arange(first, last,
interval))
    time_array=hist_array[1]
    time_array = np.arange(time_array[0]+0.5*interval,time_array[-1],interval)
#=====================================
    time_array_p = np.concatenate((time_array_p, time_array))
    print("len=",len(time_array_p))

    hist_array_p = np.concatenate((hist_array_p, hist_array[0]))
    print("len=",len(hist_array_p))
    print("hist_array_p=", hist_array_p)

    new_hist_array=[]
    for i in hist_array_p:
```

```
        new_hist_array.append(i/interval)
#================================

    new_edges = np.concatenate((start_edges, end_edges))
#=============================================================================
plt.plot(time_array_p,new_hist_array,'',linewidth=0.5)
plt.title("70094-01-03-00_1d0")
plt.xlabel("Time(s)")
plt.ylim(top = 6000)
plt.ylabel("Count/sec")
plt.savefig("test5.png")
plt.show()
=================================================================
```

用 UMAP 法降維後並用 HDBSCAN 做分群（包含計算 sample standard deviation）

分群圖：



每一群的值：

cluster label	initial time	t90	total photon number	hardness ratio
0	141.9352	1.74±0.87	1.74±0.51	4.10±3.37
1	40.53523	1.68±0.80	1.83±0.48	2.98±0.51
2	79.93948	1.51±1.17	1.79±0.81	2.95±0.51
3	730.5508	1.62±0.71	1.70±0.40	2.64±0.51
4	11.23383	1.60±0.50	1.84±0.48	3.18±0.51

程式碼

```
======================================================================
#!bin/usr/python

import hdbscan
import umap
import matplotlib.pyplot as plt
from astropy.io import fits
import ast
import numpy as np
import os
import pandas as pd
from scipy.stats import sem

#====================================

f = open('/data/resh/path_test.txt')
text = []
for line in f:
    text.append(line)
f.close()

data = { 'initial time':[],
         'time interval':[],
         'total photon number':[],
         'hardness ratio':[]}

burst_array = []
spec = [211,212,213]
```

```
for z in spec:

        text1 = text[z]
        cleaned_path = text1.strip("'\n")
        #================================================
        name_f = cleaned_path
        file_name = name_f.split('/')[5].replace('.evt.bary', '')

        path3 = f'/data/resh/edge/p0_0.01/{file_name}.txt'
        #================================================
        with open(path3, 'r', encoding='UTF-8') as file:
            content = file.read()
        content = content.replace('\n', '')

        item_list = []

        index_list = [0,10,3,6]
        new = []
        new1 = []
        for item in eval(content):
            print("item = " ,item)
            # 刪除元組元素
            for b in item:
                # 檢查當前元素是否為元組
                if isinstance(b, tuple):
                    new.extend(b)
                else:
                    new.append(b)
#============================================================
            for index in index_list:
                new1.append(list(new)[index])
            item_list.append(new1)
            new1 = []
            new = []

        item_list = np.array(item_list)
        burst_array.append(item_list)
        file.close()
#================================================
keys = ['initial time','time interval', 'total photon number', 'hardness ratio']
for n in range(3):
    if len(burst_array[n]) == 0:
        continue
    else:
        for i in range(4):
            for j in range(len(burst_array[n])):
```

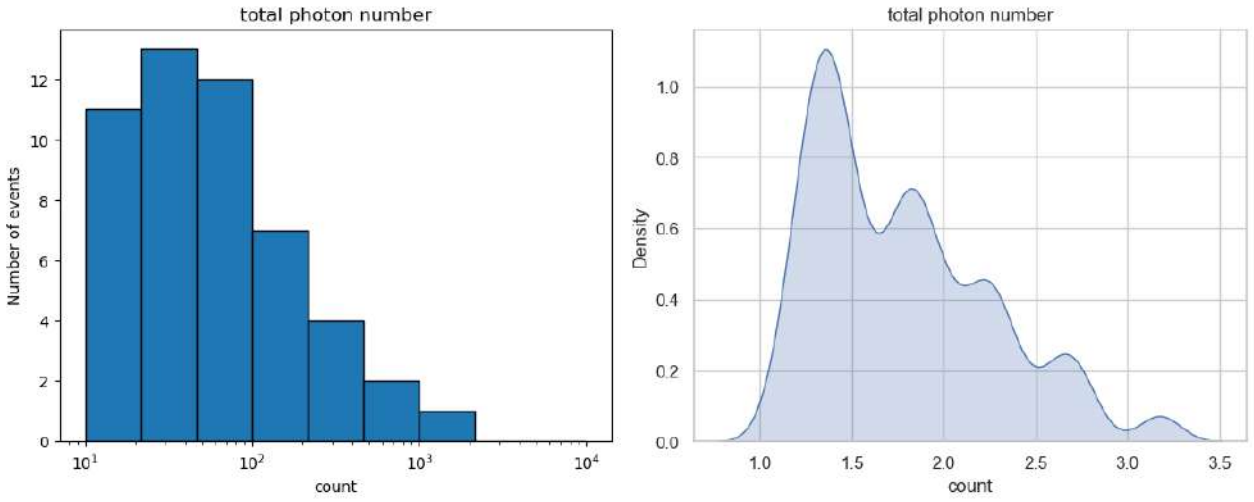
```
                data[keys[i]].append(burst_array[n][j][i])
#================================================
#有"waiting time"參數中出現'nan'值
df_pri = pd.DataFrame(data)
df_pri.replace('nan', np.nan, inplace=True)
df = df_pri.dropna()
df.reset_index(drop=True, inplace=True)
print(df)
df = df.apply(pd.to_numeric)
#================================================
df = pd.DataFrame(data)

df['total photon number'] = np.log10(df['total photon number'])
for i in range(50):
    df['initial time'][i] = df['initial time'][i]-267032400
    df['time interval'][i] = df['time interval'][i]*1000
df['time interval'] = np.log10(df['time interval'])

qw = [2,3,4,5,6]
reii = [0.3,0.4,0.5,0.6]
for wis in qw:
    for rei in reii:

        # 使用 UMAP 進行降維
        data_array = df.values
        umap_model = umap.UMAP(n_neighbors=wis, min_dist=rei, n_components=2, )
        reduced_data = umap_model.fit_transform(data_array)
        s = 8
        a = f'/data/resh/cluster{s}/neig={wis},dist={rei}'
        if not os.path.exists(a):
            os.makedirs(a)
#================================================
# 使用 HDBSCAN 進行分群
        for i in range(50):
            c = i+2

            hdbscan_model = hdbscan.HDBSCAN(min_cluster_size=c,
gen_min_span_tree=True)
            cluster_labels = hdbscan_model.fit_predict(reduced_data)
            print("cluster_labels = ", cluster_labels)

            df['cluster'] = cluster_labels
            cluster_summary = df.groupby('cluster').mean()
            cluster_summary = cluster_summary.reset_index()
            cluster_summary.columns = ['cluster label', 'initial time','time
interval', 'total photon number', 'hardness ratio']
```

```
            p = len(set(cluster_labels))
            if p <=10:

                path_cc =
f'/data/resh/complete_table{s}/neig__{wis}_dist__{rei}'
                if not os.path.exists(path_cc):
                    os.makedirs(path_cc)

                with
open(f'/data/resh/complete_table{s}/neig__{wis}_dist__{rei}/ori_size__{c}.txt',
"w") as f:
                    f.write(str(df))
                f.close()

                line_label = len(cluster_summary['cluster label'])
                ori_data1=[[] for _ in range(line_label)]
                ori_data2=[[] for _ in range(line_label)]
                ori_data3=[[] for _ in range(line_label)]
                ori_data4=[[] for _ in range(line_label)]
                for o in range(line_label):
                    for e in range(len(df['initial time'])):
                        if df['cluster'][e] == o-1:
                            ori_data1[o].append(df['initial time'][e])
                            ori_data2[o].append(df['time interval'][e])
                            ori_data3[o].append(df['total photon number'][e])
                            ori_data4[o].append(df['hardness ratio'][e])
                    o+=1
                other_data0 = { 'initial time':[],
                             'time interval':[],
                             'total photon number':[],
                             'hardness ratio':[]}
                other_data1 = { 'initial time':[],
                             'time interval':[],
                             'total photon number':[],
                             'hardness ratio':[]}
                other_data2 = { 'initial time':[],
                             'time interval':[],
                             'total photon number':[],
                             'hardness ratio':[]}

                for e in range(line_label):
                    data_new = ori_data1[e]
                    other_data0['initial time'].append(np.std(data_new, ddof=0))
                    other_data1['initial time'].append(np.std(data_new, ddof=1))
                    other_data2['initial time'].append(sem(data_new))
```

```
                for e in range(line_label):
                    data_new = ori_data1[e]
                    other_data0['time interval'].append(np.std(data_new,
ddof=0))
                    other_data1['time interval'].append(np.std(data_new,
ddof=1))
                    other_data2['time interval'].append(sem(data_new))
                for e in range(line_label):
                    data_new = ori_data1[e]
                    other_data0['total photon number'].append(np.std(data_new,
ddof=0))
                    other_data1['total photon number'].append(np.std(data_new,
ddof=1))
                    other_data2['total photon number'].append(sem(data_new))
                for e in range(line_label):
                    data_new = ori_data1[e]
                    other_data0['hardness ratio'].append(np.std(data_new,
ddof=0))
                    other_data1['hardness ratio'].append(np.std(data_new,
ddof=1))
                    other_data2['hardness ratio'].append(sem(data_new))

                df_other_data0 = pd.DataFrame(other_data0)
                df_other_data1 = pd.DataFrame(other_data1)
                df_other_data2 = pd.DataFrame(other_data2)



                with
open(f'/data/resh/complete_table{s}/neig__{wis}_dist__{rei}/size__{c}_deviation.
txt', "w") as f:
                    f.write("population standard deviation = \n")
                    f.write(str(df_other_data0))
                    f.write("\n")
                    f.write("sample standard deviation = \n")
                    f.write(str(df_other_data1))
                    f.write("\n")
                    f.write("statistical error = \n")
                    f.write(str(df_other_data2))
                f.close()

                p = len(set(cluster_labels))
#================================================
        # 繪製散點圖
                plt.scatter(reduced_data[:, 0], reduced_data[:, 1],
c=cluster_labels, cmap='Spectral', s=5)
                plt.colorbar()  # 顯示色條
```

```
                plt.title(f'Projected by UMAP and clustered by HDBSCAN
number={p}')
                plt.xlabel('UMAP Dimension 1')
                plt.ylabel('UMAP Dimension 2')
                filename = a + f'/{c}.png'
                plt.savefig(filename)
                plt.close()

    # 在命令行中顯示結果

                b = f'/data/resh/complete_table{s}/neig__{wis}_dist__{rei}'
                if not os.path.exists(b):
                    os.makedirs(b)

                with
open(f'/data/resh/complete_table{s}/neig__{wis}_dist__{rei}/size__{c}.txt', "w")
as f:
                    f.write(str(cluster_summary))
                f.close()
=========================================================================
```

爆發參數的直方圖



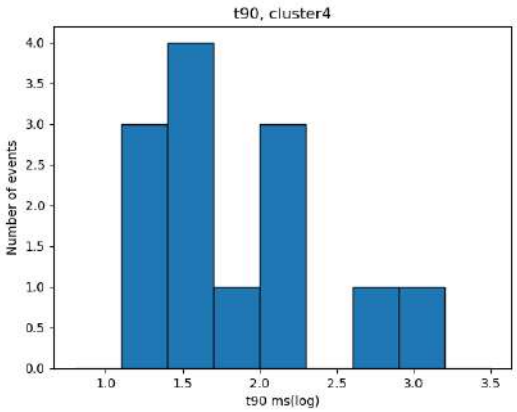
Total photon number 程式碼

```
=========================================================================
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import math
#=============================================
```

```
file_path = 'D:/com_data/report/after/分群資料/6_0_3_5/number.xlsx'
df = pd.read_excel(file_path)
#================================================
column = df['total photon number']
# 將該列轉換為一維陣列
total_photon_number = column.values

print(total_photon_number)
#================================================
#畫出直方圖
import numpy as np
def paint(x):

# 對數據取對數值
    log_x = np.log10(x)

# 設定 bin 的範圍和數量
    bins = np.logspace(np.floor(np.log10(min(x))), np.ceil(np.log10(max(x))),
num=10)

# 繪製直方圖
    plt.hist(x, bins=bins, edgecolor='black')
    plt.xscale('log')
    plt.title("total photon number")
    plt.xlabel("count")
    plt.ylabel("Number of events")
    plt.show()
paint(one_dimensional_array)
#================================================
#畫出 KED 圖
def paint(x):

    # 繪製核密度估計圖
    sns.kdeplot(x, fill=True,bw_adjust=0.5)

    plt.title("total photon number")
    plt.xlabel("count")
    plt.ylabel("Density")
    plt.show()
paint(one_dimensional_array)
=========================================================================

#每一群分群圖資料

df1 = df[['total photon number', 'cluster']]
for i in range(5):
```

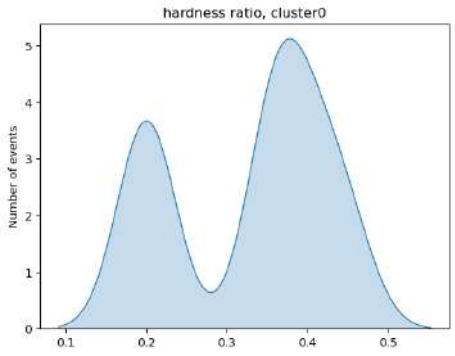
```
    # 過濾出'cluster'等於3的列
    filtered_df1 = df1[df1['cluster'] == i]
    # 取出'totalphotonnumber'列並轉換為一維陣列
    globals()['total_photon_count'+str(i)] = filtered_df1['total photon
number'].values
    print(globals()['total_photon_count'+str(i)])
#================================================
```

#畫每群分佈的直方圖方佈




```
for i in range(5):
    def paint(x):

        bins = np.linspace(0.6, 3.5, num=15)
        plt.hist(x, bins=bins,
edgecolor='black')

        plt.title(f"total photon number,
cluster{i}")
        plt.xlabel("count(log)")
        #plt.xlim(0, 4)
        plt.ylabel("Number of events")
        plt.show()
    paint(globals()['total_photon_count'+str(i)])
#================================================
```

#畫每群分佈的KED圖

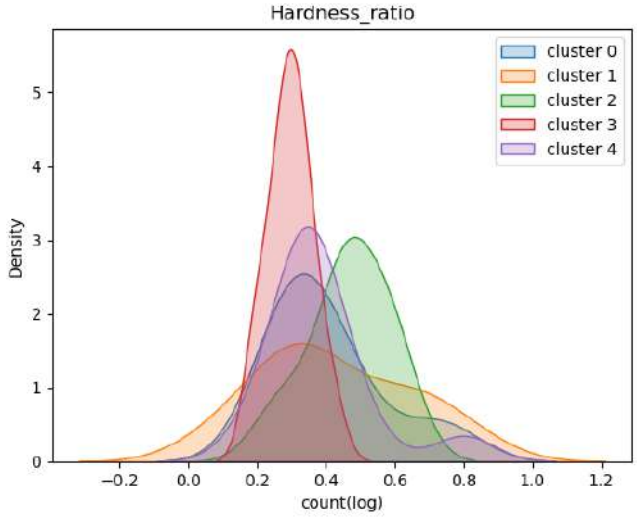



```
for i in range(4):
    def paint(x):

        # 繪製核密度估計圖
        sns.kdeplot(x, fill=True, bw_adjust=0.5)

        plt.title(f"total photon number,
cluster{i}")
        plt.xlabel("count(log)")
        plt.ylabel("Density")
        plt.show()
    paint(globals()['total_photon_count'+str(i)])
#================================================
```

#每群交疊在一張圖

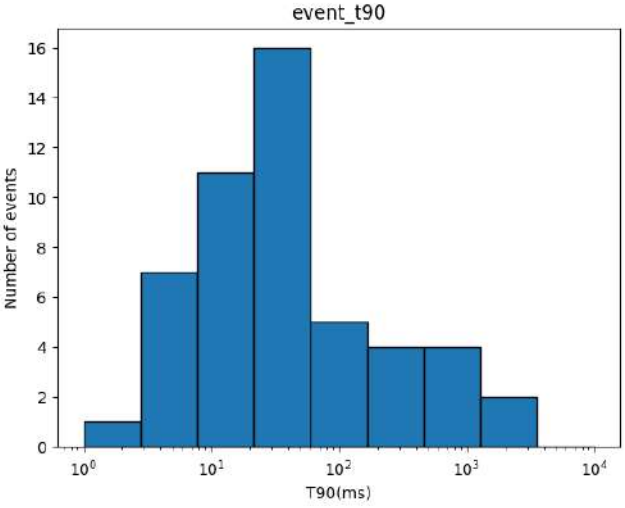
```
def paint():

    # 繪製核密度估計圖
    sns.kdeplot(total_photon_count0, label='cluster 0', fill=True)
    sns.kdeplot(total_photon_count1, label='cluster 1', fill=True)
    sns.kdeplot(total_photon_count2, label='cluster 2', fill=True)
    sns.kdeplot(total_photon_count3, label='cluster 3', fill=True)
    sns.kdeplot(total_photon_count4, label='cluster 4', fill=True)
```

```
    plt.title("total photon number")
    plt.xlabel("count(log)")
    plt.ylabel("Density")
    plt.legend()
    plt.show()
paint()
```

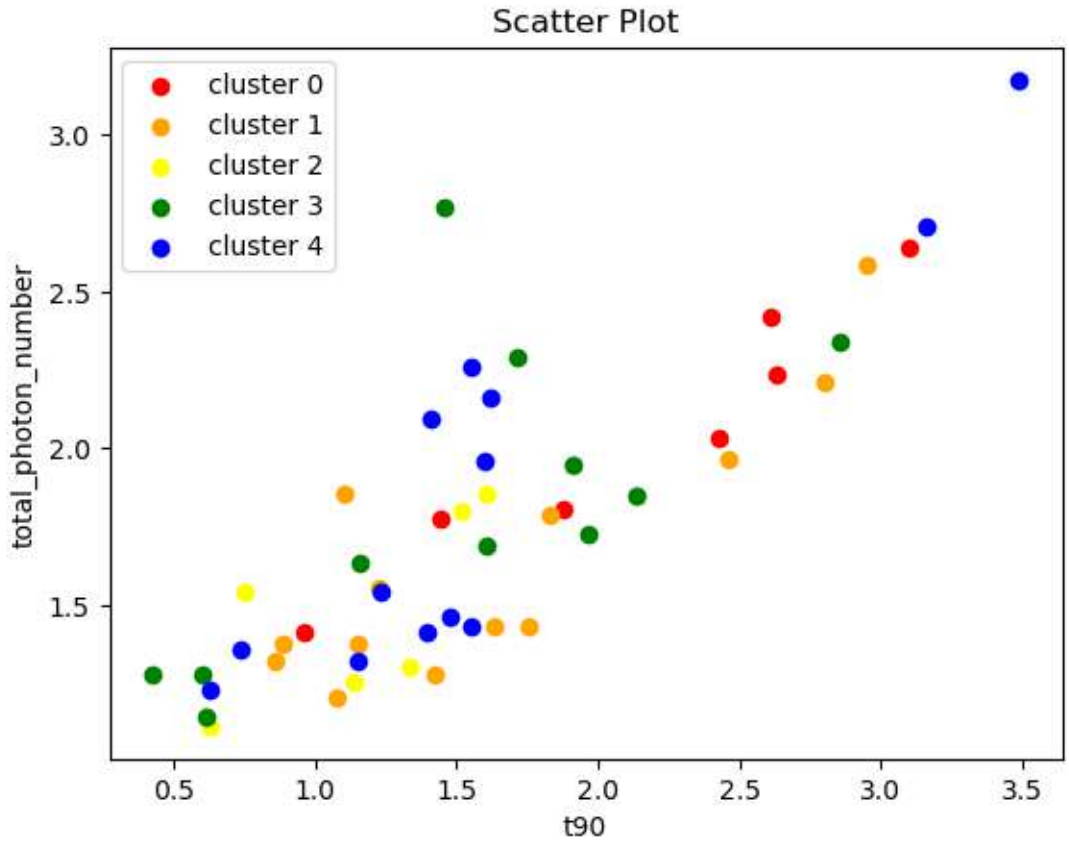


==========

之後依序畫出其它張直方圖



相關性分析



程式碼

```
=====================================================================
#散佈圖 (Scatter Plot)
color = [ 'red', 'orange', 'yellow', 'green', 'blue']
labels = ['cluster 0', 'cluster 1', 'cluster 2', 'cluster 3', 'cluster 4']
for i in range(5):
    plt.scatter(globals()['t90'+str(i)], globals()['total_photon_count'+str(i)],
c=color[i], label=labels[i])
plt.title('Scatter Plot')
plt.xlabel('t90')
plt.ylabel('total_photon_number')
# 添加圖例
plt.legend(loc='upper left')
plt.show()
=====================================================================
```

皮爾森相關係分析

	data1	data2	data3	data4
data1	1	-0.0331	-0.13282	0.110607
data2	-0.0331	1	0.911218	-0.0717
data3	-0.13282	0.911218	1	-0.1133
data4	0.110607	-0.0717	-0.1133	1

程式碼

```
==========================================================================
import pandas as pd

# 有四種資料
data = {
    'data1': initial_time,
    'data2': t90,
    'data3': total_photon_number,
    'data4': hardness_ratio
}

df = pd.DataFrame(data)
correlation_matrix = df.corr(method='pearson')
print(correlation_matrix)
==========================================================================
```

sample standard deviation =

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2}$$

Hardness ratio 的誤差值計算方法

- 公式
  =SQRT((SQRT(hard) /hard)^2 + (SQRT(soft) / soft)^2) * (hard/soft)

$$\bullet error = \frac{hard}{soft} \cdot \sqrt{\left(\frac{\sqrt{hard}}{hard}\right)^2 + \left(\frac{\sqrt{soft}}{soft}\right)^2}$$

Wright averaging

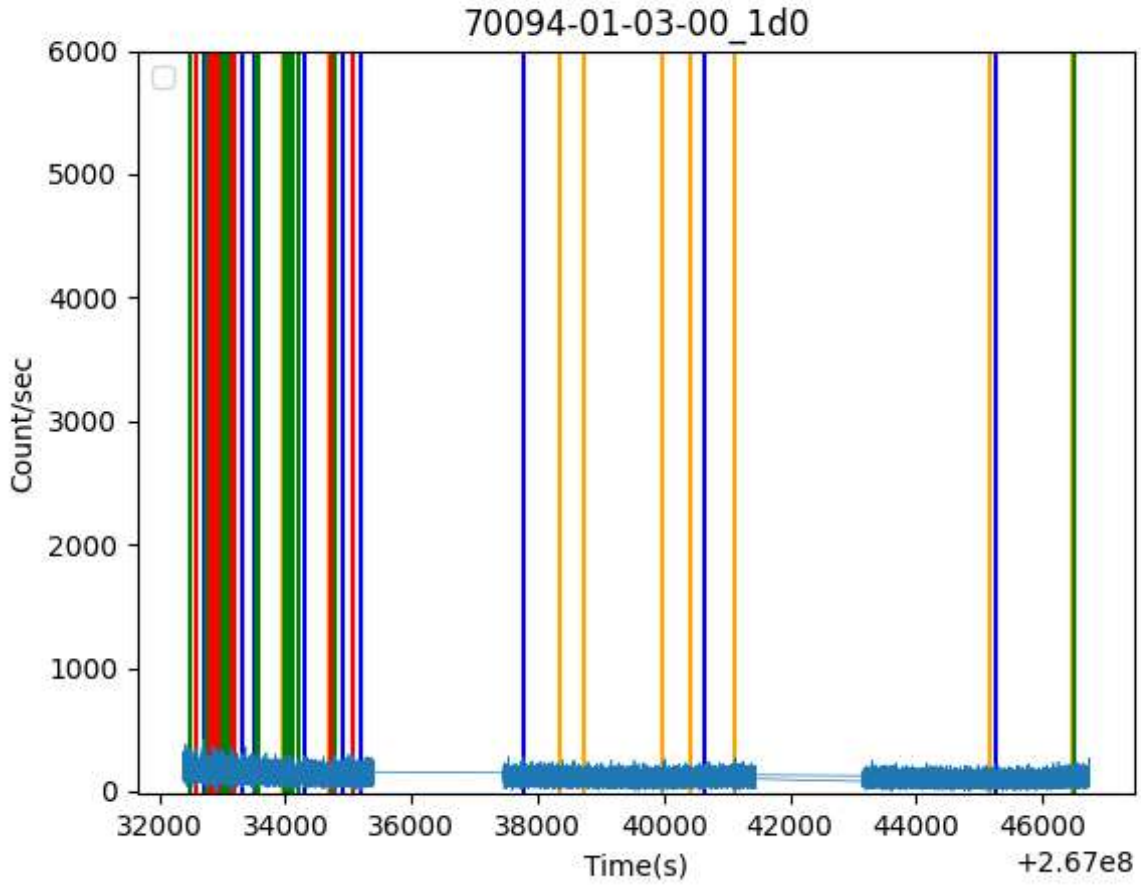
$$\overline{x}\left(weighted\right) = \frac{\sum_{i=1}^{N} \frac{x_i}{\sigma_i^2}}{\sum_{i=1}^{N} \frac{1}{\sigma_i^2}}$$

$$\sigma_{\overline{x}}^2\left(weighted\right) = \left(\sum_{i=1}^{N} \frac{1}{\sigma_i^2}\right)^{-1}$$

Result

label	x	σ2
-1	0.884532	0.294225
0	0.512175	0.041555
1	2.31121	0.013223
2	1.18768	0.042597
3	0.634758	0.047179
4	1.014565	0.054419
5	0.413685	0.094043

畫出出不同群在時間上的分佈



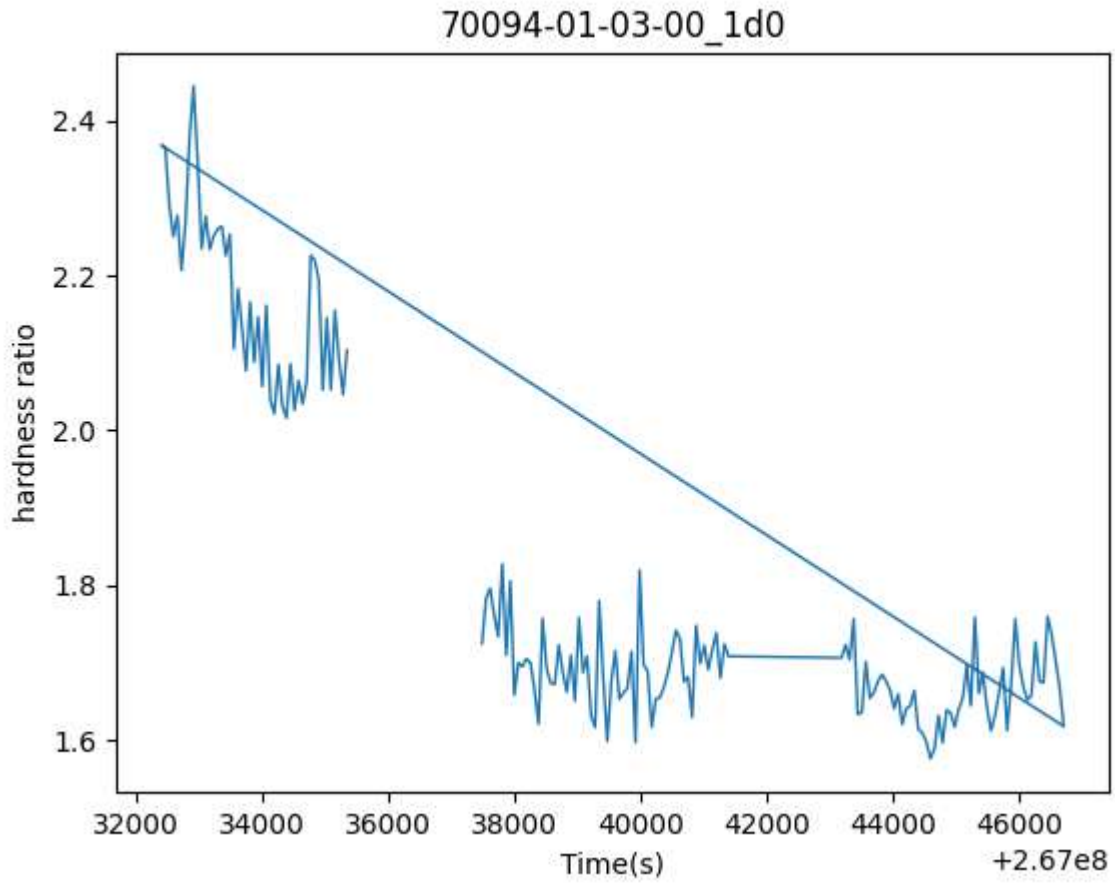
程式碼

```
================================================================================
#!/usr/bin/python

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from astropy.io import fits
import ast
#============================
f = open('/data/resh/path_test.txt')
text = []
for line in f:
        text.append(line)
f.close()

v = 'p0_0.01'
spec, time_array_p, hist_array_p, start_edges, end_edges = [211, 212, 213], [], [], [], []
for z in spec:
```

```
        text1 = text[z]
        print("text1=",text1)
        cleaned_path = text1.strip("'\n")
        print(cleaned_path)
        fits_file_path = cleaned_path
        hdul = fits.open(fits_file_path)
        data = hdul[1].data
        hdul.close()

        data_array = np.array(data)
#================================
        name_f = cleaned_path
        file_r =name_f.split('/')[2]
        file_rr =name_f.split('/')[3]
        file_name =name_f.split('/')[5].replace('.evt.bary', '')

        path1 = f'/data/resh/edge/{v}/{file_name}.txt'
        with open(path1, "r") as d:
            new_data = d.read()
            d.close()
        array = eval(new_data)

        for i in range(len(array)):
            start_edges.append(array[i][0][0])
            end_edges.append(array[i][0][1])
#================================
        threshold = 29
        rows_to_keep = []
        for row in data_array:

            if row[4] <= threshold:

                rows_to_keep.append(row)

        new_data_array = np.array(rows_to_keep)
#================================
        time_array = new_data_array['TIME']
        for i in range(len(start_edges)):
            time_array = [x for x in time_array if not (start_edges[i] <= x <= end_edges[i])]
#=========plot ==================
        interval=0.1

        first=time_array[0]
        last=time_array[-1]

        hist_array = np.histogram(time_array, bins = np.arange(first, last, interval))
        time_array=hist_array[1]
        time_array = np.arange(time_array[0]+0.5*interval,time_array[-1],interval)
```

```
#====================================
        time_array_p = np.concatenate((time_array_p, time_array))
        print("len=",len(time_array_p))

        hist_array_p = np.concatenate((hist_array_p, hist_array[0]))

        new_hist_array=[]
        for i in hist_array_p:
                new_hist_array.append(i/interval)
#=========================================================================
        new_edges = np.concatenate((start_edges, end_edges))
file_path = '/data/resh/P70094/output.xlsx'
df = pd.read_excel(file_path)

colors = [ 'blue', 'orange', 'green', 'red']
df1 = df[['time', 'cluster']]
for i in range(4):
        filtered_df1 = df1[df1['cluster'] == i]
        # 取出'totalphotonnumber'列並轉換為一維陣列
        globals()['time'+str(i)] = filtered_df1['time'].values
        print(globals()['time'+str(i)])

for i in range(4):
        for j in range(len(globals()['time'+str(i)])):
                plt.axvline(x=(globals()['time'+str(i)][j]+267032400),
                                color=colors[i]) # Plotting a vertical line

#=========================================================================
plt.plot(time_array_p,new_hist_array,'',linewidth=0.5)
plt.title("70094-01-03-00_1d0")
plt.xlabel("Time(s)")
plt.ylim(top = 6000)
plt.ylabel("Count/sec")
plt.legend(loc='upper left')
plt.savefig("test7.png")
plt.show()
=========================================================================
```

畫出 hardness ratio 在時間上的分佈



程式碼

```
================================================================================
#!/usr/bin/python

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from astropy.io import fits
import ast
#============================
f = open('/data/resh/path_test.txt')
text = []
for line in f:
        text.append(line)
f.close()

v = 'p0_0.01'
spec, time_array_p, hist_array_p, start_edges, end_edges = [211, 212, 213], [], [], [], []
```

```
for z in spec:

    text1 = text[z]
    print("text1=",text1)
    cleaned_path = text1.strip("'\n")
    print(cleaned_path)
    fits_file_path = cleaned_path
    hdul = fits.open(fits_file_path)
    data = hdul[1].data
    hdul.close()

    data_array = np.array(data)
#================================
    name_f = cleaned_path
    file_r =name_f.split('/')[2]
    file_rr =name_f.split('/')[3]
    file_name =name_f.split('/')[5].replace('.evt.bary', '')

    path1 = f'/data/resh/edge/{v}/{file_name}.txt'
    with open(path1, "r") as d:
        new_data = d.read()
        d.close()

    print("new_data = ", new_data)
    array = eval(new_data)

    for i in range(len(array)):
        start_edges.append(array[i][0][0])
        end_edges.append(array[i][0][1])
#================================
    #def count_soft():
    threshold = 9
    rows_to_keep = []
    for row in data_array:
        if row[4] <= threshold:
            rows_to_keep.append(row)
    count_soft = np.array(rows_to_keep)

#================================
    time_soft = count_soft['TIME']
    for i in range(len(start_edges)):
        time_soft = [x for x in time_soft if not (start_edges[i] <= x <= end_edges[i])]
#================================
    #def count_hard():
    threshold1 = 9
    threshold2 = 24
    rows_to_keep = []
    for row in data_array:
        if row[4] > threshold1 and row[4] <= threshold2:
```

```
                rows_to_keep.append(row)
    count_hard = np.array(rows_to_keep)
#================================
    time_hard = count_hard['TIME']
    for i in range(len(start_edges)):
        time_hard = [x for x in time_hard if not (start_edges[i] <= x <= end_edges[i])]
#================================
    interval=128
    first=count_soft['TIME'][0]
    last=count_soft['TIME'][-1]

    hist_soft = np.histogram(time_soft, bins = np.arange(first, last, interval))
    time_array=hist_soft[1]
    time_array = np.arange(time_array[0]+0.5*interval,time_array[-1],interval)

    hist_hard = np.histogram(time_hard, bins = np.arange(first, last, interval))
    hardness = hist_hard[0]/hist_soft[0]
#================================
    time_array_p = np.concatenate((time_array_p, time_array))
    print("len=",len(time_array_p))

    hist_array_p = np.concatenate((hist_array_p, hardness))
#=========================================================
plt.plot(time_array_p,hist_array_p,'',linewidth=1)
plt.title("70094-01-03-00_1d0_out")
plt.xlabel("Time(s)")
plt.ylabel("hardness ratio")
plt.savefig("time_hardness_out.png")
plt.show()
```

【評語】160029

本作品以機器學習的方法建立“磁星”短 X 射線爆發的性質，並希望藉以檢驗“磁星”短 X 射線爆發是否可能伴隨著快速電波爆發的發生．本項作品顯示作者有良好的數據處理能力，也有後續發展的空間．建議應針對磁星 SGR 1935+2154 的短 X 射線爆發性質先行探討，而且除了 1E2259 之外，可再檢驗其他目標，並可試著探討快速電波爆發與短 X 射線爆發之間可能的物理關聯。