題目: Pixel Based影像辨識

關鍵字: Pixel Based, R Language, 非監督式學習

姓名: 陳柏翰

系級、學號: 地理108 40423219L

摘要:

衛星影像辨識是現今GIS領域非常重要的一環，隨著科技的進步，電腦已經可以代替部分的人工分類。然而，大部分時候仍需要人工校正崩塌地等影像區塊，尤其是崩塌地與河道相鄰的區域。該期末作業嘗試透過非監督式分類的方式進行Pixel RGB 的分類，比較單純透過 Classification 產生的 Rules 進行分類與透過各群的 Centroid 來進行Classification，後者是否會有較佳的表現。

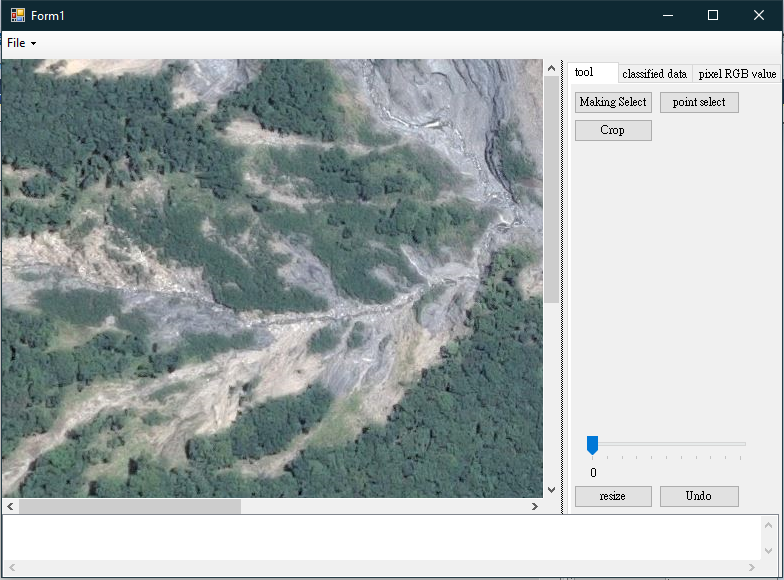
研究動機與研究目的:

身為地理系的學生，我接觸過不少同學曾經在地圖數化公司實習，實習的內容多半是透過人工的方式選擇特定的影像區塊，該作業需要投入大量的勞力與時間，即便是既有的前處理軟體仍由其極限，希望該作業嘗試製作地圖分類的前處理軟體，透過課堂所學知識來嘗試改善地圖分類的問題。

研究方法與步驟:

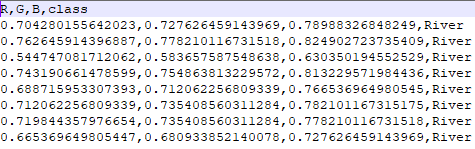
本作業主要分成4個步驟:

一，GUI 編寫: 透過C# Xaml 撰寫GUI

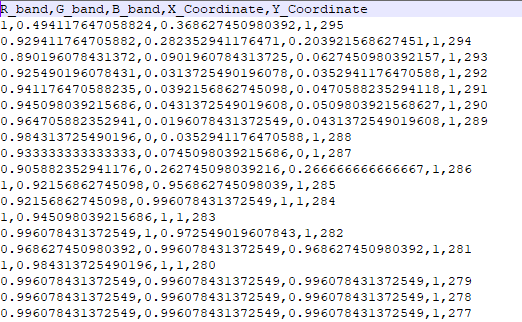


二，資料前處理:

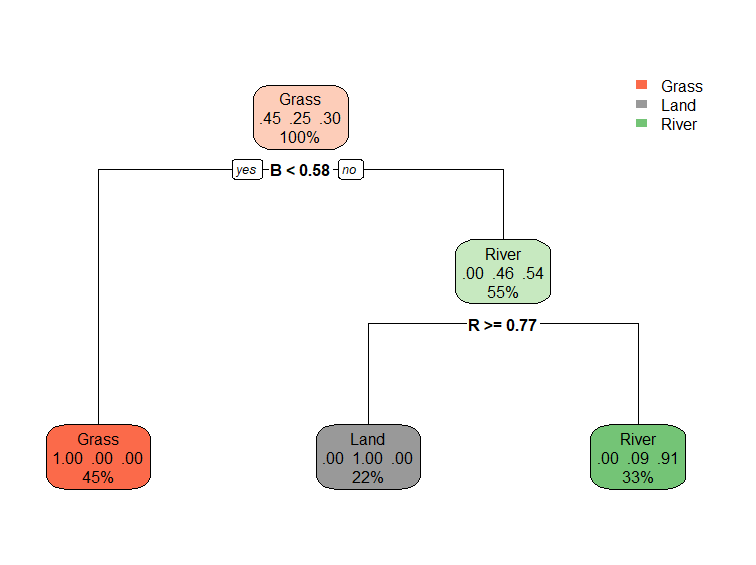
1, 透過第一步驟完成的GUI 進行圖片載入並透過人工分類選取Pixel 作為Training Data



2, 將載入的圖檔轉成 RGB 的資料架構以及X 與Y Coordinate



三，產生分類規則: 將Training Data 載入R script, 透過rpart 來產生分類規則



說明: rpart 的分類規則產生方式是透過Cross Validation以及Gain值的方式來計算使用不同的Column 何者Gain 值最大

四，將Cluster 分類: 透過kmeans 以及 DBSCAN 產生Cluster，以每個Cluster 的Centroid作為Classified 的資料

技術說明

一、資料讀取

透過Pixel 的RGB 作為影像辨識在影像辨識的領域被稱作Pixel Based，該方法所面臨的最大考驗在於如何降低處理時間，本作業的核心運算透過R 語言來進行，經過測試R 在執行Kmeans 運算上的速度對整體運行影響不大，主要的運算時間是花在讀取Pixel，為降低pixel 的讀取時間，程式最初是透過1行行的資料讀取，此方法在降低時間上有顯著效果，尤其處理的資料其比數多半超過1萬(100\*100)筆，但在運算的效果上的並無顯著的提升辨識效果。

面對以上問題，其原因應該是因為1行行讀取的Pixel檔其顏色的群聚性不高，在分群上無法將相同顏色的資料區分出來，故Pixel的讀取改由一格格的方塊(例: 30x30)來選取範圍，該方法在圖形的分群效果較前者佳，但在時間的處理上比一行行讀取費時許多，一行行讀取在處理大量資料時，所花費的時間不超過1分鐘，但區塊的讀取所花費的時間動輒10分鐘起跳，本作業嘗試降低時間的消耗，透過先讀取直行在選取橫列的方式確實有降低讀取時間，但依舊比一行行讀取來的緩慢。

二、分群: Kmeans

為節省運算時間，透過R現有kmean 套件來進行資料上的計算，為產生最佳分類效果，透過迴圈的方式計算分群個數2到15，若WSS/BSS<=0.05則採用該分群個數，分群個數在載入下一份資料的時候重新產生(例: 載入下一個圖塊)。

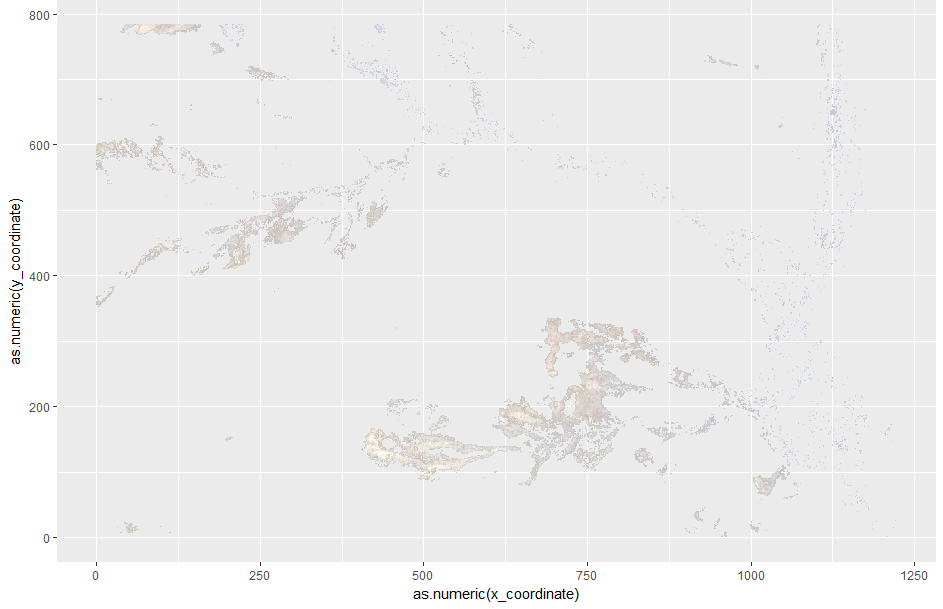
三、分群: DBSCAN

透過R現有DBSCAN 的套件處理資料，eps以及minPts 透過人工的方式調整以取得最佳值。

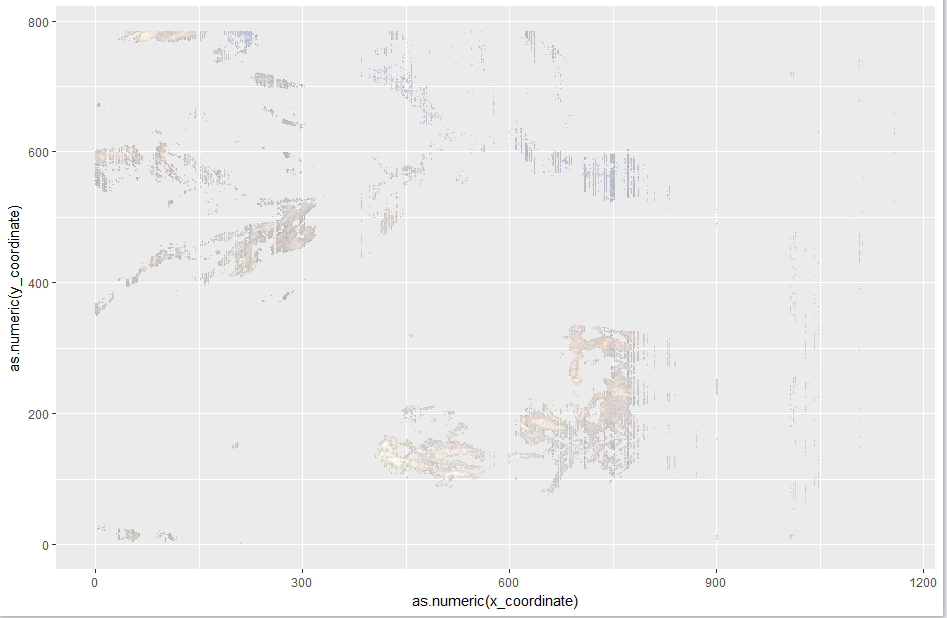
效能評估過程



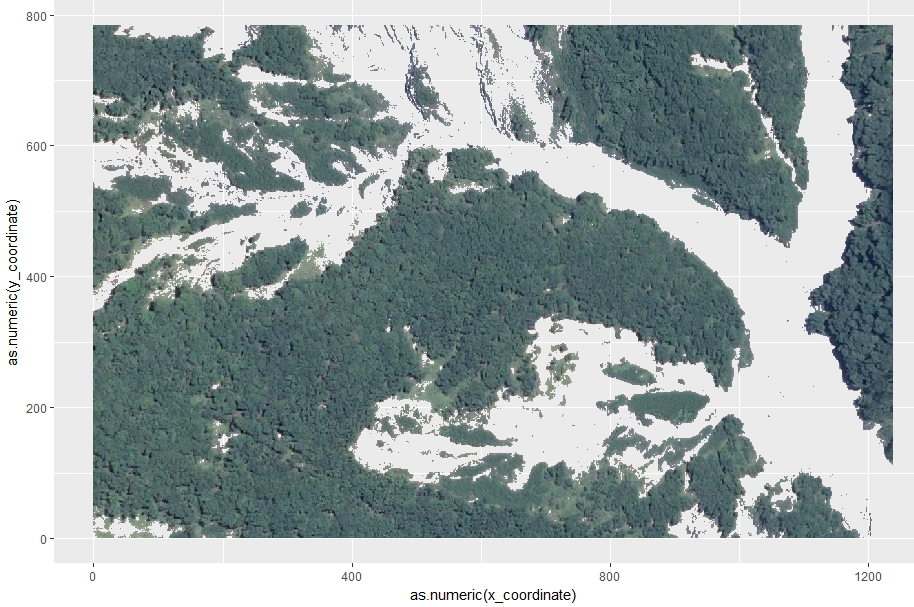
**原圖**



**對照組: Classified**



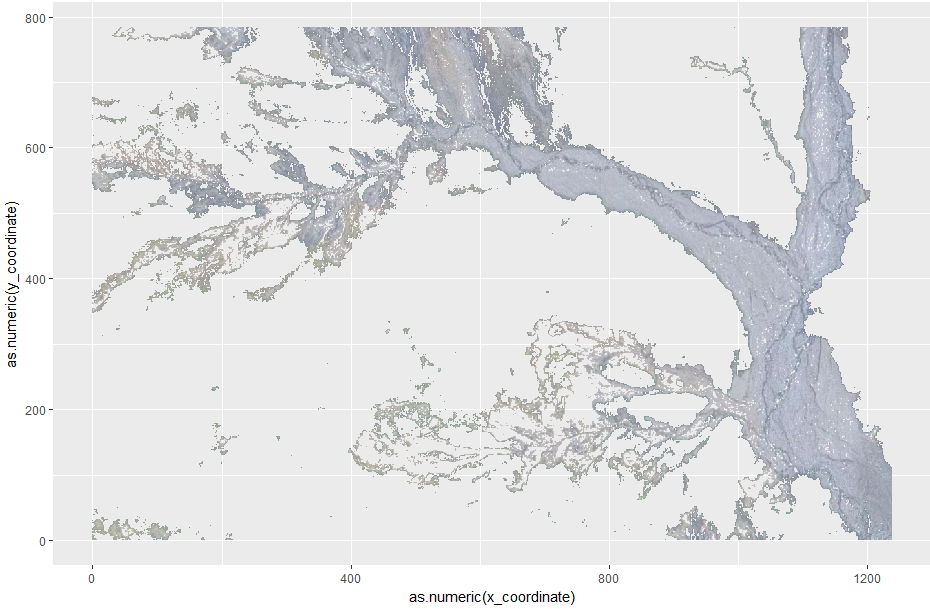
**實驗組: kmeans**



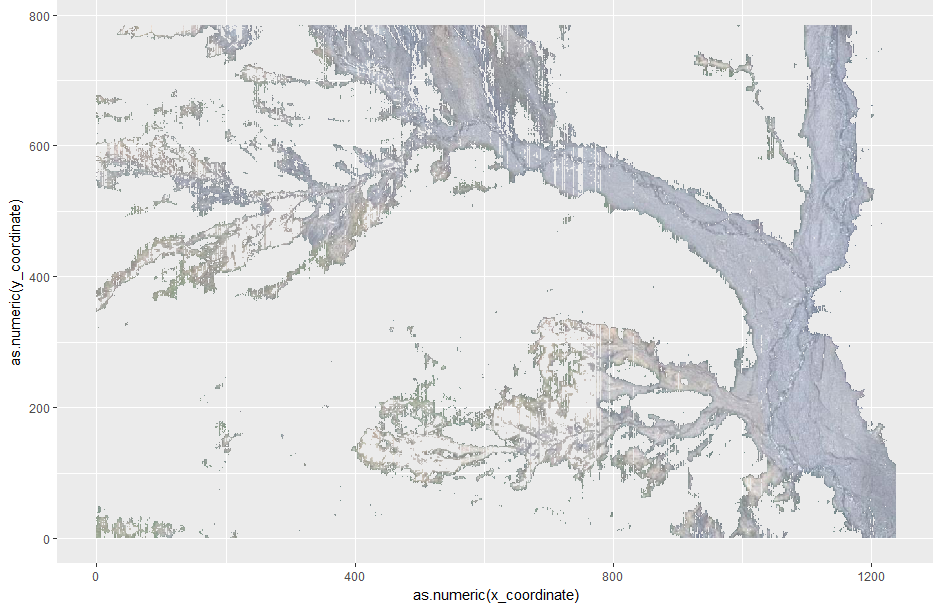
**對照組: Classified**



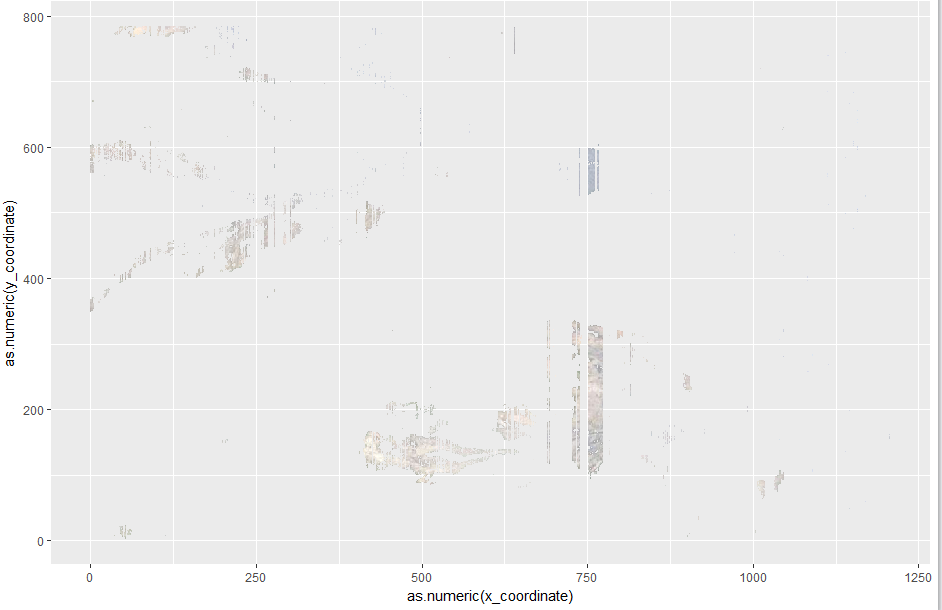
**實驗組: kmeans**



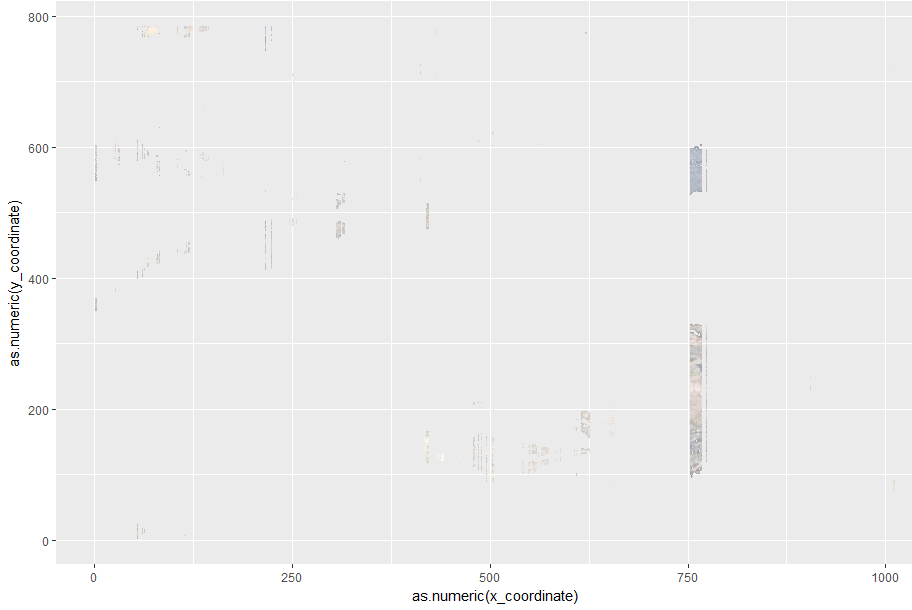
**對照組: Classified**



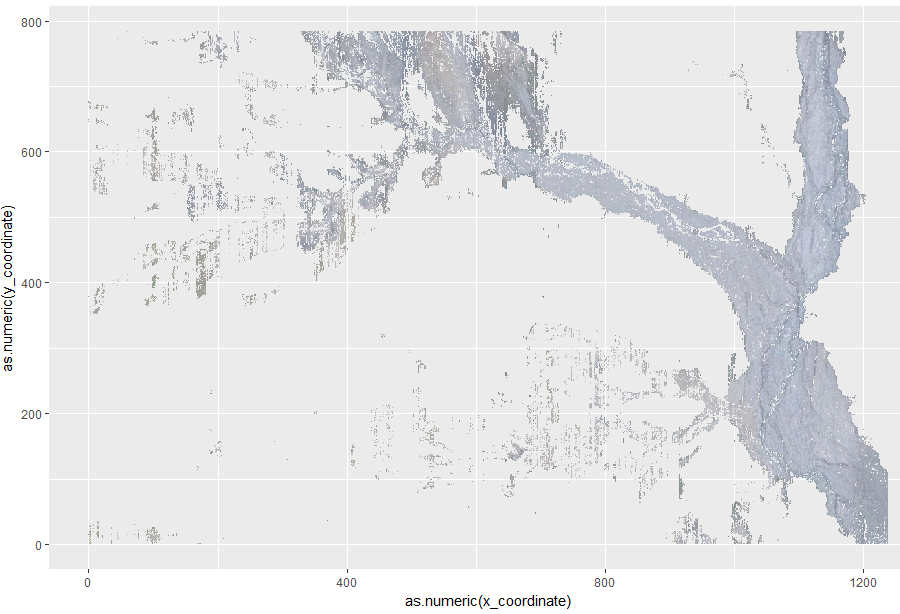
**實驗組: kmeans**



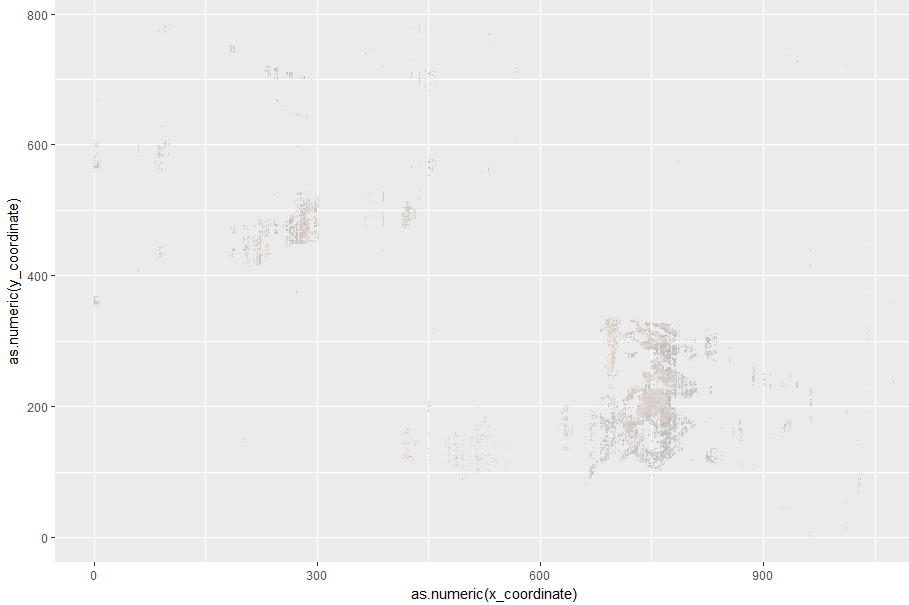
**實驗組: DBSCAN eps = .03, minPts = 5**



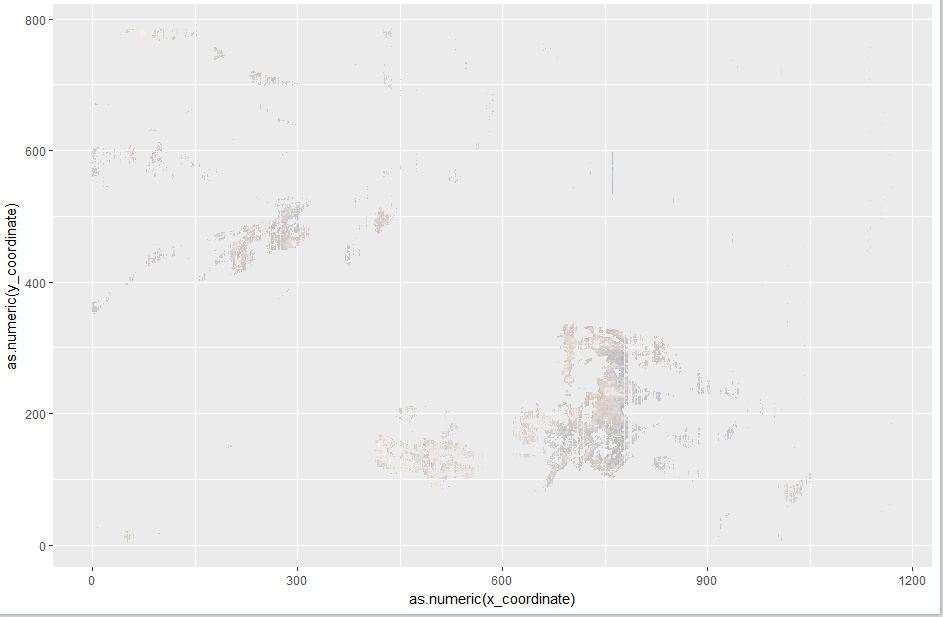
**實驗組: DBSCAN eps = .05, minPts = 10**



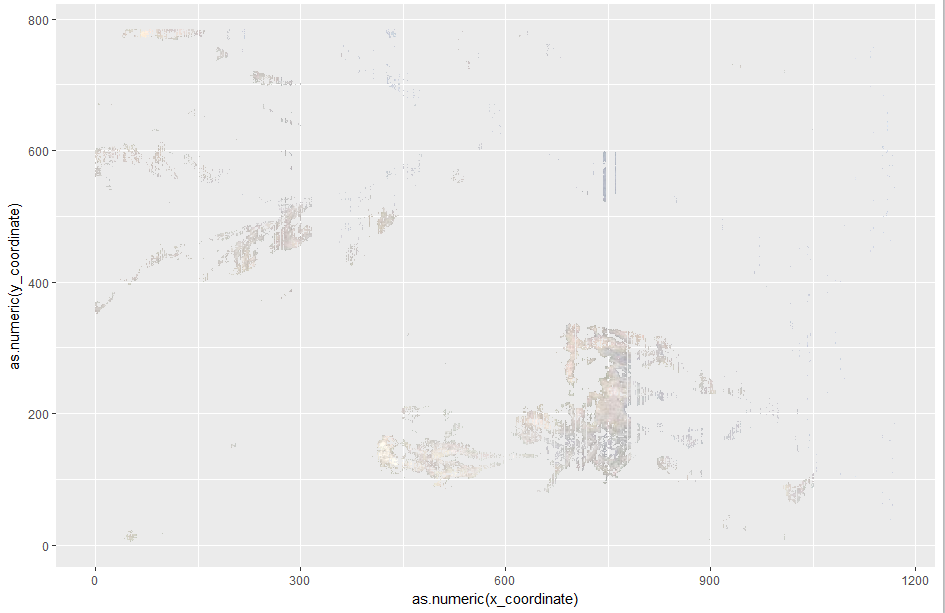
**實驗組: DBSCAN eps = .02, minPts = 10**



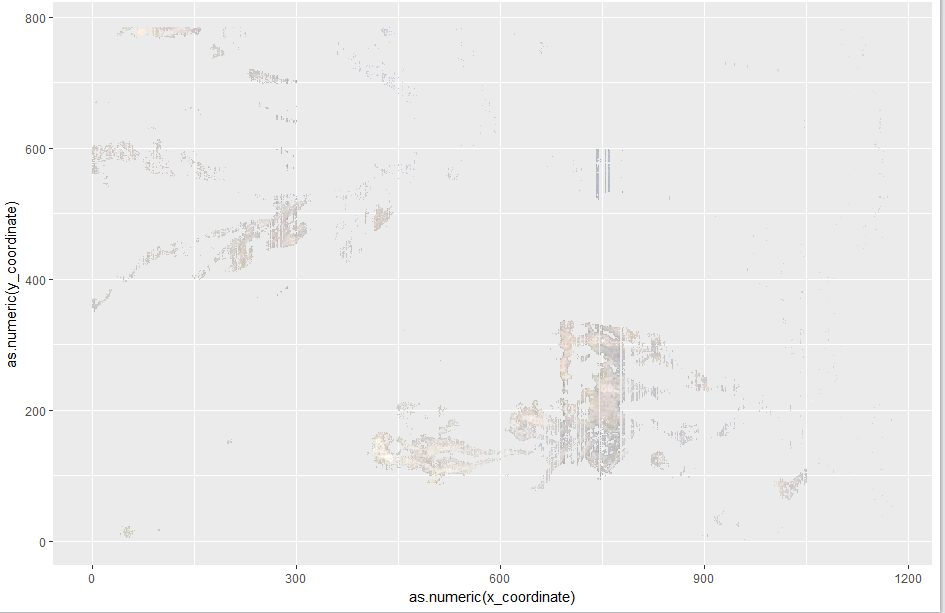
**實驗組: DBSCAN eps = .02, minPts = 10**



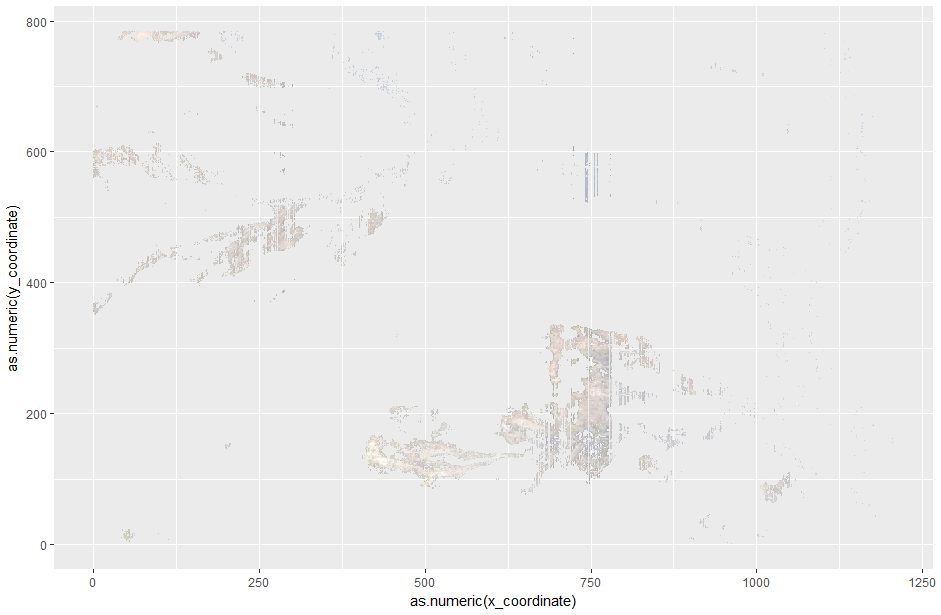
**實驗組: DBSCAN eps = .02, minPts = 7**



**實驗組: DBSCAN eps = .02, minPts = 5**



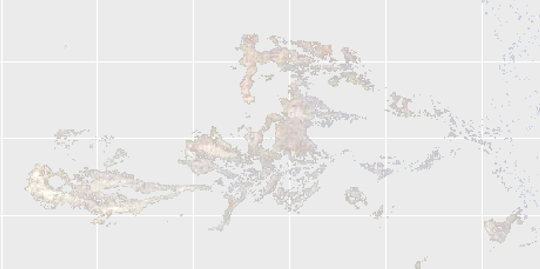
**實驗組: DBSCAN eps = .02, minPts = 4**



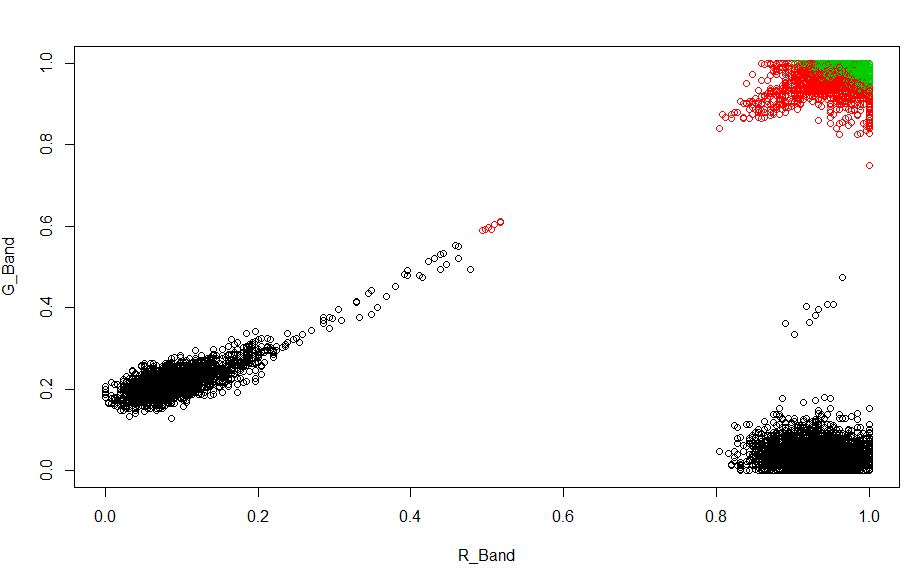
**實驗組: DBSCAN eps = .02, minPts = 3**

說明:

上述圖表為各次實驗操作過程的崩塌地選取示意圖，可以發現隨著eps以及minPts 的調整，DBSCAN 的選取範圍有漸漸涵蓋多數的崩塌地，但因為是一行行的進行資料分群，若該行的崩塌地資料筆數不足極有可能被視為雜訊，因而導致出圖的崩塌地圖形不連續。將兩個崩塌地區域進行比較，可以發現直接透過規則分類的圖形不會出現不連續的綜線，但在圖形的完整選取上，DBSCAN 所產生結果其選取的範圍相對較為完整

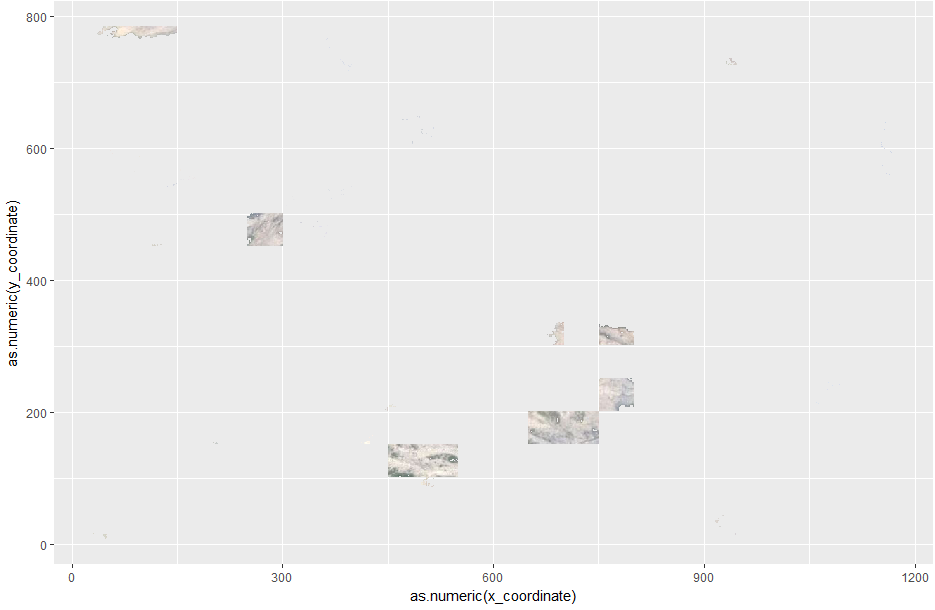
對照組: Classification

實驗組: DBSCAN

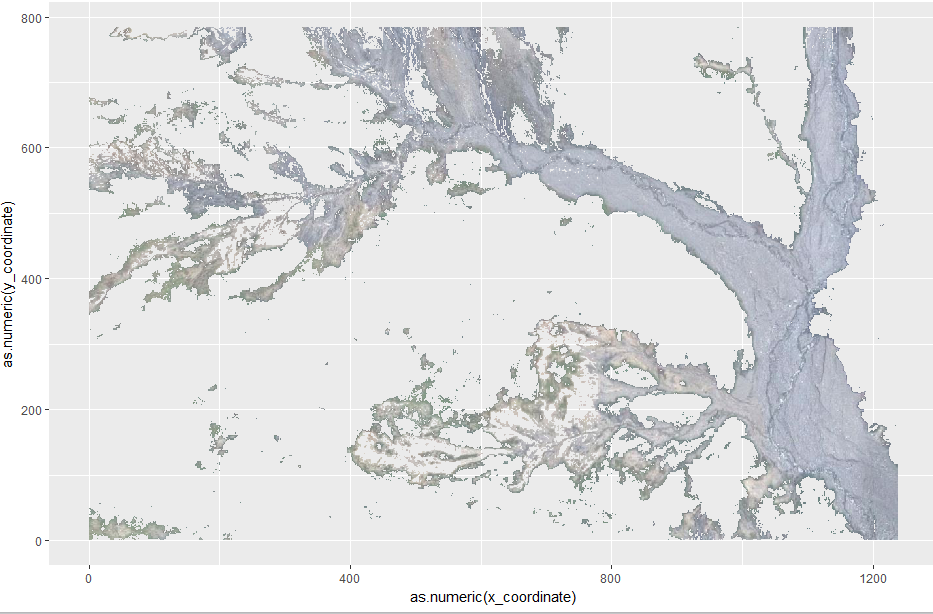


資料形式示意圖:

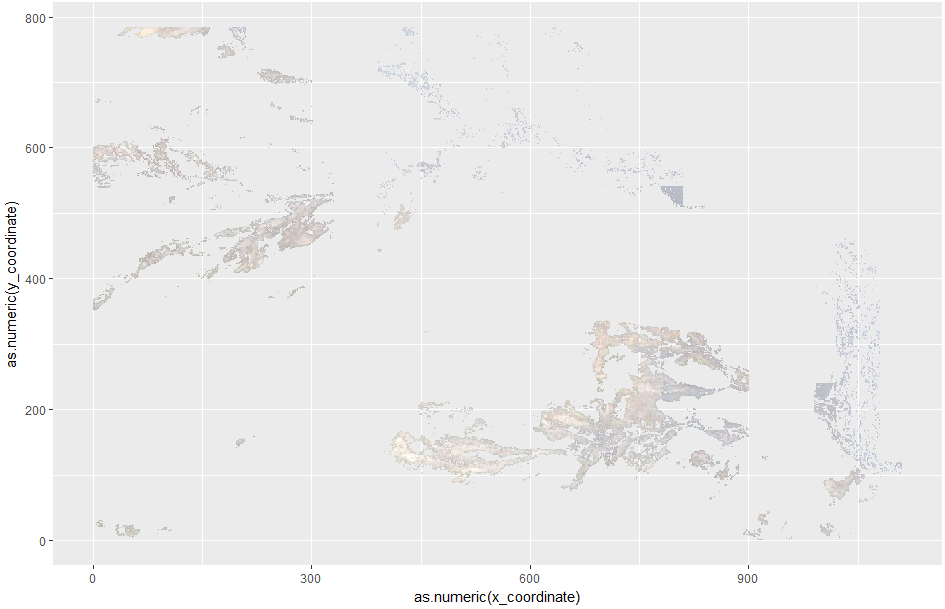
該圖表為RGB中的R\_Band以及G\_Band 濃度所產生的分布圖，每一個點即代表一個pixel，該資料並非本次實驗資料。



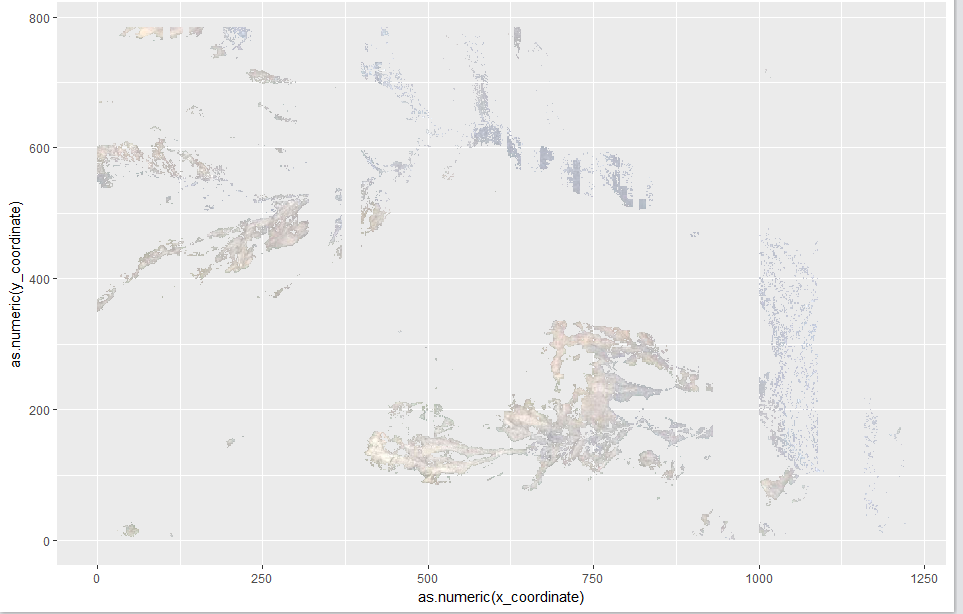
**實驗組: DBSCAN 50x50 eps = .02, minPts = 30**



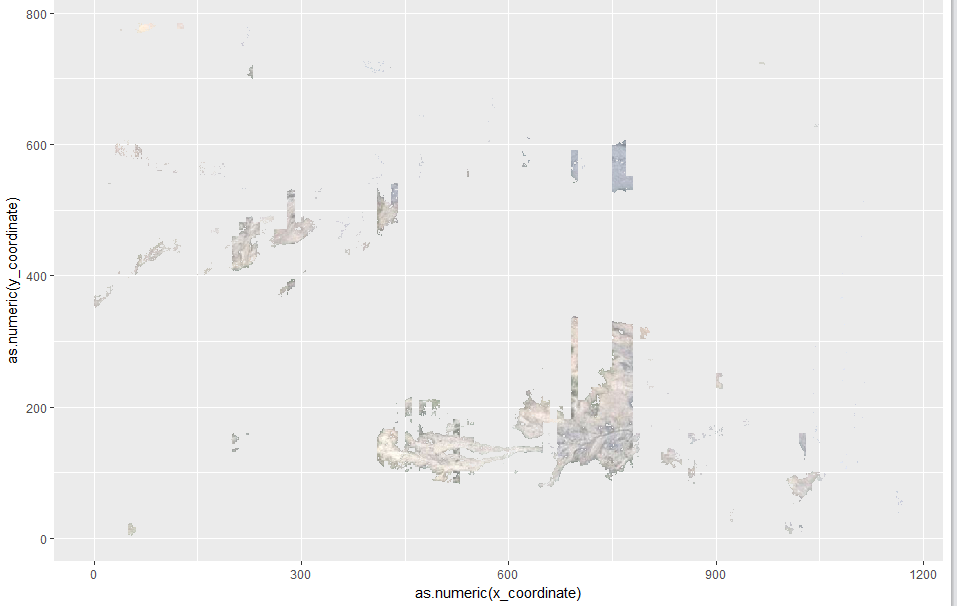
**實驗組: Kmeans 30x30**



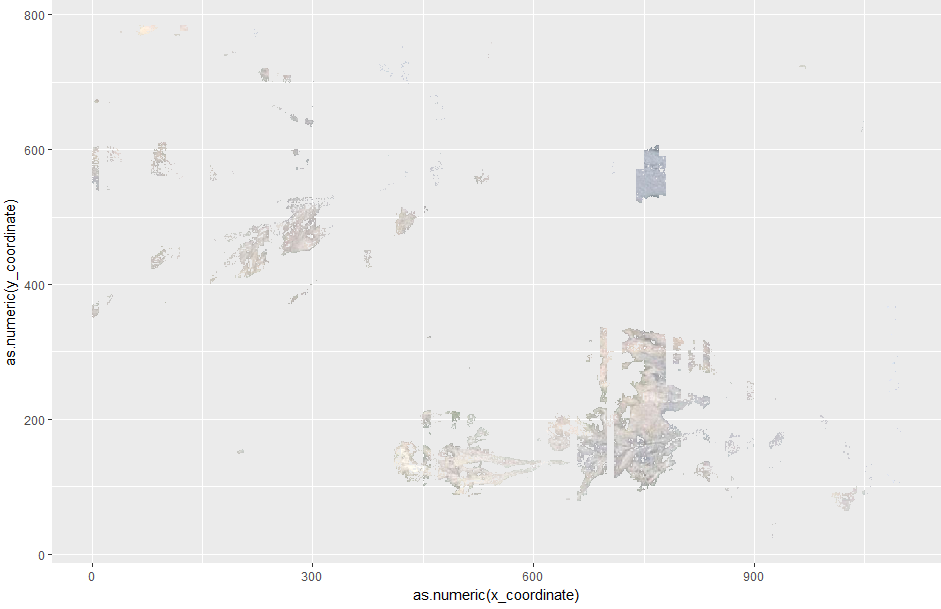
**實驗組: Kmeans 30x30**



**實驗組: Kmeans 10x10**



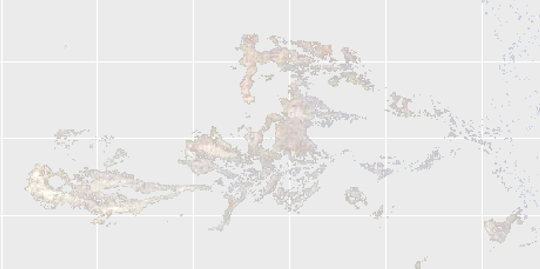
**實驗組: DBSCAN 10x10 eps = .02, minPts = 15**



**實驗組: DBSCAN 10x10 eps = .02, minPts = 40**

說明:

上述資料為改用方塊的方式來選取資料範圍進行影像分群，可以發現透過方塊的圖形分群可以保留相鄰pixel 的影響，將崩塌地的圖形大致上完整的選取，Classification產生的圖形則相對地破碎。

對照組: Classification

實驗組: DBSCAN

**結論:**

衛星影像的辨識在於降低時間，目前未止本作業所採用的圖塊載入方法所需時間都至少要10分鐘，如何在更短的時間內達到相同的效果是必須要克服的目標，尤其影像辨識所使用的圖形解析度極高，若需要大量時間才能將資料分類其結果將毫無意義。在比較的過程可以發先Classification 所需要的時間極少，如果將影像先經過Classification 後在進行影像分群或許可以大幅降低運算時間，若再將影像的RGB濃度透過Logistic function 將濃度的分界拉大可能也有助於提升辨識效果。此外，面對影像辨識的種種問題，無法辨識陰影區域以及雲朵遮蔽的地方是所有分類軟體的通病，透過Classification 後再進行Clustering 的分群可嘗試作為解決辦法。