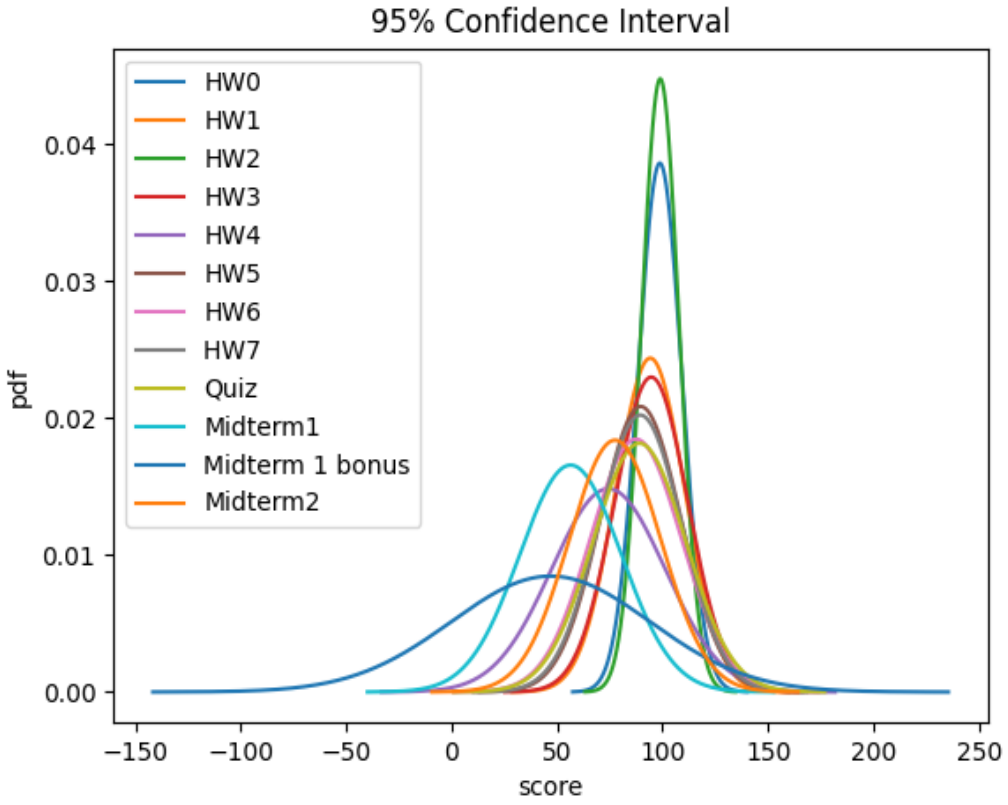
商研二 r10741056 譚庚豪

1. 資料預處理:
   - Removal: 首先我將”class name”, “submission id”這兩個columns刪除, 再將”submission_time”的格式轉成datetime以利後續分析。
   - Filter: 基於我的分析方向和PDOGS的評分方式, 且考慮到有同學可能會在deadline後上傳作業以當作練習, 於是對於每一次作業、考試, 我只取它們deadline前的最後一筆(i.e. 有被採計分數的那一次)

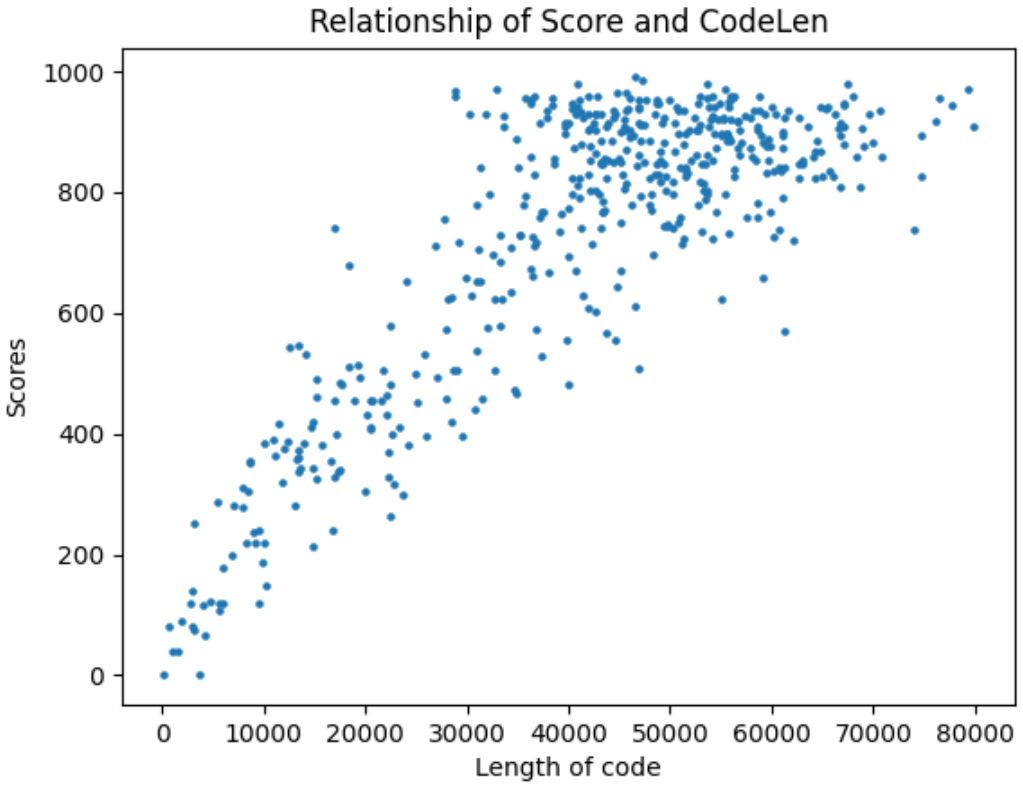
2. 每一次作業的mean distribution:



這部分的計算為, 加總每次作業、每個人所獲得的總分, 在利用總分與人數求取平均與標準差, 我的計算過程中有考慮加權(e.g. hw0的滿分為40, 將其乘以2.5、midterm2的滿分為120, 將其乘以0.83), 就平均分而言, 前三低(從最低開始)依序為「midterm1 bonus, midterm1, hw4」, 深入研究後, 發現前兩者之所以會如此低可能是因為「題目具備一定難度」加上「時間限制」, 以midterm1 bonus來說, 題目本身運用到三維陣列, 且「實際上」的寫作時間可能只有三天(因為hw4的deadline為4月1號), 這也許是它獲選為最低平均分的原因, 而就hw4來說, 首先它需要運用到Dynamic Programming的知識, 這對初學者來說不容易, 且時間上它又跟midterm1撞到, 這可能是它平均分也如此低的原因。值得注意的是, 以標準差的角度看上去, midterm1 bonus長得最寬、

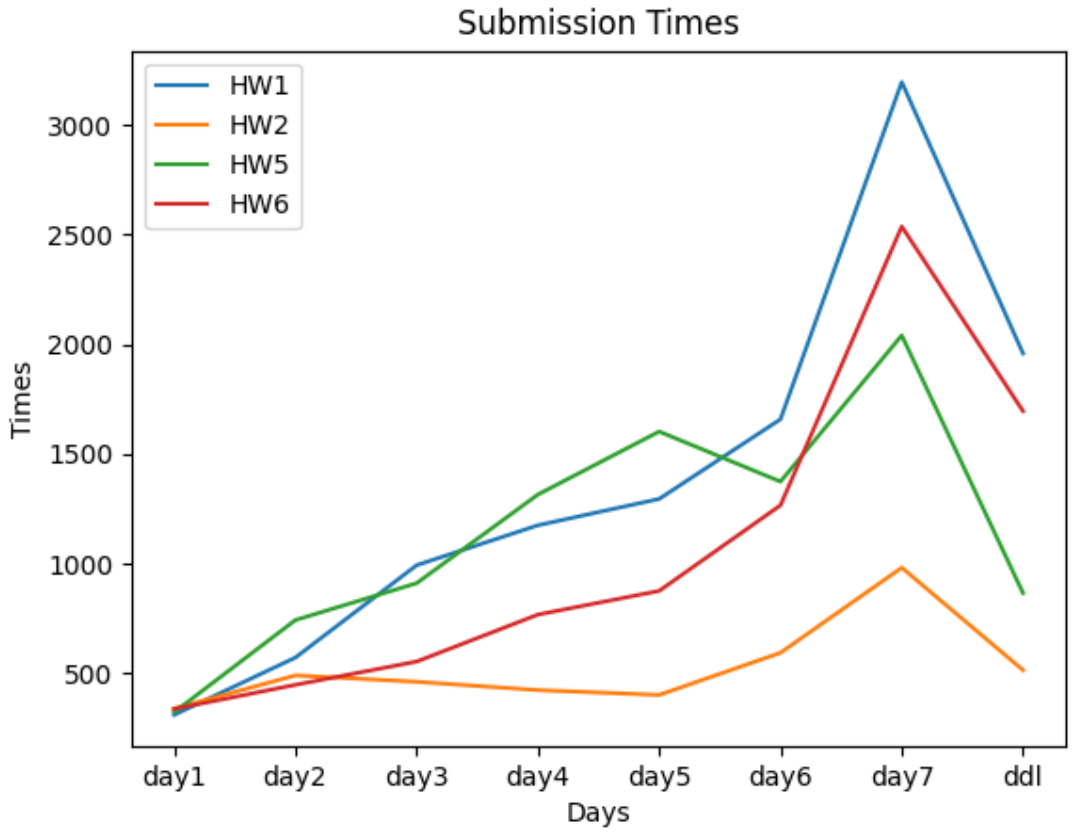
HW0及HW2長得最窄(同時這兩者也是平均分最高的)，前者應該是由於上述提及的難度，以及它其實只有一題所導致(分數可能不是100就是0)，而後者可以說是本次課程最簡單的兩次作業，也就是眾所皆知的賣禮盒題，除此之外大部分的作業都落在80~100分，所以以數據而言，這堂課的作業算是相對簡單了。

3. 分數和codes長度關係圖：



如圖，每一點代表「每一位學生本學期所獲得的總分以及他所有作業的代碼總長度」，我依據代碼的長度由短到長去排序成績，令人意外的是，點陣圖的尾部沒有呈現下墜的趨勢，因為依照傳統觀念認為，同樣一個任務，如果代碼寫得越冗長(太多沒必要的變數 or 太多沒用到函式 等等)則越容易有bugs，然而這個觀念顯然不適用於這堂課，甚至可以以一句話形容這張圖：「就是寫爆」，然而這個現象不見得是好事，對於一個程式初學者而言，「如何精確地寫好每一行程式」可能是對他再好不過的訓練，也是一個好習慣的養成，所以建立在這堂課「相對簡單」的情況下(由上一張圖)，我建議設計測資的助教們可以試著往壓低時間、空間複雜度的方向去思考(e.g. 從O(n^2)壓到O(n)之類的)，以達到讓初學者從一開始就學著去設計更好的演算法、避免迴圈的濫用等目的。

4. 上傳次數和deadline關係圖:



原本想針對每次作業都做一個線圖，但礙於篇幅限制，這邊只挑選寫作時間同樣為8天的hw1, hw2, hw5, hw6，不過我想結果應該是差不多的，首先從不同作業的高低差來看，搭配上面第一張圖可以大概推測hw2上傳次數之所以那麼少可能是因為過於簡單(也可以從day2後到day5呈現一段下降趨勢得出此結論)，相比之下hw1的題目難度有顯著上升的情況，而hw5, hw6的上傳次數之所以介於兩者中間，考慮到當時已處於學期中(考完midterm1)，可能是因為有一定比例的學生停修所致。而再看整張圖的走勢，不難看出隨著deadline的逼近，上傳次數有明顯的上升趨勢，尤其是day6~day7這一段特別明顯，然而deadline當天卻都比前一天減少許多，考慮到這四次作業deadline都是截至當天的中午12點，所以撇除大部分的同學是在day7就已經AC，我推測剩下還沒AC的同學應該是由於前一天熬夜起不來，或是已經呈現放棄狀態，所以我在這部分的結論是:「實際上認真卯起來寫作業其實也就逼近deadline那幾天」，於是延續上一段所提及的調整難度的方向，我會建議助教們或老師，如果不願從壓低複雜度的角度著手(可能由於測資的設計較麻煩)，也許可以從壓低作業時限的方向思考，以促進修課學生「以更短的時間、更有效率地寫出一個好程式」，然後可以把多出來的時間教授更深的python應用，或是新增一個期中專案給同學們更多累積實作經驗的機會。