RasMMA交接清單

**目錄**

[程式碼與資料集 2](#_Toc520823786)

[需求對照表（每個需求如何解決） 2](#_Toc520823787)

[程式碼解釋 3](#_Toc520823788)

[RasMMA演算法 3](#_Toc520823789)

[讀取RasMMA的結果 10](#_Toc520823790)

[Global Sequence Alignment 13](#_Toc520823791)

[BegC演算法(Behavior Group Clustering) 16](#_Toc520823792)

[讀取Behavior Forest資訊 18](#_Toc520823793)

[家族分類實驗 19](#_Toc520823794)

# 程式碼與資料集

程式碼位置：<https://github.com/weichih-c/Sequence_Analysis>，裡面有需要的各種程式碼

資料集路徑：Lab的FTP中的路徑－”/証鴻/偉志實驗結果交接.zip”，裡面有ReadMe.txt可以看，會解釋各種實驗input/output的資料

# 需求對照表（每個需求如何解決）

* 獲取各家族的Behavior forest及相關報表資訊：執行RasMMAExample.ipynb
* 想做BegC：先做完各家族的RasMMA後，執行BegC\_Example.ipynb
* 讀取各家族Behavior forest內容：利用CollectForestInfo.ipynb
* 以家族Behavior forest為Model進行分類實驗：
  1. 先執行RasMMA\_Classification\_Testing.ipynb 獲取每個profile在各個家族，各Behavior Tree的相似度分數
  2. 再執行RasMMA\_Classification\_MatchReader.ipynb查看最終結果與產生對應的報表（獲得match rate，Precision/Recall/F1等）

# 程式碼解釋

## RasMMA演算法

簡介：RasMMA是一個基於sequence比對的階層式分群演算法，會在過程中不斷比較任兩個profile之間的API序列相似度，如果相似程度很高，則會把兩個profile合併成一個群；不相像的profile則會被遺棄不分群。

* **執行主程式：RasMMAExample.ipynb**

參數設定: 在main cell裡設定所需變數

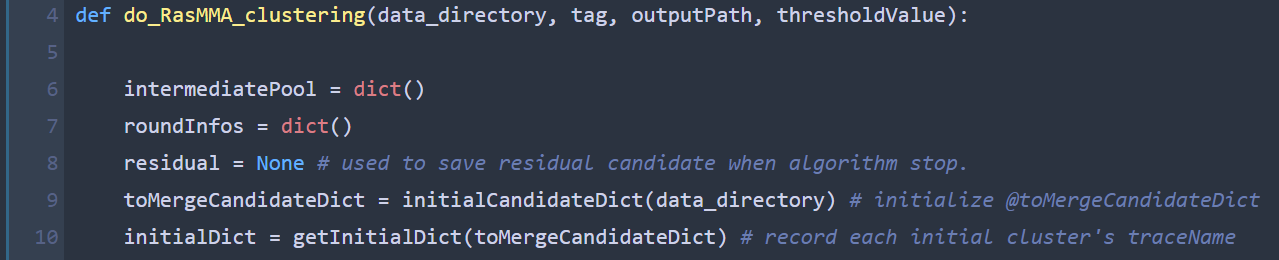
輸入：trace檔的資料夾和指定家族名

輸出：指定家族的Behavior Tree資訊，以Pickle檔存在指定outputPath之中

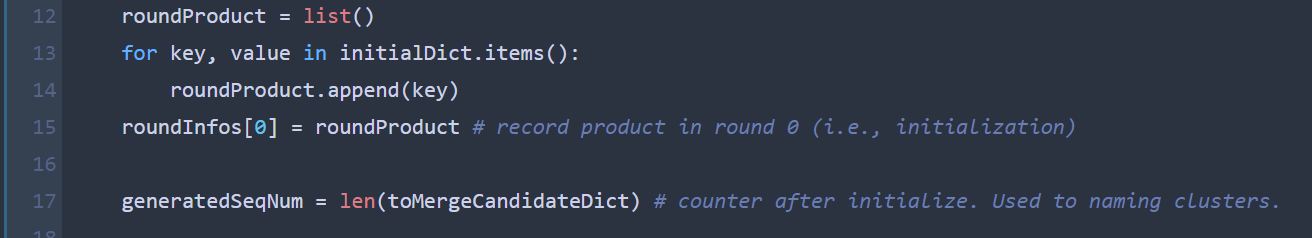


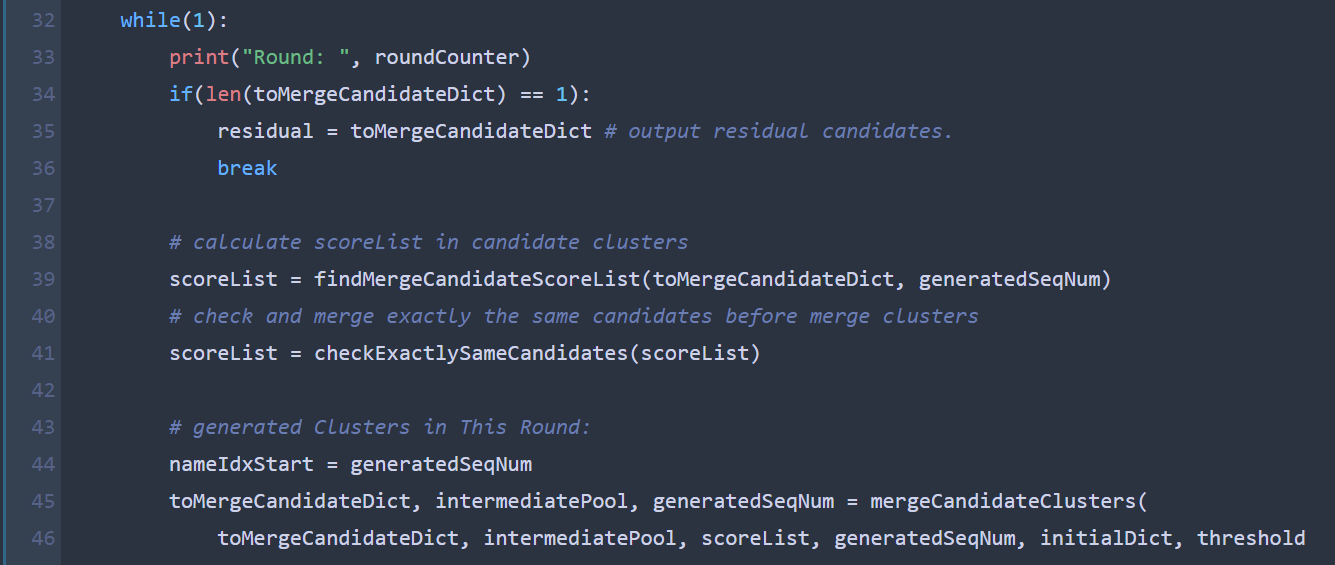
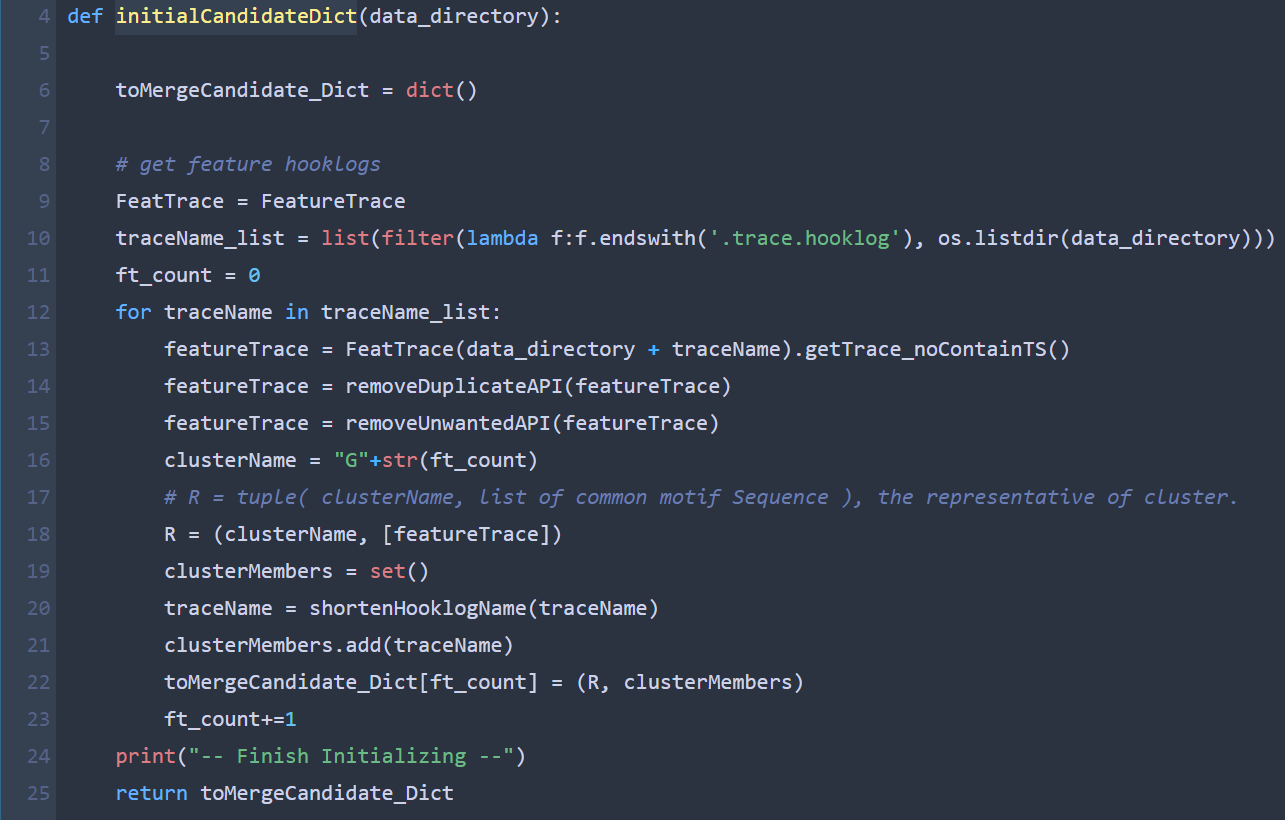
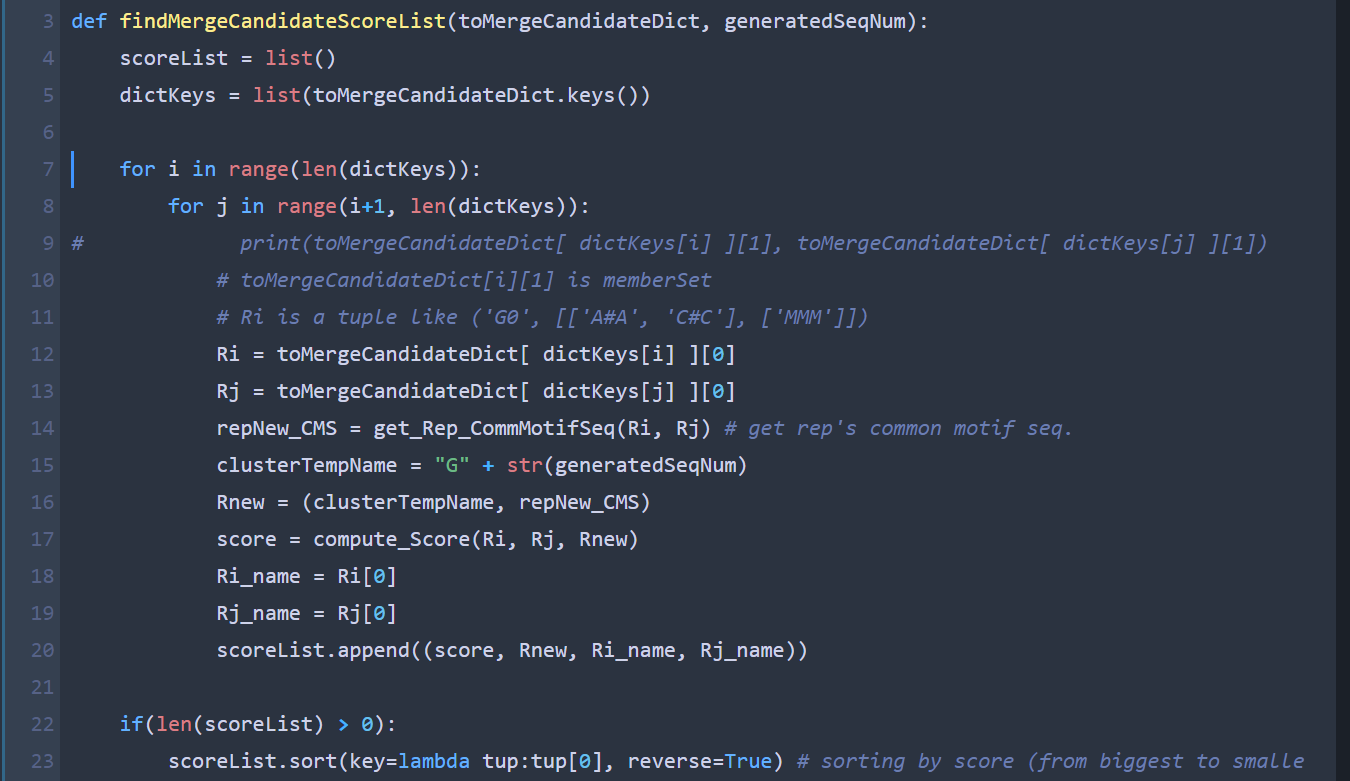
* 上圖是main cell，設定好欲執行的家族名稱和指定路徑之後即可連結RasMMA的運算邏輯，並執行程式，最後輸出到outputPath之中。
* 其中main函式裡只呼叫startClustering函數與紀錄開始／結束時間，startClustering()會用RasMMA演算法對指定資料的traces進行分群，並存成pickle檔輸出到outputPath。
* **運算邏輯：RasMMA.ipynb**

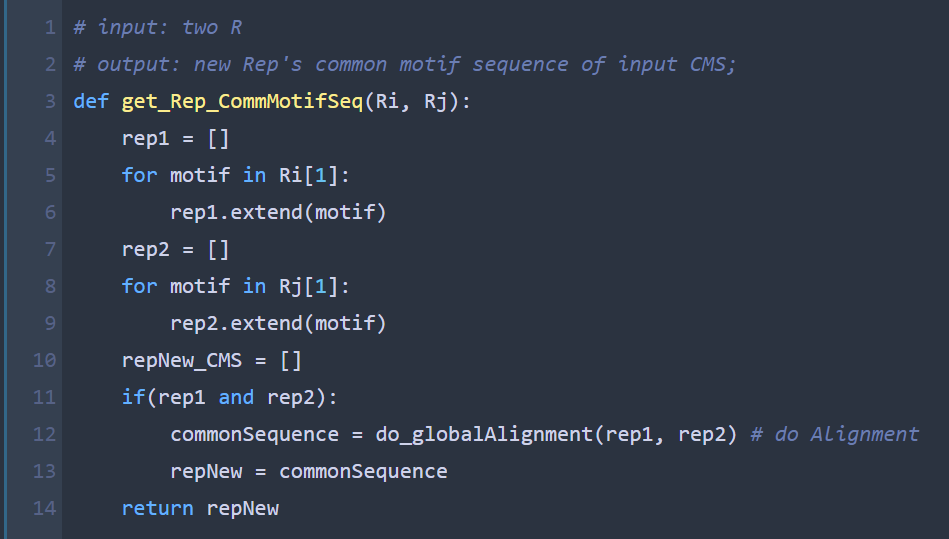
RasMMA演算法共分7個步驟（請參考論文），以下則針對七步驟的實作邏輯進行說明。

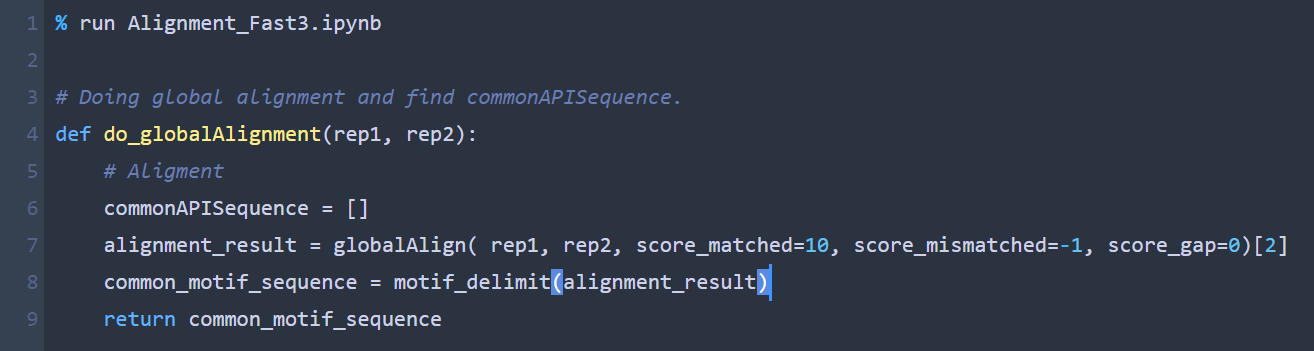
整個演算法的起點從**do\_RasMMA\_clustering()**開始，

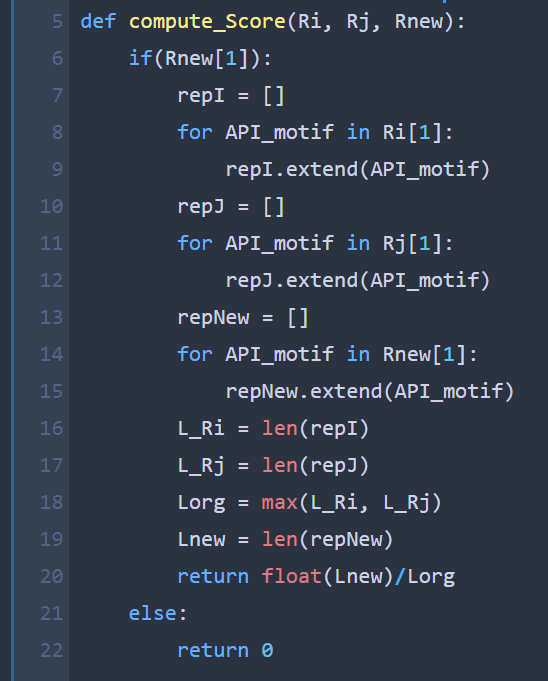
* + 第6行用來紀錄演算法每一輪執行時的中間產物
  + 第7行紀錄每一輪所產生出來的新cluster名字
  + 第9行進行初始化，將所有traces在一開始各自當成一個cluster，並塞入toMergeCandidateDict之中。toMergeCandidateDict是目前準備被合併的cluster候選人，key=第幾個產生的cluster，value=對應的cluster內容
  + 第10行紀錄初始化時每個trace檔分別對應到哪個cluster Name.



* 第12-15行紀錄初始化（也就是第０輪）的產物，即為初始clusters
* 第17行的generatedSeqNum用來紀錄目前所產生到第幾個cluster了(命名用)
* 第32行開始是一個大迴圈，用來重複演算法的第2~6步驟直到終止條件觸發為止，終止條件如下：
  1. 只剩下最後一個candidate了，沒必要繼續分群
  2. 所有candidate之間的相似度都低於threshold
* 第34行是判斷目前剩餘candidate數目
* 第39行是演算法的第2到4步驟，計算任兩個candidate clusters的共同API序列，計算相似度，以及按照大小排序組合。在此步驟是整個演算法的bottle neck，需要對所有n個cluster做次的組合運算。
* 第41行是預先把同為滿分(i.e.,完全相似)的cluster，若彼此之間有交集，則一次合併完。例如A,B相似度100%，A,C相似度100%，則A,B,C三者相似，此時在這步驟先檢查一次這種現象。因為RasMMA每次演算法只會合併兩個，且有排他性，若A被選去一個群則不能加入其他群中，但完全相似的case是特殊情況，在此步驟優化，可以加速演算法運行。
* 第45行是演算法的第5和6步驟，檢查是否有符合資格的cluster（即相似程度高於threshold且組成成員不重複出現在不同群），將新產生的cluster加入mergeCandidateDict中，代替掉舊的cluster。
* 完成上述步驟之後會回傳結果到主程式，進行存檔動作。
* **函式內容介紹－initialCandidateDict**
  + 第10行讀取資料夾內的所有trace檔檔名
  + 第13~15行用FeatureTrace這個Class去讀取trace檔，並做winnowing，最後獲取的featureTrace即為profile，也就是一串API call sequence。
  + 第16行是為每個初始cluster取名，按照建立順序做命名
  + 第18行建立一個物件R，裡面有名字和API Call Sequence，注意這邊的sequence是list型態，因為後面計算時是這個欄位會變成motif sequence，是一個list物件，因此統一。
  + 第19-21行建立cluster member，因為一開始是每個profile各自成群，所以只有自己。
  + 第22行加入candidate dict中，以產生序號做key值。
* **函式內容介紹－findMergeCandidateScoreList**
  + 第7-8行是雙層迴圈，目的是把所有的candidate兩兩配對都計算過
  + 第14行呼叫**get\_Rep\_CommMotifSeq**，用來找到pair的common motif sequence，我們是採用global sequence alignment
  + 第15-16行是建立一個可能的組合，紀錄名字和common motif sequence
  + 第17行呼叫**compute\_Score**計算組合的分數
  + 第20行紀錄組合的分數以及構成的人
  + 第23行依照分數作排序，由大到小排列
* **函式內容介紹－get\_Rep\_CommMotifSeq**



* 第4-9行把兩個common motif sequence串接起來，因為motif sequence是一個list of API list，因此用extend做串接
* 第12行呼叫**do\_globalAlignment**做global sequence alignment運算
* **函式內容介紹－do\_globalAlignment**
  + 第7行引用Alignment\_Fast3.ipynb裡面的**globalAlign**涵式進行運算，此函式將會另外做介紹。
  + 第8行依照motif進行切割，之後回傳motif sequence
* **函式內容介紹－compute\_Score**



* 此函式用來計算兩個API sequence的相似度分數，計算方法是拿他們的common sequence(Rnew)長度當分子，舊兩者的長度中，較長的當分母，相除即得到分數。

## 讀取RasMMA的結果

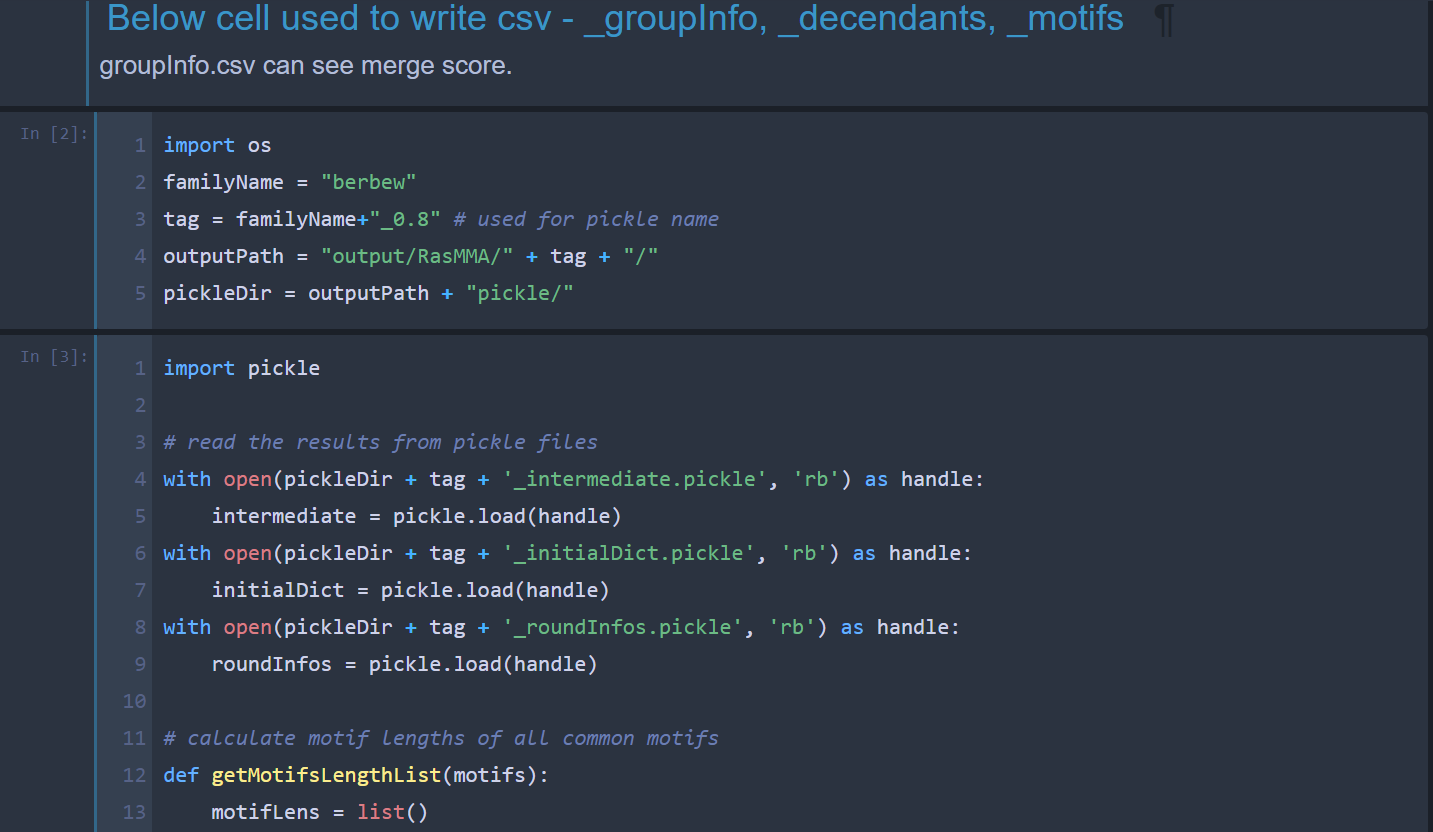
* 讀取結果轉化成報表

程式：**RasMMAExample.ipynb**

輸入：指定家族的RasMMA outputPath，目的是讀取其pickle檔。

輸出：家族的RasMMA分群結果，會產生３個csv檔可用excel開

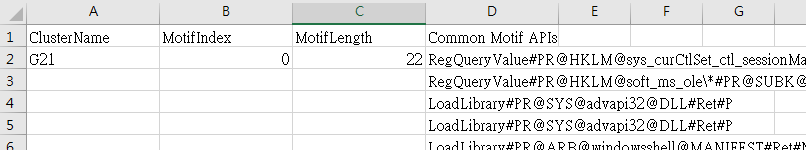
**執行以下cells即可在指定資料夾產生報表**



其中outputPath必須為RasMMA產生結果的資料夾，此兩格用途是讀取家族pickle檔，並產生對應報表－GroupInfo.csv, Descendants.csv, Motifs.csv

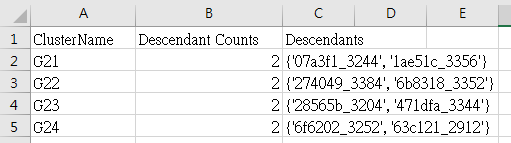
* 閱讀報表欄位
  + Motifs.csv：裡面存放每個cluster的每一個motif的API內容

**注意：Loner cluster不會出現在此處!**

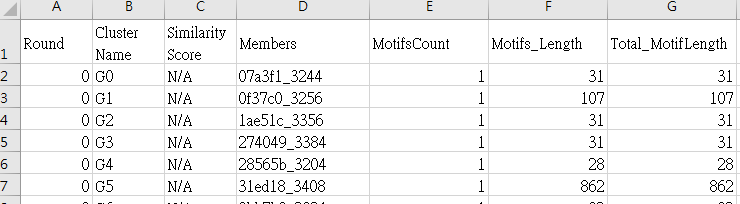


* + - clusterName是cluster名稱
    - MotifIndex是該cluster的motif sequence的編號, 從0開始
    - MotifLength是指定編號的motif裡面有幾個API calls
    - Common Motif APIs是指定motif裡頭的API內容
    - 如圖顯示：motif-0有22個API calls
  + Descendants.csv：存放每一個cluster(Behavior Tree)是由哪些traces所構成

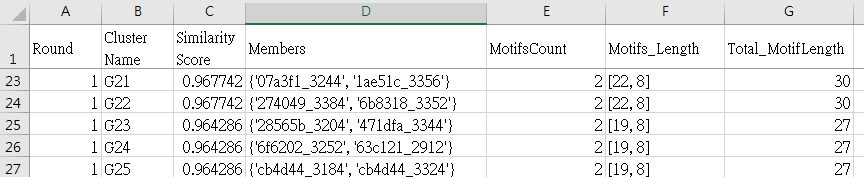
**注意：Loner cluster不會出現在此處!**



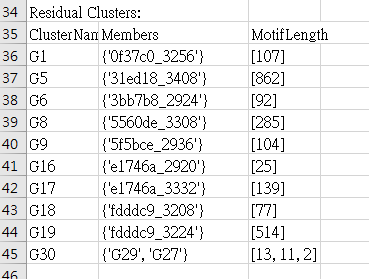
* clusterName是cluster名稱
* Descendant Counts是該cluster由幾個profile所構成
* Descendants是profiles是誰
  + GroupInfo.csv：存放合併過程中所有的資訊，包含每一群的分數以及哪些是Loner Tree，哪些是Tree Root。



* Round是代表第幾輪所產生的cluster，Round 0代表是初始化所產生的群
* Similarity Score代表是合併該群時所依據的分數，round 0是初始化所以不會有分數
* Members是代表該群包含的profiles或者群體名稱，round 0只會有自己一個
* Motif Count是代表該群裡的common motif sequence有幾個motif，每個motif是一小段的API Call
* Motifs\_Length是代表每個motif各自的長度，如果只有一個motif則只有一個長度，若多個motifs則會是一個序列
* Total\_MotifLength是將Motifs\_Length裡頭所有長度的總合



* Round1之後開始有了合併過程，以clusterName – G21為例
* 分數是0.9677
* 由兩個profiles所構成
* 有兩個Motif片段
* 每個片段長度分別為22和8
* 總和是30



* **如何尋找LonerTree與TreeRoot?**
  + - 在GroupInfo.csv下面有個[Residual Clusters]，此處存放著到演算法終止之後仍未被合併的clusters，包含tree root與loner
    - Loner：在這邊代號小於初始化代號（round0所產生）的clusters
    - TreeRoot：代號大於初始化代號者，不是初始化產物又不能跟他人合併，即為Tree root cluster。

## Global Sequence Alignment

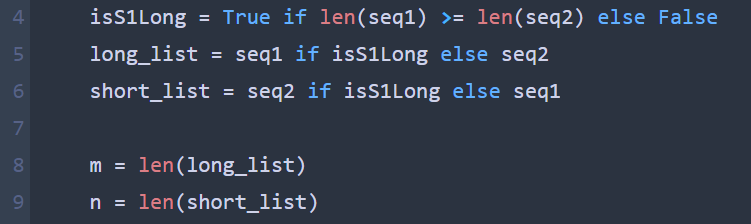
簡介：此演算法詳細邏輯可參考wiki - needleman wunsch algorithm，其用途是可將兩串序列拉到一樣長度之後，跨越不同之處並找到兩串序列中相同的部分。

* **程式所在：Alignment\_Fast3.ipynb**

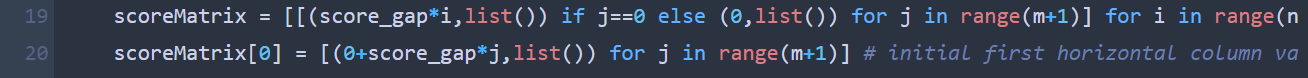
此程式純粹是邏輯運算，只是個module檔，實作Needleman Wunsch 的Global Sequence Alignment。

* 函式介紹－**globalAlign**

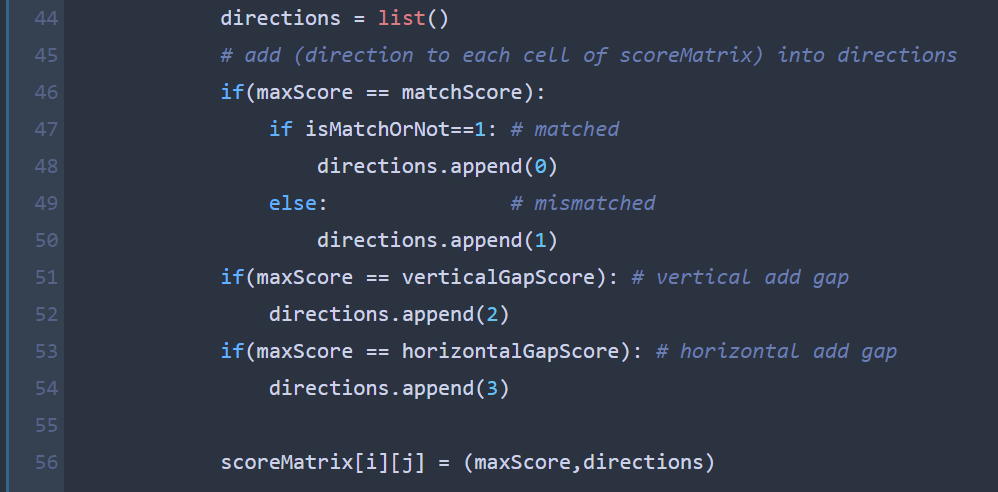
參數設定 – 需要餵入想比較的兩串序列以及matchScore, mismatchScore, gapScore。（用途請參考wiki）



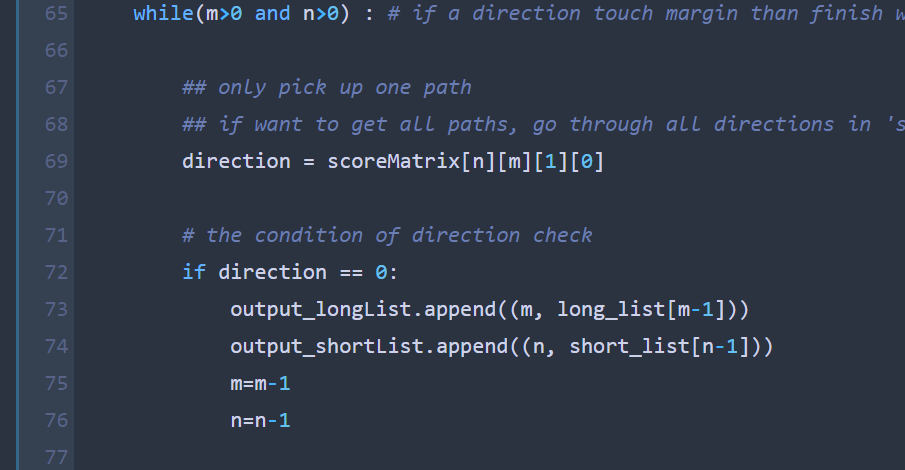
* 把長的序列當成m(橫行)，短的當成n(縱列)

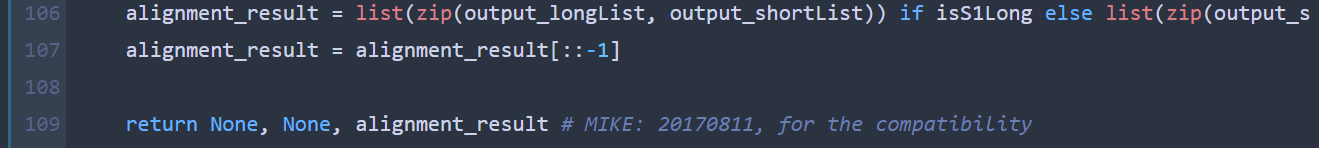


* + 建立空白欄與初始化
  + 迴圈用來計算矩陣每一格的分數(參考wiki)
  + 第27-28行計算gap方向的分數
  + 第30-39行計算斜向(match/mismatch)的分數，並計算到底是match還是mismatch(由34行判斷)
  + 第42行選出最大的分數是哪一個



* 第46~54行判斷最高分是哪一個方向的
* 第56行把方向和分數紀錄進矩陣中



* 第65行的迴圈在走回頭路找最佳common序列出來(參考wiki)
* 第69行讀取每一個該前進的方向
* 第72行以後依照方向指示存取API
* 第106行把用zip把兩個API List壓縮成一個tuple串列
* 第107行反轉串列
* 最後回傳tuple串列結果

## BegC演算法(Behavior Group Clustering)

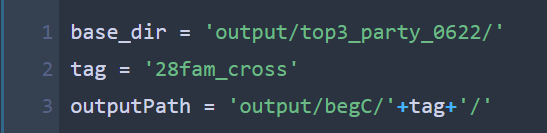
簡介：將跑完RasMMA的家族結果作為輸入，再次放進RasMMA演算法中運算，即可產生cross family behavior group。

* **執行主程式：BegC\_Example.ipynb**

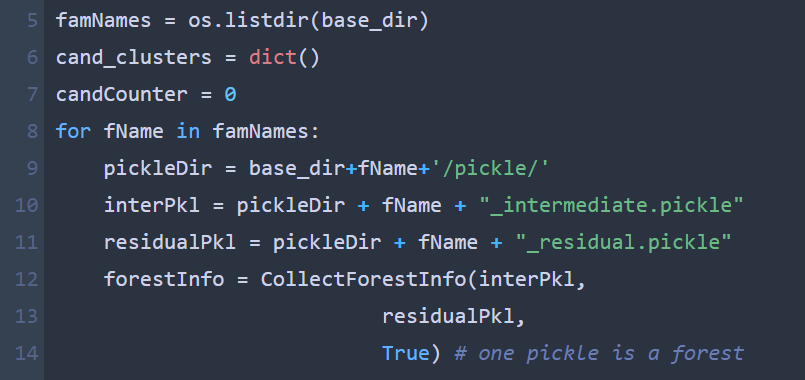
輸入：所有家族的RasMMA結果pickle檔(即所有的behavior forest資訊)

輸出：這些家族Behavior Forest的RasMMA結果，即為behavior group，一樣存成pickle檔放在輸出位置

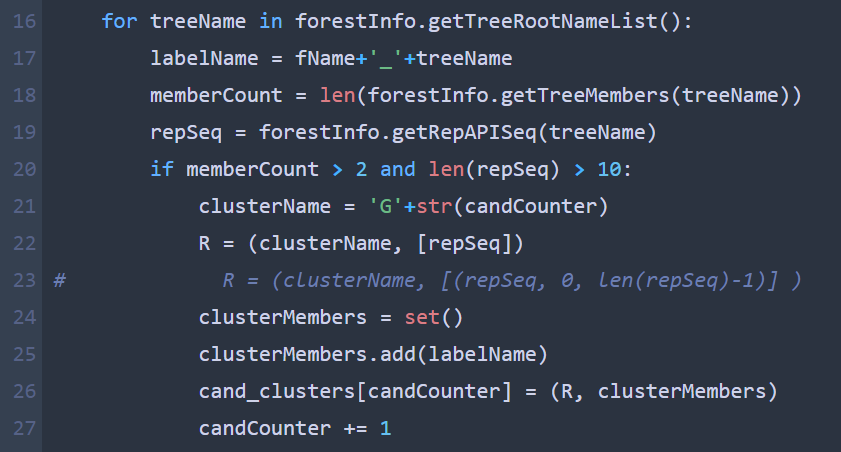
* + 參數設定: 各家族pickle檔根目錄和輸出位置

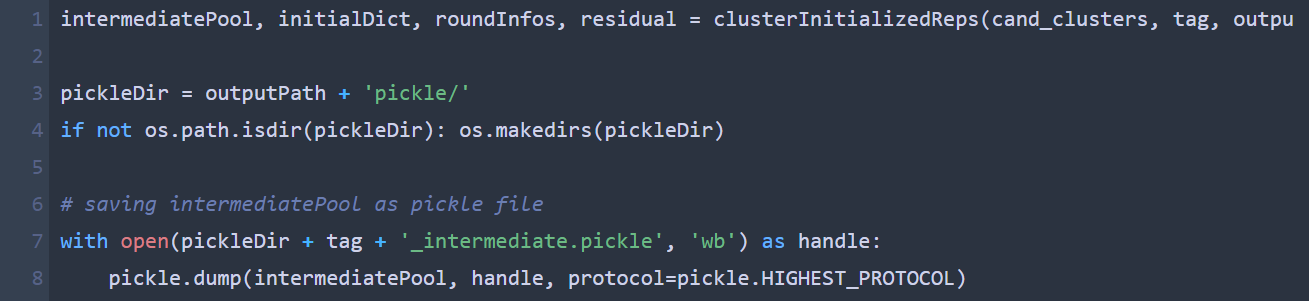


* + 初始化: 將各家族符合資格的behavior tree的repSequence讀取進記憶體中



* + 運用CollectForestInfo這個class去讀取家族pickle檔資料，會獲得forest資訊



* + 第16行開始是把某家族的forest每棵樹資料讀進來
  + 第20行是只挑選出由3個profiles以上和長度大於10的behavior tree留下
  + 第21-26行是模仿RasMMA的initialization，把每棵行為樹當成單一個trace處理
  + 完成上述初始化後呼叫**RasMMA.ipynb**裡頭有個函式–**clusterInitializedReps()** 進行分群，此函式內容和RasMMA演算法裡呼叫的**do\_RasMMAClustering**完全一樣，只是忽略了初始化步驟。
  + 分完群後一樣存成pickle檔，可用讀取家族結果的方式讀取。
  + 注意：此程式只有初始化比較特別，其餘都跟RasMMA一樣，可依照自己需求進行初始化修改，只要格式正確，且最後呼叫**clusterInitializedReps()**即可。

## 讀取Behavior Forest資訊

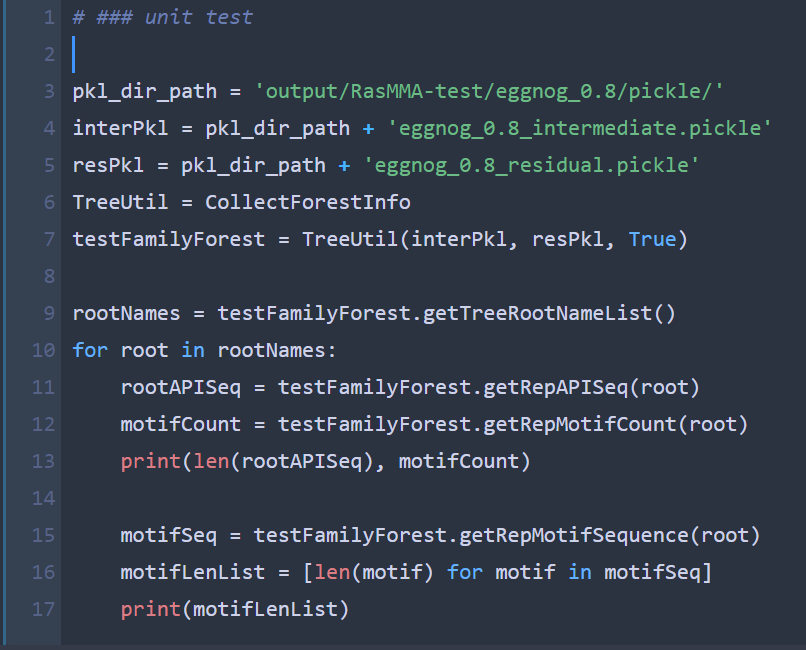
做完RasMMA家族後，除了看報表，可能會想要對Behavior forest或是裡頭的behavior tree做運算，此時可參考函式庫－CollectForestInfo

* **主要程式：CollectForestInfo.ipynb**

裡面有各種函式可以用來讀forest資訊，在最下面有個小小的unit test範例可以看。

輸入：兩個pickle檔的路徑，用於初始化class

接著可以依據任務需求呼叫各種API。



## 家族分類實驗

* **執行主程式：RasMMA\_Classification\_Testing.ipynb**

簡介：將TrainingData先執行RasMMAExample後以RasMMA的結果當成Model去對TestingData做家族分類，邏輯請參考論文。

輸入：1. 所有家族RasMMA結果的Pickle檔路徑. 2. TestingData的trace檔目錄

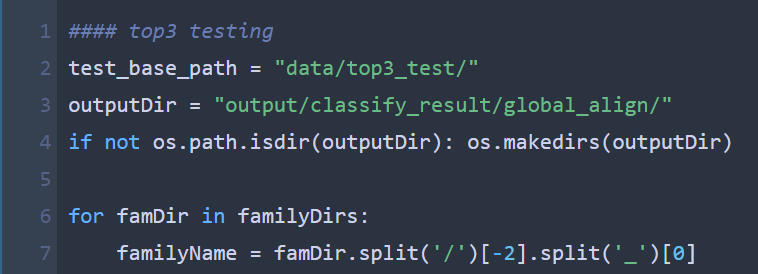
輸出：所有profile對應每個家族所有behavior tree的相似度分數，存成pickle檔

參數設定:

Model路徑



Testing trace的路徑



設定好之後，Run All! 結果會以每個Family的testing data為單位分別存成一個pickle檔(即為Family名稱是Ground Truth的意思)

讀取結果請見RasMMA\_Classification\_MatchReader.ipynb

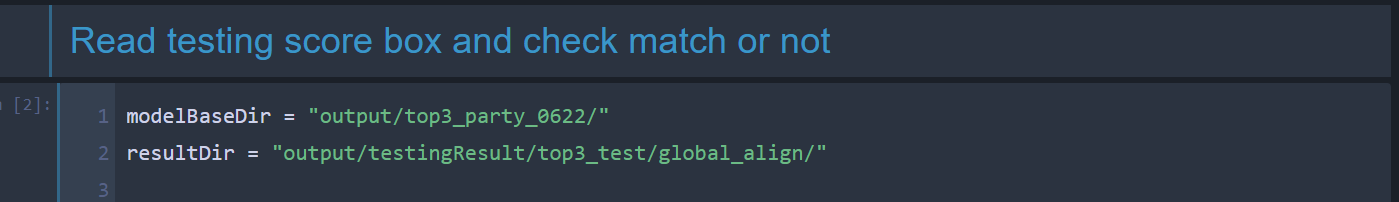
* **執行主程式：RasMMA\_Classification\_MatchReader.ipynb**

簡介：讀取Classification\_Testing.ipynb的結果，並產生match rate (i.e., Accuracy Rate)、Precision、Recall、F1等值。這支程式有點像是多個需求寫再同一份之中，以Markdown做區隔。

* 讀取Testing結果，產生Match Rate

輸入1：model的路徑

輸入2：Testing的結果output路徑



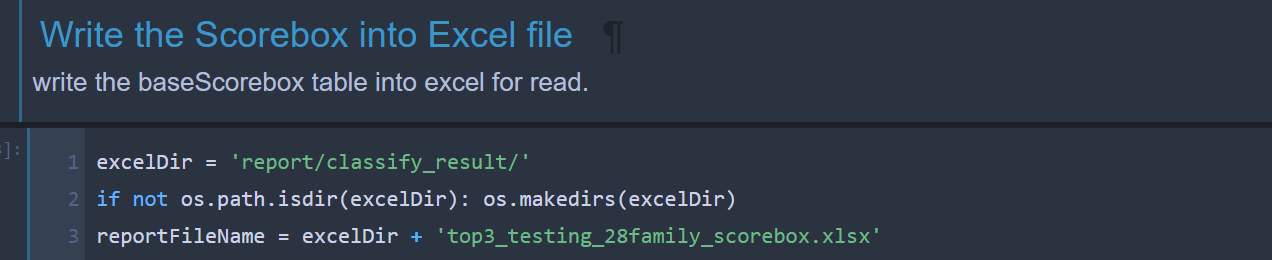
輸出：match rate



* 印出每個testing profile (by process)在各家族的得分，存成Excel

輸入：excel檔名和存放路徑，同時需要先執行前一格cell

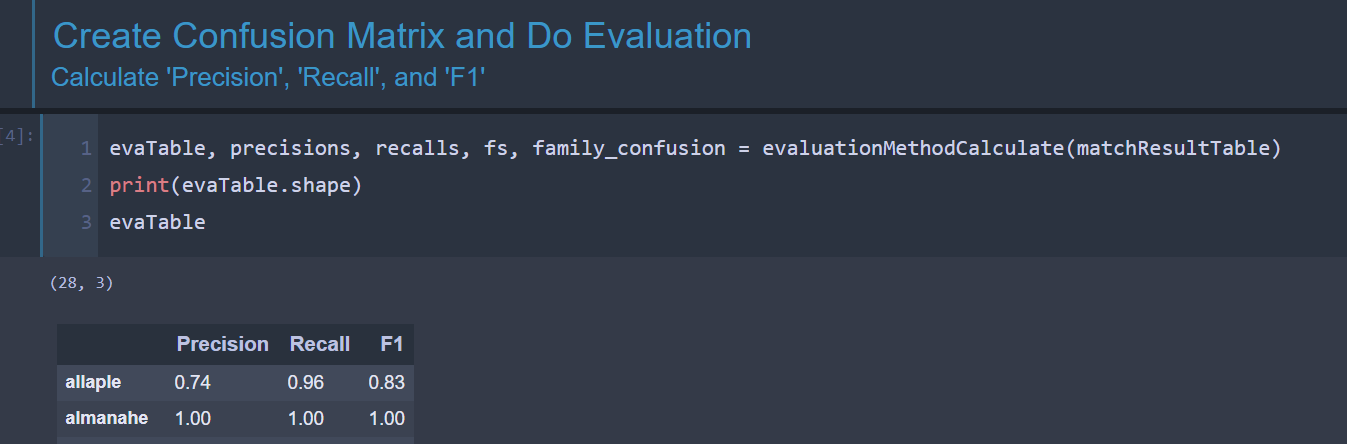
輸出：Excel得分表



* 計算各家族Testing的P/R/F

輸入：先執行前面格子即可

輸出：各種值，如果想印出的話要再自行把evaTable輸出



計算一下總平均

