RAPORT

Elżbieta Jowik

28 kwietnia 2019

1. Wstęp

Klastrowanie jest jedną z najczęściej stosowanych technik analizy eksploracyjnej danych. Ma ono na celu wykrycie w zbiorze obserwacji skupień, czyli **rozłącznych** podzbiorów zbioru, wewnątrz których obserwacje są sobie bliskie względem określonego kryterium, natomiast różne podzbiory są od siebie odległe. Innymi słowy celem analizy skupień jest znalezienie w zbiorze danych ukrytej struktury. My będziemy rozważać analizę skupień w przestrzeni euklidesowej R^p.
Implementacja algorytmu spektralnego wymaga zastosowania następujących podprocedur opisanych w treści zadania:

1. znalezienie M najbliższych sąsiadów wszystkich punktów;
2. stworzenie grafu sąsiedztwa i uspójnienie go;
3. wyznaczenie k-wektorów własnych jego laplasjanu;
4. zastosowanie algorytmu k-średnich w nowej przestrzeni danych.

O ile 3 pierwsze kroki są dość jednoznaczne i nie budzą wielu wątpliwości, o tyle zastosowanie algorytmu kmeans() jest momentem newralgicznym. Ponieważ zastosowaliśmy gotową implementację tego algorytmu, a dobór parametrów w jej wywołaniu w znacznym stopniu wpływa na otrzymywane rezultaty, przyjrzyjmy się, na czym de facto polega algorytm k-średnich.

Zgodnie z dostępną literaturą ***algorytm k-średnich*** *to iteracyjny algorytm inicjalizowany przez narzucenie początkowych położeń k środków* m_k. Mając te środki, możemy im przypisać punkty próby, w taki sposób, że dany punkt zostaje przypisany do tego środka m_k, którego jest najbliższy. W ten sposób wszystkie punkty zostają podzielone na rozłączne skupienia.
W drugim kroku algorytmu, obliczamy nowe środki skupień, równe średnim skupień powstałych w poprzednim kroku. Mając nowe środki, dokonujemy nowego podziału próby na skupienia. Ponieważ mamy nowe środki, przynajmniej niektóre punkty próby mogą zmienić skupienie, do którego przynależą. Potem powtarzamy iteracje tak długo, dopóki żaden punkt nie zostanie przeniesiony z jednego skupienia, do innego.

UWAGI!

1. Ponieważ analiza skupień zaczyna się od wyłonienia k losowo wybranych centroidów, za każdym razem, gdy funkcja jest wywoływana, można uzyskać inny rezultat. To powoduje, że funkcja spectral_clustering() dla tych samych parametrów zwraca ciągi referencyjne, które się od siebie różnią!
2. Ponadto klastrowanie jest wrażliwe na początkowy wybór centroidów. Funkcja kmeans () ma opcję nstart, która próbuje wielu początkowych konfiguracji i *wybiera* tę najkorzystniejszą. Na przykład dodanie nstart = 100 wygeneruje 100 początkowych konfiguracji. Takie podejście jest zalecane, gdyż poprawia ono trafność rezultatów zwracanych przez kmeans() i tym samym spectral_clustering() oraz PRAWDOPODOBNIE minimalizuje efekt opisany powyżej (intuicja).
3. Treść zadania przewiduje testowanie parametrów *tuningujących* poszczególne algorytmy. W przypadku algorytmu *Genie* (funkcja hclust2()), takim parametrem jest thresholdGini. Wpływ jego wartości na trafność zwracanych przypisań zbadamy na wybranych (różnorodnych) zbiorach w końcowej części raportu.

Przejdźmy do porównania różnych algorytmów analizy skupień.
Rozważać będziemy:

1. własną implementację algorytmu spektralnego,
2. wszystkie algorytmy hierarchiczne funkcji **hlust()**,
3. algorytm *Genie* z pakietu **genie**,
4. algorytm zagnieżdżania aglomeracyjnego *Agnes* z pakietu *cluster*.

Kryterium, jakie będziemy stosować do oceny trafności przypisań, zwracanych przez poszczególne algorytmy, tak jak w przypadku testów, będą etykiety referencyjne, przygotowane przez ekspertów.

Do porównania zgodności wynikowych ciągów przypisań, zwracanych przez poszczególne algorytmy, i danych z góry etykiet referencyjnych będziemy stosować:

1. **indeks Fowlkesa-Mallowsa** (dendextend::FM_index())
2. **skorygowany indeks Randa** (mclust::adjustedRandIndex())

Każdy z powyższych indeksów zwraca wartość równą **1**, jeśli dane podziały są równoważne. Im ich wartość jest dalej od 1, tym bardziej są one od siebie różne.

Jedynie samodzielnie implementowany algorytm zależy od liczby najbliższych sąsiadów, dlatego w przypadku indeksów generowanych dla tego algorytmu rozważać będziemy ich średnią arytmetyczną indeksów otrzymanych dla 20 różnych M.

Doboru 20 najbliższych sąsiadów dokonujemy na dwa sposoby:

1. kolejne M to liczby całkowite z przedziału $[1, 20]$ (**)
2. M to 20 całkowitych, równomiernie rozłożonych wartości z przedziału $[1, length(dane) - 1]$. (*)

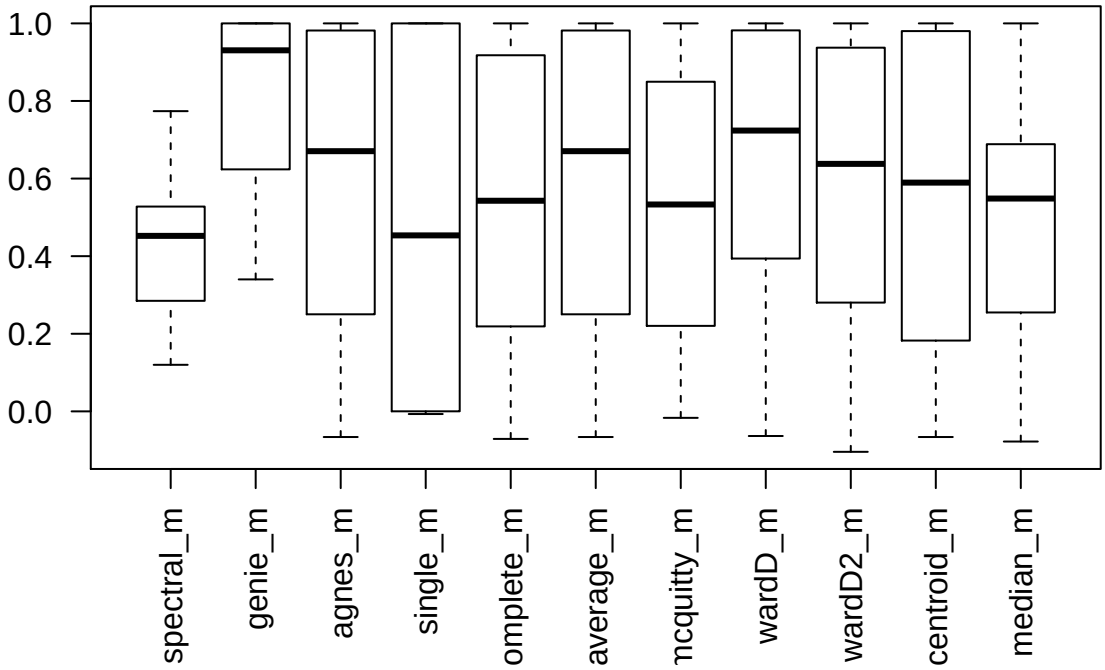
W dalszych rozważaniach wskażę, jak znaczącą różnicę otrzymanych rezultatów powoduje dobór M najbliższych sąsiadów oraz atrybut nstart funkcji kmeans(), a także rozstrzygnę, który z algorytmów działa najlepiej.

Spójrzmy na poniższe boxploty.

2. Wartości skorygowanego indeksu Randa

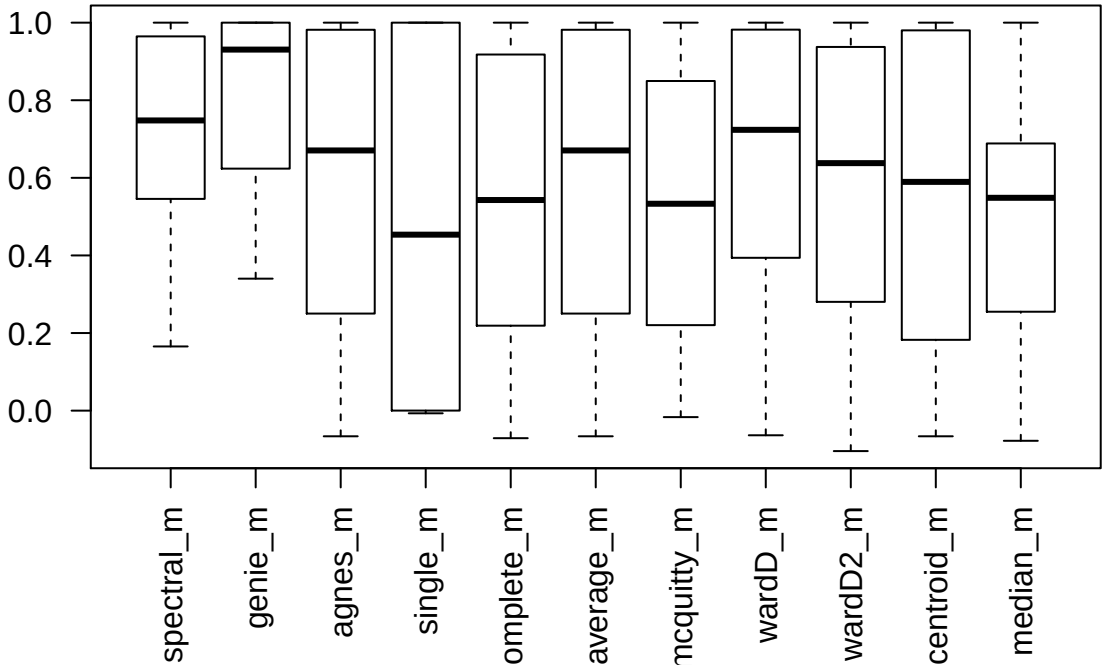
2.1 dla sąsiadów dobranych w sposób (*) i wartości atrybutu nstart = 1 (wartość domyślna).

```
boxplot(output_m, las = 2)
```



2.2 dla sąsiadów dobranych w sposób (**) i wartości atrybutu nstart = 100.

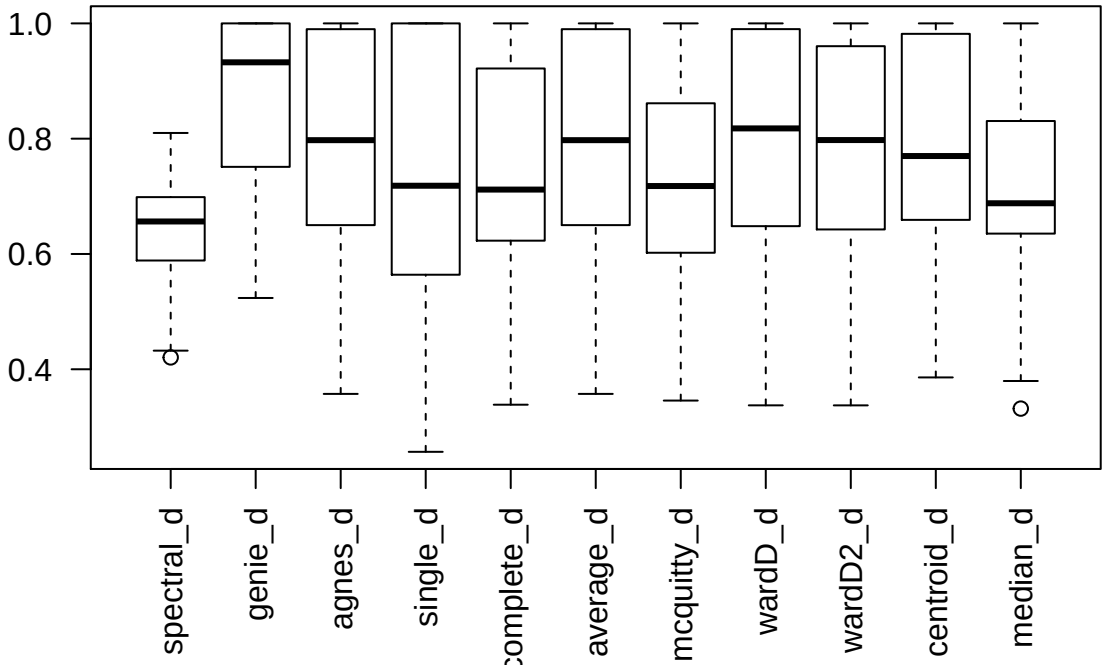
```
boxplot(output_m_2, las = 2)
```



3. Wartości indeksu Fowlkesa-Mallowsa

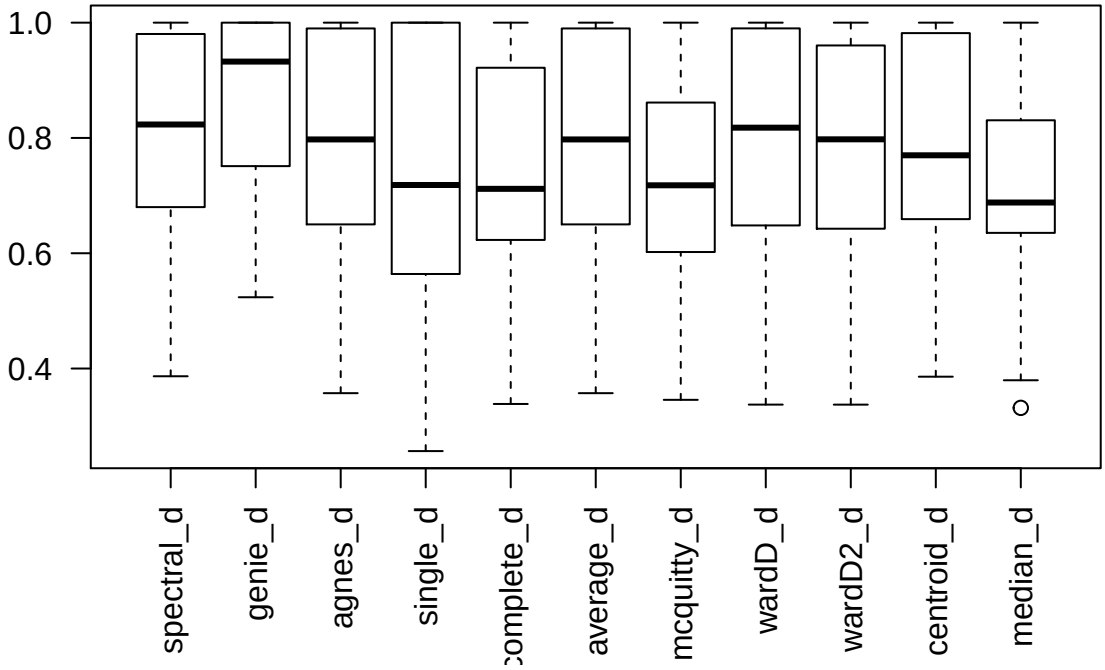
3.1 dla sąsiadów dobranych w sposób (*) i atrybutu nstart = 1 (wartość domyślna).

```
boxplot(output_d, las = 2)
```



3.2 dla sąsiadów dobranych w sposób (*) i atrybutu nstart = 100.

```
boxplot(output_d_2, las = 2)
```



4. Analiza otrzymanych rezultatów

4.1 Badanie wpływu doboru sąsiadów oraz wartości parametru nstart na otrzymane wartości

Zwróćmy uwagę na boxploty opisane jako **spectral_m** na dwóch pierwszych wykresach, oraz **spectral_d** na dwóch ostatnich.
Są to uśrednione, a następnie przedstawione w postaci wykresów skrzynkowych wartości otrzymanych indeksów dla algorytmu spektralnego.

Wówczas, gdy liczby najbliższych sąsiadów są liczbami całkowitymi z przedziału obustronnie domkniętego $[1, 20]$, a wartość parametru nstart wynosi 100, otrzymujemy **zdecydowanie trafniejsze podziały próby na skupienia** niż w drugim przypadku. Wystarczy zwrócić uwagę na położenie pudełek względem osi OY oraz wartości median, żeby zobaczyć, że **różnica sięga nawet 30%**!

Wbrew intuicji (patrz strona 1., UWAGA nr 3) modyfikacja wartości atrybutu nstart z domyślnej 1 na 100 powoduje, że otrzymane wyniki są bardziej zróżnicowane - świadczy o tym rozmiar pudełek, czyli różnica pomiędzy pierwszym i trzecim kwartylem otrzymanych rezultatów oraz długość *wąsów*, czyli wartości odchyleń.
Nie można również zapominać, jak istotny z punktu widzenia działania algorytmu jest dobór liczby najbliższych sąsiadów. Podczas analizy otrzymywanych wartości indeksów, zauważyłam tendencję, która jak się potem okazało, nie była przypadkowa. Okazuje się, że wówczas, gdy liczba sąsiadów jest niewielka, czyli gdy dzielimy próbę na mniej liczne skupienia, algorytm działa dużo sprawniej i z większą dokładnością. Gdy zwiększymy M powyżej określonej wartości zależnej od liczby rekordów w danym wolumenie danych, wartości indeksów dla otrzymanych w rezultacie ciągów referencyjnych gwałtownie spadają.

4.2 Ogólna analiza wyników poszczególnych algorytmów

W przypadku własnej implementacji algorytmu, w gruncie rzeczy interesuje nas to, czy jesteśmy w stanie tak dobrać parametry, aby uzyskane wyniki były zadowalające.
Z powyższych rozważań wynika, że dzięki odpowiedniej parametryzacji funkcji spectral_clustering() jesteśmy w stanie uzyskać zróżnicowane rezultaty.
Na potrzeby raportu, będziemy analizować te korzystniejsze.
Zatem do dalszej analizy będziemy wykorzystywać wykresy z podrozdziałów 2.2 i 3.2 .
Przy analizie algorytmów będziemy zwracać szczególną uwagę na wartości odchyleń oraz różnicę pomiędzy pierwszym a trzecim kwartylem otrzymanych wyników, gdyż to one świadczą o tym, czy algorytm działa w sposób zrównoważony i czy tym samym może być uznany za wiarygodny.

Już na pierwszy rzut oka widać, że najtrafniejszy podział próby na skupienia otrzymujemy w przypadku zastosowania algorytmu *Genie*. Niewielka różnica między kwartylami świadczy o tym, że w przypadku tego algorytmu nie mamy do czynienia z sytuacją, że w zależności od próby jeden z otrzymanych podziałów będzie ewidentnie dobry, a drugi ewidentnie zły. Algorytm Genie nie tylko gwarantuje konsekwencję zwracanych wyników, ale również zachowuje możliwość szybkiego działania.

To powoduje, że algorytm *Genie* możemy uznać za zdecydowanie najlepszy spośród rozważanych.
Dlatego od tej chwili to ten właśnie algorytm będzie stanowił punkt odniesienia w analizie pozostałych.

Własna implementacja algorytmu, omówiona szczegółowo we wstępie, pod pewnymi względami nie ustępuje algorytmowi *Genie*, co widać na wykresach, jednak różnica odchyleń w jego przypadku sięga nawet 0.8!
To oraz zdecydowanie wysoki koszt obliczeniowy, poddaje pod poważną wątpliwość jakość *spectral_clustering()*.

Algorytm **AGlomerative NESting** należy do metod aglomeracyjnych, co oznacza, że każda obserwacja początkowo traktowana jest jak osobny klaster. W kolejnych etapach grupy podobne do siebie łączone są w coraz większe grupy tak długo, aż powstanie klaster obejmujący wszystkie elementy.
(http://www.statystyka.az.pl/analiza-skupien.php).
Zwróćmy uwagę, jak odmienne od siebie są boxploty tego algorytmu, otrzymane dla dwóch indeksów. Nawet uwzględniając to, że wewnątrz indeksów zawarte są rózne formuły matematyczne, odmienność otrzymanych

rezultatów jest zaskakująca.
Wysoki koszt obliczeniowy oraz fakt, że dla najgorszych podziałów prób, wartość skorygowanego indeksu Randa oscyluje wokół 0, w mojej ocenie poddają pod wątpliwość rzetelność uzyskanych podziałów.

Skupmy się teraz na algorytmach hierarchicznych funkcji hclust(). Funkcja ta wykonuje hierarchiczną analizę skupień dla klastrowanych n obiektów.
Początkowo każdy obiekt jest przypisany do własnego klastra, a następnie algorytm przechodzi iteracyjnie, na każdym etapie łącząc dwa najbardziej podobne klastry, tak długo, aż pojawi się tylko jeden klaster. Na każdym etapie oblicza się odległości między klastrami, zgodnie z konkretną stosowaną metodą grupowania, określoną jako parametr metod. Okazuje się, że mimo iż za każdym razem wywołujemy tę samą funkcję, dobór metody powoduje, że otrzymujemy zróżnicowane wyniki.
Najwięcej wątpliwości, ze względu na brak konsekwencji skuteczności, budzi metoda single, czyli metoda pojedynczego powiązania (która jest ściśle związana z minimalnym drzewem rozpinającym) przyjmujująca strategię klastrowania „znajomi przyjaciół". Wadą tego podejścia jest podatność na obserwacje odstające oraz ewidentna skłonność do generowania niezrównoważonych podziałów.

Najkorzystniejszą z rozważanych metod funkcji hclust() jest ward.D, gdyż otrzymane dzięki niej rezultaty są najbardziej zbliżone do wyników *Genie*.

OGÓLNY WNIOSEK:

Biorąc pod uwagę ostateczną trafność zwracanych podziałów prób, zrównoważenie wyników, czyli pewną ich konsekwencję oraz koszt obliczeniowy najlepszym algorytmem klastrującym jest zdecydowanie *Genie*.

5. Analiza algorytmu *Genie* w zależności od parametru thresholdGini

Metoda Genie wymusza wcześniejsze łączenie skupień o minimalnej liczności, jeśli stopień nierówności rozkładu przekroczy pewien ustalony próg, określony przez rozważany parametr.
W celu dokonania analizy wpływu thresholdGini na otrzymane rezultaty, ponownie wywołałam algorytm *Genie* na wszystkich zbiorach, tym razem dla różnych wartości tego argumentu. Jest on liczbą rzeczywistą z przedziału $[0, 1]$ o domyślnej wartości 1. Wartości, dla których testujemy algorytm należą do zbioru:
$\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$.
Poniżej mamy nietrywialne zestawienie uśrednionych rezultatów.

5.1 Zestawienie wartości w postaci tabeli

```
knitr::kable(dane, format = "markdown")
```

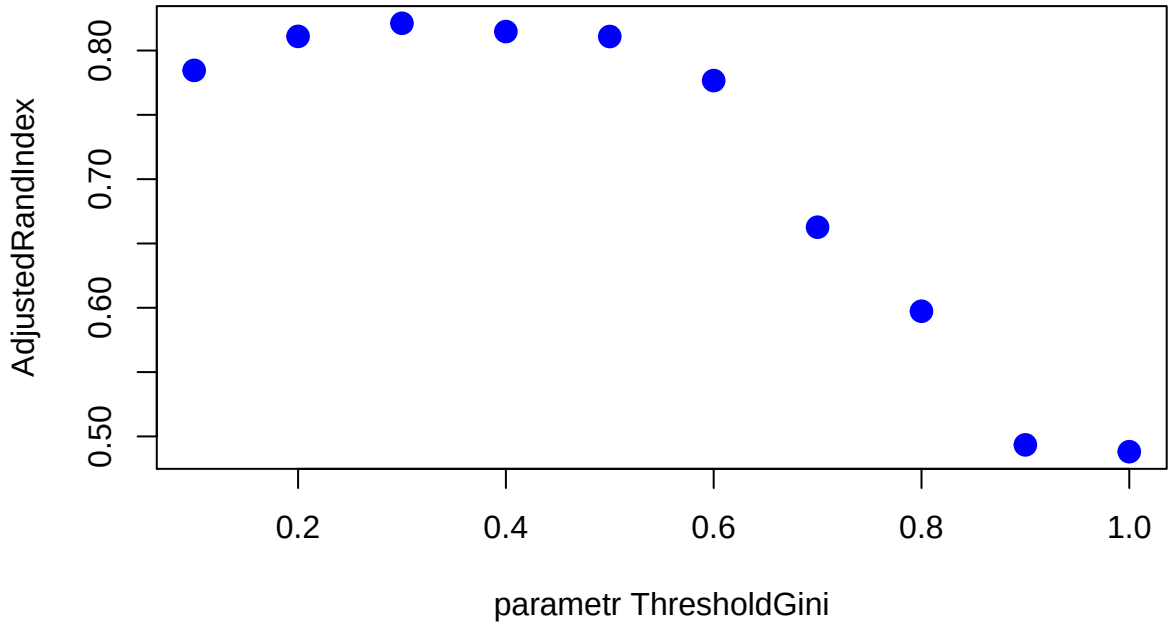
	AdjustedRandIndex	FM_Index
0.1	0.7845544	0.8599959
0.2	0.8109831	0.8736343
0.3	0.8211951	0.8757212
0.4	0.8147423	0.8723609
0.5	0.8108949	0.8692314
0.6	0.7766288	0.8524319
0.7	0.6626578	0.8077854
0.8	0.5973169	0.7788314
0.9	0.4934309	0.7416246
1	0.4881138	0.7412543

Na podstawie otrzymanych wyników widzimy, że średnio algorytm działa najlepiej, wówczas, gdy parametr przyjmuje wartości z przedziału: $[0.2, 0.5]$ (patrz wykresy na następnej stronie). Oznacza to, że uruchamiając algorytm dla takich wartości, a nie dla domyślnej 1, jak to zrobiliśmy w przypadku generowania plików csv, mogliśmy otrzymać jeszcze lepsze rezultaty.
Ponieważ jednak algorytm *Genie* i tak jest najlepszy spośród rozpatrywanych, ponowne generowanie plików csv nie jest potrzebne, gdyż o ogólnym zestawieniu badanych algorytmów nie spowodowałoby żadnych zmian. Warto jednak pamiętać, że mimo iż algorytm działa bardzo dobrze dla 1, można go dodatkowo sparametryzować tak, aby działał jeszcze lepiej.

5.2 Graficzna interpretacja tabeli

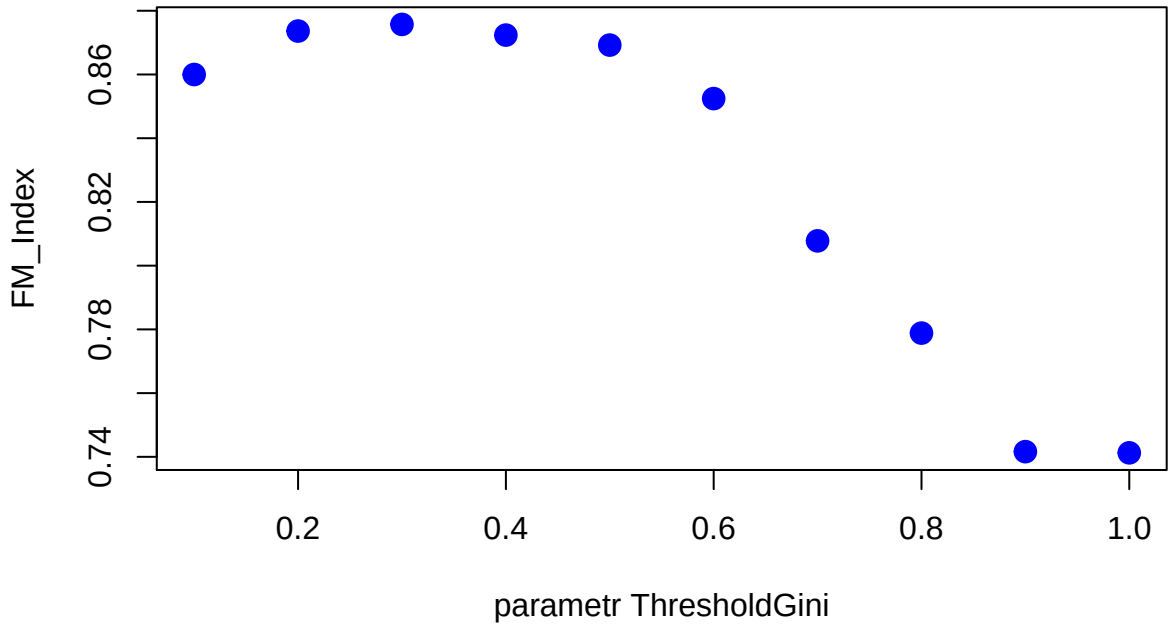
5.2.1 Skorygowany indeks Randa

```
plot3D::scatter2D(rownames(dane), dane$AdjustedRandIndex, cex = 1.5,
                  pch = 19, col = 4, ylab = "AdjustedRandIndex", xlab = "parametr ThresholdGini")
```



5.2.2 Indeks Fowlkesa-Mallowsa

```
plot3D::scatter2D(rownames(dane), dane$FM_Index, cex = 1.5, pch = 19, col = 4,
                  ylab = "FM_Index", xlab = "parametr ThresholdGini")
```



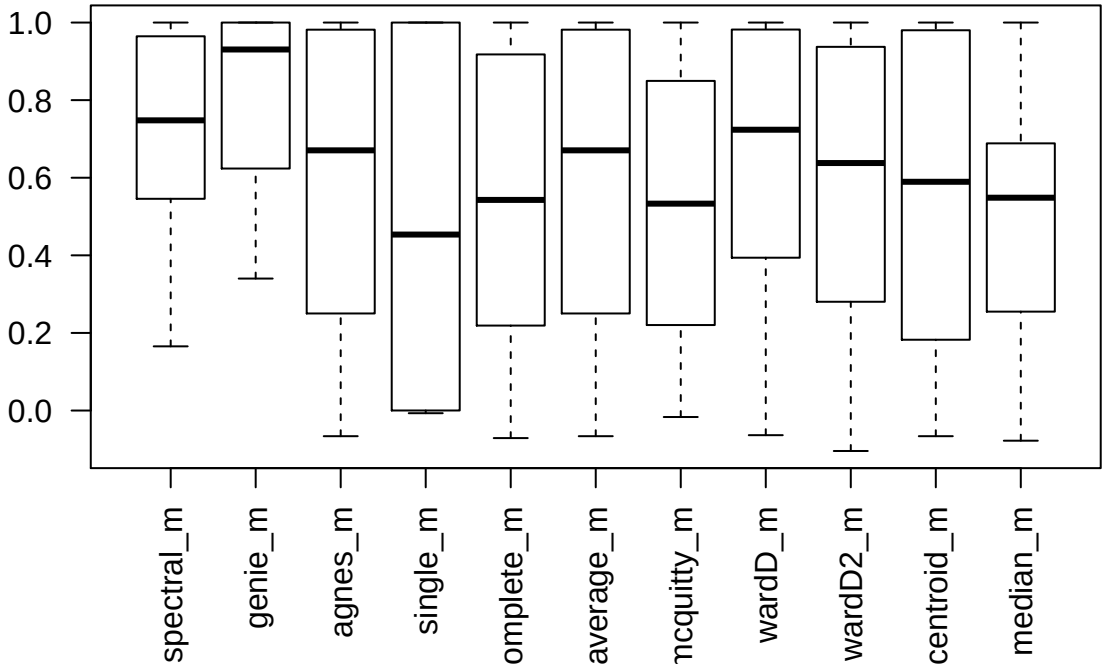
6. Wpływ standaryzacji zmiennych na jakość analizy skupień.

Na potrzeby badania wpływu standaryzacji zmiennych na jakość analizy skupień rozważmy jeszcze raz rezultaty z podrozdziałów 2.2 i 3.2 (przed standaryzacją) oraz te wygenerowane dla danych po standaryzacji.

6.1 Skorygowany indeks Randa

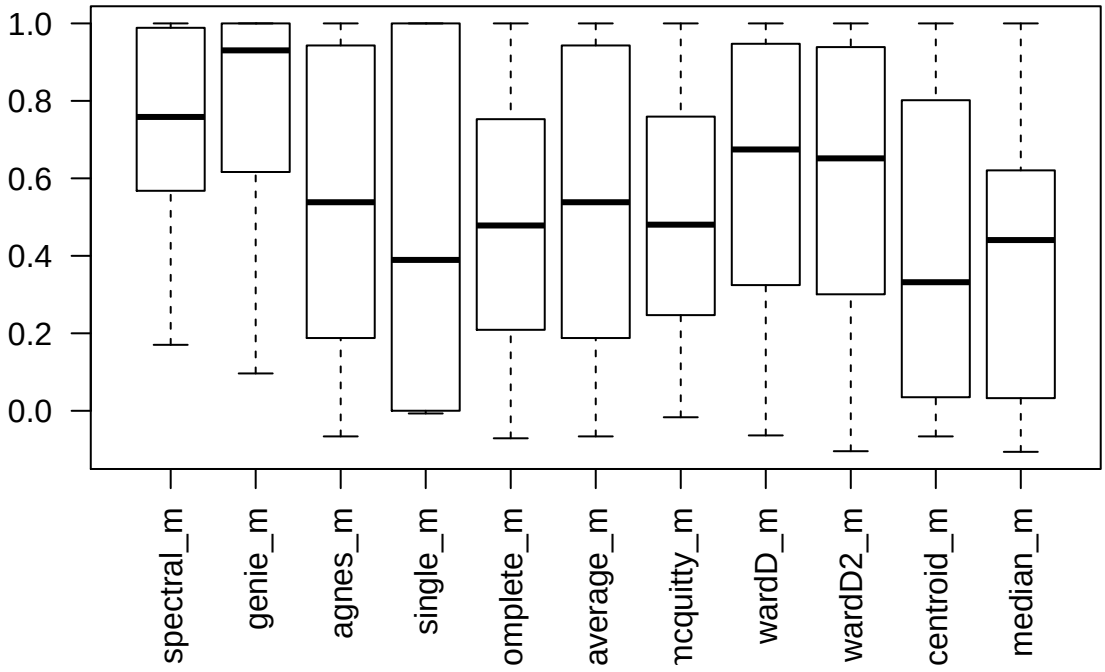
6.1.1 PRZED STANDARYZACJĄ

```
boxplot(output_m_2, las = 2)
```



6.1.2 PO STANDARYZACJI

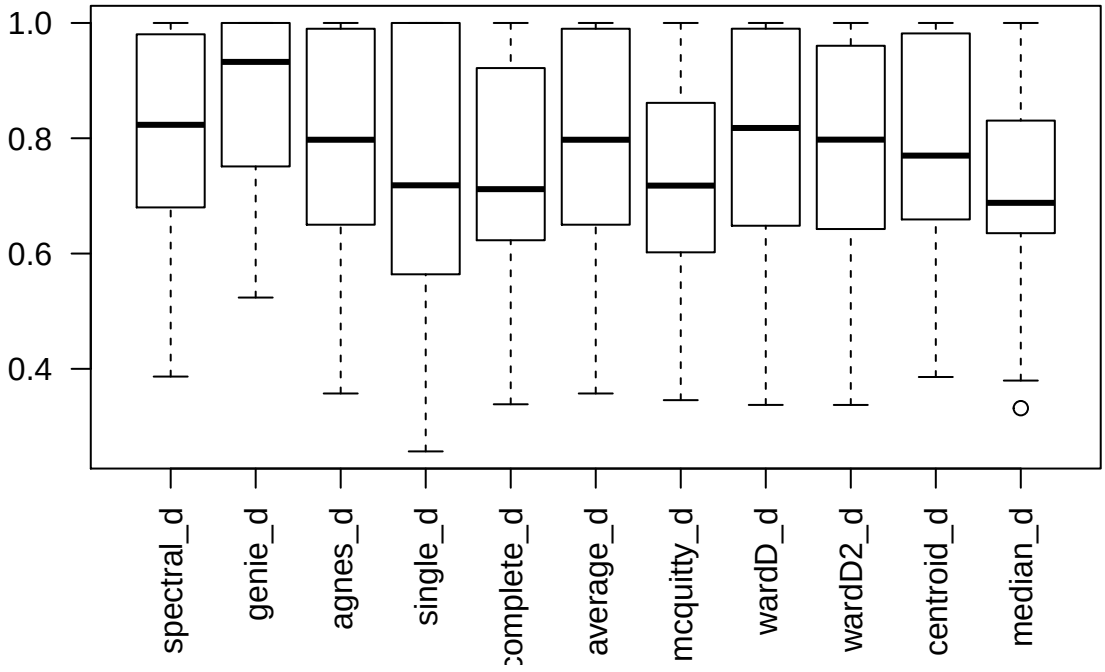
```
boxplot(output_m_s, las = 2)
```



6.2 Indeks Fowlkesa-Mallowsa

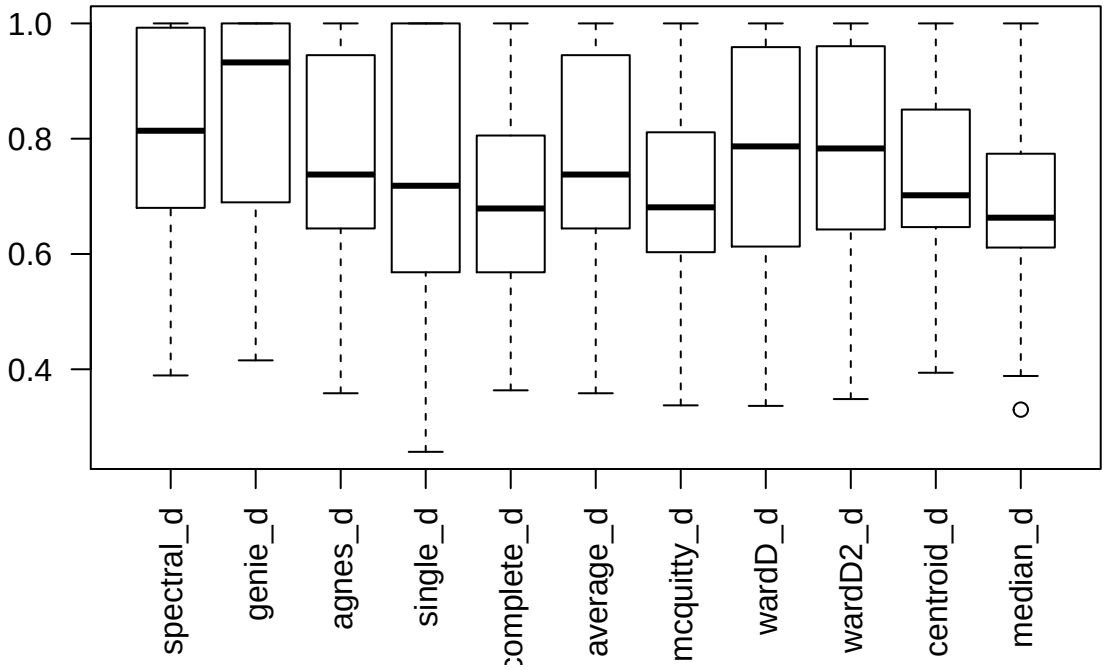
6.2.1 PRZED STANDARYZACJĄ

```
boxplot(output_d_2, las = 2)
```



6.2.2 PO STANDARYZACJI

```
boxplot(output_d_s, las = 2)
```



WNIOSKI:

W przypadku własnej implementacji algorytmu, algorytmu *Agnes* oraz większości algorytmów hierarchicznych funkcji *hclust()* standaryzacja danych wejściowych nie spowodowała większych zmian. Trzeci kwartyl nieznaczenie przesunął się w stronę jedynki, co oznacza, że niewielka część wyników uległa poprawie. Zmiany te jednak nie były gwałtowne, gdyż wartości mediany, odchyleń oraz pierwszego kwartyla pozostały identyczne. Jedynie w przypadku metod *complete*, *centroid* i *median* wyniki przed i po standaryzacji różnią się w znacznym stopniu. W szczególności wartości kwartyli uległy zdecydowanemu pogorszeniu - w przypadku metod *centroid* i *median* obniżyły się watości obydwu kwartyli oraz median, a w przypadku metody *complete* zauważalnemu obniżeniu uległ jedynie trzeci kwartyl. Dla wszystkich dotychczas algorytmów wartości odchyleń pozostały niezmienione.

Charakter zmian, jakie zaszły dla rezultatów zwracanych przez *Genie*, są zupełnie odmienne. Przede wszystkim dotyczą one ich zróżnicowania. Wartości kwartyli oraz mediany zostały niezmienione. Jedynie zmiany wartości odchyleń są zauważalne. Wydłużony dolny wąs świadczy o osłabionej konsekwencji zwracanych wyników, co zdecydowanie działa na niekorzyść rozważanego algorytmu.