1 Clustering

Clustering è suddividere un dataset di un certo numero di elementi in sottoparti chiamate cluster sulla base della loro affinità.

Il problema è che diversi algoritmi di clustering non sono in grado di fornire una metrica oggettiva per determinare quanto il clustering che hanno indotto sia effettivamente rappresentativo della struttura del dataset, o se sia semplicemente un raggruppamento arbitrario. Inoltre, diversi algoritmi come K-Means richiedono il numero di cluster come iperparametro, rendendo il discorso ancora più complesso, perché in genere nel clustering non supervisionato non vi è a disposizione alcuna "ground truth".

La metrica Silhouette si propone di rispondere alle seguenti domande:

- Il clustering è di buona qualità? In altre parole, gli elementi di uno stesso cluster sono fra di loro "vicini" ed al contempo "lontani" dagli elementi di tutti gli altri cluster?
- Quali sono gli elementi ben classificati, ovvero quelli che probabilmente si trovano nel cluster "giusto"?
- Quali sono gli elementi che è difficile stabilire con certezza in quali cluster vadano collocati, ovvero quelli che stanno "nel mezzo" fra più cluster?
- Il numero di cluster scelto è effettivamente rappresentativo del dataset o è 'artificioso'?

2 Silhouette

2.1 Setup

Si supponga di avere a disposizione un dataset di dimensione $NtimesM$, dove N indica il numero degli elementi e M è il numero di attributi. Per comodità, si assuma che gli attributi siano tutti dati numerici (altezze, lunghezze, capacità, ecc...). Per ogni elemento, tutti i valori di ciascun attributo sono noti.

A partire da tale dataset è possibile costruire quella che viene chiamata **matrice delle distanze**. Tale matrice ha dimensione $N \times N$ e, in ciascuna cella (i, j), è presente un valore indicato con $d(i, j)$ che rappresenta il grado di "dissomiglianza" fra l'elemento i e l'elemento j del dataset.

Tale grado di dissomiglianza è calcolato mediante una **funzione di distanza**, usando come input i valori degli attributi di i e di j. Un esempio di funzione di distanza è la **distanza Euclidea**, definita come segue:

$$d(i,j) = \sqrt{\sum_{m=1}^{M} (f_{i,m} - f_{j,m})^2} = \sqrt{(f_{i,1} - f_{j,1})^2 + \cdots + (f_{i,M} - f_{j,M})^2} \quad (1)$$

Dove $f_{i,m}$ e $f_{j,m}$ indicano il valore del m-esimo attributo per, rispettivamente, l'i-esimo ed il j-esimo elemento del dataset.

Altri esempi di distanze sono la **distanza di Manhattan** e la **distanza di Minkowski** (dal punto di vista di Silhouette, quale funzione di distanza venga usata è irrilevante).

"1"	"2"	"3"	"4"	"5"	"6"
0	0.54	0.51	0.65	0.14	0.62
0.54	0	0.3	0.33	0.61	1.09
0.51	0.3	0	0.24	0.51	1.09
0.65	0.33	0.24	0	0.65	1.17
0.14	0.61	0.51	0.65	0	0.62
0.62	1.09	1.09	1.17	0.62	0

Table 1: Matrice delle distanze per il dataset `iris`. Per questioni di spazio sono presenti solamente i primi 6 elementi.

Una volta nota la matrice delle distanze, si supponga di applicare un algoritmo di clustering (K-Means, ad esempio) per suddividere il dataset in un certo numero di cluster, sia questo K. Per ciascun cluster, è interamente noto sia il numero di suoi elementi, sia a quale cluster ciascun elemento del dataset è stato assegnato.

2.2 Costruzione di $a(i)$ e $b(i)$

Per poter calcolare il coefficiente di Silhouette, è prima necessario introdurre due quantità per ciascun elemento i del dataset, indicate rispettivamente con $a(i)$ e $b(i)$.

Preso un elemento i del dataset, sia A il cluster in cui l'algoritmo lo ha riposto. Ammesso che A contenga altri elementi all'infuori di i, è possibile definire $a(i)$ come distanza media fra i e tutti gli elementi di A escluso i stesso:

$$a(i) = \frac{1}{|A| - 1} \sum_{j \in \{A - \{i\}\}} d(i, j) \tag{2}$$

Tale valore misura quanto un cluster è *coeso*, nel senso che se tale valore è piccolo per tutti gli elementi del cluster, questi si trovano fra loro vicini. Per tale motivo, $a(i)$ viene anche chiamata **distanza intra-cluster**.

Dopodiché, in maniera simile, per un cluster C diverso da A è possibile definire $D(i, C)$ come la distanza media fra i (che appartiene ad A) e gli elementi di C:

$$D(i, C) = \frac{1}{|C|} \sum_{j \in C} d(i, j)$$

Tale valore misura quanto un cluster è *separato*, nel senso che se tale valore è grande per tutti gli elementi del cluster a cui i appartiene, il cluster nel suo

complesso si trova molto distante da tutti gli altri. Per tale motivo $D(i, C)$ viene anche chiamata **distanza inter-cluster**.

Assumendo che il numero di cluster sia più di uno, per uno stesso elemento i è possibile calcolare la distanza inter-cluster per ogni possibile cluster C distinto da A. Fra questi $K - 1$ cluster, è di particolare interesse il cluster che ha il più piccolo valore di distanza inter-cluster per i, chiamato **neighboring cluster**. Questo perché tale cluster è quello che, se il cluster A non esistesse, sarebbe la miglior scelta per catalogare i, dato che è quello i cui elementi sono i più vicini ad i.

Se il neighboring cluster per i è il cluster C', la distanza inter-cluster $D(i, C')$ viene indicata con $b(i)$:

$$b(i) = min_{C \neq A} D(i, C) \tag{3}$$

2.3 Calcolo di $s(i)$

Una volta calcolato $a(i)$ e $b(i)$ per l'elemento i del dataset, è possibile assegnarvi un valore di Silhouette $s(i)$, così calcolato:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{4}$$

Se l'elemento i si trova in un cluster che contiene solamente sé stesso, per convenzione il valore $s(i)$ viene posto a 0 (è una scelta arbitraria, ma è anche quella più neutra).

È facile verificare che, per qualsiasi elemento i:

$$-1 \leq s(i) \leq 1$$

Si assuma infatti che $b(i) \geq a(i)$. L'espressione diventa:

$$s(i) = \frac{b(i) - a(i)}{b(i)} = \frac{b(i)}{b(i)} - \frac{a(i)}{b(i)} = -\frac{a(i)}{b(i)}$$

Avendo assunto che $b(i)$ sia maggiore di $a(i)$, tale frazione è una frazione propria, e pertanto il suo valore è racchiuso nell'intervallo $[-1, 0]$.

Si assuma invece $a(i) > b(i)$. L'espressione diventa:

$$s(i) = \frac{b(i) - a(i)}{a(i)} = \frac{b(i)}{a(i)} - \frac{a(i)}{a(i)} = \frac{b(i)}{a(i)}$$

Avendo assunto che $a(i)$ sia maggiore di $b(i)$, tale frazione è una frazione propria, e pertanto il suo valore è racchiuso nell'intervallo $[0, 1]$.

Per farsi una migliore idea del significato di $s(i)$, può essere utile considerare alcune situazioni estreme.

Quando $s(i)$ è approssimativamente 1 si ha che $b(i)$ è molto più grande di $a(i)$, e quindi la distanza fra i ed i membri del cluster a cui appartiene è molto più piccola della distanza fra i ed i membri degli altri cluster. Questo significa

0	1	2
1	2	3
3	1	0.85
3	1	0.82
3	1	0.83
3	1	0.81
3	1	0.85
3	1	0.75
3	1	0.82
3	1	0.85
3	1	0.75
3	1	0.83

0	1	2
1	2	3
3	1	0.85
1	2	0.85
1	2	0.38
2	1	0.85
1	2	0.59
1	2	0.37
1	2	0.59
1	2	0.28
1	3	0.27
1	2	0.34
1	2	0.58

0	1	2
1	2	3
1	2	0.63
2	1	0.5
1	2	0.23
2	1	0.61
2	1	0.36
2	1	0.56
2	1	0.54
1	2	0.47
2	1	0.56
2	1	0.44
2	1	0.56

Table 2: Valori di $s(i)$, cluster e neighboring cluster per i primi 10 elementi dei tre cluster. Si noti come i valori di $s(i)$ del primo cluster siano più alti ed il neighboring cluster sia sempre lo stesso, mentre gli altri due cluster hanno valori più variegati.

che la scelta di aver posto i in quel cluster è una buona scelta, perché persino la "seconda scelta" è di netto inferiore alla prima.

Quando $s(i)$ è approssimativamente 0 si ha che $b(i)$ e $a(i)$ hanno lo stesso ordine di grandezza, e quindi la distanza fra i ed i membri del cluster a cui appartiene è comparabile a quella fra i ed i membri del suo neighboring cluster. Questo significa che la scelta di aver posto i in quel cluster è inconclusiva, nel senso che se fosse stato invece scelto il neighboring cluster si avrebbe avuto sostanzialmente lo stesso risultato.

Quando $s(i)$ è approssimativamente -1 significa che $a(i)$ è molto più grande di $b(i)$, e quindi la distanza fra i ed i membri del cluster a cui appartiene è molto più grande della distanza fra i ed i membri degli altri cluster. Questo significa che la scelta di aver posto i in quel cluster è discutibile, perché vi sono cluster con cui i ha più in comune rispetto a quello in cui si trova.

2.4 Silhouette Plot

I valori $s(i)$ non sono, di per loro, particolarmente informativi. È però possibile costruire un Silhouette plot di ciascun cluster come un bar chart dove ciascuna colonna i-esima ha altezza proporzionale a $s(i)$, ordinato dalla colonna più alta a quella più bassa. Tale grafico può essere arricchito riportando informazioni ulteriori per ciascun elemento come, ad esempio: in quale cluster si trova, qual'è il suo neighboring cluster, il suo valore di $s(i)$.

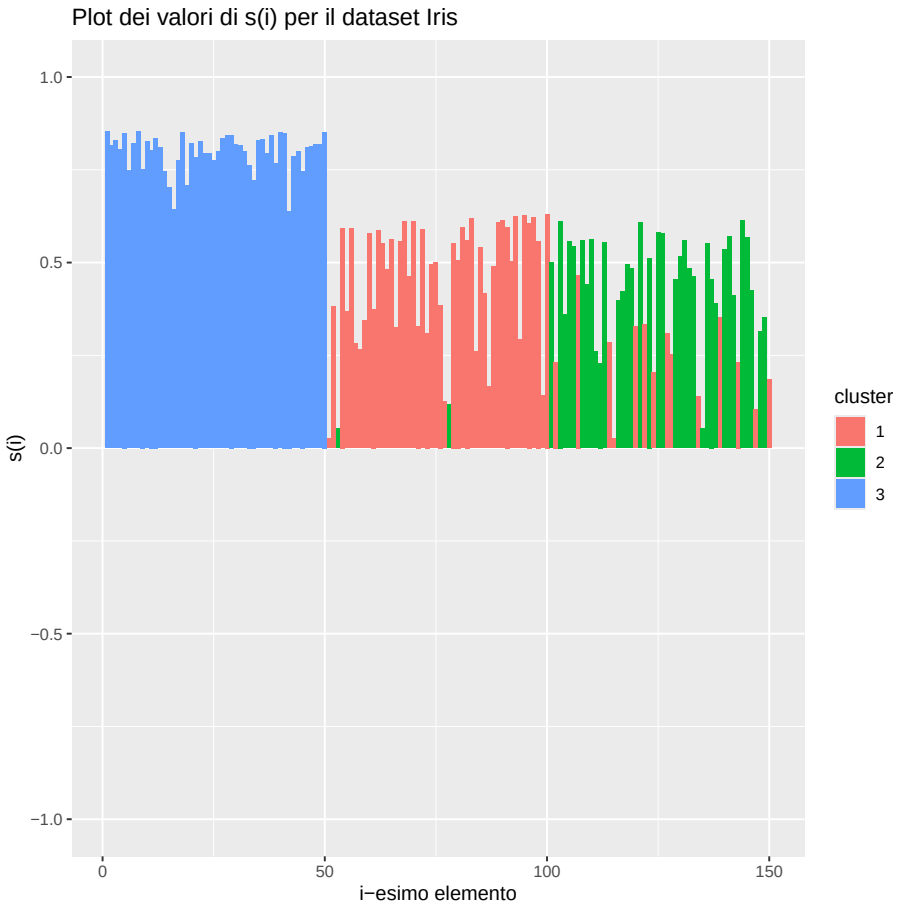

Figure 1: Silhouette plot per il dataset `iris`.

2.5 Come interpretare Silhouette

Il vantaggio di Silhouette è che non dipende da quale algoritmo è stato usato per effettuare il clustering. Per tale motivo, può essere usato per valutare "a posteriori" il risultato dell'algoritmo, provando a modificare il valore degli iperparametri per valutare quale combinazione di iperparametri restituisce il risultato più coerente. Questo riesce particolarmente bene negli algoritmi in cui il numero di cluster figura fra gli iperparametri, come K-Means.

Per esempio, si supponga che un dataset abbia effettivamente delle aree molto dense separate da aree ampie vuote. Operando un clustering in cui il numero di cluster è più basso del numero "naturale" di cluster, delle aree molto distanti tra loro vengono inglobate in un cluster unico nonostante vi siano considerevoli distanze nel mezzo. Silhouette può evidenziare questa situazione perché il valore di $a(i)$ tende ad essere molto alto, essendo i membri del dataset

molto distanti dai loro centroidi.

Si supponga invece di operare un clustering in cui il numero di cluster è più alto del numero "naturale" di cluster. In tale situazione, anche aree dense vengono spezzate in cluster diversi. Silhouette può evidenziare questa situazione perché il valore di $b(i)$ tende ad essere molto basso, dato che elementi molto vicini vengono separati forzosamente.

In generale, l'operato di un algoritmo di clustering può considerarsi ottimale se il valore si $s(i)$ tende ad essere molto alto per tutti gli elementi del dataset. A tale scopo, è possibile calcolare la Silhouette media per un certo cluster C come la media tutti gli $s(i)$ per ciascun elemento i che appartiene a C. Se tale valore medio è alto, il cluster nel suo complesso è ben formato.

Se si ha invece interesse a sapere qual'è il numero ottimale di cluster, è possibile considerare la Silhouette media complessiva come la media di tutti gli $s(i)$ per ogni elemento dell'intero dataset. L'idea è quella di testare diverse combinazioni di iperparametri dell'algoritmo di clustering e scegliere la combinazione che restituisce la Silhouette media complessiva più grande: tale combinazione sarà quella che restituisce il clustering che meglio interpreta il dataset.

Si noti come un valore della Silhouette media complessiva pari a 0 non significa necessariamente che il clustering non sia andato a buon fine. Può infatti anche indicare che effettivamente il dataset non ha alcuna struttura di clustering naturale, e che quindi l'algoritmo di clustering ha comunque fornito un risultato corretto, dato che effettivamente qualsiasi risultato vale l'altro.

2.6 Sviluppi futuri

A partire dall'idea di base di Silhouette, è possibile costruirne infinite varianti. Gli autori citano:

- Per determinare il miglior numero di cluster non è strettamente necessario utilizzare la Silhouette media complessiva. Sarebbe infatti possibile anche combinare gli $s(i)$ in modo diverso;
- La Silhouette media complessiva può essere essa stessa usata come funzione obiettivo da massimizzare direttamente all'interno di un algoritmo di clustering, anziché effettuare una valutazione a posteriori;
- Se l'algoritmo di clustering si basa sulla costruzione di centroidi o sulla elezione di rappresentanti, si potrebbe usare la distanza da tali centroidi o rappresentanti come grado di dissomiglianza anziché calcolare $a(i)$ o $D(i, C)$ per ogni i-esimo elemento, semplificando il procedimento. Naturalmente, questo approccio renderebbe Silhouette dipendente dal tipo di algoritmo usato.

Tutte e tre le varianti sono toccate da articoli citati.

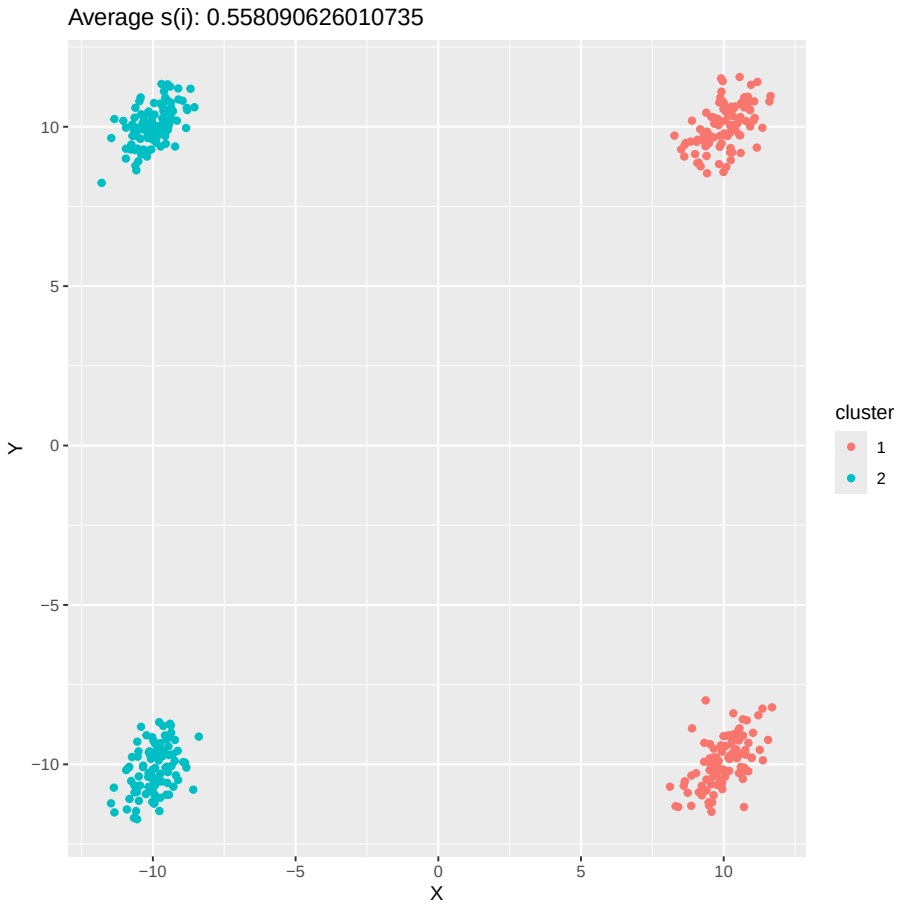

Figure 2: Clustering con K-Means usando K = 2 per un dataset auto generato in cui la struttura di cluster è ben visibile.

3 Scelta dei pacchetti

Il linguaggio `R` offre diverse implementazioni del calcolo della Silhouette. Ho cercato alcune fra queste utilizzando il sito `https://rdrr.io` e scegliendone quante più possibili che provenissero dal repository CRAN. In particolare, sono stati scelti `cluster`, `drclust`, `tidyclust` e `Kira`. Pacchetti noti come `fpc`, che pure implementano Silhouette, li ho esclusi a priori perché non aggiornati da tempo.

Oltre a questi, come controprova ho utilizzato l'implementazione della Silhouette presente nel pacchetto `scikit-learn` per `Python`. Questo è stato fatto attraverso il pacchetto `reticulate`, che permette di chiamare funzioni `Python` all'interno di codice `R`.

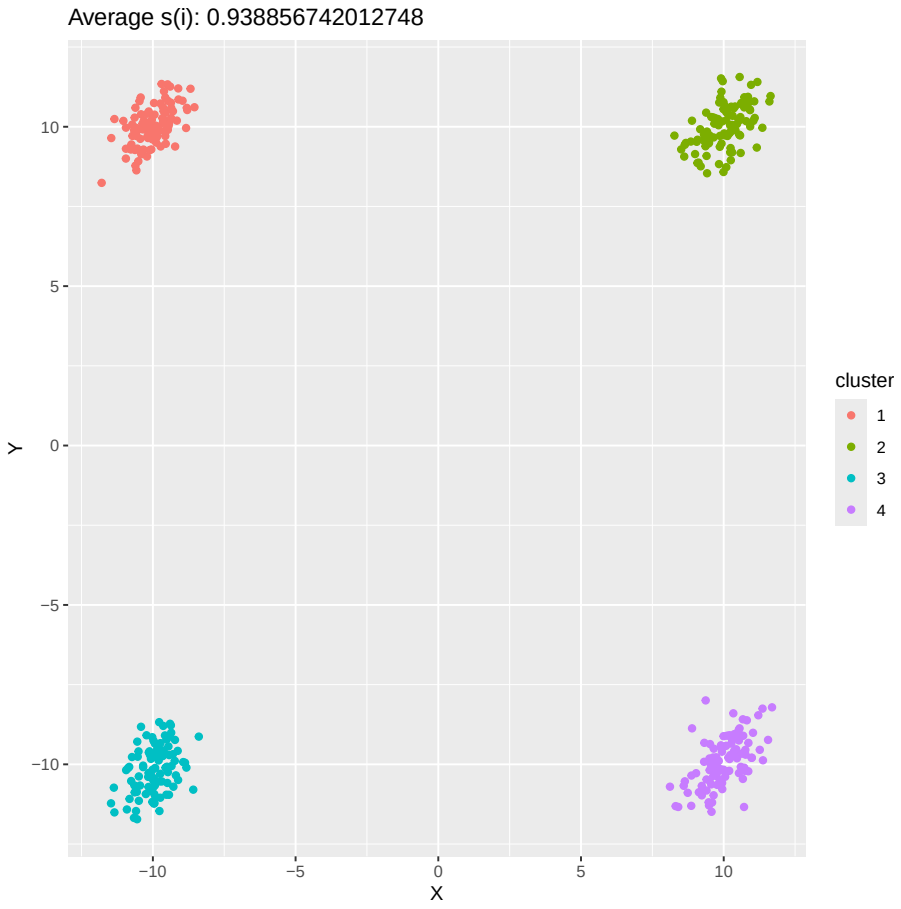

Figure 3: Clustering con K-Means usando K = 4 per un dataset auto generato in cui la struttura di cluster è ben visibile.

Per comparare le performance delle diverse implementazioni di Silhouette ho eseguito due test, uno chiamato "sanity check" ed uno chiamato "matrice binaria".

3.1 Sanity check

Il test Sanity check prevede di utilizzare due dataset creati artificialmente, il primo dove la struttura di cluster è estremamente evidente ed il secondo dove la struttura di cluster è completamente assente, applicare su questi l'algoritmo di clustering K-Means e calcolare sul risultato di quest'ultimo la Silhouette media.

Lo scatter plot dei due dataset è riportato in Figure 5 e Figure 6. Li ho costruiti utilizzando la funzione `rnorm`, che genera dei punti casuali a partire da

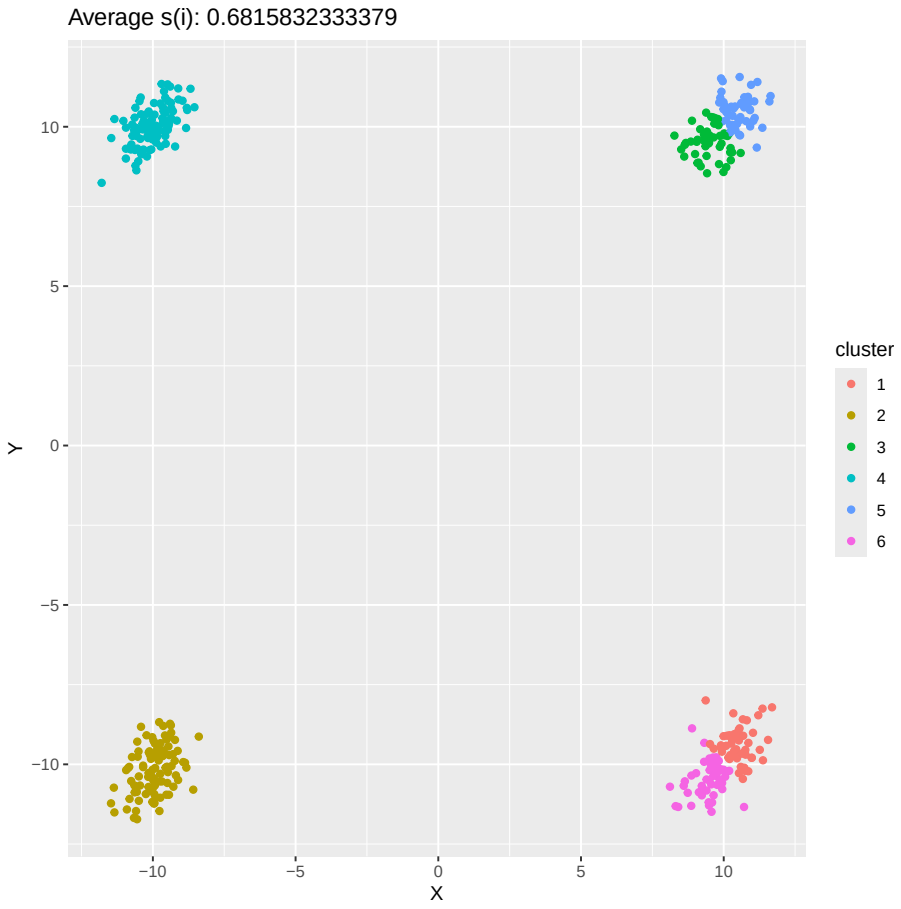

Figure 4: Clustering con K-Means usando K = 6 per un dataset auto generato in cui la struttura di cluster è ben visibile.

una distribuzione normale bivariata.

Il primo dataset è stato costruito usando due distribuzioni normali fuse insieme, entrambe con una deviazione standard molto bassa, di modo che i punti siano concentrati attorno alla media.

Il secondo dataset è costituito da una sola distribuzione normale centrata in $(0, 0)$ con una deviazione standard molto ampia, di modo che i punti siano molto dispersi.

I risultati per il test sanity check sono riportati di seguito. Il test sanity check non é stato particolarmente conclusivo, perché tutti e cinque i pacchetti hanno fornito valori molto simili (circa 0.9 per il primo dataset e circa 0.4 per il secondo). Questo é un risultato atteso, perché il test era appositamente costruito per escludere pacchetti problematici.

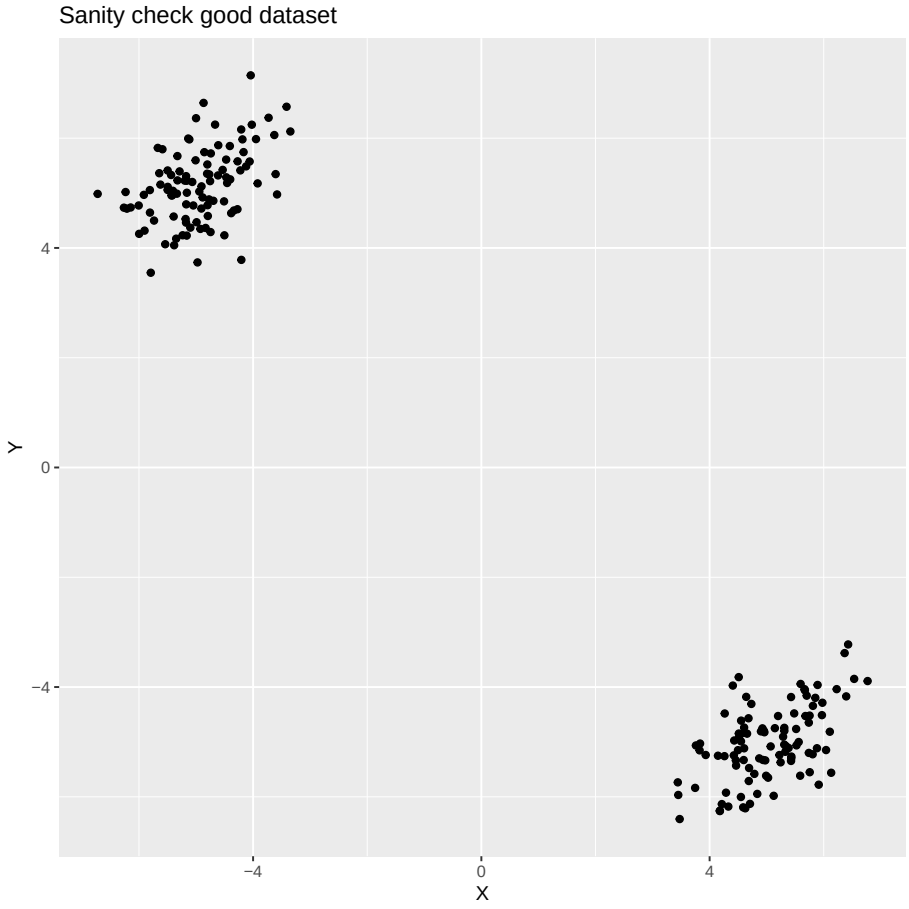

Figure 5: Plot del dataset per il test sanity check, in cui la struttura di cluster è ben visibile.

3.2 Matrice binaria

Il test matrice binaria inizia con una matrice di dimensione $2ntimesn$, costituita per metà da 0 e per metà da 1. Su tale matrice viene applicato K-Means con iperparametro $K = 2$ e si calcola la Silhouette media complessiva del risultato. Dopodiché, una qualsiasi delle righe viene sostituita con un valore scelto casualmente nell'intervallo $(0, 1)$ e si ripete il procedimento. Tale test viene ripetuto esattamente $2n$ volte, ogni volta con una riga diversa.

I $2n$ valori della Silhouette media complessiva così trovati sono poi riportati in un plot; l'idea è che tali valori debbano essere alti nelle prime istanze del test quando gli 0 e gli 1 sono ben separati e piano piano perdano si riducano così come il dataset perde coesione. Ci si aspetta che i punti sul grafico approssimino una distribuzione lineare.

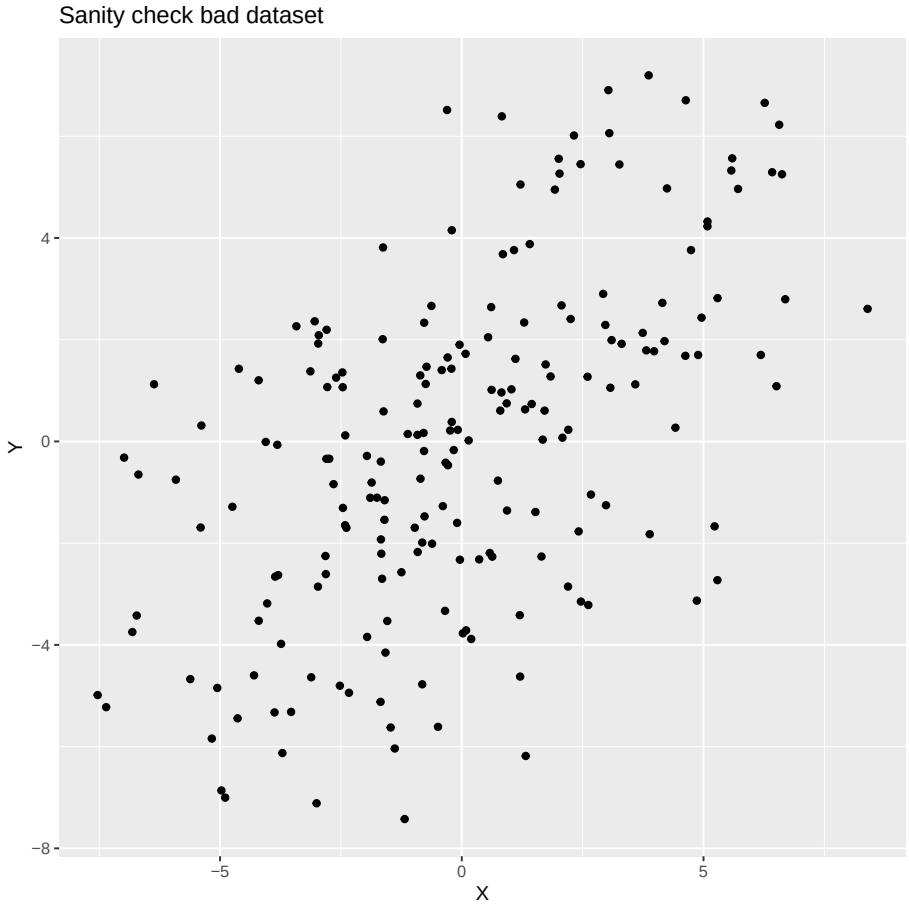

Figure 6: Plot del dataset per il test sanity check, in cui la struttura di cluster è completamente assente.

I risultati per il test matrice binaria sono riportati di seguito. Il test é stato piú informativo del precedente, perché i valori restituiti avevano delle differenze evidenziabili. In particolare, `Kira` é stato il pacchetto con le performance peggiori, perché i valori della Silhouette media complessiva sono rimasti pressoché identici. I pacchetti `cluster`, `drclust` e `tidyclust` hanno invece avuto risultati molto simili. In particolare, `cluster` e `tidyclust` hanno avuto risultati perfettamente identici, segno che probabilmente l'uno usa l'altro come subroutine.

3.3 Considerazioni

Alla luce dei due test considerati, il pacchetto `Kira` é stato immediatamente escluso, perché i risultati forniti dal test della matrice binaria non sono affatto

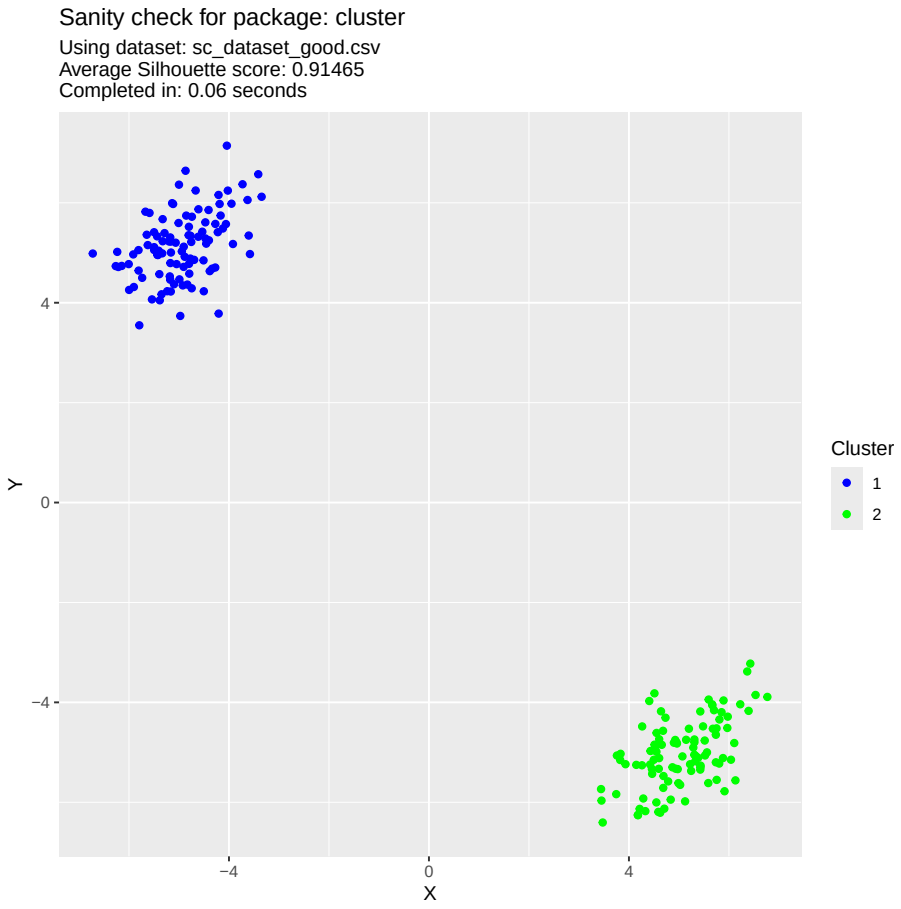

Figure 7: Plot del test sanity check per il pacchetto `cluster`, usando il dataset con struttura di cluster ben visibile.

incoraggianti. Dato che i pacchetti `cluster` e `tidyclust` hanno fornito un risultato identico, fra i due é stato preferito `cluster`, perché fra i due era quello con il tempo di esecuzione piú basso. Il pacchetto `scikit-learn` attraverso `reticulate` non é stato preso in considerazione, perché era stato incluso semplicemente come metro di paragone.

Fra `cluster` e `drclust` ho preferito scegliere `cluster`. Questo sia perché il tempo di esecuzione é inferiore, sia perché il pacchetto `cluster` si trova spesso giá incluso nelle installazioni di `R` (garanzia di affidabilitá) sia perché é l'unico pacchetto il cui input non dipende dall'algoritmo usato. Infatti, `cluster` ha in input il risultato di un qualsiasi algoritmo di clustering ed una matrice delle distanze, mentre gli altri pacchetti richiedono in input espressamente il risultato dell'applicazione di una loro implementazione di un algoritmo.

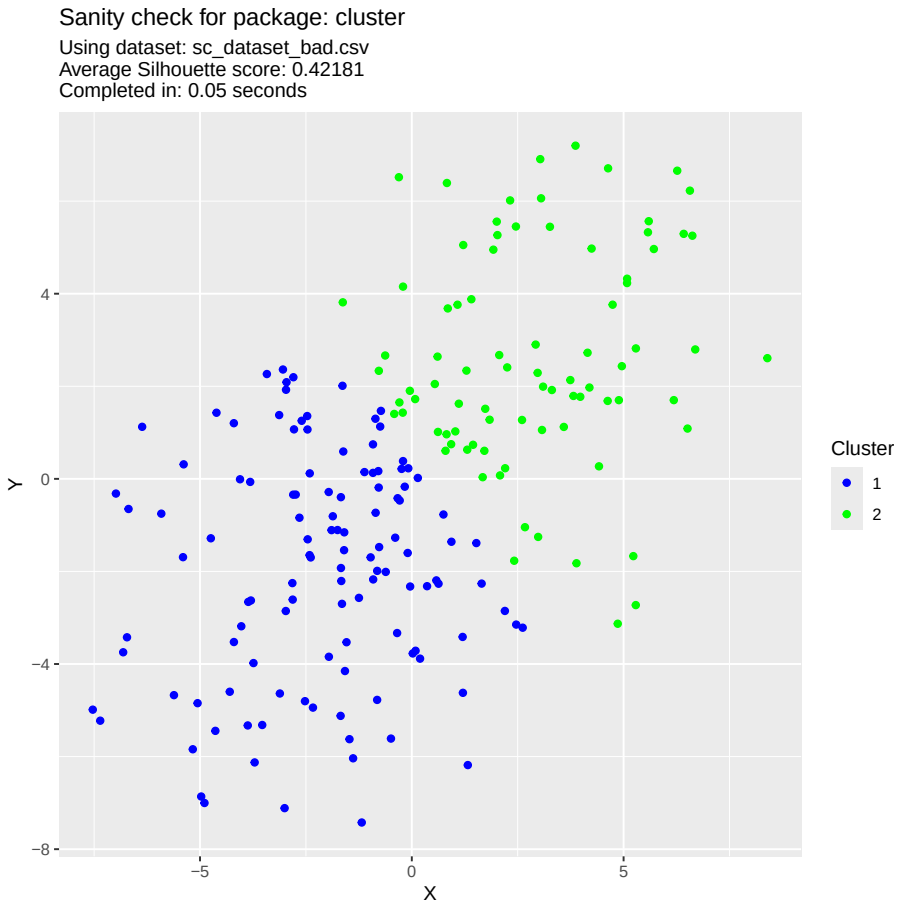

Figure 8: Plot del test sanity check per il pacchetto `cluster`, usando il dataset con struttura di cluster assente.

4 Uso di Silhouette su EHR

5 Aspetti teorici di Silhouette

6 Dataset reali

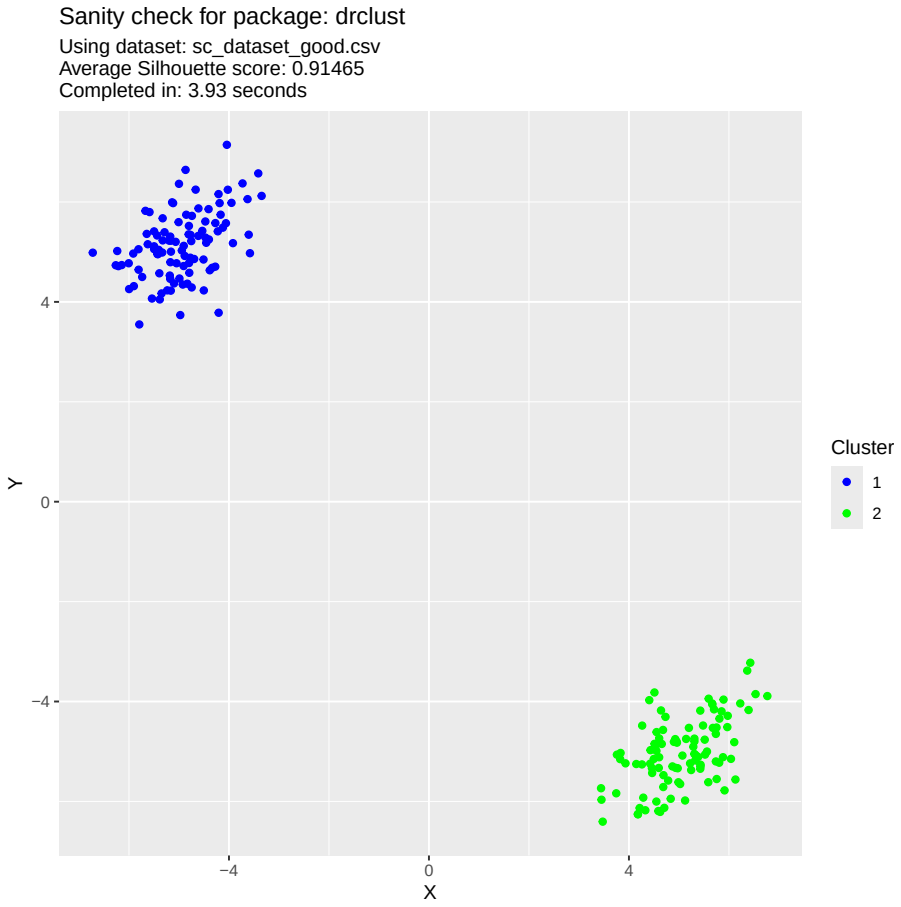

Figure 9: Plot del test sanity check per il pacchetto `drclust`, usando il dataset con struttura di cluster ben visibile.

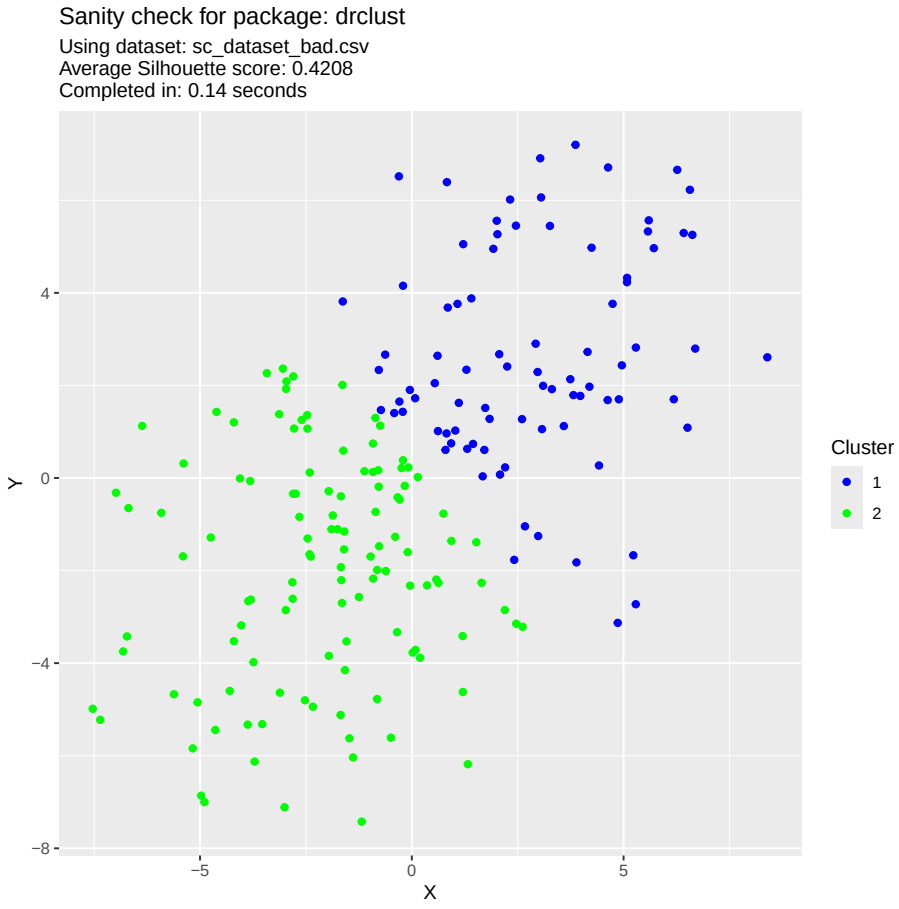

Figure 10: Plot del test sanity check per il pacchetto `drclust`, usando il dataset con struttura di cluster assente.

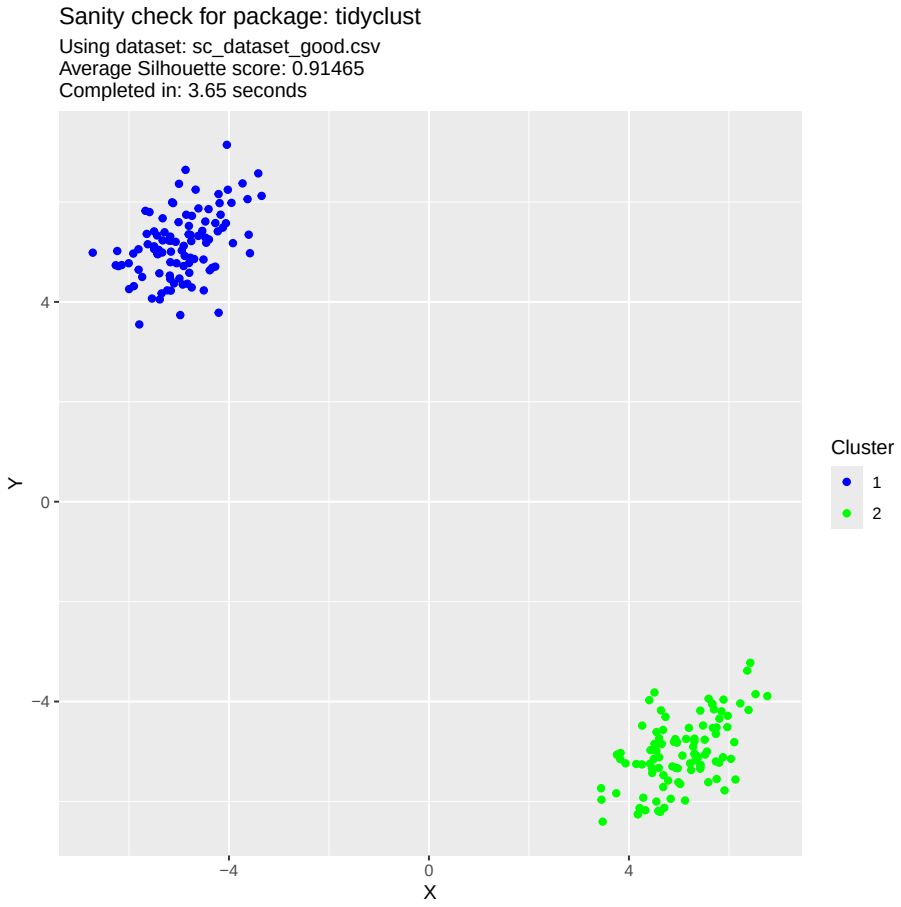

Figure 11: Plot del test sanity check per il pacchetto `tidyclust`, usando il dataset con struttura di cluster ben visibile.

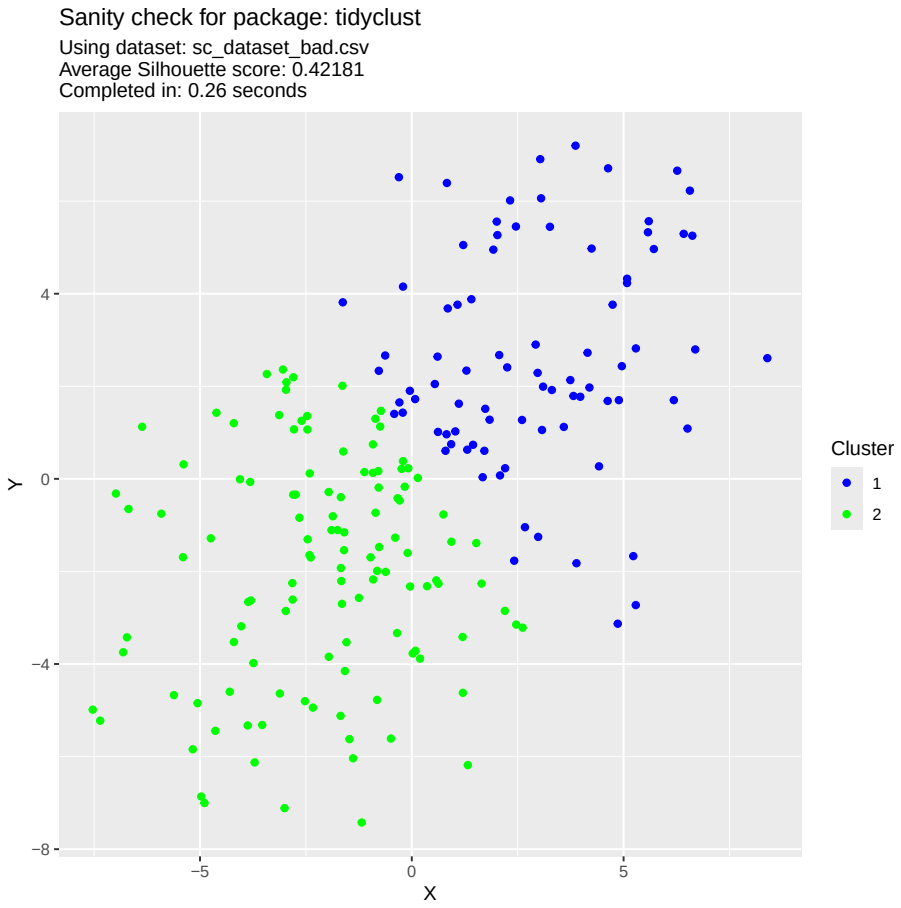

Figure 12: Plot del test sanity check per il pacchetto `tidyclust`, usando il dataset con struttura di cluster assente.

Figure 13: Plot del test sanity check per il pacchetto `Kira`, usando il dataset con struttura di cluster ben visibile.

Figure 14: Plot del test sanity check per il pacchetto `Kira`, usando il dataset con struttura di cluster assente.

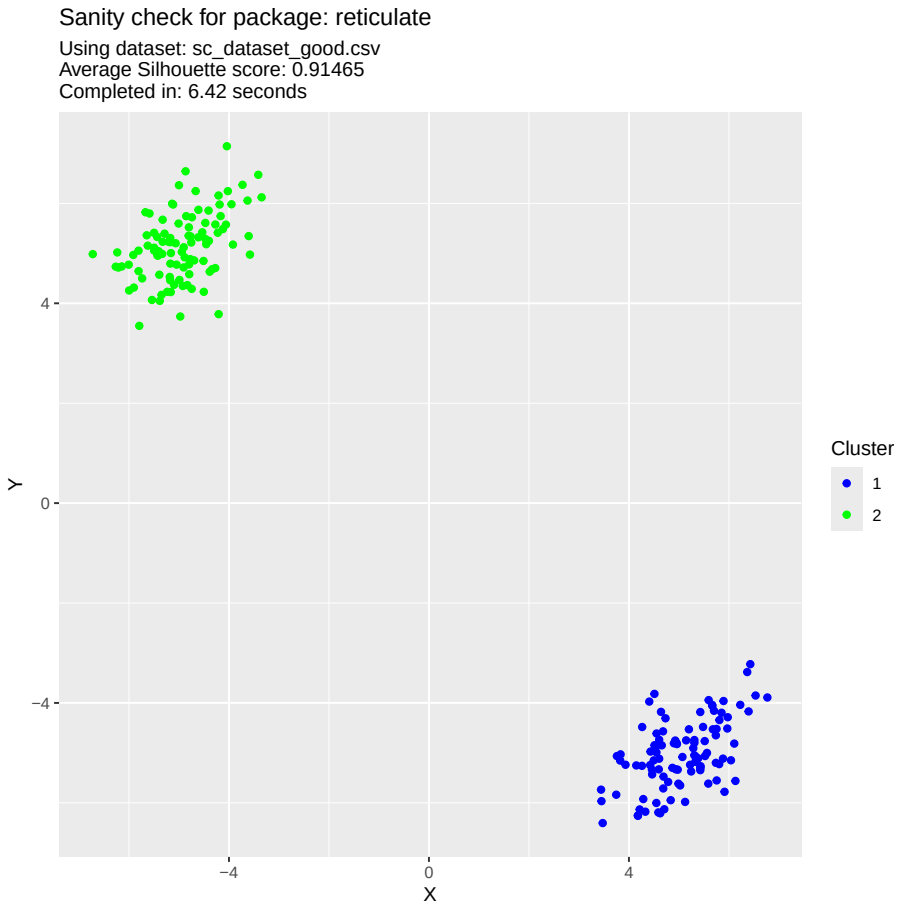

Figure 15: Plot del test sanity check usando `scikit-learn` attraverso `reticulate`, usando il dataset con struttura di cluster ben visibile.

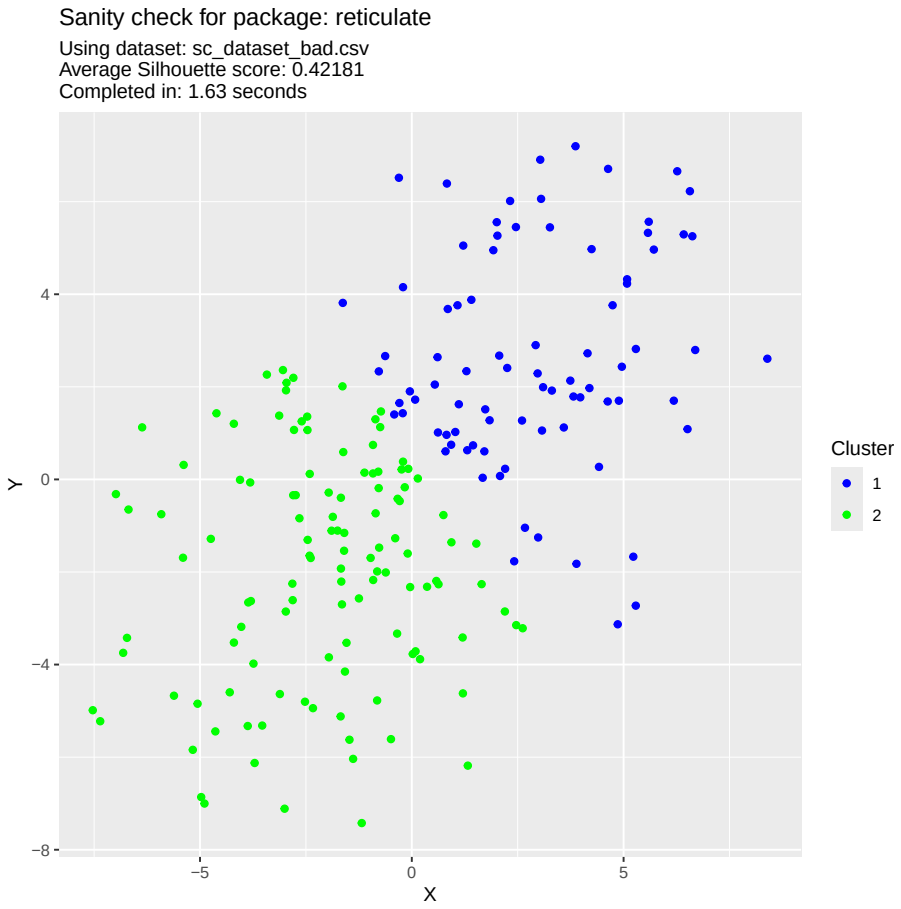

Figure 16: Plot del test sanity check usando `scikit-learn` attraverso `reticulate`, usando il dataset con struttura di cluster assente.

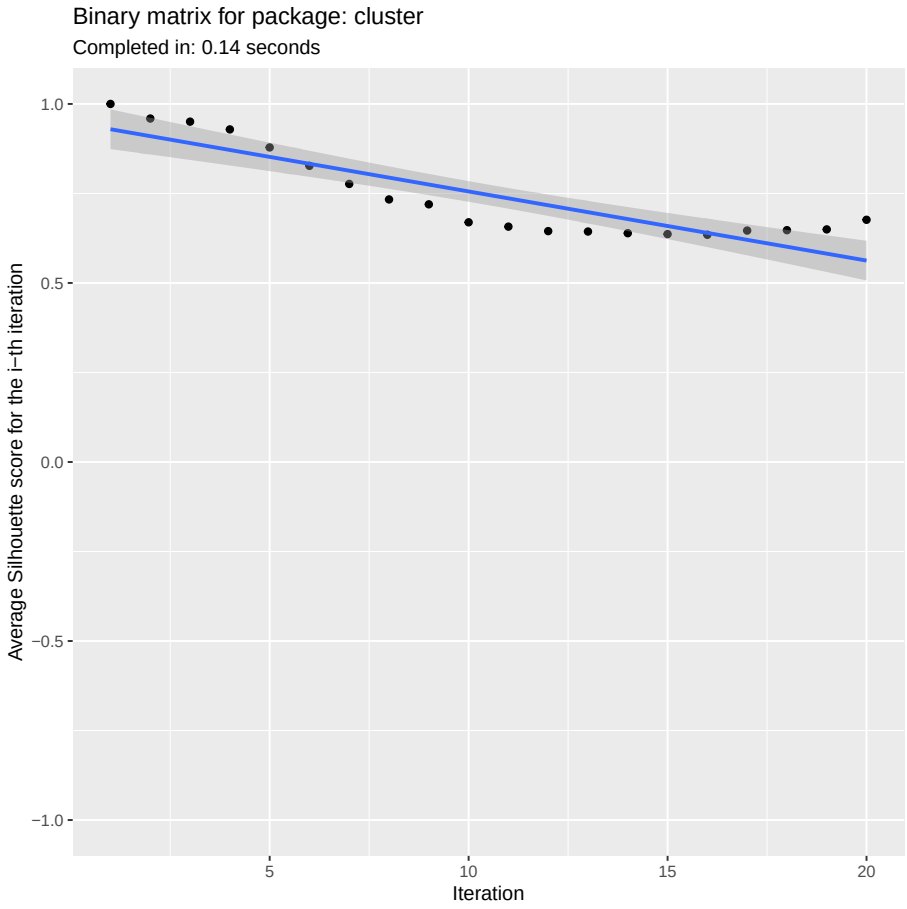

Figure 17: Plot del test matrice binaria per il pacchetto `cluster`

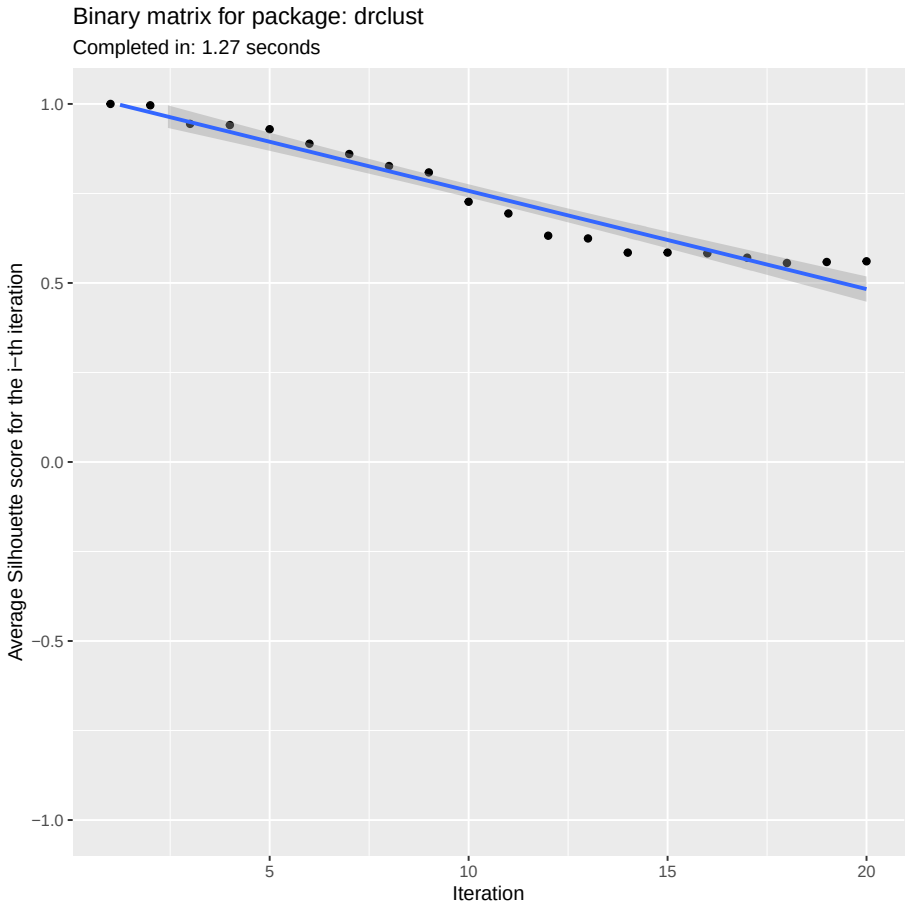

Figure 18: Plot del test matrice binaria per il pacchetto `drclust`

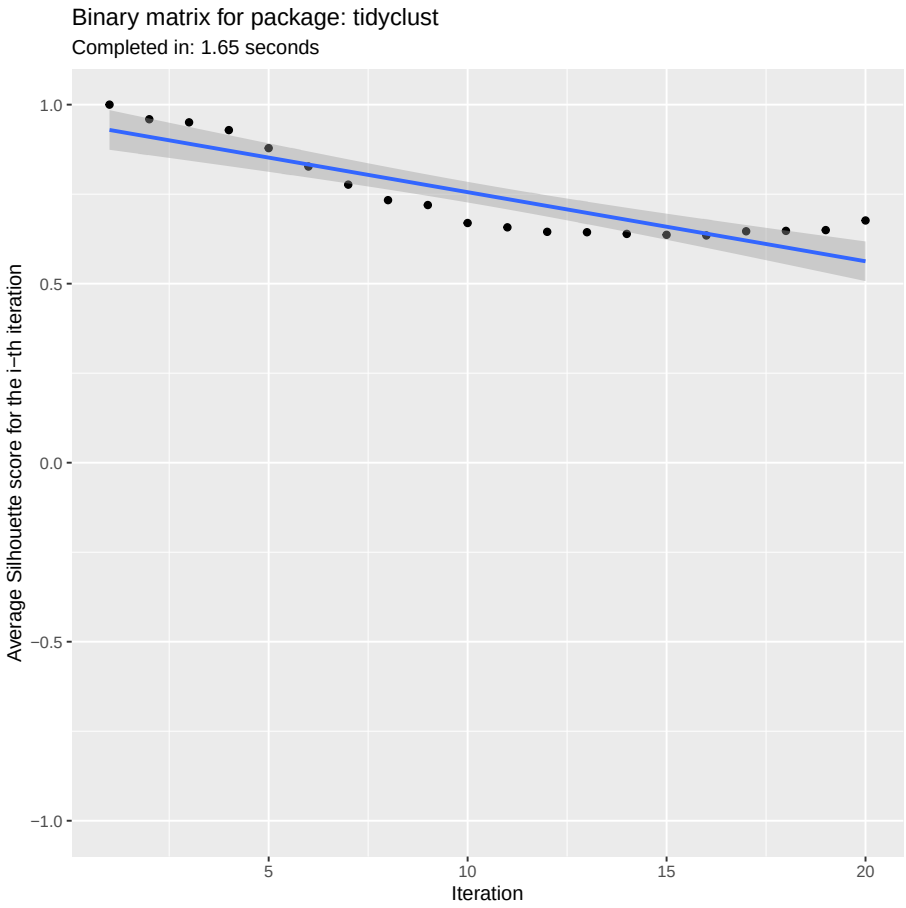

Figure 19: Plot del test matrice binaria per il pacchetto `tidyclust`

Figure 20: Plot del test matrice binaria per il pacchetto `Kira`

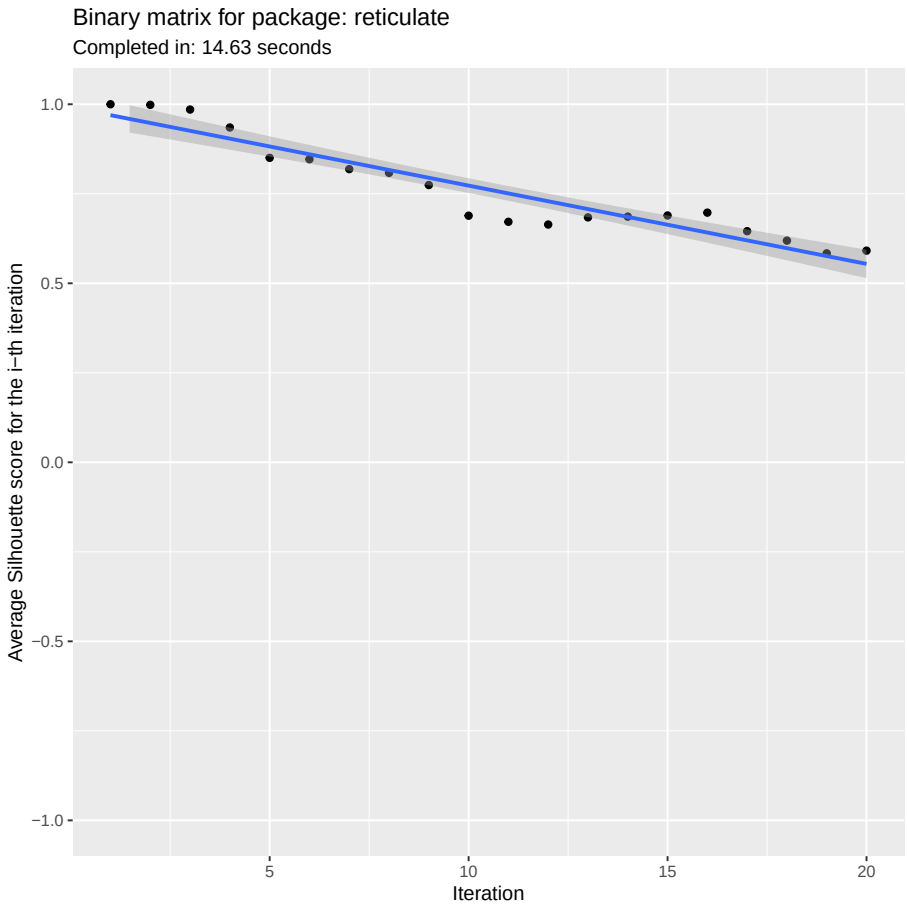

Figure 21: Plot del test matrice binaria per il pacchetto `scikit-learn` (attraverso `reticulate`)