POLITECHNIKA POZNAŃSKA

WYDZIAŁ INFORMATYKI I TELEKOMUNIKACJI
Instytut Informatyki

Praca dyplomowa inżynierska

PORÓWNANIE METOD KLASTROWANIA DANYCH GENOMICZNYCH W PYTHONIE I R

Marta Czarkowska, 151576
Oliwia Kwaśna, 153736

Promotor
dr hab. inż. Aleksandra Świercz

POZNAŃ 2025

Spis treści

Rozdział 1

Wstęp

1.1 Wprowadzenie

Współczesna analiza danych genomicznych odgrywa fundamentalną rolę w biologii molekularnej, ponieważ umożliwia głębsze zrozumienie złożonych mechanizmów biologicznych zakodowanych w genomie. Dynamiczny rozwój technologii sekwencjonowania DNA znacząco zwiększył dostępność wielkoskalowych zbiorów danych. Z jednej strony otworzyło to nowe możliwości badawcze, a z drugiej stworzyło istotne wyzwania analityczne. W szczególności, wysoko wymiarowe dane genomiczne wymagają zastosowania zaawansowanych metod bioinformatycznych, które umożliwiają ich efektywne porządkowanie oraz wyciąganie biologicznie istotnych wniosków. W tym kontekście kluczową rolę odgrywa klastrowanie, czyli metoda umożliwiająca grupowanie obiektów na podstawie ich wzajemnych podobieństw.

Klastrowanie znajduje szerokie zastosowanie w badaniach biologicznych, obejmując m.in. identyfikację współdziałających genów, klasyfikację typów komórek, a także wyodrębnianie podtypów chorób np. nowotwory. Chociaż metody klastrowania pozwalają na uproszczenie analizy danych genomicznych i odkrywanie ukrytych wzorców, ich skuteczność oraz interpretowalność zależą od zastosowanych algorytmów oraz narzędzi analitycznych dostępnych w różnych środowiskach programistycznych.

Obecnie w analizie danych genomicznych dominują dwa języki programowania: R oraz Python. Dostarczają one różnorodne metody i algorytmy przeznaczone do analizy danych wielowymiarowych, w tym algorytmy klasteryzacji (np. K-means, hierarchiczne podejście, DBSCAN) oraz techniki pozwalające na badanie współekspresji genów. Język R, dzięki pakietowi Bioconductor, oferuje specjalistyczne narzędzia dla biologii molekularnej, w tym Weighted Gene Co-expression Network Analysis (WGCNA), które umożliwia szczegółową analizę sieci współekspresji genów. Z kolei Python, korzysta z biblioteki scikit-learn, która wspiera implementację metod: K-means, DBSCAN i hierarchicznego klastrowanie.

Zarówno R oraz Python umożliwiają przeprowadzanie zaawansowanych analiz danych genomicznych. Istnieją pomiędzy nimi fundamentalne różnice w podejściu do oceny podobieństwa między genami lub próbkami. WGCNA wykorzystuje korelacje między poziomami ekspresji genów, poprzez konstruowanie sieci współekspresji. W przeciwieństwie do tego, metody K-means oraz klastrowanie hierarchiczne często bazują na odległości euklidesowej, która może prowadzić do innego sposobu grupowania danych. W konsekwencji, klasyfikacja uzyskana przy użyciu algorytmów Pythonowych może wykazywać większe zagęszczenie wewnętrzne, podczas gdy WGCNA grupuje geny w sposób uwzględniający ich biologiczne współdziałanie. Zrozumienie tych różnic jest kluczowe dla właściwej interpretacji wyników oraz porównywania ich w różnych podejściach analitycznych.

Jednym z największych wyzwań w klastrowaniu danych genomicznych pozostaje ich wysoka wymiarowość, która utrudnia zarówno skuteczne grupowanie, jak i biologiczną interpretację wyników. W związku z tym istotnym kierunkiem badań pozostaje rozwój metod umożliwiających nie tylko precyzyjne klastrowanie, ale również integrację wyników z dostępną wiedzą biologiczną. Odpowiedni dobór narzędzi i algorytmów analitycznych stanowi kluczowy element dalszego postępu w bioinformatyce oraz w badaniach nad funkcjonalnym znaczeniem genomu.

1.2 Cel i zakres pracy

Celem niniejszej pracy jest porównanie wyników różnych metod klastrowania danych genomicznych, zaimplementowanych w językach Python i R. Analiza obejmuje ocenę wydajności obliczeniowej algorytmów, ich interpretowalności oraz jakości wyników. Istotnym elementem pracy jest analiza wyników klastrowania uzyskanych przy użyciu różnych algorytmów w językach Python i R oraz zrozumienie, jak różnice w implementacji metod wpływają na strukturę i interpretację klastrów. Praca ma na celu odpowiedź na bieżące wyzwania związane z analizą danych genomicznych, które charakteryzują się wysoką wymiarowością, obecnością szumów i wartości odstających. Dzięki porównaniu możliwości R i Pythona, praca dostarcza praktycznych wskazówek dla bioinformatyków i naukowców. Wyniki analizy mogą być wykorzystane w badaniach nad ekspresją genów, klasyfikacją próbek biologicznych oraz identyfikacją współdziałających grup genów, przyczyniając się do lepszego zrozumienia procesów biologicznych i odkrywania nowych biomarkerów chorób.

Struktura pracy została zaplanowana w sposób systematyczny, aby umożliwić kompleksowe przedstawienie tematyki klastrowania danych genomicznych oraz analizę porównawczą metod stosowanych w językach Python i R.

Przy realizacji projektu podział zagadnień w pracy został dokonany w następujący sposób:

Oliwia Kwaśna: implementacja w Pythonie metod klastrowania, w tym K-Means, DBSCAN oraz klasteryzacji hierarchicznej, a także analiza porównawcza sposobów grupowania danych w języku Python względem metod dostępnych w R.

Marta Czarkowska: implementacja w R metody klastrowania WGCNA oraz przeprowadzenie analizy wzbogacenia funkcjonalnego dla danych klastrowanych zarówno w Pythonie oraz R.

Wspólne opracowanie: przygotowanie wstępu, wprowadzenia do metod klastrowania oraz sformułowanie końcowych wniosków wynikających z przeprowadzonej analizy.

Dzięki takiemu podziałowi pracy każda z osób mogła skupić się na dogłębnej analizie wybranych metod oraz ich implementacji umożliwiło to szczegółowe porównanie podejść stosowanych w Pythonie i R w kontekście analizy danych genomicznych.

Rozdział 1 stanowi wprowadzenie do problematyki pracy, określenie jej celu oraz zakresu przeprowadzonych badań. Przedstawiona została motywacja podjęcia analizy klastrowania danych genomicznych oraz omówione zostały główne cele badawcze i praktyczne aspekty, jakie miała spełniać niniejsza praca.

Rozdział 2 poświęcony jest metodologii klastrowania danych genomicznych. Rozpoczyna się od omówienia podstaw teoretycznych związanych z klasteryzacją, następnie przedstawiony zostaje przegląd popularnych algorytmów stosowanych w języku Python, K-Means, DBSCAN oraz klasteryzacja hierarchiczna, a także metody klastrowania dostępne w języku R, ze szczególnym uwzględnieniem algorytmu WGCNA. Szczegółowe omówienie metod implementacyjnych w obu językach pozwala na bezpośrednie porównanie podejść oraz ocenę ich efektywności w kontekście analizy danych genomicznych.

Rozdział 3 zawiera opis implementacji algorytmów klastrowania. Na początku przedstawiona zostaje charakterystyka wykorzystanych zbiorów danych genomicznych oraz opis procesu ich wstęp-

nego przetwarzania (preprocessing). Następnie szczegółowo opisano implementację wybranych metod klastrowania w językach Python i R, uwzględniając kroki obliczeniowe, parametryzację algorytmów oraz wizualizację uzyskanych wyników.

Rozdział 4 stanowi podsumowanie analizy porównawczej metod klastrowania zastosowanych w obu językach programowania. Dokonano oceny jakości uzyskanych klastrów oraz porównania efektywności wybranych algorytmów. Na podstawie przeprowadzonych badań sformułowano wnioski dotyczące praktycznej użyteczności metod klastrowania w analizie danych genomicznych oraz wskazano potencjalne kierunki dalszych badań i usprawnień w zakresie stosowanych technik analizy.

Rozdział 5 podsumowywuje przeprowadzone badania oraz wskazuje potencjalne kierunki dalszych analiz. Dokonano w nim oceny skuteczności metod klastrowania w analizie danych genomicznych oraz ich przydatności w identyfikacji biologicznie istotnych grup genów. Poruszone w nim zostają zarówno zalety, jak i ograniczenia zastosowanych podejść, wskazując na istotne różnice między metodami klasycznymi a podejściem sieciowym. Sformułowano również propozycje przyszłych badań, obejmujące integrację wyników z danymi eksperymentalnymi, analizę dynamicznych zmian ekspresji genów oraz rozwój metod automatycznej optymalizacji parametrów klastrowania.

Rozdział 2

Metodyka klastrowania danych genomicznych

2.1 Podstawy teoretyczne klastrowania

Klastrowanie jest jedną z metod uczenia maszynowego, która odnosi się do grupy technik uczenia nienadzorowanego. Nazywane jest również techniką analizy skupień, która pozwala grupować dane w taki sposób, aby elementy w jednej grupie, w tym przypadku w klastrze, były bardziej podobne do siebie niż do elementów w innych grupach. Pomaga w zrozumieniu struktury danych oraz organizowaniu ich w logiczne grupy w zależności od ich cech. Głównym celem klastrowania jest więc, dzielenie danych wejściowych na odrębne grupy na zasadzie wspólnych właściwości. Proces ten polega na odkrywaniu wzorców w zbiorze danych bez ingerencji zewnętrznej użytkownika. Wynik w danych uczących w tej metodzie nie jest w żadnym stopniu określony z góry – nie potrzeba wstępnie zdefiniowanych kategorii ani etykiet. W efekcie umożliwia zidentyfikowanie odstających wartości lub szumów, czyli nieustrukturyzowanych danych, które stanowią zazwyczaj znaczną część danych wejściowych. Powstałe klastry, mogą mieć niejednorodne kształty, różną gęstość oraz rozmiar. Mogą być rozłączne, stykać się lub niekiedy częściowo na siebie nachodzić.

Przed przystąpieniem do klastrowania konieczne jest przygotowanie danych poprzez proces zwany preprocessingiem. Preprocessing obejmuje działania mające na celu poprawę jakości danych i dostosowanie ich do wymagań algorytmu klastrowania. Typowe etapy preprocessingu to normalizacja danych (skalowanie cech do porównywalnych zakresów), usuwanie wartości odstających, radzenie sobie z brakującymi danymi oraz redukcja wymiarowości w przypadku danych o dużej liczbie cech. Przygotowanie danych w takich spsoób, pozwala uzyskać dokładniejsze wyniki i zmniejszyć wpływ szumów na proces analizy skupień.

Klastrowanie jest powszechnie stosowaną technika w celu odkrywania ukrytych wzorców w danych wejściowych i rozwiązywania złożonych problemów analitycznych. Aby lepiej zrozumieć temat klastrowania, należy przedstawić kilka kluczowych pojęć.

Do scharakteryzowania klastrów często wykorzystuje się pojęcie centroidu jako reprezentatywnego punktu, uśrednionego dla wszystkich punktów danych. W klastrowaniu metodą K-means odgrywa kluczową rolę, ponieważ służy jako początkowy punkt, czyli położenie dla klastrów, które są iteracyjnie dopasowywane w celu zminimalizowania odległości punktu danych od najbliższego centroidu. Pierwszym krokiem w tym procesie jest wyznaczenie wstępnych lokalizacji centroidów. Każdy punkt danych zostaje przypisany do grupy, czyli klastra, w którym znajduje się najbliższy centroid względem odległości. Odległość ta jest mierzona zazwyczaj przy użyciu metryki euklidesowej. Gdy wszystkie punkty zostaną przypisane, obliczane są nowe położenia centroidów,

wyznaczane jako średnia arytmetyczna współrzędnych punktów należących do danego klastra. Wspomniany proces jest powtarzany iteracyjnie aż do spełnienia kryterium zbieżności, czyli do momentu, w którym położenia centroidów przestają się widocznie zmieniać. Po zakończeniu iteracyjnego procesu klastrowania obliczana jest odległość każdego punktu od wszystkich centroidów i przypisywany jest on do najbliższego klastra. Celem klastrowania jest uzyskanie uporządkowanych klastrów, w których punkty znajdują się bliżej danego centroidu niż innych, skutkując logicznym i spójnym pogrupowaniem danych.

Kolejnym ważnym pojęciem jest odległość euklidesowa, czyli miara odległości wyrażona w linii prostej pomiędzy dwoma punktami w przestrzeni euklidesowej, inaczej mówiąc, długość odcinka łączącego punkty. W klastrowaniu genów stanowi kluczowe narzędzie, ponieważ pozwala grupować geny na podstawie ich podobieństwa. Stosuje się ją dla zmiennych ilościowych i definiuje jako pierwiastek z sumy kwadratów różnic wartości na poszczególnych wymiarach. W algorytmie K-means odległość euklidesowa jest wykorzystywana do określania najbliższego centroidu, co umożliwia przypisanie genu do klastra reprezentowanego przez ten centroid. Dzięki temu tworzone są naturalne grupy genów na podstawie ich podobieństwa. Dodatkowo odległość euklidesowa znajduje zastosowanie w ocenie jakości klastrów, np. przy użyciu wskaźnika *silhouette score*, który określa, czy punkty danych są bliżej swojego klastra niż innych. Zaletą tej miary jest prostota obliczeń oraz intuicyjność, co sprawia, że jest szeroko stosowana w algorytmach wymagających miary odległości. Jednak ograniczeniem przy korzystaniu z odległości euklidesowej może być przypadek, w którym występują cechy o dużych wartościach, gdyż może to prowadzić do niepoprawnych wyników. Rozwiązaniem jest normalizacja danych, czyli preprocessing (przetwarzanie wstępne). Odległość euklidesowa jest szeroko wykorzystywana w wielu technikach uczenia maszynowego oraz podczas analizy danych, szczególnie w metodach opierających się na podobieństwie punktów w przestrzeni.

Odległość euklidesowa w przestrzeni 2D: $d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

Klastry można zdefiniować jako podzbiory, w których elementy są przyporządkowane na podstawie ich centroidów. W metodzie K-means liczba klastrów musi być z góry ustawiona. Zbyt mała liczba klastrów prowadzi do zbyt ogólnego podziału genów, co może nie przynieść satysfakcjonujących wyników, ponieważ istnieje ryzyko utraty istotnych informacji. Z drugiej strony, zbyt duża liczba klastrów może powodować nadmierny podział danych, co utrudnia prawidłową interpretację wyników i wprowadza szum informacyjny.

Dendrogram to kolejne pojęcie, wykorzystywane w kontekście wizualizacji. Jest to graficzne przedstawienie procesu hierarchicznego klastrowania danych, które pokazuje sposób łączenia obiektów w grupy na różnych poziomach podobieństwa. Wykres przypomina drzewo, gdzie liście reprezentują poszczególne obserwacje, a węzły oznaczają punkty połączenia, pokazując moment złączenia klastrów w większą grupę. Wysokość połączeń na osi Y odzwierciedla odległość lub miarę podobieństwa zastosowaną podczas grupowania, umożliwiając identyfikację liczby klastrów poprzez wyznaczenie poziomej linii cięcia na określonej wysokości. Dendrogram znajduje zastosowanie głównie w hierarchicznym klastrowaniu, które może być realizowane w sposób aglomeracyjny (łączenie małych grup w większe) lub dzielący (podział dużej grupy na mniejsze). Zapewnia również analizę sposobu grupowania elementów danych oraz identyfikację etapu klastrowania, na którym obserwacje zaczęły się różnicować.

W przypadku gdy istotna jest ocena jakość grupowania powszechnie stosuje się wskaźnik sylwetkowy *silhouette score* . Definiowany jako jakość klastrowania oceniającą rozmieszczenie punktów i przypisanie ich do swoich klastrów. Zasada jego działania, polega na porównywaniu średnich odległości punktu od punktów w danym klastrze i od punktów w najbliższym sąsiednim klastrze.

Ocena jakości klastrowania odbywa się w przedziale od -1 do 1, co oznacza, że wartości bliższe -1 są uważane za źle przypisane i powinny znajdować się w innym klastrze. Wartości bliskie 0 oznaczają natomiast, że punkt znajduje się na granicy między klastrami, a wartości, które są bliskie 1, wskazują wysoką wartość wskaźnika i oznaczają, że punkt jest dobrze przypisany do danego klastra. W efekcie tego możliwe jest lepsze zrozumienie, jak klastry są od siebie oddzielone i czy są w odpowiedni sposób umieszczone wewnątrz klastra. Dzięki temu wskaźnikowi, można ocenić kilka metod klastrowania według ich skuteczności. Podczas pracy z dużymi, wielowymiarowymi danymi, obliczenie wskaźnika Silhoutte może być sporym wyzwaniem i być kosztowne obliczeniowo. Korzyścią jest fakt, że wskaźnik umożliwia sprawdzenie różnych algorytmów klastrowania, ocenienie spójności wewnątrz klastrów, jak i separację między klastrami.

Silhouette Score oblicza się według wzoru:

$$S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{2.1}$$

gdzie: - $a(i)$ to średnia odległość punktu i do innych punktów w tym samym klastrze, - $b(i)$ to minimalna średnia odległość punktu i do punktów w innym klastrze (najbliższego sąsiedniego klastra).

Adjusted Rand Index (ARI) to miara zgodności dwóch podziałów zbioru danych, która uwzględnia losowe dopasowania. Wartość ARI wynosi 1 w przypadku pełnej zgodności klasteryzacji z referencyjnym podziałem oraz zbliża się do 0 dla losowego przypisania elementów do klastrów.

Wzór na *Adjusted Rand Index* jest następujący:

$$ARI = \frac{\sum_{ij} n_{ij}2 - \left[\sum_i a_i 2 \sum_j b_j 2\right] /n2}{\frac{1}{2}\left[\sum_i a_i 2 + \sum_j b_j 2\right] - \left[\sum_i a_i 2 \sum_j b_j 2\right] /n2} \tag{2.2}$$

gdzie: - n_{ij} to liczba elementów wspólnych w klastrach i i j, - a_i to suma elementów w klastrze i, - b_j to suma elementów w klastrze j, - n to całkowita liczba elementów.

ARI jest szczególnie użytecznym wskaźnikiem do porównania różnych metod klasteryzacji, ponieważ uwzględnia zarówno poprawne przypisania elementów do klastrów, jak i wpływ losowych dopasowań.

Obie miary – *ARI* i *Silhouette Score* – pozwalają na ocenę jakości klasteryzacji z różnych perspektyw. *ARI* umożliwia porównanie wyników z rzeczywistym podziałem referencyjnym, natomiast *Silhouette Score* analizuje spójność i separację klastrów na podstawie metryk odległościowych. Wspólne wykorzystanie obu metryk pozwala na kompleksową ocenę wyników grupowania danych genomicznych.

Zrozumienie tych podstawowych pojęć i miar oceny jakości klastrowania jest kluczowe dla skutecznej analizy danych i wyboru odpowiednich metod klastrowania w zależności od charakterystyki danych.

2.2 Metody klastrowania w Pythonie

Algorytm K-means, opracowany przez MacQueen (1967), jest jedną z najczęściej stosowanych metod klastrowania w bioinformatyce [6]. Działa iteracyjnie, ponieważ dzieli dane na nieprzecinające się klastry w celu minimalizacji wariancji wewnątrz grup, czyli sumy kwadratów odległości punktów od centroidów. Liczba klastrów jest określana przez użytkownika, a początkowe centroidy wybierane są losowo. Najczęściej stosowaną metryką jest odległość euklidesowa. Algorytm przypisuje punkty do najbliższych centroidów, aktualizuje ich pozycje jako średnie wartości współrzędnych przypisanych punktów i powtarza proces, aż do momentu, gdy centroidy przestaną się

zmieniać lub osiągnięta zostanie maksymalna liczba iteracji. K-Means jest szybki i łatwy w implementacji, ale wrażliwy na wartości odstające oraz szum, co wpływa na potencjalne obniżenie jakości wyników. Algorytm zakłada, że klastry mają kształt sferyczny i podobną wielkość, co nie zawsze odpowiada rzeczywistym danym biologicznym. Przed zastosowaniem K-Means kluczowe jest przeprowadzenie normalizacji danych, aby zapobiec zaburzeniom wyników przez różny zakres wartości.

Kolejnym algorytmem jest DBSCAN (*ang. Density-Based Spatial Clustering of Applications with Noise*), opracowany przez Ester et al. (1996) [2]. Jest to algorytm klastrowania oparty na gęstości, który skutecznie grupuje klastry o nieregularnych kształtach i radzi sobie z punktami odstającymi, klasyfikując je jako szum. Klaster jest definiowany jako zbiór punktów o wysokiej gęstości, oddzielony od innych klastrów obszarami o niskiej gęstości. Działanie algorytmu opiera się na dwóch parametrach: epsilon – maksymalna odległość uznawana za sąsiedztwo, oraz minPts – minimalna liczba punktów wymagana do uznania punktu za rdzeniowy. W DBSCAN można wyróżnić trzy rodzaje punktów: punkty rdzeniowe, czyli punkty posiadające co najmniej minPts sąsiadów, punkty brzegowe, które są nliskimi punktami rdzeniowymi, ale o mniejszej liczbie sąsiadów oraz punkty szumu, nieprzypisane do żadnego klastra. Algorytm rozpoczyna od losowego punktu i sprawdza, czy dany punkty jest punktem rdzeniowym, i jeśli tak, tworzy nowy klaster, do którego iteracyjnie dodaje sąsiednie punkty spełniające kryteria. Proces powtarza się do momentu przetworzenia wszystkich punktów. DBSCAN nie wymaga wcześniejszego określenia liczby klastrów, a ich liczba wynika z rozkładu gęstości danych. Algorytm jest odporny na wartości odstające, ale jego skuteczność zależy od właściwego doboru parametrów. Niewłaściwe wartości epsilon lub minPts mogą prowadzić do błędnej segmentacji danych.

Klasteryzacja hierarchiczna to jedna z podstawowych metod grupowania danych, która została szczegółowo opisana przez Everitta i współpracowników w książce Cluster Analysis (2011) [3]. Algorytm ten pozwala na organizowanie danych w strukturę hierarchiczną, co znajduje szerokie zastosowanie w analizie danych wielowymiarowych. Wynik procesu jest przedstawiany w postaci dendrogramu, wizualizującego zależności między punktami danych oraz klastrami. Wyróżnia się dwa rodzaje klasteryzacji hierarchicznej: aglomeracyjną (łączenie) oraz dzielącą (podział). Klasteryzacja aglomeracyjna to podejście "od dołu do góry". W tym podejściu każdy obiekt (gen) rozpoczyna jako odrębny klaster. Następnie iteracyjnie łączą się najbardziej podobne do siebie klastry, aż wszystkie dane zostaną połączone w jeden nadrzędny klaster lub osiągnięta zostanie z góry określona liczba klastrów. W każdej iteracji oblicza się miarę podobieństwa dla wszystkich par klastrów. Najczęściej stosowaną miarą jest odległość euklidesowa. Dwa najbardziej podobne klastry są łączone w jeden, a miary podobieństwa są aktualizowane, aby uwzględnić nowo powstały klaster. Proces ten powtarza się do momentu osiągnięcia ustalonego kryterium zakończenia. Klasteryzacja dzieląca to podejście od góry do dołu. Rozpoczyna się od określenia całego zbioru danych jako jednego, większego nadrzędnego klastra, a potem klaster jest iteracyjnie dzielony na mniejsze, aż osiągnie określoną liczbę klastrów.

Proces klastrowania danych, niezależnie od wybranej metody, może napotkać kilka wyzwań, które wpływają na jakość wyników i efektywność obliczeń. Losowy wybór centroidów może prowadzić do nieoptymalnych wyników, więc kluczowe jest dobranie odpowiednich parametrów, czyli liczby klastrów i pozycje centroidów. Szum i wartości odstające mogą zakłócać wyniki klastrowania, szczególnie w algorytmie K-Means, który zakłada istnienie dobrze zdefiniowanych grup. W przypadku algorytmu DBSCAN problem ten jest znacznie zredukowany, ponieważ punkty o niskiej gęstości są traktowane jako szum i nie są przypisywane do żadnego klastra. Przy klastrowaniu danych o dużej liczbie cech, czyli danych genomicznych, rośnie trudność w skutecznym tworzeniu klastrów. Wysoka wymiarowość powoduje, że dane mogą stać się rzadkie i trudniejsze do porów-

nywania. Aby poprawić efektywność, często stosuje się metody redukcji wymiarowości PCA, czyli analiza głównych składowych powodująca zmniejszenie liczby cech przy jednoczesnym zachowaniu istotnych informacji. Algorytm klasteryzacji hierarchicznej charakteryzuje się wysoką złożonością obliczeniową, zwłaszcza dla dużych zbiorów danych, ponieważ wymaga obliczenia odległości dla wszystkich par punktów. W przypadku dużych danych bardziej efektywne mogą być algorytmy o niższej złożoności, czyli K-Means lub DBSCAN, które są szybsze i lepiej skalują się do dużych zbiorów danych.

2.3 Metody klastrowania w R

Weighted Gene Co-Expression Network Analysis (WGCNA) to zaawansowana technika analizy danych genomicznych, która umożliwia badanie współekspresji genów i grupowanie ich w moduły na podstawie podobieństw wzorców ekspresji. Została obszernie zdefiniowana w artykule [8] i jest metodą szeroko stosowaną w genomice funkcjonalnej i badaniach biomedycznych, cenioną za swoją elastyczność i zdolność do analizy złożonych, wielowymiarowych danych. W języku R zgodnie z tym co opisał [5] , WGCNA pozwala na efektywne zarządzanie dużymi zbiorami danych oraz identyfikację kluczowych genów i powiązań w sieciach biologicznych. Proces analizy rozpoczyna się od wyznaczenia macierzy korelacji między genami, która określa siłę powiązań między ich profilami ekspresji w badanych próbkach. Aby uwzględnić zmienność danych i ograniczyć wpływ skrajnych korelacji, metoda ta stosuje tzw. soft-thresholding, czyli przekształcenie korelacji w wagi poprzez podniesienie ich do potęgi (parametr β). Tak przeprowadzona transformacja wzmacnia silne korelacje, pomagając w identyfikacji kluczowych zależności, jednocześnie tłumiąc słabe korelacje, co redukuje szum w danych. Optymalna wartość parametru β jest dobierana na podstawie analizy topologii sieci w celu osiągnięcia struktury topologii bezskalowej (*ang. scale-free topology*), charakterystycznej dla sieci biologicznych, w których rozkład stopni węzłów podlega funkcji potęgowej.

W kolejnym etapie budowana jest macierz Topological Overlap Matrix (TOM), która uwzględnia zarówno bezpośrednie, jak i pośrednie połączenia między genami. Macierz TOM zgodnie z artykułem [4] wzmacnia strukturę modułów i eliminuje przypadkowe połączenia, co pozwala na bardziej precyzyjne grupowanie genów. Na jej podstawie geny są grupowane za pomocą hierarchicznego klastrowania, które wykorzystuje metodę średnią (*ang. average linkage*) i oblicza odległości między genami jako $1 - TOM_{ij}$. Dendrogram powstały w tym procesie odzwierciedla zależności między genami, a dynamiczne cięcie drzewa (*ang. Dynamic Tree Cut*) dzieli dendrogram na moduły, dostosowując rozmiar klastrów do charakterystyki danych. Dzięki temu możliwe jest wykrywanie modułów o różnej wielkości i kształcie, co jest szczególnie istotne w przypadku dużych i heterogenicznych zbiorów danych genomicznych.

Każdy moduł jest reprezentowany przez eigengen, czyli pierwszy główny komponent (PCA) profilu ekspresji genów w module, który pełni rolę metagenu. Eigengeny umożliwiają analizę relacji między modułami oraz ich hierarchii. Na podstawie macierzy odmienności eigengenów, np. korelacji ujemnej, tworzy się dendrogram modułów, co pozwala na identyfikację zależności między nimi. Moduły, których eigengeny wykazują wysoką korelację, na przykład powyżej 0,75, mogą być scalane w większe moduły, co upraszcza analizę i redukuje redundancję. Po identyfikacji modułów możliwa jest bardziej szczegółowa analiza sieci, która uwzględnia jej topologię lokalną i globalną. Analiza stopnia połączeń węzłów pozwala na wskazanie kluczowych genów, tzw. „*hub genes*," które pełnią istotne role biologiczne w module. Dodatkowo graficzne przedstawienie modułów i ich powiązań umożliwia łatwiejszą interpretację wyników oraz identyfikację kluczowych interakcji między genami.

Metoda WGCNA łączy w sobie techniki klastrowania, analizy sieci i redukcji wymiarowości w jednym spójnym frameworku. Dzięki zastosowaniu transformacji korelacji i analizy topologicznej jest odporna na szum w danych i dostosowana do wymagań dużych zbiorów genomicznych. Dzięki hierarchicznemu klastrowaniu, dynamicznemu cięciu drzewa oraz analizie eigengenów możliwe jest precyzyjne grupowanie genów w moduły, które następnie można analizować w kontekście biologicznym. WGCNA znalazła szerokie zastosowanie w badaniach takich jak identyfikacja biomarkerów chorób, analiza różnic w ekspresji genów w różnych warunkach eksperymentalnych oraz badania nad interakcjami genów i szlakami metabolicznymi.

Rozdział 3

Przeprowadzenie analizy

3.1 Przygotowanie danych do analizy (preprocessing)

3.1.1 Opis danych wejściowych

Dane wykorzystane w analizie pochodzą z genomu Homo sapiens i obejmowały informacje dotyczące ekspresji genów. Zostały również wstępnie znormalizowane. Zbiór danych pobrano z publicznej bazy Gene Expression Omnibus (*GEO*) i jest dostępny pod unikalnym identyfikatorem projektu GSE116428. Eksperyment[7], z którego pochodzą dane, miał na celu analizę różnic w ekspresji genów między próbkami kontrolnymi a próbkami pobranymi od pacjentów z preeklampsją (PE), osobno dla kobiet i mężczyzn. Plik wejściowy składał się z 32 kolumn i 28 677 wierszy. Liczba wierszy odpowiadała liczbie genów. W pierwszych 5 kolumnach oprócz nazw genów znajdowały się dodatkowe metryki, takie jak średnia ekspresja (*mean*), wariancja (*var*), współczynnik zmienności (*coeff of v.*) oraz klasyfikacja genów na długie niekodujące RNA (lncRNA). W kolejnych 27 kolumnach znajdowały się informacje dotyczące ekspresji genów w poszczególnych próbkach. Fragment pliku wejściowego został przedstawiony w Tabeli 3.1

Name	**lncRNA**	**mean**	**var**	**coeff of var**	**SRR7451446**
A2M-AS1	1	5.14	30.25	1.07	0.00
AADACL2-AS1	1	2.58	19.84	1.73	2.71
AATBC	1	20.88	64.05	0.38	20.36

Tabela 3.1: Fragment pliku wejściowego.

Szczegółowe informacje na temat próbek znajdowały się w osobnym pliku *pheno_samples.txt*, którego fragment został zamieszczony w Tabeli 3.2.

Sample	**Sample Name**	**Type**	**Type2**	**Sex**
SRR7451446	NT_F_rep1	NT_F	NT	F
SRR7451447	NT_F_rep2	NT_F	NT	F
SRR7451448	NT_F_rep3	NT_F	NT	F
SRR7451449	NT_F_rep4	NT_F	NT	F
SRR7451450	NT_F_rep5	NT_F	NT	F
SRR7451451	NT_F_rep6	NT_F	NT	F
SRR7451452	NT_F_rep7	NT_F	NT	F
SRR7451453	NT_F_rep8	NT_F	NT	F
SRR7451454	NT_M_rep1	NT_M	NT	M
SRR7451455	NT_M_rep2	NT_M	NT	M

Sample	Sample Name	Type	Type2	Sex
SRR7451456	NT_M_rep3	NT_M	NT	M
SRR7451457	NT_M_rep4	NT_M	NT	M
SRR7451458	NT_M_rep5	NT_M	NT	M
SRR7451459	NT_M_rep6	NT_M	NT	M
SRR7451460	NT_M_rep7	NT_M	NT	M
SRR7451461	NT_M_rep8	NT_M	NT	M
SRR7451462	PE_F_rep1	PE_F	PE	F
SRR7451463	PE_F_rep2	PE_F	PE	F

Tabela 3.2: Fragment tabeli zawierającej próbki i ich charakterystyki.

Próbki zostały sklasyfikowane na podstawie typu (*type*), podtypu (*type2*) oraz płci (*sex*), co pozwoliło na podział na grupy: NT_F (kontrolne od kobiet), NT_M (kontrolne od mężczyzn), PE_F (przypadki od kobiet) oraz PE_M (przypadki od mężczyzn).

3.1.2 Preprocessing danych w obu środowiskach

Pierwszym etapem preprocessingu było wczytanie znormalizowanych danych z pliku w formacie .xlsx, zawierającego informacje o ekspresji genów w poszczególnych próbkach. Następnie obliczono wartości średniej wariancji dla poszczególnych kolumn, co pozwoliło na przeprowadzenie selekcji genów o odpowiednio wysokiej zmienności. W celu eliminacji genów o niskiej wariancji, które mogłyby wprowadzać szum do analizy i nie wnosiły istotnych informacji biologicznych, ustalono próg wariancji na poziomie 10. W efekcie liczba analizowanych genów została zredukowana z 28 677 do 22 247, co umożliwiło skupienie się na najbardziej zmiennych i potencjalnie istotnych biologicznie genach. Ofiltrowano 3 473 z 20 788 genów kodujących białka.

Warto zaznaczyć, że szczególną uwagę poświęcono genom oznaczonym jako lncRNA (długie niekodujące RNA), ze względu na ich istotną dla biologów rolę w regulacji ekspresji genów oraz procesach komórkowych. Jednakże ze względu na ich charakterystycznie niską ekspresję i wariancję, zastosowanie zbyt restrykcyjnego progu wariancji mogłoby prowadzić do ich nadmiernej eliminacji. Aby temu zapobiec, wartość progowa również została dobrana tak, aby zminimalizować utratę biologicznie istotnych lncRNA, przy jednoczesnym odfiltrowaniu genów o zbyt niskiej zmienności. W wyniku tego procesu usunięto 2 957 z 7 889 genów lncRNA.

Kluczowym etapem była selekcja kolumn zawierających wartości ekspresji dla poszczególnych próbek. Ponieważ pierwsze pięć kolumn zawiera metadane lub inne informacje pomocnicze, zostają one pominięte w analizie. W tym celu do zmiennej available_samples przypisane zostały wszystkie kolumny zbioru danych z wyłączeniem pierwszych pięciu, co pozwala uzyskać listę nazw kolumn reprezentujących próbki biologiczne analizowane w dalszych częściach skryptu.

3.1.3 Dodatkowy preprocessing w Pythonie: Normalizacja i PCA

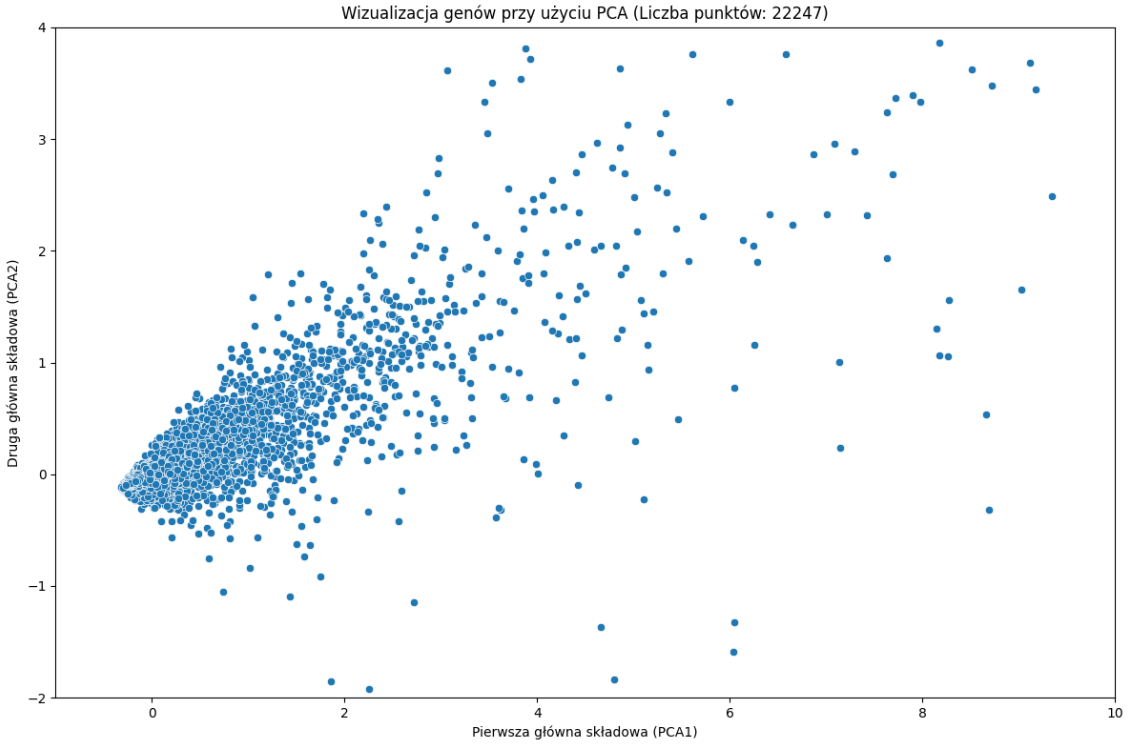
Chociaż dane były wstępnie znormalizowane, przeprowadzono dodatkową normalizację w Pythonie przy użyciu narzędzia StandardScaler z biblioteki NumPy. Normalizacja ta polegała na standaryzacji danych, co oznacza, że dla każdej cechy obliczono średnią oraz odchylenie standardowe, a następnie przekształcono dane tak, aby miały średnią równą 0 i odchylenie standardowe równe 1. Znormalizowane dane zostały zapisane w macierzy NumPy, co pozwoliło na ich późniejsze wykorzystanie w analizie statystycznej i biologicznej.

Aby zredukować wymiarowość danych genowych i umożliwić ich wizualizację, zastosowano metodę analizy głównych składowych PCA (Principal Component Analysis). PCA pozwala na reprezentację danych w przestrzeni o mniejszej liczbie wymiarów poprzez wyznaczenie nowych osi głównych, na których zmienność danych jest największa. W analizie wykorzystano wartości ekspresji genów dla próbek od SRR7451446 do SRR7451472. Przed zastosowaniem PCA dane zostały znormalizowane przy użyciu metody StandardScaler, co pozwoliło na przeskalowanie wszystkich wartości do rozkładu o średniej równej 0 i odchyleniu standardowym równym 1. PCA wykonano z liczbą głównych składowych równą 10, co umożliwiło wyznaczenie dziesięciu nowych wymiarów opisujących największe zmiany w danych. Wyniki zapisano w nowej Tabeli 3.3 jako kolumny PCA1, PCA2, ..., PCA10, które reprezentują kolejne główne składowe.

Aby zwizualizować dane, wygenerowano wykres punktowy (Rysunek 3.1), który przedstawia dane w przestrzeni wyznaczonej przez pierwszą (PCA1) i drugą główną składową (PCA2). Osie wykresu odpowiadają wartościom tych składowych, które wyjaśniają największą część zmienności danych. PCA1 odpowiada za 92.05% całkowitej wariancji, natomiast PCA2 za 6.06%, co łącznie daje 98.11

Dla zapewnienia przejrzystości wykresu ograniczono zakres osi X do wartości od -1 do 10, a zakres osi Y do wartości od -2 do 4. Poszczególne punkty na wykresie reprezentują geny, a ich rozmieszczenie odzwierciedla różnice w wartościach głównych składowych.

Dodatkowo, obliczono wartości wyjaśnionej wariancji dla kolejnych składowych oraz sumaryczną wartość dla pierwszych dziesięciu składowych. Łączna wyjaśniona wariancja przez 10 głównych składowych wyniosła 99.77%, co wskazuje na wysoką skuteczność redukcji wymiarowości przy zachowaniu większości informacji o zmienności danych.



Rysunek 3.1: PCA

Tabela 3.3: Procent wariancji wyjaśnionej przez pierwsze 10 głównych składowych PCA.

Główna składowa	**Wyjaśniona wariancja**
PCA1	92.05%
PCA2	6.06%
PCA3	0.70%
PCA4	0.31%
PCA5	0.22%
PCA6	0.13%
PCA7	0.11%
PCA8	0.08%
PCA9	0.05%
PCA10	0.05%
Całkowita wyjaśniona wariancja	**99.77%**

3.2 Implementacja algorytmów klastrowania w Pythonie

Algorytm K-means

Pierwszym algorytmem wykorzystanym do klasteryzacji i porównania sposobu grupowania danych genomicznych jest K-means z biblioteki scikit-learn. Klastrowanie zostało przeprowadzone dla zakresu liczby klastrów od 40 do 65 z krokiem co 5, analizując wartości: 40, 45, 50, 55, 60 i 65. Następnie został utworzony obiekt K-means dla każdej liczby klastrów z parametrami *random_state=42* i *n_init=50*, aby zredukować losowość oraz przeprowadzić 50 inicjalizacji algorytmu, zwiększając stabilność wyników. Algorytm dopasowywał się do danych za pomocą metody *fit*. W obiekcie *DataFrame* uzyskane etykiety klastrów dla każdego punktu zapisano jako nowe kolumny, np. *Cluster_40*. Po klastrowaniu wykorzystano funkcję *silhouette_score* do obliczenia współczynnika *silhouette*, który ocenia, jak dobrze punkty danych pasują do swoich klastrów w porównaniu z klastrami sąsiednimi. Funkcja *silhouette_samples*, określiła wartości współczynnika *silhouette* dla poszczególnych punktów oraz ich średnie dla każdego klastra. Dzięki temu było możliwe ocenienie poziomu wewnętrznej spójności grup.

Tabela 3.4 przedstawia średni wskaźnik *Silhouette* i liczebność punktów danych przypisanych do każdego klastra. Liczba klastrów k=40 została wybrana jako początkowy krok analizy ze względu na możliwość lepszego odwzorowania różnorodności danych przy umiarkowanej liczbie klastrów.

Średni wskaźnik *Silhouette* dla klastrów był zróżnicowany, ponieważ niektóre klastry mają wysokie wartości wskaźnika, co wskazuje na dobrą separację i spójność wewnętrzną (np. klastry o wysokiej liczebności), a niektóre niską lub zerową. Klastry singletony (pojedyncze punkty) charakteryzowały się wartościami równymi zero. Jest to naturalne zjawisko, ponieważ współczynnik *silhouette* nie działa w przypadku pojedynczych klastrów, dlatego, że nie jest możliwe odniesienie do innych punktów w obrębie tego samego klastra lub do punktów w innych klastrach. Obecność klastrów singletonów oraz outlierów wynika z właściwości algorytmu K-means, który nie zawiera w sobie wbudowanego mechanizmu obsługującego wartości odstające, czyli geny, które różnią się od pozostałych i są trudne do przypisania do większych grup.

Na wykresie przedstawiającym rozkład klastrów K-means dla k=40 (Rysunek 3.2) można zauważyć znaczące nieproporcjonalności w rozkładzie liczby punktów danych między klastrami. Niektóre klastry były bardzo liczne, a inne składały się tylko z kilku punktów. Te nieproporcjonalności mogą być wynikiem różnorodności struktury danych wejściowych lub wpływu punktów odstających. Zwizualizowanie rozkladu genów, pozwala zidentyfikować problem. Dzięki temu możliwe jest

lepsze zrozumienie charakteru danych i ograniczenia algorytmu w obecnym ustawieniu parametrów. Nie podjęto dalszych kroków w celu eliminacji tych nieproporcjonalności, ponieważ K-means w swojej standardowej implementacji nie pozwala na ich bezpośrednią korektę. Wizualizacja liczebności klastrów dostarczyła jednak cennych informacji o strukturze danych i ich potencjalnych problemach, co może być uwzględnione w kolejnych etapach analizy.

Biblioteka pandas oraz moduł google.colab zostały wykorzystane do zapisywania wyników klastrowania w formie CSV. Przy pomocy funkcji to_csv() wyeksportowane zostały kolumny, które zawierają ID genów oraz kolumna, która przechowuje przypisane etykiety klastrów dla podziału na 40 grup. Komunikat potwierdzający pomyślny zapis wyników wraz z wskazaniem nazwy wygenerowanego pliku kończy proces.

Rysunki 3.3, 3.4 i 3.5 przedstawiają poziomy ekspresji genów w poszczególnych próbkach dla klastrów 29, 15 i 12, uzyskanych przy użyciu metody K-means (40 klastrów) w postaci heatmap. Przedstawione w pracy heatmapy, w tym dla klastrów 12, 15 i 29, stanowią wybrane przykłady wizualizacji, które najlepiej oddają różnice w strukturze i charakterystyce klastrów. Łącznie wygenerowano większą liczbę heatmap, jednak do szczegółowej analizy wybrano te, które najlepiej ilustrują różnorodność w wynikach klasteryzacji i odmienność między grupami genów. Do zwizualizowania wyników klasteryzacji genów wykorzystano biblioteki *seaborn* oraz *matplotlib*. W celu zapewnienia czytelności i reprezentatywności pominięto klastry zawierające mniej niż 50 genów. Pozwoliło to uniknąć tworzenia nieczytelnych wizualizacji. Heatmapy obrazują różnorodność ekspresji genów w ramach wybranych klastrów, a paleta kolorów *viridis* wskazuje na intensywność ekspresji (oś X reprezentuje próbki, a oś Y — geny).

Klaster 29 (Rysunek 3.3) zawiera 229 genów i charakteryzuje się najwyższą wartością współczynnika *silhouette* spośród analizowanych klastrów, ponieważ wynosi ona 0.9297. Heatmapa wskazuje na wysoce jednorodny wzorzec ekspresji genów w próbkach, z przeważającym udziałem genów o niskiej ekspresji, przedstawionych na wykresie w ciemnych odcieniach fioletu. Taki rozkład sugeruje spójność i dobrą separację tej grupy, co może świadczyć o jednolitej funkcji biologicznej genów wchodzących w skład klastra. Wysoka wartość współczynnika *silhouette* sugeruje, że klaster 29 jest dobrze odizolowany od pozostałych klastrów.

Klaster 15(Rysunek 3.4) obejmuje 446 genów, a jego współczynnik *silhouette* wynosi 0.4849. Można wywnioskować niższą spójność w porównaniu do klastra 29, lecz trzeba mieć na uwadze, że ten klaster jest większy. Na heatmapie widoczna jest większa różnorodność ekspresji genów, co wskazuje na bardziej heterogeniczną strukturę klastra. Obecność jaśniejszych i ciemniejszych obszarów może świadczyć o różnicach funkcji genów w tej grupie, co czyni ją bardziej zróżnicowaną w porównaniu z pozostałymi klastrami.

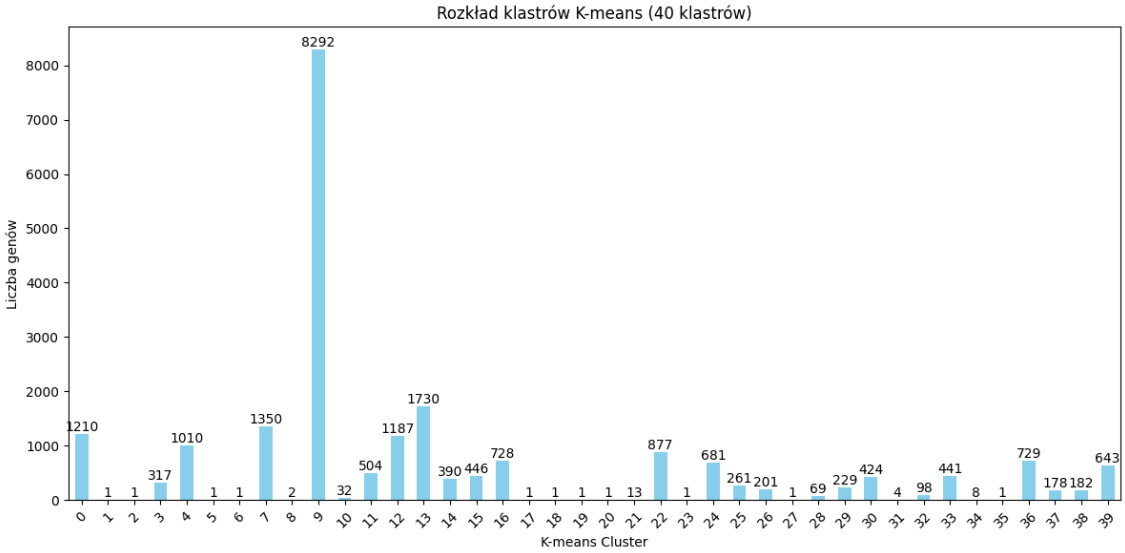
Klaster 12 (Rysunek 3.5) zawiera 1187 genów i cechuje się wysoką wartością współczynnika *silhouette* wynoszącą 0.8255. Jednak jego znaczna wielkość może sugerować, że klaster ten pełni funkcję grupy o charakterze ogólnym, do której przypisywane są geny niewykazujące wyraźnej przynależności do bardziej specyficznych klastrów. Taka sytuacja może wynikać z ograniczeń algorytmu, który nie zawsze jest w stanie precyzyjnie rozdzielić dane o bardziej złożonych relacjach. Dominują ciemne odcienie fioletu na heatmapie, które wskazują na niską ekspresję większości genów, ale widoczne są również sporadyczne jaśniejsze obszary, sugerujące wyższą ekspresję w niektórych próbkach. Takie wzorce wskazują na względną homogeniczność grupy z kilkoma wyjątkami, które mogą pełnić specyficzne role regulacyjne w wybranych próbkach.

Porównanie trzech klastrów wskazuje na istotne różnice w ich strukturze i spójności. Klaster 12 wyróżnia się największą liczebnością genów, co sugeruje, że obejmuje geny o bardziej uniwersalnej funkcji w różnych warunkach. Klaster 29 cechuje się najwyższą spójnością, co może wiązać się z specyficzną rolą biologiczną jego genów. Klaster 15 odznacza się większą różnorodnością ekspresji,

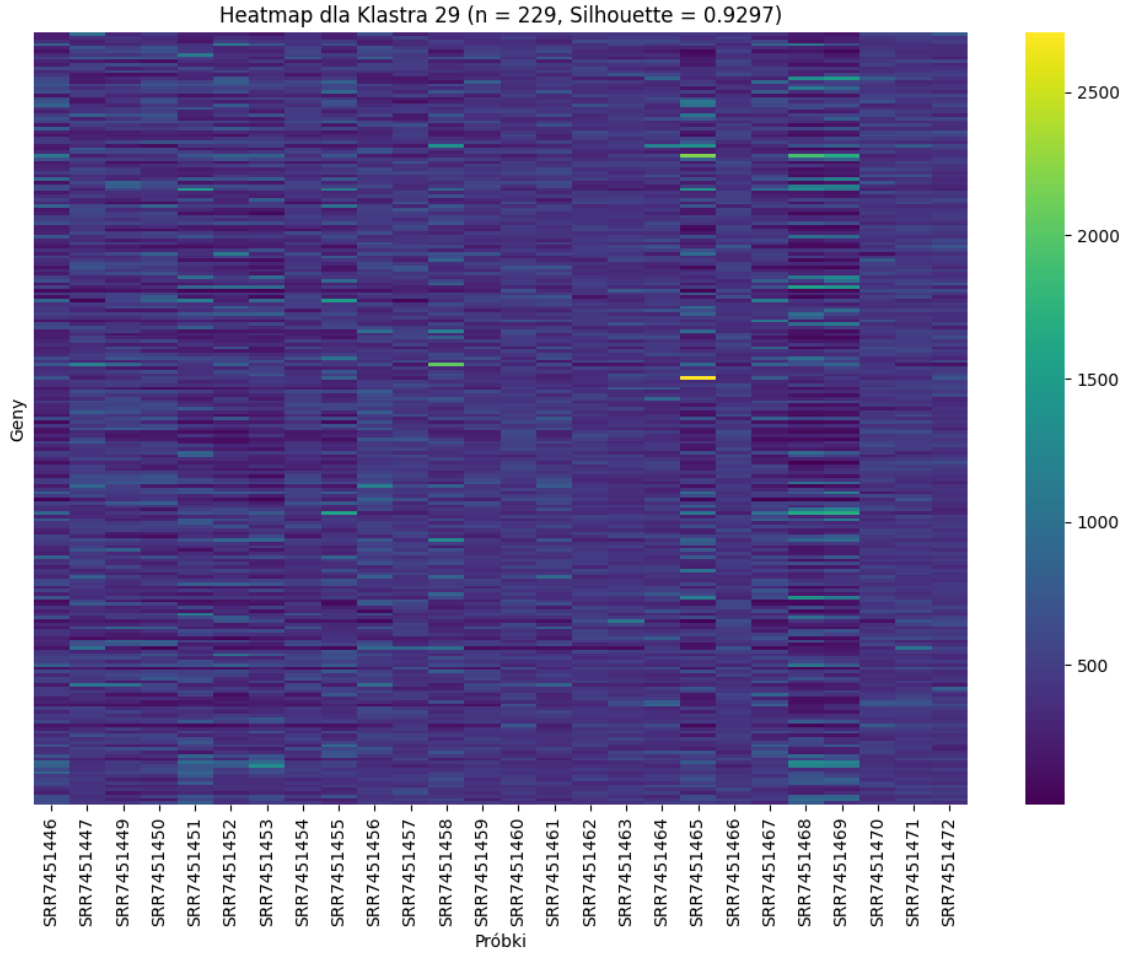
wskazuje to na bardziej zróżnicowane funkcje genów w tej grupie. Takie różnice w strukturze klastrów mogą dostarczać cennych informacji o funkcjonalnych powiązaniach między genami oraz potencjalnych procesach biologicznych, w których uczestniczą.

Cluster	**Silhouette**	**Count**
9	0.870688	8292
13	0.000000	1730
7	0.000000	1350
0	0.224439	1210
12	0.595880	1187
4	0.000000	1010
22	0.000000	877
36	0.662184	729
16	0.049585	728
24	0.973440	681
39	0.145729	643
11	0.217296	504
15	0.829501	446
33	0.913241	441
30	0.275172	424
14	0.456424	390
3	0.734065	317
25	0.000000	261
29	0.000000	229
26	0.000000	201
38	0.000000	182
37	0.114956	178
32	0.512103	98
28	0.000000	69
10	0.542251	32
21	0.873223	13
34	0.658203	8
31	0.000000	4
8	0.030916	2
17	0.936807	1
23	0.467620	1
27	0.325333	1
5	0.428609	1
1	0.695311	1
19	0.128532	1
20	0.000000	1
6	0.657840	1
2	0.406363	1
35	0.416079	1

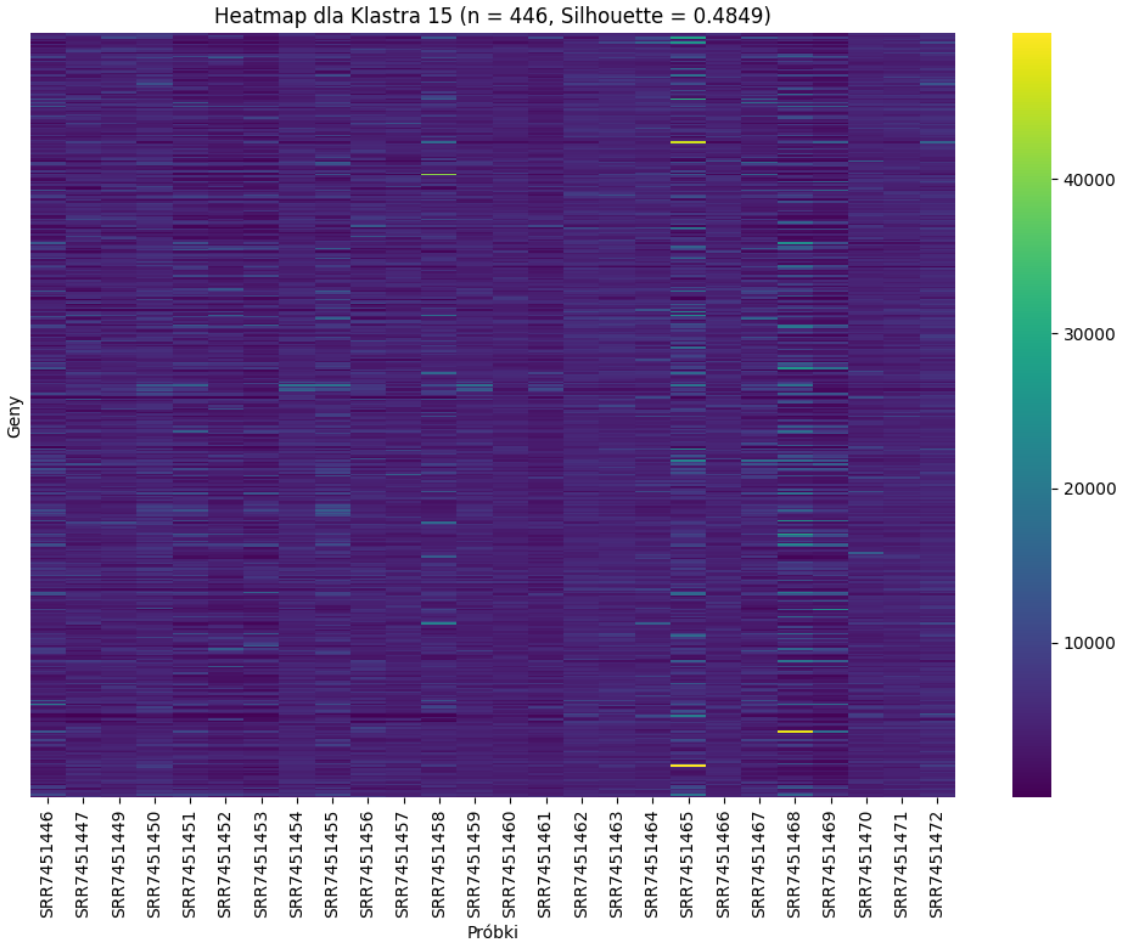
Tabela 3.4: Średni wskaźnik Silhouette i liczebność klastrów dla każdego klastra.



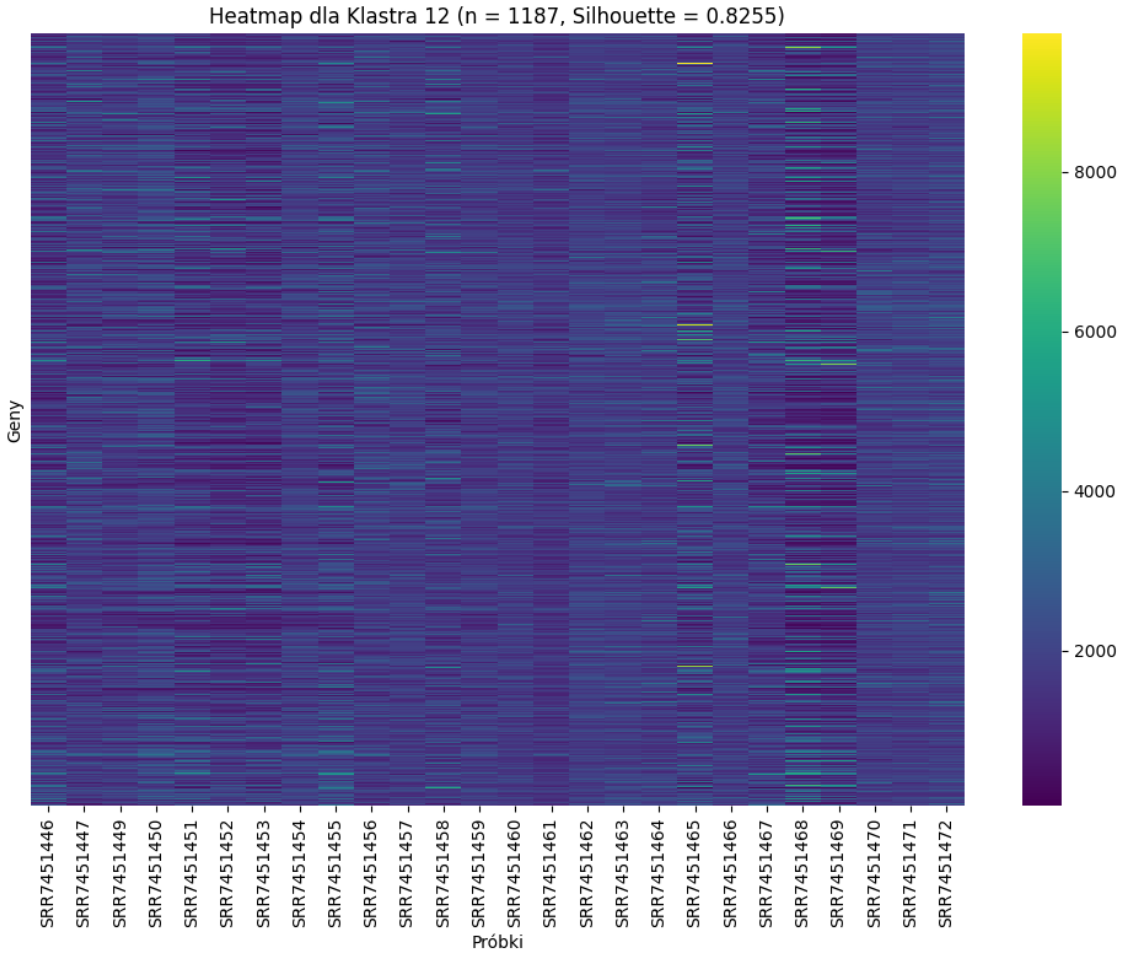
Rysunek 3.2: Rozkład klastrów K-means (40 klastrów).



Rysunek 3.3: Heatmapa dla Klastra 29 ($n = 229$, Silhouette = 0.9297).



Rysunek 3.4: Heatmapa dla Klastra 15 ($n = 446$, Silhouette = 0.4849).



Rysunek 3.5: Heatmapa dla Klastra 12 ($n = 1187$, Silhouette = 0.8255).

Algorytm Hierarchiczny

Innym algorytmem wykorzystanym do klasteryzacji genów jest algorytm hierarchiczny z biblioteki scikit-learn, który bazuje na metodzie aglomeracyjnej. Proces ten polega na łączeniu pojedynczych genów w coraz większe grupy w oparciu o miary odległości. Na początkowym etapie każdemu genowi został przypisany osobny klaster, a następnie najbardziej podobne grupy były łączone w kolejnych krokach, aż do uzyskania określonej liczby klastrów, którą w tej analizie ustawiono na 50. Liczba klastrów została dobrana jako kompromis między szczegółowością wyników a ich interpretowalnością. Pozwoliło to na wyodrębnienie wystarczającej liczby grup genów, aby uchwycić różnorodność struktury danych genomicznych, jednocześnie nie komplikując analizy nadmierną liczbą klastrów.

Przed rozpoczęciem klasteryzacji dane zostały znormalizowane, co zapewniło jednolitą skalę wartości i wyeliminowało wpływ jednostek miar na działanie algorytmu. Po zakończeniu klasteryzacji geny otrzymały etykiety klastrów, które zapisano jako nową kolumnę w tabeli danych.

Jakość klasteryzacji oceniono za pomocą współczynnika *silhouette*. W tabeli 3.5 są wyniki z tej analizy, które ujawniają odchylenia w podziale genów między klastrami. Obliczone średnie wartości współczynnika *silhouette* dla każdego klastra umożliwiają również ocenę ich jakości i jednorodności.

W kolejnym kroku przeprowadzono analizę liczebności genów w poszczególnych klastrach. Na wykresie słupkowym (Rysunek 3.6) przedstawiono wyniki tej analizy. Oś X oznacza identyfikatory

klastrów, a oś Y liczbę genów przypisanych do każdego klastra. Słupki wykresu zostały oznaczone jasno-niebieskim kolorem, a ich wysokości dodatkowo opisano liczbą genów, co ułatwia interpretację danych i umożliwia identyfikację klastrów o dużej liczebności oraz tych zawierających jedynie kilka genów.

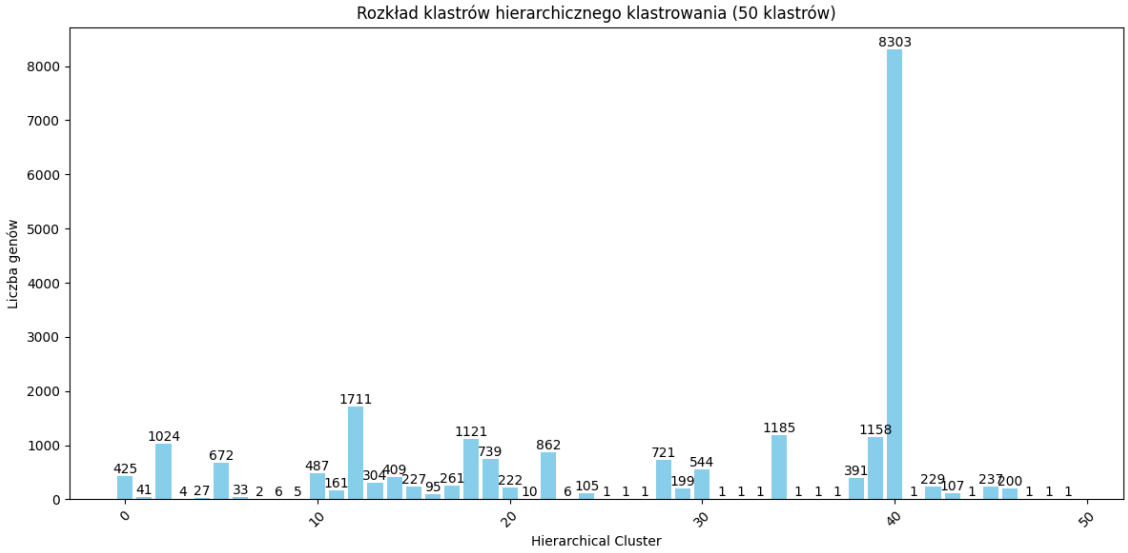
Aby lepiej zrozumieć proces grupowania genów, wygenerowano dendrogram (Rysunek 3.7), który przedstawia sposób łączenia poszczególnych genów i grup w strukturze drzewa hierarchicznego. Dendrogram ten obrazuje stopniowe łączenie obserwacji na podstawie miary odległości i umożliwia wizualizację etapów grupowania. W analizie zastosowano metodę Warda, która minimalizuje sumę kwadratów odchyleń wewnątrz klastrów, co sprzyja powstawaniu grup o niskim rozrzucie wartości.

Proces generowania dendrogramu rozpoczęto od obliczenia macierzy powiązań, która opisuje relacje między obserwacjami w zbiorze znormalizowanych danych. Macierz ta zawiera informacje o odległościach między punktami oraz kolejności ich łączenia w hierarchicznym procesie grupowania. Na osi X dendrogramu znajdują się identyfikatory klastrów w końcowych etapach łączenia, natomiast oś Y przedstawia wartości odległości, czyli różnice między grupami w momencie ich scalenia. Wyższe wartości na osi Y oznaczają bardziej odległe grupy przed ich połączeniem, co odzwierciedla ich różnorodność.

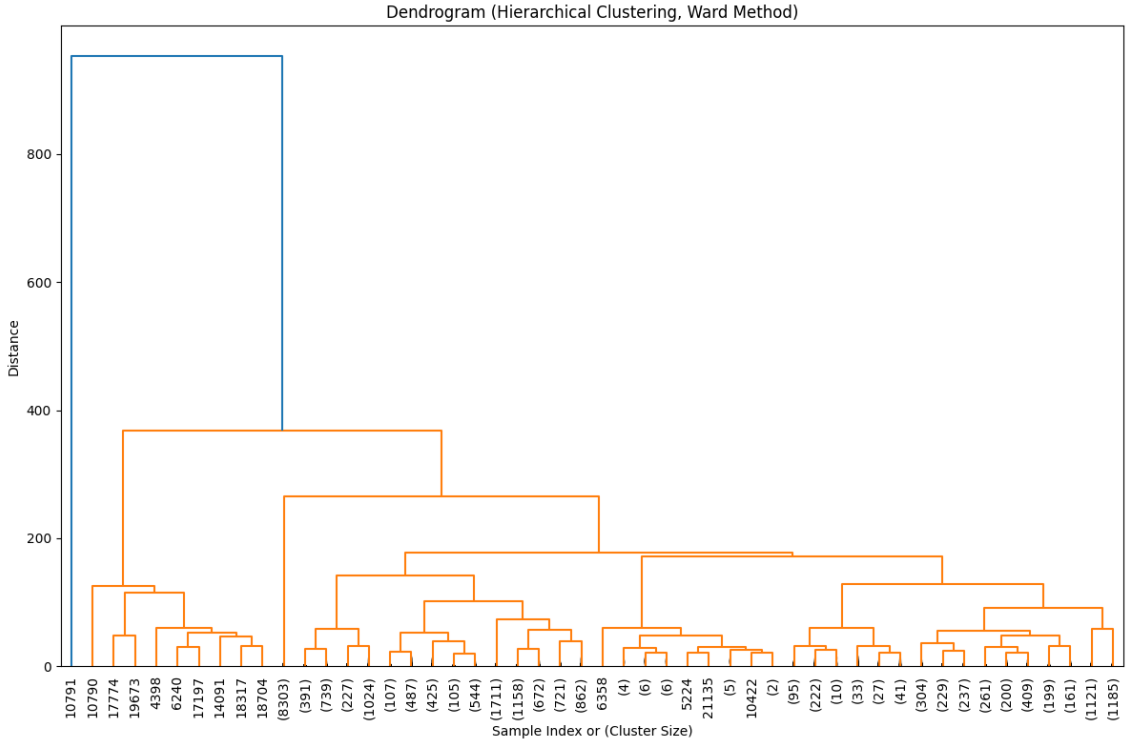
Wyniki klasteryzacji hierarchicznej, przedstawione w Tabeli 3.5, dostarczają cennych informacji o strukturze danych genomicznych. Wygenerowany dendrogram (Rysunek 3.7) pozwala zidentyfikować kluczowe podobieństwa i różnice między grupami genów, co może być istotne dla dalszych analiz biologicznych.

Cluster	Silhouette	Count
0	0.207122	425
1	0.180060	41
2	0.574123	1024
3	0.274367	4
4	0.241017	27
5	0.386914	672
6	0.102919	33
7	0.124069	2
8	0.187148	6
9	0.244634	5
10	0.130055	487
11	0.496422	161
12	0.921726	1711
13	0.379643	304
14	0.457928	409
15	0.948001	227
16	0.470340	95
17	0.871161	261
18	0.856213	1121
19	0.708074	739
20	0.244139	222
21	-0.020257	10
22	0.511339	862
23	0.214778	6
24	0.709795	105
25	0.000000	1
28	0.671125	721
29	0.665333	199
30	0.702598	544
31	0.000000	1
32	0.000000	1
33	0.000000	1
34	0.830329	1185
35	0.000000	1
36	0.000000	1
37	0.000000	1
38	0.856379	391
39	0.899778	1158
40	0.971001	8303
41	0.000000	1
42	0.891354	229
43	0.554428	107
44	0.000000	1
45	0.536799	237
46	0.310111	200
47	0.000000	1
48	0.000000	1
49	0.000000	1

Tabela 3.5: Średni wskaźnik Silhouette i liczebność klastrów dla Agglomerative Clustering (50 klastrów).



Rysunek 3.6: Rozkład klastrów hierarchicznego klastrowania (50 klastrów).



Rysunek 3.7: Dendrogram dla hierarchicznego klastrowania (metoda Warda).

Algorytm DBSCAN

Algorytm DBSCAN (Density-Based Spatial Clustering of Applications with Noise) dostępny w bibliotece scikit-learn to metoda klasteryzacji oparta na gęstości punktów, która umożliwia wykrywanie grup o nieregularnych kształtach oraz identyfikację obserwacji uznanych za szumy (outliers). Główne parametry algorytmu to *eps*, który określa maksymalną odległość między punktami uzna-

wanymi za sąsiadujące, oraz *min_samples*, definiujący minimalną liczbę punktów w sąsiedztwie wymaganą do utworzenia klastra. W analizie przyjęto wartości $eps = 0.5$ oraz $min\_samples = 10$. Parametry te zostały dobrane na podstawie wielokrotnych prób z różnymi konfiguracjami, a wybrane ustawienia zapewniły najlepsze wyniki pod względem identyfikacji klastrów oraz rozróżniania punktów szumu. Dane zostały uprzednio znormalizowane, co zapewniło jednolitą skalę wartości i pozwoliło algorytmowi na prawidłowe dopasowanie do danych. Wyniki klasteryzacji zapisano jako dodatkową kolumnę w przefiltrowanym zbiorze danych.

Wyniki zostały przedstawione w Tabeli 3.6, która zawiera liczebność poszczególnych klastrów oraz liczbę punktów oznaczonych jako szumy (wartość -1 została wyróżniona pogrubieniem). Największy klaster oznaczony numerem 0 obejmował 8950 genów. Znaczna lliczebność może wskazywać na grupę genów o wspólnych, ale niespecyficznych cechach, które nie zostały jednoznacznie przypisane do mniejszych, bardziej jednorodnych klastrów. Najmniejsze klastry, czyli klaster 28 (12 genów) oraz klaster 25 (7 genów), wskazują na bardziej niszowe grupy genów, które mogą być biologicznie istotne, ale są rzadziej reprezentowane. Punkty oznaczone jako szumy stanowiły 674 przypadki, co stanowi istotną część analizy, ponieważ mogą reprezentować nietypowe wartości ekspresji genów i potencjalnie zakłócają proces klasteryzacji.

Algorytm DBSCAN umożliwia obliczenie średniego współczynnika *silhouette* jednakże, w przypadkach, gdy większość punktów zostanie oznaczona jako szumy lub algorytm utworzy jedynie pojedynczy klaster, obliczenie współczynnika *silhouette* jest niemożliwe. W tej analizie DBSCAN nie wygenerował stabilnych klastrów przy przyjętych parametrach, co zostało uwzględnione w opisie wyników

Cluster	Count
0	8950
4	2223
3	1472
6	1189
9	1148
2	886
1	811
7	720
-1	**674**
5	620
8	498
17	393
12	346
18	262
10	261
19	229
20	226
22	195
13	194
14	160
16	157
21	127
24	126
15	125
11	105
23	67
26	50
27	14
28	12
25	7
Nie można obliczyć wskaźnika Silhouette dla DBSCAN (za mało klastrów lub obecność outlierów).	

Tabela 3.6: Ilość genów w każdym klastrze dla DBSCAN. Pogrubiona wartość −1 wskazuje outliery.

3.3 Implementacja algorytmów klastrowania w R

Implementacja programu rozpoczyna się od weryfikacji, czy niezbędne pakiety są dostępne w systemie. W tym celu stosuję się funkcję *requireNamespace()*, która sprawdza obecność pakietu, a jeśli go brakuje, automatycznie instaluje go za pomocą *install.packages()* dla standardowych pakietów R lub *BiocManager::install()* dla pakietów z Bioconductor. Dzięki temu możliwe staję się unikanie każdorazowej, ręcznej instalacji i zapewnia się spójność środowiska analitycznego. Pakiety wykorzystywane w analizie można podzielić na kilka głównych kategorii. Pierwszą z nich są pakiety służące do przetwarzania i manipulacji danymi, takie jak *dplyr*, który dostarcza funkcje do filtrowania, grupowania i transformacji danych, oraz *readxl*, umożliwiający importowanie danych z plików Excela. Drugą grupą są pakiety odpowiedzialne za analizę ekspresji genów i przetwarzanie danych genomowych, do których należą WGCNA, służący do analizy współekspresji genów i budowy sieci korelacyjnych, oraz preprocessCore, zapewniający funkcje normalizacji danych, co jest kluczowe w badaniach genomowych. Kolejną kategorią są pakiety wspierające analizę funkcjonalną genów, takie jak *clusterProfiler*, który umożliwia przeprowadzanie analizy wzbogacenia funkcjonalnego genów, *GO.db*, zawierający bazę danych Gene Ontology, oraz *org.Hs.eg.db*, dostarczający

informacje na temat genów ludzkich, co pozwala na mapowanie i konwersję identyfikatorów genów. Do tej grupy należy również *biomaRt*, będący interfejsem do BioMart, umożliwiającym pobieranie informacji o genach z baz danych biologicznych. Istotną rolę pełni także enrichplot, który dostarcza narzędzia do wizualizacji wyników analizy wzbogacenia. Kolejna kategoria obejmuje pakiety do analizy klasteryzacji i wizualizacji danych. Pakiet *cluster* dostarcza algorytmy klasteryzacji, co umożliwia grupowanie genów na podstawie ich ekspresji, *pheatmap* pozwala na tworzenie map cieplnych z szerokimi możliwościami dostosowywania ich wyglądu, a *gplots* oferuje dodatkowe funkcje graficzne przydatne w analizie danych. Przykładowo *ggplot2* stanowi jedno z najpowszechniejszych narzędzi do tworzenia wykresów, które pozwala na kompleksową prezentację wyników analiz. W ramach przygotowania środowiska po instalacji pakiety zostały załadowane do przestrzeni roboczej za pomocą funkcji *library()*. Dzięki temu wszystkie funkcje i narzędzia stają się dostępne do dalszej analizy. Pakiety zostały załadowane w kolejności zgodnej z ich zastosowaniem w analizie: najpierw pakiety do manipulacji danymi i wczytywania plików, następnie pakiety do wizualizacji danych, dalej pakiety do analizy klasteryzacji i budowy sieci genowych, a na końcu narzędzia do analizy wzbogacenia i mapowania genów.

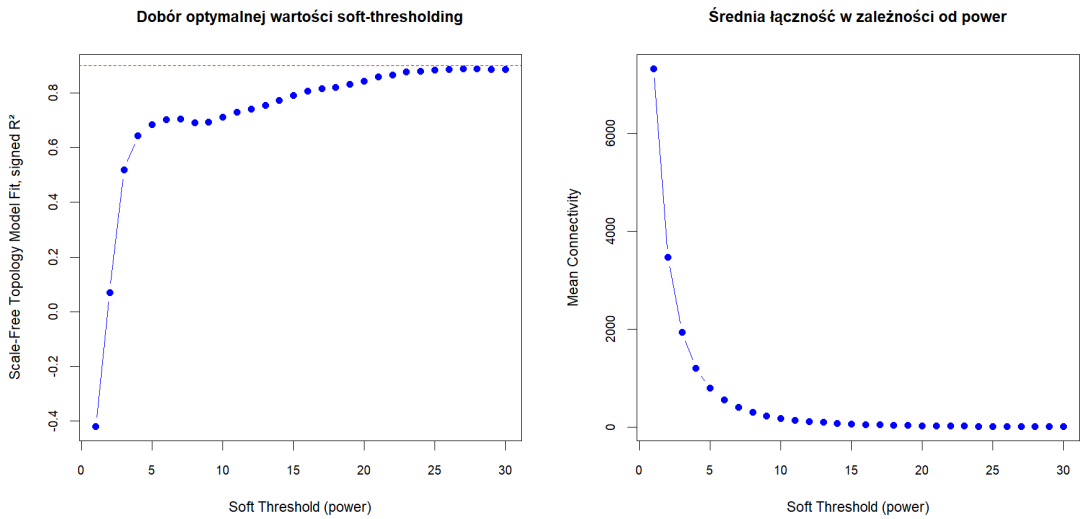
Dane wejściowe zapisywane są do zmiennej *summary_htseq* przy użyciu funkcji *read_excel()*, która umożliwia wczytanie zestawu ekspresji genów wraz z metadanymi próbek. Po zakończeniu etapu preprocessingu (opisanego w rozdziale 3.1), tworzona jest macierz ekspresji genów *gene_expression_matrix*, która zawiera wartości ekspresji dla wybranych próbek biologicznych. Konwersja do formatu ramki danych zapewnia kompatybilność z dalszymi etapami analizy, a identyfikatory genów z kolumny *Name* są przypisywane do wierszy, umożliwiając jednoznaczną identyfikację każdej jednostki.

Aby zapobiec błędom w dalszej analizie, zaimplementowana zostaję weryfikacja struktury danych, która sprawdza obecność kolumny *Name* w zbiorze *filtered_genes*. W przypadku jej braku, skrypt generuje komunikat błędu i przerywa działanie za pomocą funkcji *stop()*. Takie podejście minimalizuje ryzyko błędów wynikających z nieoczekiwanych zmian w danych wejściowych.

Analiza ekspresji genów opiera się na identyfikacji współekspresyjnych modułów genowych za pomocą pakietu WGCNA. Po przypisaniu nazw genów do wierszy w obiekcie *filtered_genes*, sprawdzona zostaje ich unikalność. Jeśli w zbiorze występują duplikaty, funkcja *make.unique()* automatycznie nadaje im odpowiednie oznaczenia, aby uniknąć konfliktów w dalszych etapach analizy. Następnie skonstruowana zostaje macierz ekspresji genów *datExpr* poprzez transpozycję, która pozwala na uzyskanie struktury wymaganej przez pakiet WGCNA, czyli gdzie wiersze odpowiadają próbkom, a kolumny reprezentują geny.

Kolejnym etapem jest kontrola jakości danych, obejmująca weryfikację jakości genów oraz próbek przy użyciu funkcji *goodSamplesGenes()*. Taka analiza pozwola na dodatkową identyfikację genów oraz próbek zawierających zbyt wiele brakujących wartości lub wykazujących nietypowe wzorce ekspresji. W przypadku wykrycia problematycznych próbek lub genów, dochodzi do ich usunięcia z dalszej analizy. W tym przypadku, żadne geny ani próbki nie zostały usunięte. Świadczy to o fakcie, że dobrane parametry filtracji preprocessingowej okazały się wystarczające do wyizolowania kluczowych genów.

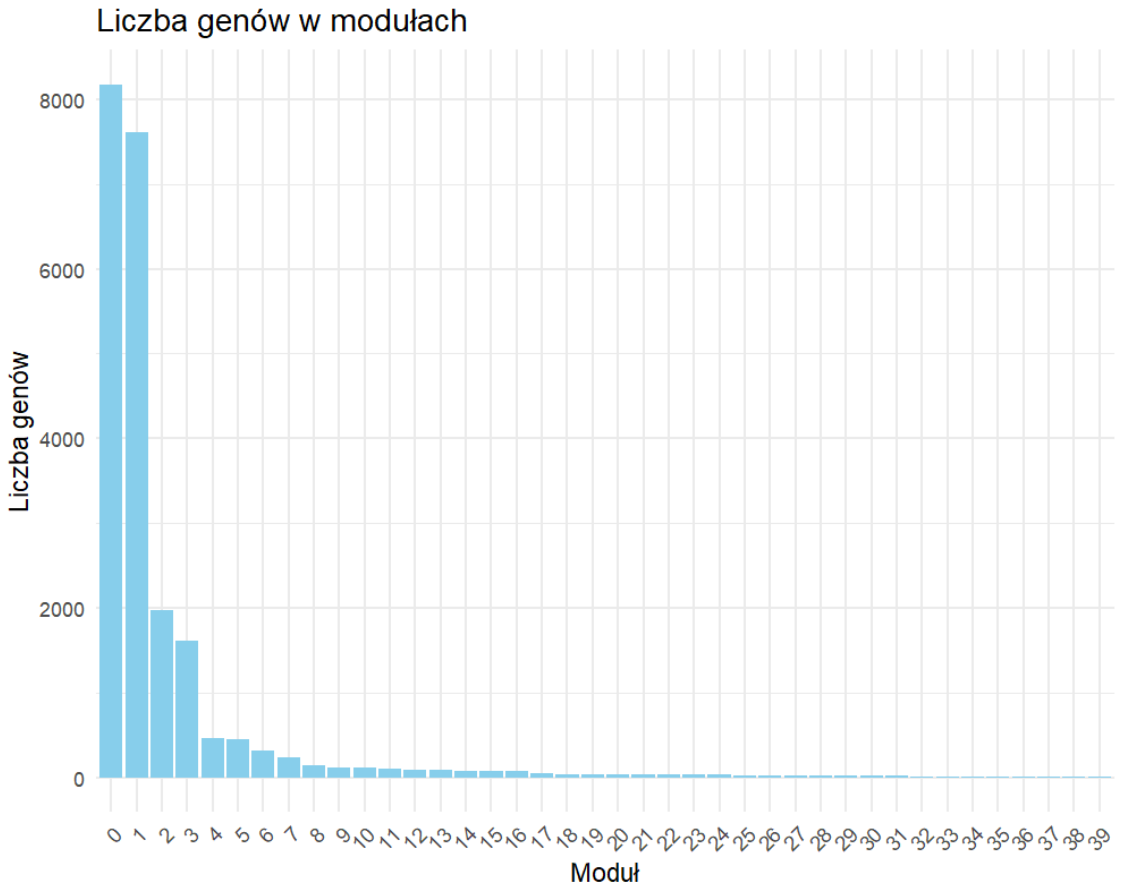
Następnie przeprowadzona zostaje procedura wyboru optymalnej wartości *soft-thresholding power*, która ma kluczowe znaczenie w konstruowaniu sieci współekspresji genów. Określenie tej wartości odbywa się za pomocą funkcji *pickSoftThreshold()*, która testuje różne wartości parametru i ocenia ich wpływ na skalowalność sieci. W wyniku analizy wybierana jest wartość, która zapewnia najlepszą zgodność z założeniami modelu, a jej wynik zostaje zapisany w zmiennej softPower.



Rysunek 3.8: Dobór optymalnej wartości parametru soft-thresholding w analizie współekspresji genów

Dane procedury wyboru optymalnej wartości soft-thresholding power są wizualizowane na rysunku 3.8, który przedstawia wykresy zależność współczynnika Scale-Free Topology Model Fit (*signed* R^2) oraz średniego stopnia łączności od wartości parametru power. Na pierwszym wykresie z lewej, obserwujemy, że wraz ze wzrostem wartości power współczynnik R^2 rośnie, stabilizując się na poziomie powyżej 0.8, co wskazuje na zgodność sieci z modelem skali swobodnej. Jednocześnie drugi wykres ukazuje, że średnia łączność sieci maleje wraz ze wzrostem wartości power, co oznacza ograniczenie nadmiernych połączeń między genami. Optymalna wartość *softPower* zostaje wybrana na podstawie równowagi między wysokim dopasowaniem do modelu skali swobodnej a zachowaniem odpowiedniej struktury sieci. Po wyznaczeniu optymalnej wartości *soft-thresholding power* równej 21 następuje budowa sieci współekspresji genów metodą analizy blokowej (*blockwiseModules*). W celu zwiększenia wydajności obliczeniowej analiza przeprowadzana jest w blokach o maksymalnym rozmiarze określonym przez zmienną *maxBlockSize*, w tym przypadku 556. Algorytm tworzy moduły genowe na podstawie wyznaczonej wcześniej wartości *softPower*, stosując metodę niezorientowanej macierzy topologicznej (*TOMType = "unsigned"*). Minimalny rozmiar modułu ustalony zostaje na 10 genów. Oznacza to, że mniejsze grupy nie są uznawane za odrębne moduły. Dodatkowo algorytm scala moduły, których odległość nie przekracza progu (*mergeCutHeight = 0.25*). Ostatecznie przypisane kolory modułów oraz ich eigengeny są przechowywane w obiektach *moduleColors_blockwise* oraz *MEs_blockwise* Wyniki analizy są zapisywane do pliku *blockwiseModules_results.RData*, dzięki czemu mogą one zostać później wykorzystane bez konieczności ponownego przeprowadzania obliczeń. W wyniku analizy powstaje czterdzieści klastrów, ponumerowanych od 0 do 39. Liczba genów przypisanych do każdego klastra znajduję się Tabeli 3.7, w kolumnie *Gene Count*.

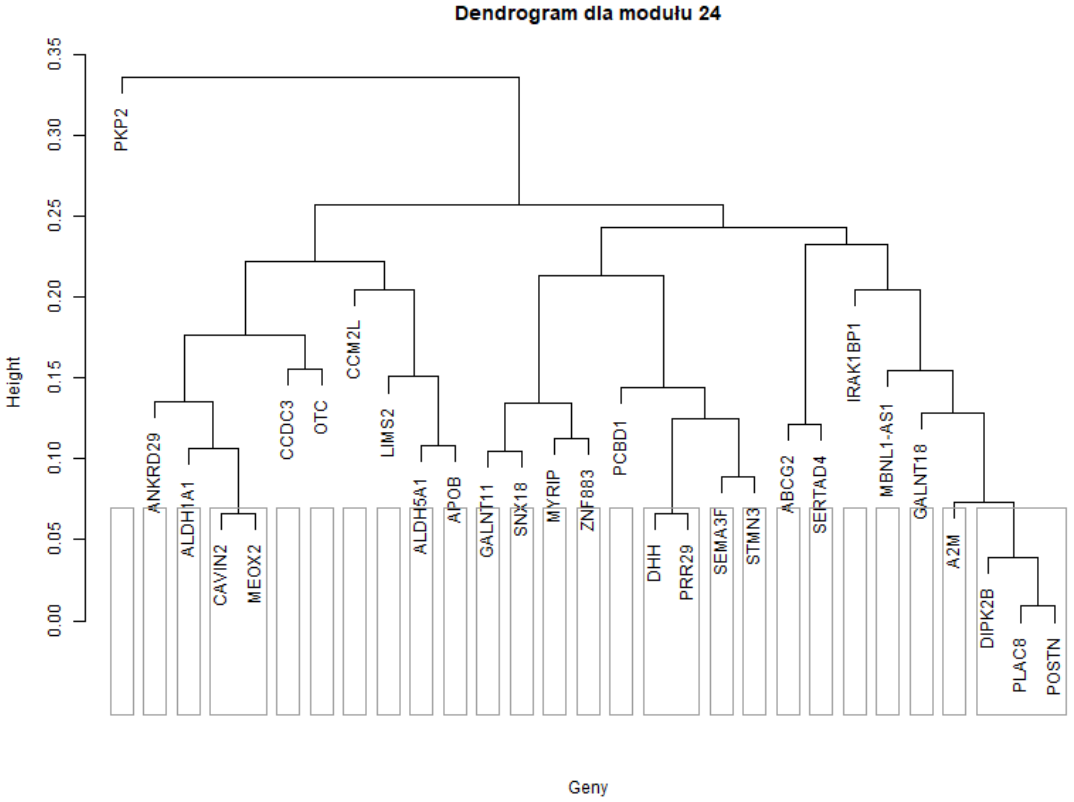
W celu dalszej analizy wyników budowy sieci współekspresji genów przeprowadzana jest wizualizacja danych w postaci wykresów słupkowych oraz map cieplnych. Pierwszym krokiem jest utworzenie wykresu przedstawiającego liczebność genów w poszczególnych modułach (Rysunek 3.9) za pomocą funkcji *ggplot()* z pakietu *ggplot2*. Na osi X znajdują się poszczególne moduły, a na osi Y liczba przypisanych do nich genów. Słupki reprezentujące wielkość modułów są wypełnione kolorem *skyblue*, a całość wykresu sformatowana jest w minimalistycznym stylu *theme_minimal()*.



Rysunek 3.9: Wykres słupkowy rozkładu genów w klastrach dla WGCNA

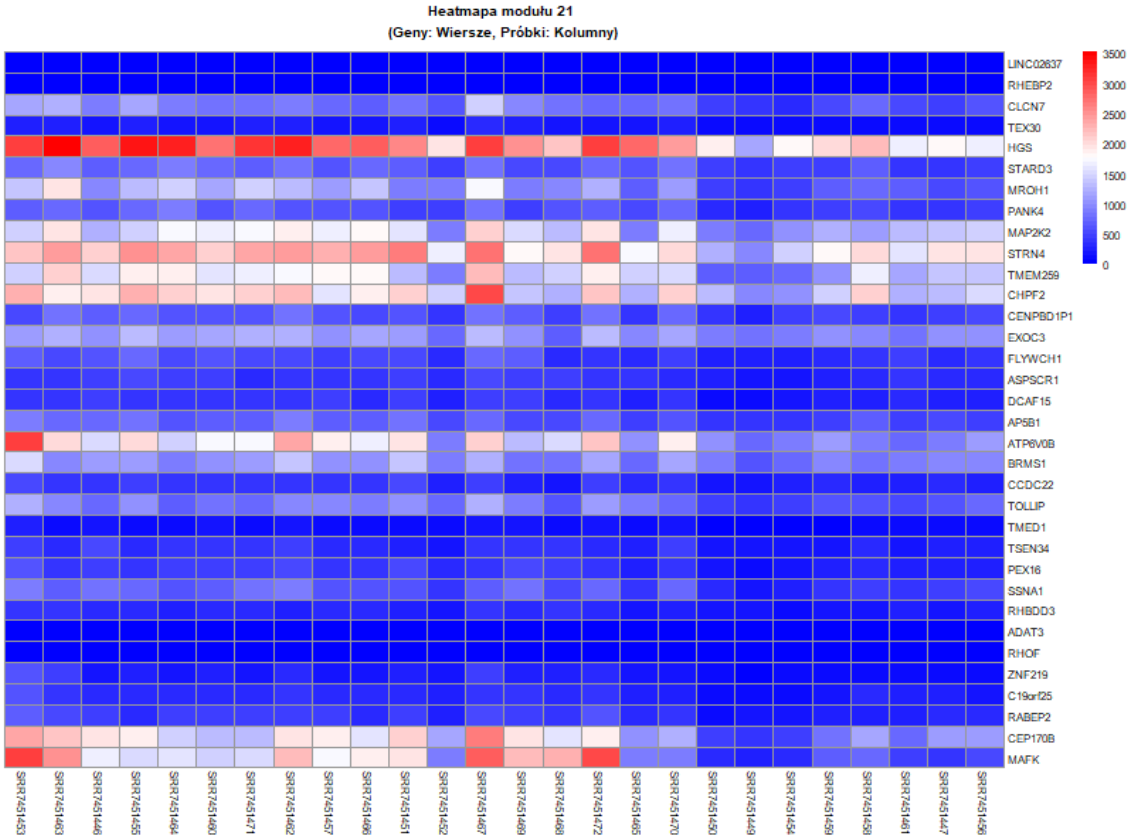
Następnym etapem analizy jest generowanie map cieplnych, które umożliwiają szczegółową wizualizację wzorców ekspresji genów w ramach poszczególnych modułów. W tym celu tworzona jest funkcja *generateHeatmapForModules()*, która tworzy mapy cieplne dla każdego modułu oddzielnie. Na początku funkcja identyfikuje geny przypisane do danego modułu, sprawdzając ich obecność w zmiennej *moduleColors*. Jeśli w danym module znajduje się mniej niż dwa geny, proces generowania mapy cieplnej zostaje pominięty, ponieważ minimalna liczba wymaganych elementów do analizy jest niewystarczająca. Kolejnym krokiem jest sprawdzenie, czy wszystkie geny przypisane do danego modułu faktycznie występują w danych ekspresji. W przypadku brakujących genów skrypt wyświetla odpowiedni komunikat ostrzegawczy, a dalsze kroki dla tego modułu zostają przerwane.

Po upewnieniu się, że dane są kompletne, następuje ekstrakcja wartości ekspresji genów w danym module. Dane te są następnie transponowane, tak aby geny znajdowały się w wierszach, a próbki w kolumnach, co jest standardowym formatem dla wizualizacji w postaci heatmapy. Kolejnym krokiem jest obliczenie macierzy odległości dla genów i próbek, aby umożliwić późniejszą hierarchiczną klasteryzację. Odległości w macierzy wyznaczane są na podstawie korelacji między profilami ekspresji genów (*1 - cor(t(module_data))* dla genów i *1 - cor(module_data)* dla próbek), a następnie wykorzystywane do konstrukcji dendrogramów metodą średnich połączeń (*hclust(method = "average")*). Wynikowe dendrogramy umożliwiają uporządkowanie genów i próbek w taki sposób, aby ujawnić podobieństwa w ich wzorcach ekspresji. Przykład wynikowego dendogramu na rysunku 3.10.

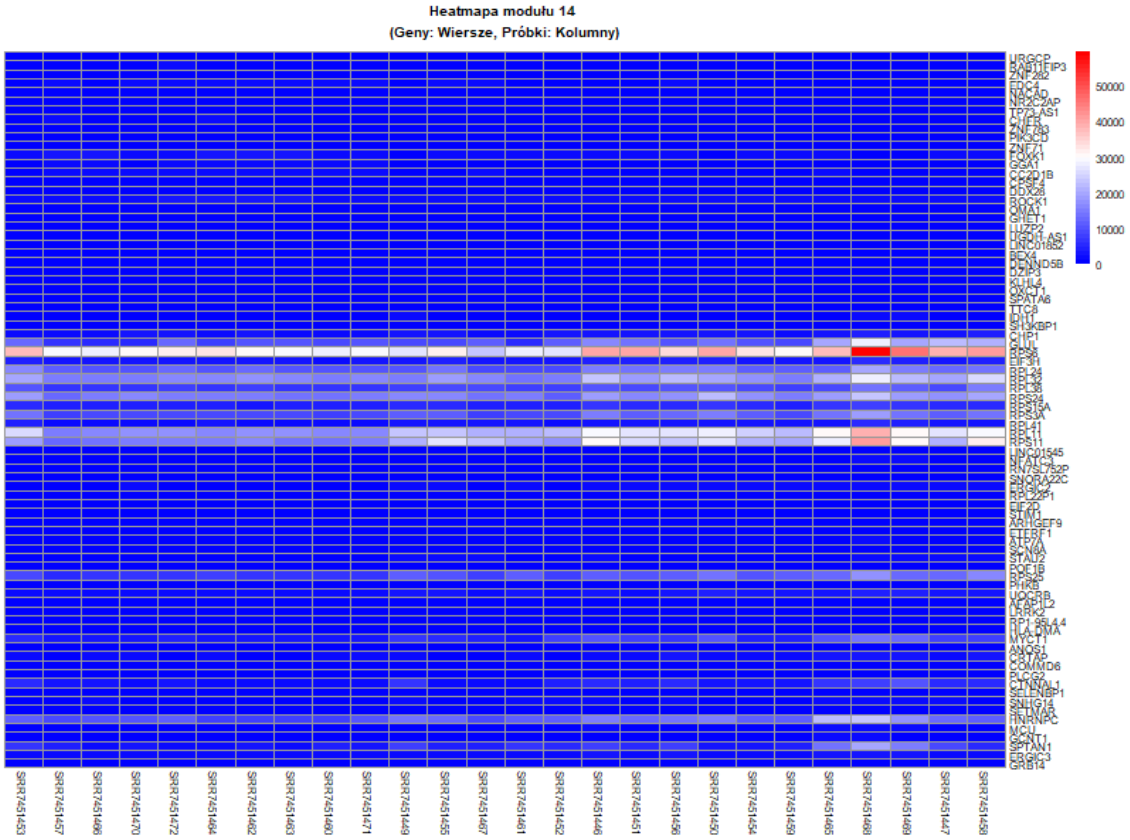


Rysunek 3.10: Dendogram dla modułu: 24.

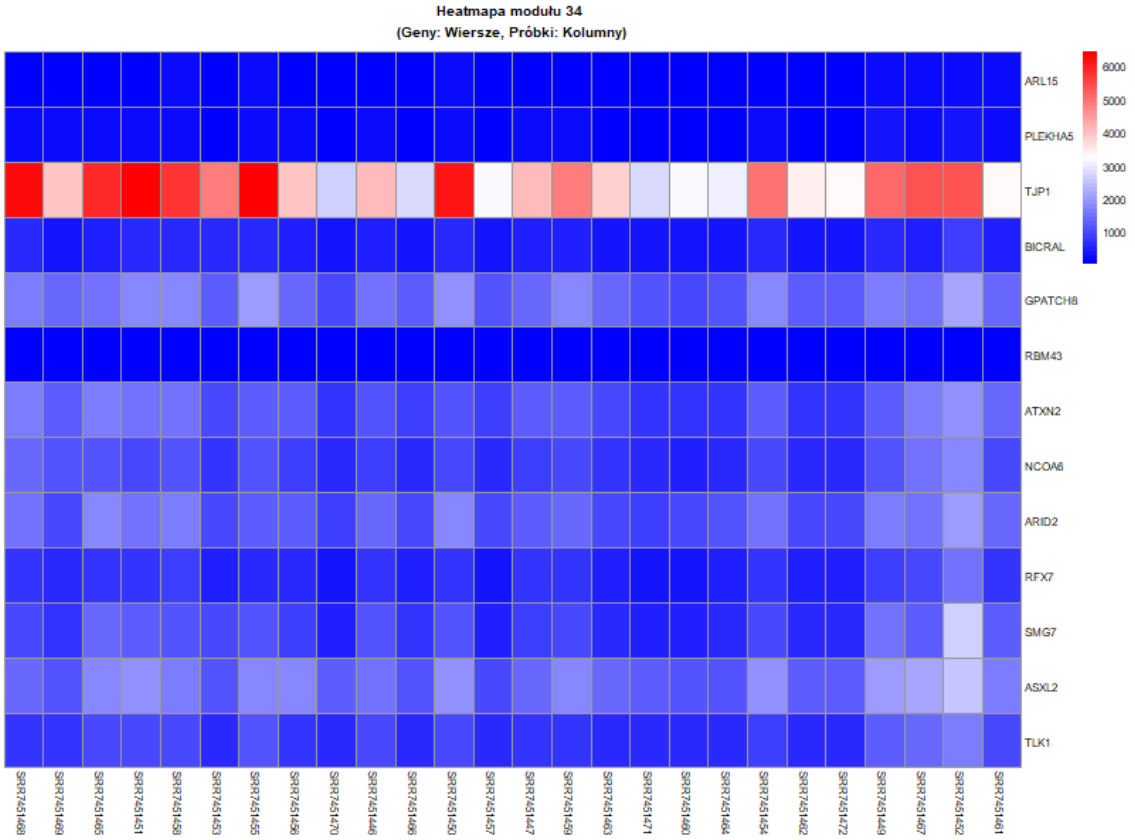
Posortowane dane są następnie wykorzystywane do wygenerowania map cieplnych za pomocą funkcji *pheatmap()*, przykładowa heatmapa dla modułu o numerze 21 znajduję się na rysunku 3.11. Kolory na mapie przechodzą od niebieskiego (niska ekspresja), przez biały (średnia ekspresja), aż do czerwonego (wysoka ekspresja), co pozwala na intuicyjne rozróżnienie poziomów ekspresji między próbkami.W celu wygenerowania heatmap dla wszystkich modułów, w skrypcie znajduje się pętla iterująca przez wszystkie unikalne moduły zidentyfikowane w analizie. Każdy moduł jest analizowany oddzielnie. Wygenerowana mapa cieplna jest zapisywana do pliku graficznego PNG, którego nazwa zawiera numer modułu, co pozwala na późniejszą łatwą identyfikację wyników. Dodatkowo, mapa cieplna jest również wyświetlana bezpośrednio w konsoli, co umożliwia natychmiastową inspekcję wyników.



Rysunek 3.11: Heatmapa dla modułu o numerze 21.



Rysunek 3.12: Heatmapa dla modułu o numerze 14.



Rysunek 3.13: Heatmapa dla modułu o numerze 34.

Heatmapy przedstawione na rysunkach 3.11, 3.12, 3.13 są wynikiem iteracyjnej analizy modułów przeprowadzonej w ramach skryptu, w którym każdy moduł został poddany osobnej wizualizacji i zapisany w formie graficznej. Wizualizacje te umożliwiają szczegółową eksplorację struktury modułów oraz identyfikację potencjalnych biologicznie istotnych wzorców współekspresji genów.

Rysunek 3.11 przedstawia heatmapę modułu 21, w którym zaobserwowano zróżnicowaną ekspresję genów, z wyraźnie aktywnymi regionami wskazującymi na ich potencjalne funkcjonalne powiązania. Rysunek 3.12 ilustruje moduł 14, charakteryzujący się bardziej jednolitym wzorcem ekspresji, z pojedynczymi obszarami o podwyższonej aktywności. Z kolei rysunek 3.13 przedstawia moduł 34, gdzie można dostrzec wyraźnie wyróżniające się geny o wysokiej ekspresji w określonych próbkach.

Analiza uzyskanych heatmap pozwala na ocenę spójności ekspresji w poszczególnych modułach oraz identyfikację genów kluczowych dla badanych procesów biologicznych.

Wyżej opisany etap analizy dostarcza wartościowych informacji na temat organizacji modułów współekspresyjnych. Wykresy słupkowe pozwalają na ogólną ocenę liczebności modułów, natomiast mapy cieplne ujawniają bardziej szczegółowe relacje między genami i próbkami, co może być przydatne w kontekście identyfikacji potencjalnych biomarkerów lub kluczowych ścieżek regulacyjnych w badanym układzie biologicznym.

Aby ocenić jakości klasteryzacji modułów genowych przeprowadzane jest obliczanie współczynnika Silhouette (wyniki zamieszczone zostały w tabeli 3.7), który mierzy, jak dobrze dany gen pasuje do swojego modułu w porównaniu z innymi modułami. Pierwszym krokiem w tym procesie jest transpozycja macierzy ekspresji genów *datExpr*, aby geny znajdowały się w wierszach, a próbki w kolumnach. Tak przekształcona macierz *datExpr_genes* jest następnie dopasowywana do ramki

modules_and_genes, zawierającej przypisania genów do modułów. W celu zapewnienia zgodności zestawów danych identyfikowane są wspólne geny poprzez *intersect()*, a także wyszukiwane są geny, które znajdują się w jednym zbiorze, ale nie występują w drugim. Lista brakujących genów jest wypisywana do konsoli, co pozwala na ewentualne wykrycie problemów z danymi wejściowymi. Następnie *datExpr_genes* oraz *modules_and_genes* są filtrowane tak, aby zawierały wyłącznie wspólne geny. Moduły genowe są konwertowane na wartości numeryczne, co umożliwia ich wykorzystanie w dalszych analizach. Aby zapobiec niespójnościom, sprawdzana jest zgodność liczby genów w *datExpr_genes* i *gene_clusters*, a w przypadku rozbieżności skrypt przerywa działanie, zwracając stosowny komunikat.

Po przygotowaniu danych obliczano macierz odległości oparta na korelacji pomiędzy genami. W tym celu wykorzystywana jest funkcja *cor()*, która wyznacza macierz korelacji dla transponowanej macierzy ekspresji genów. Wartości te są następnie konwertowane do formatu macierzy odległości poprzez zastosowanie operacji *1 - cor_matrix*, gdzie wysoka korelacja oznacza mniejszą odległość między genami, a niska korelacja wskazuje na większe różnice między ekspresjami. Tak przygotowana macierz jest podstawą do wyznaczenia współczynnika Silhouette, który obliczany jest za pomocą funkcji *silhouette()*. Uzyskane wartości pozwalają na ocenę jakości przypisania poszczególnych genów do modułów, wskazując, czy dany gen bardziej pasuje do swojego klastra, czy też powinien zostać przypisany do innego modułu.

Dla każdego modułu obliczana jest średnia wartość współczynnika Silhouette, zobrazowana w Tabeli 3.7 co pozwala na porównanie różnych modułów pod względem ich jakości. Klaster 0 ma najniższą wartość wskaźnika Silhoutte ponieważ znajdują się w nim geny które nie zostały przypisane do innych, mniejszych i bardziej specyficznych klastrów. Wartości współczynnika są grupowane według modułu i zapisane w ramce danych *silhouette_avg*, po czym eksportowane do pliku CSV `Silhouette_Scores_Genes.csv`, co umożliwia ich dalszą analizę i archiwizację wyników.

Cluster	**Silhouette**	**Gene Count**
0	-0.48283124	8172
1	-0.107501524	7615
2	-0.178297985	1975
3	-0.131610103	1614
4	0.252022302	467
5	-0.068230311	453
6	0.138019205	310
7	0.218455899	240
8	0.178731061	137
9	0.104649868	122
10	0.135397122	114
11	0.207034151	99
12	0.323699208	94
13	-0.048166833	86
14	-0.372004319	80
15	0.300841495	74
16	0.590784611	70
17	0.060484851	44
18	0.637471909	38
19	-0.157315006	36

Cluster	Silhouette	Gene Count
20	0.367247028	36
21	0.277672897	34
22	0.554137108	33
23	0.422386846	31
24	0.476534661	29
25	0.286297444	28
26	0.425554000	25
27	-0.298316343	23
28	0.527085036	21
29	0.682409446	18
30	0.600814258	18
31	0.163021592	16
32	0.628944285	14
33	0.457341612	13
34	0.458571134	13
35	0.442037268	12
36	0.414137923	12
37	0.502643319	11
38	0.595774727	10
39	0.204726630	10

Tabela 3.7: Średni wskaźnik Silhouette klastrów dla WGCNA (40 klastrów) oraz liczba genów w każdym module.

W dalszej części kodu przeprowadzane jest sprawdzenie struktury modułów oraz zapisanie informacji o przypisaniach genów do pliku *modules_and_genes.csv*. Każdy moduł jest również analizowany pod kątem liczby zawartych w nim genów, a moduły liczące mniej niż 30 genów są identyfikowane i wypisywane w konsoli, co pozwala na wychwycenie potencjalnie słabo zdefiniowanych klastrów. Analiza ta jest istotnym elementem oceny jakości klasteryzacji, ponieważ małe moduły mogą wskazywać na błędne przypisania genów lub konieczność ponownej optymalizacji parametrów analizy. Podsumowując, proces ten umożliwia ilościową ocenę jakości modułów genowych za pomocą współczynnika Silhouette, wizualizację wyników oraz identyfikację problematycznych klastrów. Uzyskane wyniki przysłużyły się dalszej optymalizacji metod analizy współekspresji genów oraz pomogły w lepszym zrozumieniu struktury badanych modułów genowych.

Rozdział 4

Ocena i porównanie wyników

4.1 Porównanie K-means z WGCNA

Pierwszym porównaniem wyników dwóch metod grupowania danych genomicznych jest porównanie K-means (z 40 klastrami) z WGCNA. Wybrana została liczba klastrów n=40 w K-means, ponieważ zapewnia odpowiedni kompromis między jakością grupowania a interpretowalnością wyników. Wyniki klastrowania genów w obu metodach zostały zestawione w tabeli (Tabela 4.1), w której geny przypisane do poszczególnych klastrów w K-means zostały porównane z ich przynależnością do modułów WGCNA. To zestawienie umożliwiło dokładne porównanie wyników obu technik na poziomie genów.

Aby zobrazować zgodność między metodami, stworzono macierz kontyngencji (Rysunek 4.1). Macierz ta przedstawia liczbę genów wspólnych między klastrami K-means a modułami WGCNA, co pozwala na ocenę stopnia nakładania się wyników oraz różnic w grupowaniu. W szczególności narzędzie to umożliwia identyfikację głównych grup genów o wysokim podobieństwie oraz różnic w mniejszych klastrach.

Wyniki wskazują, że największe podobieństwo między metodami występuje w dużych klastrach. Przykładowo, klaster 9 w K-means, który zawiera 8292 geny, ma 4591 wspólnych genów z modułem 1 WGCNA, obejmującym 7615 genów. Klaster 13 w K-means (1730 genów) ma spore nakładanie się z modułem 0 WGCNA (8172 geny), z 713 wspólnymi genami. Moduł 0 w WGCNA charakteryzuje się zbieraniem genów, które nie wykazują wyraźnych wzorców współekspresji, co powoduje różnice w strukturze grupowania między metodami.

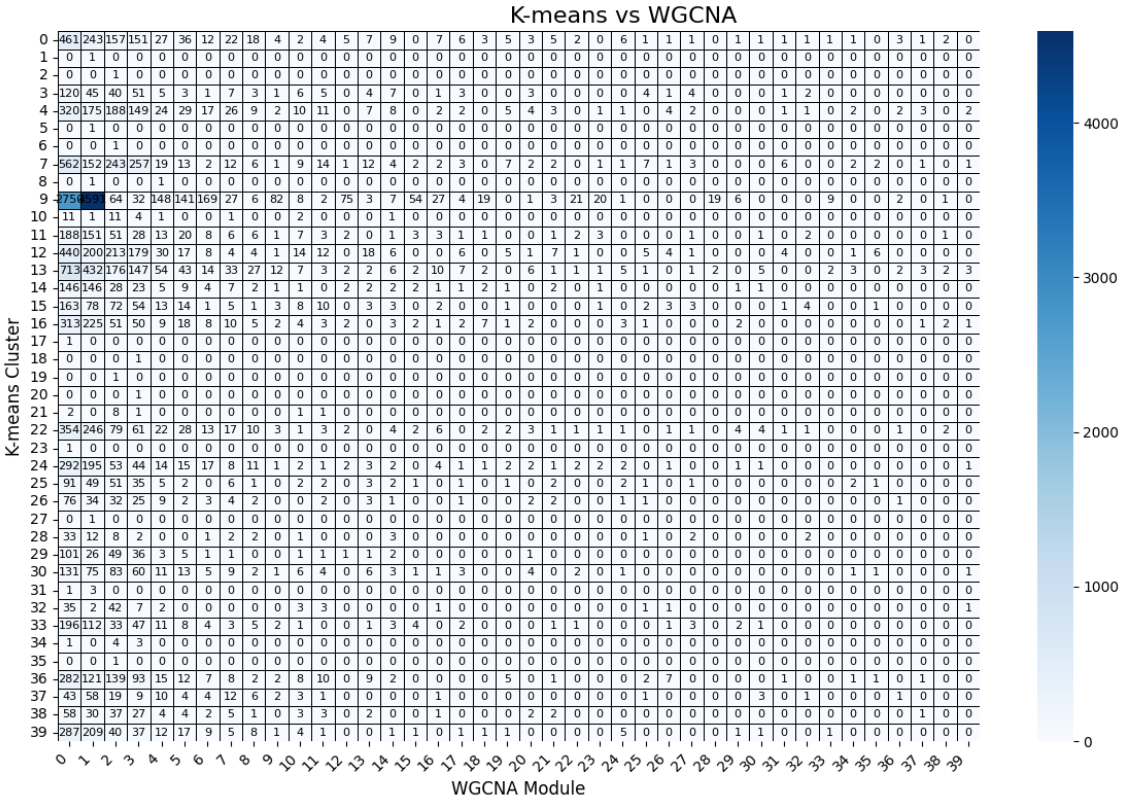
Jednocześnie analiza wykazuje, że wiele mniejszych klastrów, czyli np. klaster 31 w K-means (4 geny), nie znajduje odpowiedników w modułach WGCNA. Wiele komórek macierzy ma wartość zero, co wskazuje na brak wspólnych genów między klastrami i modułami. Wynika to z różnic w miarach podobieństwa stosowanych przez obie metody. K-means opiera się na odległości euklidesowej, podczas gdy WGCNA wykorzystuje korelację współekspresji genów, przez co prowadzi to do rozbieżności w przypisaniach genów, zwłaszcza w przypadku mniejszych grup.

Tabela 4.1 przedstawia szczegółowe zestawienie najlepszych dopasowań klastrów K-means do modułów WGCNA, mając na uwadze liczbę wspólnych genów w każdej parze. Przykładowo, klaster 7 w K-means (1350 genów) jest dopasowany do modułu 0 w WGCNA, z 562 wspólnymi genami. Porównania podkreślają podobieństwa oraz różnice w liczebności oraz strukturze grup genów w obu metodach.

Do oceny globalnej zgodności wyników zastosowano Adjusted Rand Index (ARI), którego wartość wynosi 0.1187. Wskazuje on na niską zgodność między metodami, jest to uwarunkowane ich odmiennymi założenia i podejściami. K-means optymalizuje podział danych w przestrzeni wielo-

wymiarowej, minimalizując odległości w ramach klastrów, natomiast WGCNA grupuje geny na podstawie współekspresji, uwzględniając biologiczne relacje między nimi.

Wnioski z analizy macierzy kontyngencji, tabeli porównawczej oraz wartości ARI sugerują, że metody te mogą być komplementarne. K-means umożliwia tworzenie jednorodnych grup, które jest korzystne w przypadku analizy przestrzeni wielowymiarowej, natomiast WGCNA lepiej uwzględnia biologiczne powiązania między genami. Obie metody mają różne zastosowania, w zależności od celu analizy genomicznej.



Rysunek 4.1: Macierz kontyngencji: K-means vs WGCNA.

Klaster K-means	Liczba genów	Moduł WGCNA	Liczba genów	Wspólne geny
9	8292	1	7615	4591
13	1730	0	8172	713
7	1350	0	8172	562
0	1210	0	8172	461
12	1187	0	8172	440
22	877	0	8172	354
4	1010	0	8172	320
16	728	0	8172	313
24	681	0	8172	292
39	643	0	8172	287
36	729	0	8172	282
33	441	0	8172	196
11	504	0	8172	188

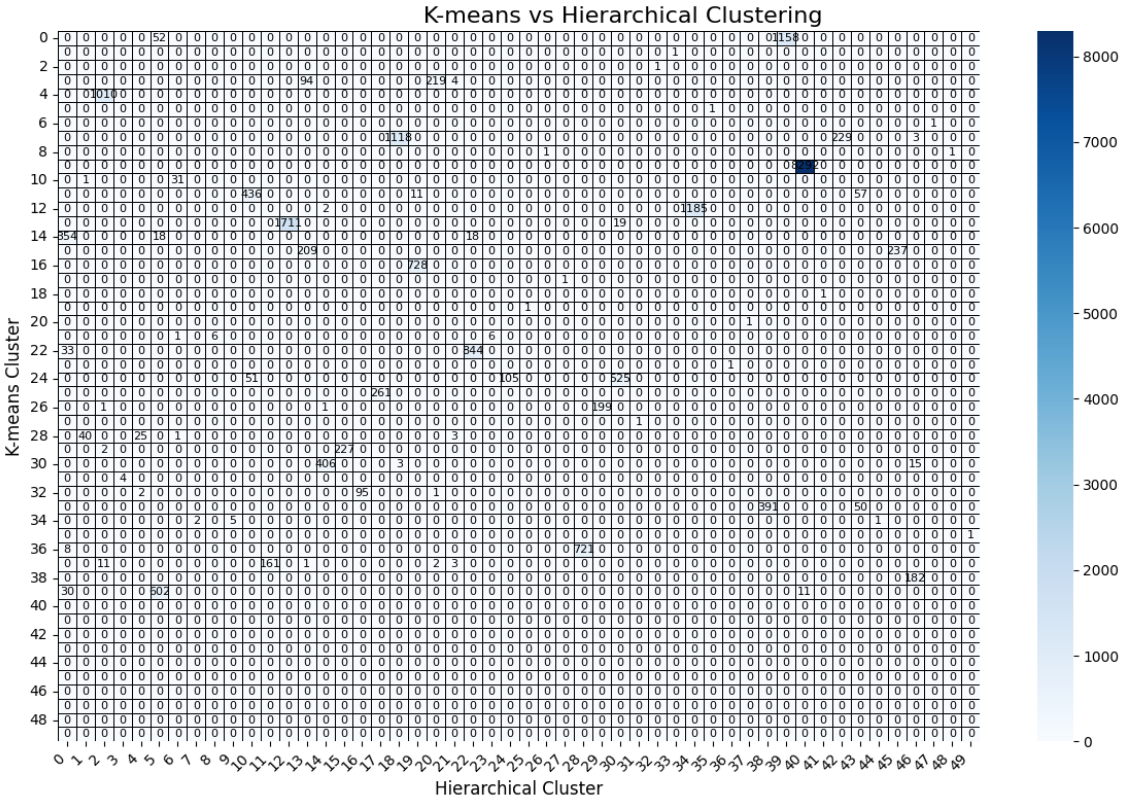
15	446	0	8172	163
14	390	0	8172	146
30	424	0	8172	131
3	317	0	8172	120
29	229	0	8172	101
25	261	0	8172	91
26	201	0	8172	76
38	182	0	8172	58
37	178	1	7615	58
32	98	2	1975	42
28	69	0	8172	33
10	32	0	8172	11
21	13	2	1975	8
34	8	2	1975	4
31	4	1	7615	3
2	1	2	1975	1
23	1	0	8172	1
27	1	1	7615	1
1	1	1	7615	1
19	1	2	1975	1
18	1	3	1614	1
17	1	0	8172	1
35	1	2	1975	1
5	1	1	7615	1
6	1	2	1975	1
8	2	1	7615	1
20	1	3	1614	1

Tabela 4.1: Najlepsze dopasowanie klastrów K-means do modułów WGCNA.

4.2 Porównanie K-means z klastrowaniem hierarchicznym

Porównanie metod K-means (40 klastrów) oraz Hierarchical Clustering (50 klastrów), zaimplementowanych w języku Python to kolejny etap analizy mającej na celu ocenę skuteczności i różnic w podejściu obu technik grupowania danych genomicznych. Do wizualizacji zgodności wyników wykorzystano ponownie macierz kontyngencji (Rysunek 4.2), która ukazuje liczbę genów wspólnych między klastrami K-means i Hierarchical Clustering. Analiza wykazała wysoką zgodność w przypisaniu genów do dużych klastrów, na przykład klaster 9 w K-means (8292 geny) niemal w całości odpowiada klastrowi 40 w Hierarchical Clustering (8303 geny). Podobnie klaster 13 w K-means (1730 genów) znajduje swój odpowiednik w klastrze 12 w Hierarchical Clustering (1711 genów). Jednak różnice w wynikach obu metod stają się widoczne w przypadku mniejszych klastrów. Przykładowo, klaster 11 w K-means (504 geny) został rozproszony pomiędzy kilka klastrów w Hierarchical Clustering (klaster 10, 43 czy 19). Sytuacje takie mogą wynikać z różnic w algorytmach. K-means optymalizuje odległość euklidesową, która tworzy zwarte grupy, a Hierarchical Clustering bazuje na hierarchicznej strukturze, która pozwala na uchwycenie bardziej złożonych relacji między genami. Tabela 4.2 przedstawia szczegółowe dopasowania między klastrami K-means i Hierarchical Clustering, uwzględniając liczbę genów w poszczególnych klastrach oraz ich pokry-

wanie się. Na przykład klaster 7 w K-means (1350 genów) został dobrze dopasowany do klastra 18 w Hierarchical Clustering, z 1118 wspólnymi genami. Do porównania wyników zastosowano Adjusted Rand Index (ARI), którego wartość wyniosła 0.9866. Wysoki wynik ARI wskazuje na dużą zgodność wyników między metodami, szczególnie w przypadku dużych klastrów. W mniejszych klastrach różnice te mogą mieć istotne znaczenie w szczegółowych analizach biologicznych, podkreślając potencjalną przewagę Hierarchical Clustering w uchwyceniu bardziej subtelnych relacji między genami. Podsumowując, porównanie K-means z Hierarchical Clustering wskazuje, że obie metody skutecznie identyfikują główne struktury w danych genomicznych. K-means oferuje bardziej zwarte i jednorodne grupy, natomiast Hierarchical Clustering umożliwia dokładniejsze uchwycenie złożonych zależności między genami. Wybór odpowiedniej metody powinien być dostosowany do specyfiki danych i celów badania.



Rysunek 4.2: Macierz kontyngencji: K-means vs Hierarchical Clustering.

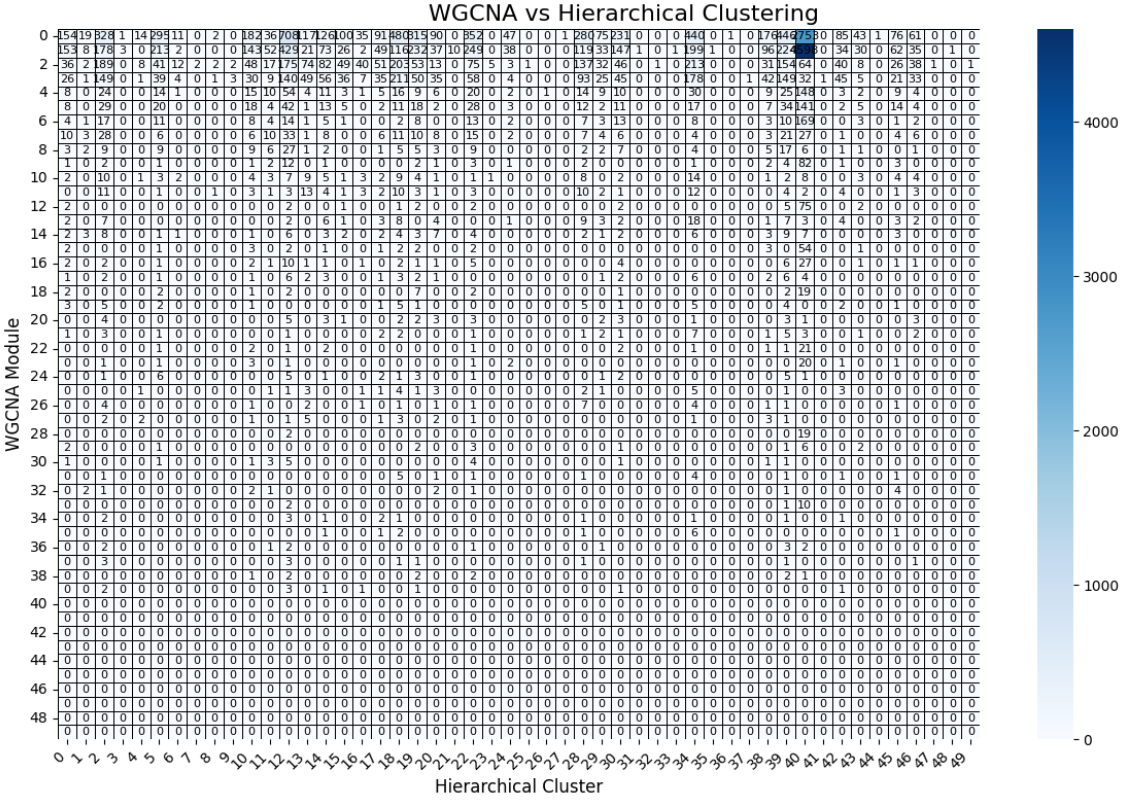
Klaster K-means	Liczba genów	Klaster hierarchiczny	Liczba genów	Wspólne geny
0	1210	39	1158	1158
3	317	20	222	219
4	1010	2	1024	1010
7	1350	18	1121	1118
9	8292	40	8303	8292
12	1187	34	1185	1185
13	1730	12	1711	1711
14	390	0	425	354
15	446	45	237	237

16	728	19	739	728
22	877	22	862	844
24	681	30	544	525
25	261	17	261	261
26	201	29	199	199
30	424	14	409	406
32	98	16	95	95
33	441	38	391	391
36	729	28	721	721
38	182	46	200	182
39	643	5	672	602

Tabela 4.2: Najlepsze dopasowanie klastrów K-means do klastrów Hierarchical Clustering.

4.3 Porównanie WGCNA z klastrowaniem hierarchicznym

Porównano wyniki grupowania danych genomicznych uzyskane za pomocą metod WGCNA oraz Hierarchical Clustering (50 klastrów), koncentrując się na ocenie zgodności grup genów w obu podejściach. Analiza macierzy kontyngencji (Rysunek 4.3) pozwoliła zidentyfikować najbardziej zgodne grupy genów oraz uwidocznić różnice w sposobie tworzenia klastrów przez obie metody. Największe pokrywanie wyników zaobserwowano w dużych modułach i klastrach. Na przykład moduł WGCNA 1 (7615 genów) i klaster 40 w Hierarchical Clustering (8303 geny) dzieliły 4598 wspólnych genów. Podobnie moduł WGCNA 0 (8172 geny) pokrywał się częściowo z klastrem 40, z 2753 wspólnymi genami. Takie wyniki wskazują, że obie metody skutecznie identyfikują główne grupy genów, choć podejścia te różnią się w strukturze generowanych grup. Mniejsze moduły i klastry wykazały mniejszą zgodność. Przykładowo, moduł WGCNA 24 (29 genów) miał zaledwie 6 wspólnych genów z klastrem 5 (672 geny). Tego rodzaju różnice są wynikiem odmiennych algorytmów: WGCNA grupuje geny na podstawie współekspresji, tworząc większe, bardziej ogólne moduły, podczas gdy Hierarchical Clustering bardziej szczegółowo rozdziela dane w struktury hierarchiczne. Zidentyfikowano także brak wspólnych genów między niektórymi modułami i klastrami, co jest efektem różnych miar podobieństwa stosowanych w obu metodach. Tabela 4.3 przedstawia szczegółowe dopasowania między modułami i klastrami, umożliwiając analizę jakości grup w kontekście liczebności genów. Wyniki te pokazują, że Hierarchical Clustering może dokładniej uchwycić subtelne różnice w danych, podczas gdy WGCNA lepiej odwzorowuje szerokie wzorce współekspresji. Adjusted Rand Index (ARI), wynoszący 0.1197, wskazuje na niską zgodność między metodami, co jest naturalnym wynikiem ich różnic w założeniach i algorytmach. WGCNA grupuje geny w większe moduły o wspólnej współekspresji, podczas gdy Hierarchical Clustering rozdziela dane w bardziej szczegółowe struktury hierarchiczne. Podsumowując, każda z metod oferuje unikalne perspektywy analizy danych genomicznych. WGCNA lepiej odzwierciedla ogólne współekspresje między genami, co jest użyteczne w analizach biologicznych na poziomie sieciowym. Z kolei Hierarchical Clustering umożliwia identyfikację bardziej precyzyjnych zależności między genami, co sprawdza się w szczegółowych badaniach. Wyniki te wskazują na komplementarność obu podejść, co może być wykorzystane w różnych kontekstach analizy biologicznej.



Rysunek 4.3: Macierz kontyngencji: WGCNA vs Hierarchical Clustering.

Moduł WGCNA	Liczba genów	Klaster hierarchiczny	Liczba genów	Wspólne geny
1	7615	40	8303	4598
24	29	5	672	6
0	8172	40	8303	2753
2	1975	34	1185	213
19	36	2	1024	5
6	310	40	8303	169
5	453	40	8303	141
3	1614	18	1121	211
7	240	12	1711	33
23	31	40	8303	20
4	467	40	8303	148
8	137	12	1711	27
9	122	40	8303	82
10	114	34	1185	14
22	33	40	8303	21
16	70	40	8303	27
17	44	12	1711	6
12	94	40	8303	75
15	74	40	8303	54
11	99	13	304	13
25	28	34	1185	5

21	34	34	1185	7
13	86	34	1185	18
37	11	2	1024	3
39	10	12	1711	3
14	80	39	1158	9
27	23	13	304	5
18	38	40	8303	19
28	21	40	8303	19
34	13	12	1711	3
20	36	12	1711	5
29	18	40	8303	6
36	12	39	1158	3
31	16	18	1121	5
26	25	28	721	7
35	12	34	1185	6
30	18	12	1711	5
38	10	19	739	2
33	13	40	8303	10
32	14	45	237	4

Tabela 4.3: Najlepsze dopasowanie modułów WGCNA do klastrów Hierarchical Clustering.

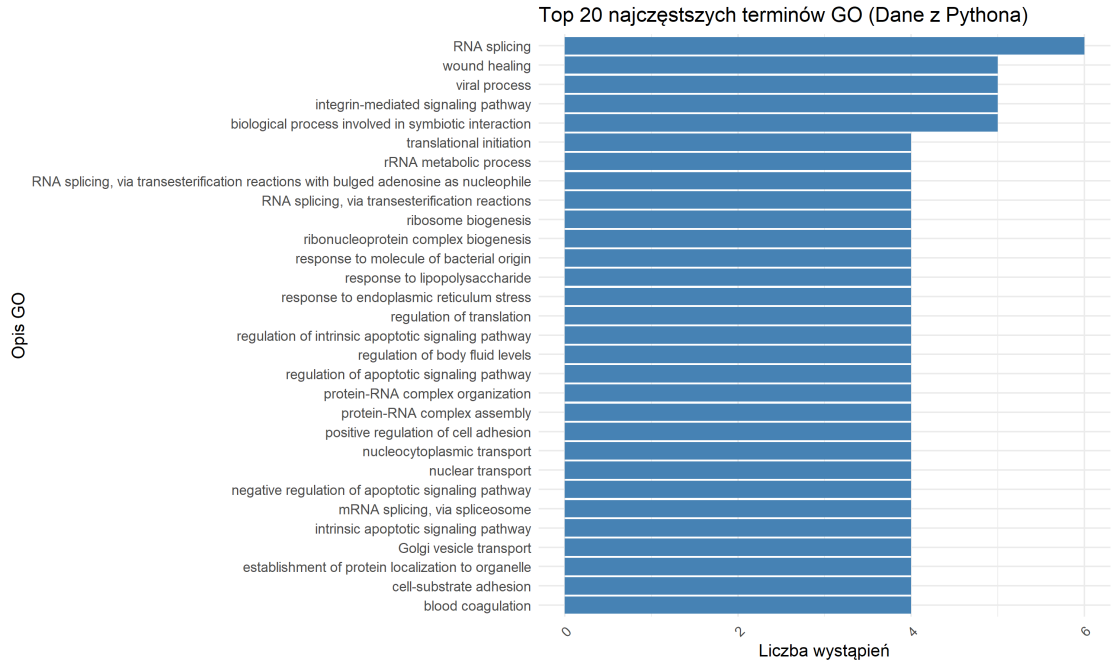
4.4 Analiza wzbogacenia

Analiza wzbogacenia jest metodą pozwalającą na interpretację biologicznej istotności zbiorów genów poprzez identyfikację nadreprezentowanych procesów biologicznych, ścieżek metabolicznych oraz innych funkcjonalnych kategorii [5]. W kontekście analizy współekspresji genów analiza wzbogacenia pozwala określić, jakie funkcje pełnią geny w poszczególnych modułach oraz jakie procesy biologiczne mogą być istotne dla badanej struktury. Główną ideą analizy wzbogacenia jest porównanie zestawu genów, np. modułu genowego, z referencyjną bazą danych funkcjonalnych, takich jak Gene Ontology (*GO*) czy KEGG. Dzięki temu można określić, które procesy biologiczne występują w analizowanej grupie genów częściej, niż można by się spodziewać na podstawie losowego doboru. Analiza ta opiera się na testach statystycznych, które oceniają znaczenie wzbogacenia danej kategorii funkcjonalnej.

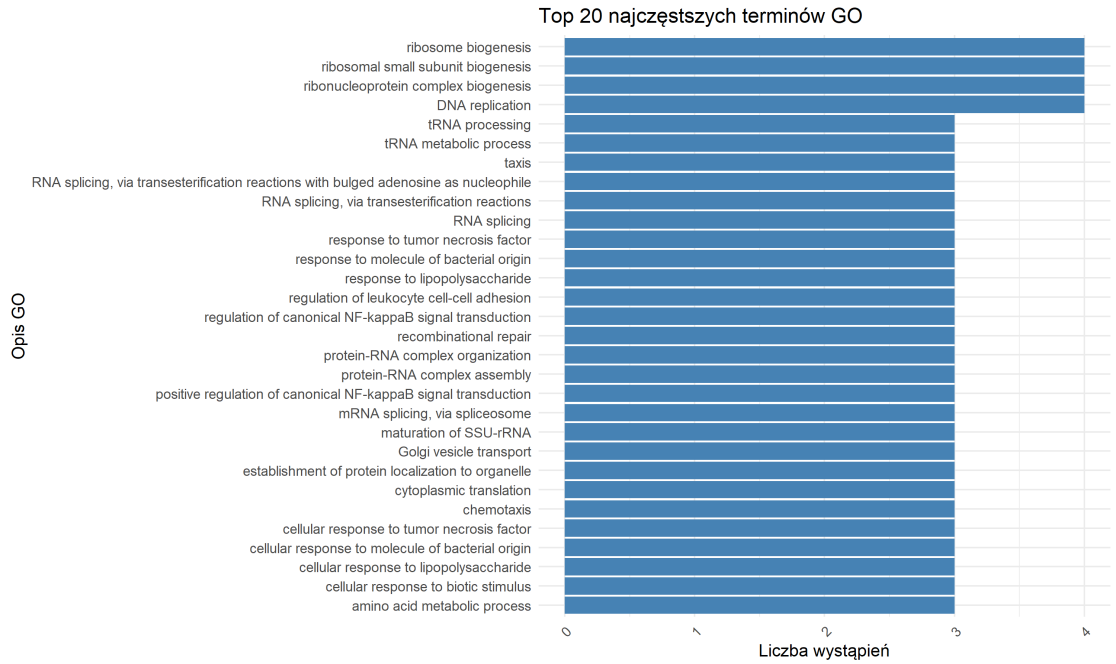
Pierwszym etapem analizy jest konwersja identyfikatorów genów danych wejściowych na standardowe formaty, umożliwiające porównanie ich funkcjonalności z bazami referencyjnymi. Do tego celu wykorzystano bazę *org.Hs.eg.db* [1], zawierającą adnotacje dla ludzkich genów. W przypadku niektórych genów, które nie mają bezpośrednich odpowiedników w bazie, ich identyfikatory są odrzucane, a analiza kontynuowana jest tylko na nieodrzuconych zestawach genów.

Kolejnym krokiem jest przeprowadzenie testu wzbogacenia, który pozwala określić, czy dany moduł zawiera geny uczestniczące w specyficznych procesach biologicznych częściej, niż można by się tego spodziewać losowo. Do tego celu wykorzystywane są metody statystyczne, takie jak test hipergeometryczny czy analiza wzbogacenia zestawu genów (*GSEA*). Aby zminimalizować ryzyko uzyskania fałszywie pozytywnych wyników zastosowano metody korekcji na wielokrotne testowanie, takie jak metoda Benjamini-Hochberga, która minimalizuje ryzyko uzyskania fałszywie pozytywnych wyników. Analiza wzbogacenia została przeprowadzona zarówno na modułach uzy-

skanych metodą WGCNA, jak i na klastrach genów utworzonych w Pythonie przy użyciu metody k-średnich (*ang. k-means*). Porównanie wyników uzyskanych obiema metodami pozwala na lepszą ocenę stabilności oraz biologicznej spójności wykrytych modułów. W analizie zastosowano filtrację wyników na podstawie wartości progowej *p-value <= 0.05* oraz *q-value <= 0.2*, co pozwala na eliminację przypadkowych wzbogaceń i skoncentrowanie się na biologicznie istotnych ścieżkach. Analiza obejmowała wyłącznie geny, dla których możliwe było jednoznaczne przypisanie identyfikatorów w bazach referencyjnych, a moduły zawierające mniej niż dwa geny zostały wykluczone ze względu na brak wystarczających danych do przeprowadzenia testów statystycznych.



Rysunek 4.4: Rozkład wystąpień najczęstych ról biologicznych dla wzbogacenia modułów k-średnich



Rysunek 4.5: Rozkład wystąpień najczęstych ról biologicznych dla wzbogacenia modułów WGCNA

Na Rysunku 4.4 przedstawiono rozkład najczęściej występujących terminów Gene Ontology (GO) dla wzbogacenia modułów uzyskanych metodą k-średnich. Najczęściej wzbogaconymi kategoriami są procesy związane z RNA splicing, wound healing, biological process involved in symbiotic interaction, a także różne mechanizmy RNA metabolic process. Wskazuje to na silną reprezentację procesów związanych z metabolizmem RNA oraz mechanizmami naprawy i regulacji transkrypcyjnej w zidentyfikowanych klastrach. Z kolei Rysunek 4.5 prezentuje wyniki analizy wzbogacenia dla modułów wyodrębnionych przy użyciu metody WGCNA. Wśród najczęściej wzbogaconych procesów GO można zauważyć kategorie związane z ribosome biogenesis, RNA processing, translational regulation, a także procesy związane z chromatin organization i cellular response to hypoxia. Oznacza to, że moduły WGCNA w większym stopniu odzwierciedlają procesy związane z ekspresją genów oraz organizacją chromatyny. Porównanie obu wykresów wskazuje na różnice w interpretacji danych – podczas gdy metoda k-średnich grupuje geny w sposób bardziej ogólny, identyfikując szerokie kategorie procesów metabolicznych i naprawczych, metoda WGCNA bardziej precyzyjnie wyodrębnia moduły współekspresji, które mogą odzwierciedlać specyficzne funkcjonalne kompleksy regulacyjne.

Istotnym elementem wyników analizy jest zbiorcze podsumowanie wzbogacenia z Tabeli 4.4, które łączy kluczowe informacje na temat wyników analizy wzbogacenia oraz wartości współczynnika Silhouette dla każdego modułu, co pozwala na kompleksową ocenę zarówno spójności klastrów, jak i biologicznej istotności przypisanych im procesów. Dla każdej z zastosowanych metod analizy WGCNA oraz klasteryzacji k-średnich w Pythonie, utworzono osobne pliki podsumowujące: Hierarchiczne Podsumowanie WGCNA.xlsx, zawierające wyniki dla modułów wykrytych metodą WGCNA, oraz Hierarchiczne Podsumowanie Python.xlsx, przedstawiające analizę klastrów genów uzyskanych w Pythonie za pomocą metody k-średnich. Każdy z plików zawiera identyfikator analizowanego klastra (kolumna *Klaster*), który umożliwia porównanie wyników między różnymi podejściami, oraz liczbę genów przypisanych do danego modułu (*L_genow*). Dodatkowo podano średnią wartość współczynnika Silhouette dla każdego modułu (*Miara_sil*).

Klaster	L_genów	Miara_sil	Term_GO	p-value	GeneRatio	BgRatio
Klaster 0	8172	-0.4828	RNA splicing	1.43e-05	189/5098	484/18986
			double-strand break repair	0.00012	131/5098	323/18986
			regulation of DNA repair	0.00012	98/5098	227/18986
			positive regulation of DNA metabolic process	0.00012	121/5098	295/18986
			macroautophagy	0.00013	146/5098	372/18986
			DNA recombination	0.00026	138/5098	352/18986
			double-strand break repair via homologous recombination	0.00083	81/5098	188/18986
			cytoplasmic translation	0.00083	73/5098	165/18986
			recombinational repair	0.00159	82/5098	194/18986
			transcription by RNA polymerase III	0.00182	34/5098	62/18986
			positive regulation of autophagy	0.00369	71/5098	166/18986
			vacuole organization	0.00534	95/5098	240/18986
			DNA replication	0.00534	109/5098	283/18986
			cytoskeleton-dependent intracellular transport	0.00648	84/5098	208/18986
			vesicle organization	0.00659	141/5098	386/18986
			Wnt signaling pathway	0.00659	169/5098	476/18986
			Golgi vesicle transport	0.00717	116/5098	308/18986
Klaster 1	7615	-0.1075	cilium movement	1.13e-08	102/3901	258/18986
			microtubule-based movement	1.52e-08	162/3901	480/18986
			nuclear division	1.52e-08	154/3901	451/18986
			cilium movement involved in cell motility	7.46e-08	86/3901	215/18986
			cilium or flagellum-dependent cell motility	1.16e-07	86/3901	218/18986
			mitotic sister chromatid segregation	6.61e-07	77/3901	194/18986
			microtubule cytoskeleton organization involved in mitosis	1.41e-06	68/3901	167/18986
			mitotic spindle organization	1.41e-06	58/3901	134/18986
			chromosome segregation	2.46e-06	139/3901	429/18986
			spindle organization	2.46e-06	78/3901	204/18986
			nuclear chromosome segregation	2.58e-06	111/3901	324/18986
			axoneme assembly	2.73e-06	48/3901	105/18986
			regulation of membrane potential	4.77e-06	147/3901	466/18986
			microtubule bundle formation	4.84e-06	57/3901	136/18986
			spindle assembly	6.87e-06	56/3901	134/18986
			flagellated sperm motility	6.94e-06	74/3901	196/18986
			sperm motility	6.94e-06	74/3901	196/18986

Tabela 4.4: Fragment podsumowującej tabeli zbiorczej dla klastrów z WGCNA

Pliki zawierają również informacje o nazwie wzbogaconego procesu biologicznego (*Term_GO*), czyli funkcjonalnej kategorii Gene Ontology (*GO*). Oprócz tego podano surową wartość p (*p-value*), określającą statystyczną istotność wzbogacenia danego procesu biologicznego, im niższa wartość, tym silniejsze powiązanie procesu z analizowanym zestawem genów. Dla każdego modułu zamieszczono wskaźnik *GeneRatio*, który określa stosunek liczby genów przypisanych do danej kategorii biologicznej do liczby wszystkich genów w module, które zostały odnalezione w bazie Gene Ontology (GO). Oznacza to, że jeśli dany moduł zawiera 8000 genów, ale tylko 5000 z nich ma przypisaną kategorię biologiczną w bazie GO, to *GeneRatio* oblicza się względem tych 5000 genów, a nie całej liczby genów w module. Dodatkowo uwzględniono wskaźnik *BgRatio*, który odnosi się do liczby genów przypisanych do danej kategorii biologicznej w całej bazie referencyjnej w stosunku do całkowitej liczby genów zidentyfikowanych w bazie dla całego pliku wejściowego. Dzięki temu wskaźnik ten pozwala określić, jak często dany proces GO występuje w populacji genów odnalezionych w bazie dla analizowanego zbioru danych. Taka struktura podsumowania umożliwia nie tylko ocenę istotności wzbogacenia procesów biologicznych, ale także ich porównanie między klastrami oraz kontekstualizację wyników względem całego zbioru danych. Dzięki temu

możliwa jest bardziej precyzyjna interpretacja funkcjonalna wykrytych modułów i ich znaczenia w analizowanych warunkach biologicznych.

Rozdział 5

Wnioski

5.1 Podsumowanie pracy

Celem niniejszej pracy było porównanie metod grupowania danych genomicznych w celu oceny ich skuteczności w identyfikacji biologicznie istotnych grup genów. W pracy przeprowadzono szczegółową analizę trzech podejść do grupowania: klasycznego algorytmu `K-means` oraz bardziej zaawansowanych metod, takich jak `WGCNA` i `Hierarchical Clustering`. Dla każdej z metod wykonano analizy jakościowe i ilościowe, które obejmują porównanie wyników przy użyciu macierzy kontyngencji oraz wskaźnika `Adjusted Rand Index (ARI)`.

Przeprowadzone badania wykazały, że każda z analizowanych metod charakteryzuje się unikalnymi cechami, które wpływają na sposób grupowania danych genomicznych. Metoda `K-means`, oparta na minimalizacji odległości w przestrzeni wielowymiarowej, dobrze sprawdza się w przypadku identyfikacji mniejszych, jednorodnych grup. Z kolei `WGCNA`, która wykorzystuje współekspresję genów, umożliwia wykrycie większych modułów biologicznie istotnych, co może być bardziej przydatne w analizie relacji sieciowych. `Hierarchical Clustering`, jako metoda hierarchiczna, oferuje elastyczność w analizie danych, szczególnie w przypadku dużych zbiorów genów, pozwalając na dokładniejsze uchwycenie ich struktury.

Do wizualizacji zgodności między metodami zastosowano macierze kontyngencji, które umożliwiły porównanie liczby genów wspólnych między grupami utworzonymi przez różne metody. Analiza wykazała, że największe pokrywanie wyników występuje w dominujących klastrach i modułach, co potwierdza, że każda z metod skutecznie identyfikuje główne grupy genów. W mniejszych grupach zaobserwowano jednak różnice w przypisaniu genów, co uwidacznia odmienny charakter poszczególnych algorytmów.

Wyniki wskaźnika ARI ujawniły różny stopień zgodności między metodami. Porównanie K-means z WGCNA wykazało niską zgodność (ARI = 0.1187), które odzwierciedla fundamentalne różnice w podejściu tych metod. Natomiast porównanie K-means z Hierarchical Clustering wykazało wysoką zgodność (ARI = 0.9866). Świadczy to o podobieństwu w sposobie identyfikacji głównych struktur danych. Zgodność między WGCNA a hierarchicznym klastrowaniem (ARI = 0.1197) była niska, wynika to z różnic w założeniach obu metod.

Analiza wyników WGCNA wykazała, że metoda ta skutecznie identyfikuje moduły genów o biologicznym znaczeniu, grupując je na podstawie współekspresji zamiast geometrycznej bliskości. WGCNA pozwoliło na identyfikację większych modułów genowych o wysokiej spójności. Wyniki wskazują, że metoda może być bardziej przydatna w badaniach nad interakcjami genów, w przeciwieństwie do metod klasycznych, które lepiej sprawdzają się przy analizie mniejszych, izolowanych grup.

Zastosowanie analizy wzbogacenia funkcjonalnego na wynikach WGCNA dodatkowo potwierdziło znaczenie uzyskanych modułów w kontekście biologicznym. Moduły genowe zidentyfikowane przez WGCNA wykazywały wzbogacenie w kluczowe ścieżki biologiczne, co sugeruje, że metoda ta może lepiej odzwierciedlać naturalne zależności funkcjonalne między genami. Wyniki analizy pokazały, że WGCNA może być szczególnie przydatna w badaniach sieciowych genów i złożonych systemów biologicznych.

Należy jednak zaznaczyć, że wyniki klasteryzacji, ocenione za pomocą wskaźnika ARI, nie zawsze umożliwiają przeprowadzenie analizy wzbogacenia funkcjonalnego. W szczególności duża liczba klastrów jednoelementowych utrudnia uzyskanie biologicznie istotnych wyników, co uwidacznia ograniczenia niektórych metod klasteryzacji w kontekście analizy funkcjonalnej.

Niski współczynnik ARI pomiędzy WGCNA a klasycznymi metodami klasteryzacji wynika z fundamentalnych różnic w algorytmach. K-means oraz hierarchiczne klastrowanie działają na zasadzie optymalizacji odległości w przestrzeni wielowymiarowej, co oznacza, że przypisują próbki do klastrów na podstawie ich bliskości geometrycznej. W przeciwieństwie do tego, WGCNA konstruuje sieć współekspresji genów i identyfikuje moduły na podstawie powiązań korelacyjnych, co sprawia, że uzyskane grupy nie zawsze pokrywają się z klastrami wykrytymi metodami klasycznymi.

Dodatkowo, WGCNA generuje moduły o zmiennej wielkości, podczas gdy klasyczne metody klasteryzacji często wymuszają podobne rozmiary klastrów. Wyniki wskazują, że każda z metod podkreśla różne aspekty struktury danych genomicznych, co sprawia, że ich wybór powinien być dostosowany do specyfiki analizowanego problemu.

Podsumowując, praca dostarcza kompleksowej analizy porównawczej metod grupowania danych genomicznych, podkreślając zarówno ich mocne strony, jak i ograniczenia. Wyniki te mogą stanowić cenną wskazówkę przy wyborze odpowiednich metod w przyszłych badaniach z zakresu analizy danych biologicznych.

5.2 Propozycje dalszych badań

Przeprowadzone badania dostarczyły wielu cennych informacji na temat sposobów grupowania danych genomicznych, ale jednocześnie wskazały nowe możliwości dalszych analiz i rozwoju opisywanych metod. W przyszłości warto byłoby rozszerzyć analizę o porównanie uzyskanych wyników wzbogacenia z danymi eksperymentalnymi, na przykład z wynikami badań proteomicznych lub metabolomicznych, co mogłoby dostarczyć dodatkowych dowodów na funkcjonalne znaczenie wyodrębnionych klastrów genowych. Innym kierunkiem dalszych badań mogłaby być analiza czasowa wzbogaconych procesów biologicznych, która pozwoliłaby na identyfikację dynamicznych zmian w regulacji genów oraz ich potencjalnych powiązań z określonymi stanami fizjologicznymi lub patologicznymi. Dodatkowo, wykorzystanie bardziej zaawansowanych metod statystycznych, takich jak analiza bayesowska, mogłoby umożliwić lepsze oszacowanie istotności wzbogacenia oraz uwzględnienie niepewności w interpretacji wyników. Automatyzacja wyboru parametrów klastrowania, czyli liczba klastrów, mogłaby być kolejnym krokiem w kierunku ulepszania analiz. Włączenie algorytmów automatycznego dostrajania parametrów w obu środowiskach mogłoby uprościć proces analizy, zwiększyć jej obiektywność oraz dostarczyć praktycznych wskazówek dotyczących najlepszego podejścia w różnych przypadkach. Dalsze prace mogłyby również skupić się na porównaniu wydajności, precyzji wyników oraz łatwości użycia metod grupowania implementowanych w Pythonie i R. Takie badania dostarczyłyby praktycznych wskazówek dla badaczy, ułatwiając wybór odpowiedniego narzędzia w zależności od specyfiki danych oraz celu analizy. Podsumowując, praca dostarcza wstępnej analizy porównawczej metod grupowania danych genomicznych implementowanych w Pythonie i R, podkreślając zarówno ich mocne strony, jak i ograniczenia. Dalsze

badania mogłyby przyczynić się do rozwoju metodologii klastrowania danych genomicznych, a także dostarczyć nowych możliwości ich praktycznego zastosowania, wspierając zarówno rozwój nauk biomedycznych oraz rozwój narzędzi analitycznych dla bioinformatyki.

Załączniki i kod źródłowy

Kod źródłowy wykorzystany w pracy, zawierający implementacje algorytmów klasteryzacji oraz analizę wyników, jest dostępny w dwóch repozytoriach GitHub:

- Implementacja w Pythonie: `https://github.com/okwasna/Praca_dyplomowa`
- Implementacja w R: `https://github.com/mczarkowska/Praca_dyplomowa_R`

W repozytoriach znajdują się skrypty w odpowiednich językach programowania oraz dodatkowe materiały użyte w analizie, w tym dane wejściowe, wyniki eksperymentów oraz wizualizacje klasteryzacji.

Bibliografia

[1] Marc Carlson. *org.Hs.eg.db: Genome wide annotation for Human*. R package version 3.20.0. 2025. URL: https://bioconductor.org/packages/org.Hs.eg.db/.

[2] Martin Ester i in. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". W: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*. AAAI Press, 1996, s. 226–231.

[3] Brian Everitt i in. *Cluster Analysis*. 5th. Chichester, UK: Wiley, 2011. ISBN: 978-0-470-74991-3.

[4] Peter Langfelder i Steve Horvath. *TOMsimilarity function in WGCNA R package*. Accessed: 2024-01-27. 2023. URL: https://rdrr.io/cran/WGCNA/man/TOMsimilarity.html.

[5] Peter Langfelder i Steve Horvath. "WGCNA: an R package for weighted correlation network analysis". W: *BMC Bioinformatics* 9.1 (2008), s. 559. DOI: 10.1186/1471-2105-9-559. URL: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-9-559.

[6] J. MacQueen. "Some Methods for Classification and Analysis of Multivariate Observations". W: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Berkeley, CA, USA: University of California Press, 1967, s. 281–297.

[7] GEO Accession Viewer. *GSE116428: RNA-Seq of human placentas from normotensive and preeclamptic pregnancies*. NCBI Gene Expression Omnibus. Accessed: 2025-01-29. 2018. URL: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE116428.

[8] Bin Zhang i Steve Horvath. "A general framework for weighted gene co-expression network analysis". W: *Statistical Applications in Genetics and Molecular Biology* 4.1 (2005), Article17. DOI: 10.2202/1544-6115.1128. URL: https://pubmed.ncbi.nlm.nih.gov/16646834/.

Instytut Informatyki, Wydział Informatyki i Telekomunikacji
Politechnika Poznańska

Skład przy użyciu systemu LaTeX na platformie Overleaf.