Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Julia Herman-Iżycka

Analiza różnicowej ekspresji genów w populacjach bakterii.

rozprawa doktorska

Promotor pracy
dr hab. Bartosz Wilczyński
Instytut Informatyki
Uniwersytet Warszawski

grudzień 2021

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza rozprawa została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu doktorskiego.

Data Podpis kierującego pracą

Oświadczenie autora pracy

Świadoma odpowiedzialności prawnej oświadczam, że niniejsza rozprawa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego lub naukowego.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data Podpis autora pracy

Streszczenie

Bakterie oraz inne mikroorganizmy można spotkać w praktycznie każdym środowisku. Zamieszkują nie tylko glebę czy oceany, ale też ludzkie ciało. Poszczególne regiony ciała, takie jak skóra, jama ustna, czy jelita, zapewniają różne warunki środowiskowe, więc są zamieszkiwane przez odrębne, wielogatunkowe populacje mikroorganizmów – mikrobioty. Skład gatunkowy mikrobioty jelitowej różni się pomiędzy poszczególnymi osobnikami i zależy od wielu czynników, np. diety. Ma również wpływ na stan zdrowia. Niedawny rozwój technologii sekwencjonowania znacznie ułatwił badanie zmienności takich wielogatunkowych populacji, jak również ekspresji ich genów. Problem postawiony w tej rozprawie dotyczy właśnie identyfikacji różnicowej ekspresji na podstawie krótkich odczytów pochodzących z sekwencjonowania metatranskryptomu.

Jednym z elementów tej pracy jest analiza danych eksperymentalnych pochodzących z sekwencjonowania mRNA wyekstrahowanego z mysich odchodów. Próbki te pochodzą z eksperymentu, w którym myszy karmione są dwoma rodzajami karmy – standardową oraz wysokotłuszczową, w celu identyfikacji różnic w mikrobiomie wynikających z diety. W tej pracy przedstawiony jest przegląd istniejących metod dostosowanych do analizy danych metagenomicznych oraz transkryptomicznych, a także porównane jest ich działanie na danych metatranskryptomicznych. Pokazana jest także możliwość wykorzystania pseudomapowania do zliczania odczytów oraz asemblacja *de novo*. Następnie przedstawione jest porównanie skuteczności opublikowanych metod asemblacji bazujących na dwóch najważniejszych koncepcjach: grafie *de Bruijna* i grafie nałożeń sekwencji (*overlap-layout-consensus*).

Rozdział 4 przedstawia szczególny wariant problemu asemblacji – asemblacja różnicowych transkryptów i wykazuje, że jedną z trudności z rozwiązaniem tego problemu przy użyciu grafu *de Bruijna* jest nieaddytywność zliczeń odczytów w ścieżce. Bierze się ona z tego, że jeden odczyt może być przypisany do więcej niż jednego wierzchołka. Przeszkoda ta nie występuje dla grafu nałożeń, w którym wierzchołek odpowiada jednemu odczytowi, ani dla uproszczonej jego wersji, w której jeden wierzchołek może reprezentować więcej niż jeden odczyt. Dla grafu nałożeń jest natomiast potencjalnie wykładnicza liczba rozwiązań optymalnych. Z tego powodu w sekcji 4.2 zaprezentowane są różne warianty zachłannych heurystycznych algorytmów do rozwiązania problemu znajdowania różnicowych transkryptów bezpośrednio w grafie nałożeń. Podejścia maksymalizujące długość kontigu zwracają inne sekwencje niż wersja maksymalizująca krotność zmiany – zbiory wykorzystanych w ścieżkach krawędzi różnią się. Chociaż sumaryczna długość oraz liczba sekwencji uzyskanych przez dowolną z heurystyk jest mniejsza niż najlepszej spośród metod asemblacji, na podstawie porównania z bazami sekwencji referencyjnych możemy stwierdzić, że zawierają one prawdopodobnie mniej kontigów hybrydowych, w związku z czym są one bardziej wiarygodne jako prawidłowe transkrypty.

Abstract

Bacteria and other microorganisms can be found in virtually any environment. They live not only in the soil and oceans, but also in the human body. Different regions of the body, such as the skin, mouth and intestines, provide different environmental conditions, so they are inhabited by distinct, multispecies populations of microorganisms - microbiota. Species composition of so called gut microbiota varies between individuals and depends on many factors, such as diet, and also affects health. Recent advances in sequencing technologies facilitated studying variability of such multispecies populations as well as their gene expression. The problem raised in this dissertation concerns precisely the identification of differential expression using short reads from metatranscriptome sequencing.

First part of this thesis presents analysis of experimental data from sequencing of mRNA extracted from mouse faeces. These samples were derived from an experiment in which mice are fed two types of food – standard and high-fat, to identify differences in the microbiome resulting from the diet. This thesis provides an overview and comparison of existing methods that are commonly used for metagenomic and transcriptomic data analysis on experimental metatranscriptomic data. The possibility of using pseudo-mapping to count reads is also presented, as well as a comparison of the effectiveness of the published assembly methods based on the two most important concepts: *de Bruijn* graphs and string graphs (overlap-layout-consensus graphs).

Chapter 4 presents a special variant of the assembly problem – the assembly of differential transcripts, and shows that one of the difficulties with solving this problem using the *de Bruijn* graph is the non-additivity of the read counts in the path. This is because one read may be assigned to more than one vertex. This obstacle does not exist for the overlap graph, in which the vertex corresponds to one read, nor for its simplified version in which one vertex can represent more than one read. For the overlap graph, however, there is a potentially exponential number of optimal solutions. For this reason, section 4.2 presents various greedy heuristic approaches to solving the problem of finding differential transcripts directly in the graph. The approach that maximize the contig length return different sequences than the fold-change version - the sets used in the edge paths differ. Although the total length and number of sequences obtained by any of the heuristics is smaller than the best of the assembly methods, their reliability seems to be higher thanks to lower expected fraction of hybrid contigs.

Słowa kluczowe

mikrobiom, metatranskryptomika, NGS, ekspresja genów, asemblacja, graf *de Bruijna*, graf nałożeń

Klasyfikacja tematyczna

Applied computing → Life and medical sciences → Bioinformatics → Computational biology

Tytuł pracy w języku angielskim

Differential expression analysis in populations of bacteria.

Podziękowania

Dziękuję wszystkim, którzy dzieląc się swoją wiedzą, doświadczeniem i innymi rodzajami wsparcia przyczynili się pośrednio lub bezpośrednio do powstania tej pracy – rodzicom, przyjaciółkom i przyjaciołom, nauczycielom i wykładowcom, oraz całej społeczności świata nauki.

W szczególności dziękuję mojemu promotorowi dr hab. Bartkowi Wilczyńskiemu za nieustające zachęty do pracy naukowej, wskazywanie jej kierunku oraz wszelkie wsparcie. Dziękuję dr Ilonie Grabowicz za inspirację do wybrania akurat tego tematu badań oraz wspólne zmagania ze stawianymi przez niego wyzwaniami, a wraz ze współpracowniczkami z Center for Genomic Regulation w Barcelonie – prof. Dierssen i dr Martą Fructuoso za owocną współpracę. Dziękuję współpracownikom i współpracowniczkom z grupy Biologii Obliczeniowej na Wydziale Matematyki, Informatyki i Mechaniki za wiele inspirujących rozmów i intensywną wymianę wiedzy.

Dziękuję również mojej mikrobiocie jelitowej za jej prawdopodobnie ogromny wkład w moje zdrowie i funkcjonowanie mojego mózgu.

Praca powstała dzięki finansowaniu z grantów:

- 2014/12/W/NZ1/00463 udzielonego przez Narodowe Centrum Nauki
- DEC-2012/05/B/NZ2/00567 udzielonego przez Narodowe Centrum Nauki
- ERA-NET-NEURON/10/2013 udzielonego przez Narodowe Centrum Badań i Rozwoju, z którego m. in. było finansowane sekwencjonowanie mRNA.

Spis treści

1. Wprowadzenie 9
1.1. Podstawowe pojęcia z genetyki 10
1.1.1. Miejsce bakterii w systematyce życia na ziemi 11
1.1.2. Organizacja genomów eukariotycznych i bakteryjnych 13
1.1.3. Proces transkrypcji u eukariontów i u bakterii 15
1.2. Mikroorganizmy i ich znaczenie 16
1.2.1. Rozmiar i zróżnicowanie mikrobioty człowieka 17
1.2.2. Wpływ mikrobiomu jelitowego na gospodarza 19
1.3. Techniczne aspekty badań genomu i transkryptomu – sekwencjonowanie 20
1.3.1. Powielanie DNA – metoda PCR 20
1.3.2. Sekwencjonowanie metodą Sangera 21
1.3.3. Sekwencjonowanie nowej generacji 22
1.3.4. Sekwencjonowanie trzeciej generacji 23
1.3.5. Sekwencjonowanie RNA 24
1.3.6. Błędy w sekwencjonowaniu nowej generacji 24
1.3.7. Główne wyzwania w badaniu metatranskryptomu 25

2. Obliczeniowe metody analizy wyników eksperymentów NGS . . . 28
2.1. Asemblacja genomu *de novo* 28
2.1.1. Sformułowania problemu asemblacji *de novo* 29
2.1.2. Overlap-Layout-Consensus i graf nałożeń (String Graph) . . . 34
2.1.3. Graf *de Bruijna* 37
2.1.4. Asemblacja dwuniciowego DNA 40
2.1.5. Usuwanie błędów i upraszczanie grafu 41
2.1.6. Rekonstrukcja sekwencji na podstawie grafu 43
2.1.7. Przykłady implementacji rozwiązań problemu asemblacji *de novo* 45
2.2. Mapowanie odczytów na genom i ilościowa analiza odczytów 50
2.2.1. Sformułowanie problemu mapowania i zliczania odczytów . . 50
2.2.2. Implementacje rozwiązań problemu mapowania 53
2.2.3. Zliczanie poprzez pseudomapowanie — program `kallisto` . . 57
2.2.4. Badanie różnic w ekspresji genów 59

3. Badanie różnicowej ekspresji genów w mikrobiomie jelitowym myszy na różnych dietach 62
3.1. Pochodzenie próbek — opis eksperymentu 62
3.2. Mapowanie na genom referencyjny 64
3.2.1. Ocena mapowania 64
3.2.2. Genomy bakteryjne z bazy RefSeq 65

3.2.3. Genomy bakterii jelitowych 66
3.2.4. Identyfikacja gatunków przy pomocy programu Kraken 68
3.3. Asemblacja transkryptomu *de novo* 69
3.3.1. Sposób asemblacji 69
3.3.2. Sekwencje otrzymane w wyniku asemblacji 70
3.3.3. Ocena wyników asemblacji poprzez mapowanie na kontigi . . 70
3.4. Identyfikacja różnicowych kontigów 72
3.5. Identyfikacja różnicowo eksprymowanych genów 74
3.5.1. Identyfikacja eksprymowanych genów na podstawie kontigów 74
3.5.2. Porównanie procedury identyfikacji różnicowych genów przy użyciu kontigów i odczytów 76
3.5.3. Geny o zmienionej ekspresji przy zmianie diety 76
3.5.4. Geny o różnej ekspresji pomiędzy genetycznymi populacjami . 79
3.5.5. Podsumowanie 80

4. Asemblacja z jednoczesną identyfikacją różnicowych sekwencji . . 81
4.1. Problem asemblacji różnicowych kontigów 81
4.1.1. Definicja problemu 81
4.1.2. Różnicowe kontigi w asemblacji z użyciem grafu *de Bruijna* . 83
4.1.3. Różnicowe kontigi w asemblacji z użyciem grafu nałożeń . . . 84
4.2. Graf nałożeń dla przykładowych danych transkryptomicznych 86
4.2.1. Działanie `SGA` 86
4.2.2. Właściwości grafu dla rzeczywistych danych metatranskryptomicznych 88
4.2.3. Upraszczanie grafu 89
4.3. Identyfikacja różnicowych kontigów w grafie nałożeń - heurystyki . . 90
4.3.1. Opis heurystyk wyszukiwania różnicowych kontigów 90
4.3.2. Wyniki asemblacji przy pomocy heurystyk 91
4.3.3. Porównanie kontigów `longestfc` oraz `MEGAHIT` i `SGA` poprzez mapowanie 92
4.3.4. Wiarygodność kontigów z `MEGAHIT` 95
4.3.5. Implementacja 97

5. Podsumowanie 98

ROZDZIAŁ 1

Wprowadzenie

Różnorodność form życia zadziwia jego badaczy od wielu stuleci. Ziemię zamieszkują miliony gatunków organizmów, od mikroskopijnych po kilkudziesięciometrowe, różniące się sposobami odżywiania czy rozmnażania. Pomimo różnic w budowie czy sposobie życia organizmów, można dostrzec wiele wspólnych cech zarówno w strukturze jak i sposobie funkcjonowania, szczególnie na poziomie pojedynczej komórki.

Bakterie i inne mikroorganizmy można zaobserwować w praktycznie każdym środowisku: w oceanach, glebie, korzeniach roślin, różnych miejscach ciała człowieka (od skóry po jelita), a nawet w tak, wydawałoby się, niedostępnych dla życia miejscach ze względu na ekstremalne temperatury lub zanieczyszczenia jak gorące gejzery, lodowce, czy skażone kopalnie. Mają one kluczowe znaczenie dla zamieszkiwanych przez nie nisz środowiskowych. Dzięki niedawnemu rozwojowi sekwencjonowania wysokoprzepustowego zwiększyły się możliwości badania szczególnie tych spośród nich, których do tej pory nie udawało się hodować w warunkach laboratoryjnych. Od roku 1995, kiedy to poznano genom pierwszej bakterii, liczba zsekwencjonowanych genomów bakteryjnych ciągle rośnie i przekroczyła już 11 tys. gatunków. Wciąż jednak wiele z nich nie jest poznanych, a szybkie tempo mutacji u bakterii utrudnia ich katalogowanie. Wykorzystanie nowych technologii sekwencjonowania oraz technik obliczeniowych umożliwiło ostatnio badanie nie tylko pojedynczych gatunków, ale także **mikrobiomów**, czyli genomów całych populacji mikroorganizmów zamieszkujących daną niszę środowiskową. Choć organizmy te bywają nazywane florą bakteryjną czy mikroflorą, to nie należą do niej rośliny, ani nie zawiera ona jedynie bakterii, a również grzyby, archeony czy wirusy, dlatego lepszym określeniem tych mikroorganizmów jest nazwa **mikrobiota**. Jednak bakterie często stanowią znaczną część **mikrobioty**, szczególnie mikrobioty jelitowej, stąd częste skupienie się jedynie na tej grupie organizmów.

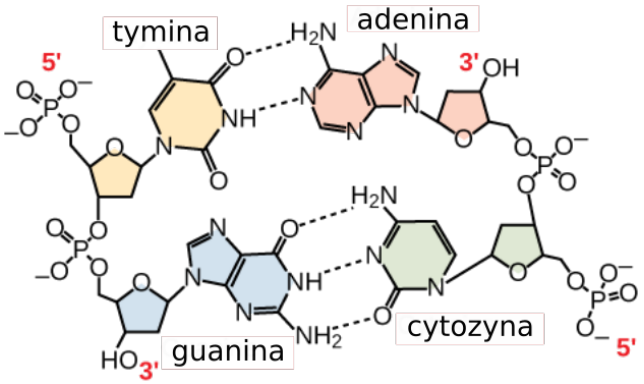
Wraz z lepszym poznaniem genetycznej różnorodności bakterii w rozmaitych środowiskach rozszerzyły się możliwości przewidywania pełnionych przez te organizmy funkcji, np. poprzez identyfikację genów obecnych w metagenomie. Jednak faktyczna aktywność bakterii w danych warunkach zależna jest od całego zestawu czynników i przejawia się poprzez ekspresję genów, której pierwszym etapem jest **transkrypcja** genu zapisanego w DNA, w wyniku której powstaje RNA zwane transkryptem. Ogół transkryptów tworzonych przez mikrobiom w danych warunkach środowiskowych, czyli **metatranskryptom**, jest zatem dynamiczny, a jego przemiany w wyniku zmiany warunków mogą wskazać, jak mikroorganizmy reagują na tę zmianę. Zbadanie metatranskryptomu stało się również możliwe dzięki rozwijającym się ostatnio technologiom sekwencjonowania.

W tym rozdziale przedstawiam biologiczne i technologiczne tło badań metatranskryptomicznych. Najpierw omawiam podstawowe pojęcia z genetyki (w tym ge-

netyki bakterii), które są przydatne do zrozumienia specyficznych problemów występujących podczas badania metatranskryptomów oraz rolę bakterii dla zdrowia człowieka. W dalszej części rozdziału omawiam też techniki eksperymentalne, m.in. wysokoprzepustowe sekwencjonowanie nowej generacji, które jest podstawą współczesnych badań genetycznych i transkryptomicznych.

1.1. Podstawowe pojęcia z genetyki

Genom zawiera całą informację potrzebną do budowy i funkcjonowania organizmu. Informacja ta jest zakodowana w DNA – dwuniciowym kwasie nukleinowym, który znajduje się w prawie każdej komórce organizmu. DNA jest polimerem zbudowanym z nukleotydów - związków zawierających resztę cukrową (w DNA jest to deoksyryboza), resztę fosforanową, oraz jedną z 4 zasad azotowych: adeninę (**A**), cytozynę (**C**), guaninę (**G**) lub tyminę (**T**) (patrz rysunek 1.1.1). To właśnie kolejność występowania tych zasad wzdłuż łańcucha cukrowego jest często określana jako sekwencja genomu, i zapisywana jako słowo nad alfabetem $\{A, C, T, G\}^*$. DNA jest cząsteczką dwuniciową a jej nici połączone są poprzez wiązania pomiędzy zasadami azotowymi, zgodnie z zasadą komplementarności - **A** z **T** i **G** z **C** (patrz rysunek 1.1.1). Stąd na podstawie jednej nici możemy odtworzyć sekwencję drugiej. Nici mają odwrotną orientację określaną przez miejsca wiązania pomiędzy resztami cukrowymi (kierunek od węgla 5' do węgla 3' w danym nukleotydzie) a rozróżnienie na główną (tzn. tę, której sekwencję zapisujemy) i do niej komplementarną jest często tylko umowne. Aby uzyskać sekwencję drugiej nici na podstawie głównej musimy znaleźć sekwencję **odwrotnie komplementarną**, czyli zamienić wszystkie nukleotydy na ich komplementarne pary, a do tego odwrócić kolejność.



Rysunek 1.1.1: Schemat budowy DNA.
Rysunek zaadaptowany (przetłumaczony) z materiałów Openstax [Clark et al., 2018b] na podstawie licencji CC BY 4.0.

Jednym z najważniejszych i najlepiej zbadanych elementów genomu są **geny**. Starsze definicje określały gen jako podstawową jednostkę dziedziczenia. Ta definicja z biegiem lat została zastąpiona bardziej precyzyjną definicją molekularną – gen to spójny fragment DNA, który ulega ekspresji powodując powstanie funkcjonalnej cząsteczki. W zależności od tego jaka cząsteczka jest końcowym produktem powstającym na bazie genu mówimy, że gen koduje np. białko. Chociaż definicja genu wciąż się zmienia pod wpływem odkryć nowych fenomenów biologicznych [Gerstein et al.,

2007], dla potrzeb tej pracy wystarczy powyższa uproszczona definicja wiążąca fragment genomu z powstającą na podstawie jego sekwencji cząsteczką.

Pierwszym etapem ekspresji genu jest **transkrypcja**, czyli zbudowanie cząsteczki jednoniciowego **RNA** (kwasu rybonukleinowego, różniącego się od DNA resztą cukrową oraz jedną z zasad - zamiast tyminy używany jest uracyl (**U**)). RNA powstaje na podstawie fragmentu DNA przy zachowaniu zasady komplementarności zasad. RNA kodujące białko jest nazywane matrycowym RNA (messenger RNA, mRNA), ponieważ jest używane w procesie **translacji** do produkcji białka jako matryca opisująca sekwencję aminokwasów w białku. Liczba cząsteczek danego mRNA obecnych w komórce ma duży związek z tym, ile cząsteczek białka powstanie na jego podstawie, dlatego wiele sposobów regulacji ekspresji genów działa poprzez regulację transkrypcji lub regulację degradacji mRNA. Poza replikacją, podczas której powstaje duplikat cząsteczki DNA, przepływ informacji genetycznej odbywa się dwuetapowo, z DNA do RNA i z RNA do białka. O ile zdarzają się odstępstwa od schematu:

$$\circlearrowright\text{DNA} \rightarrow \circlearrowright\text{RNA} \rightarrow \text{białko}$$

(np. odwrotna transkrypcja DNA←RNA), to nie znaleziono dotychczas przypadku, żeby informacja genetyczna przepływała z białka w kierunku kwasów nukleinowych. Ta dwu-etapowość i rozdzielność przepływu informacji stanowi tzw. **Centralny Dogmat Biologii**. W tej pracy skupiam się środkowym elemencie, czyli obecności i ilości mRNA.

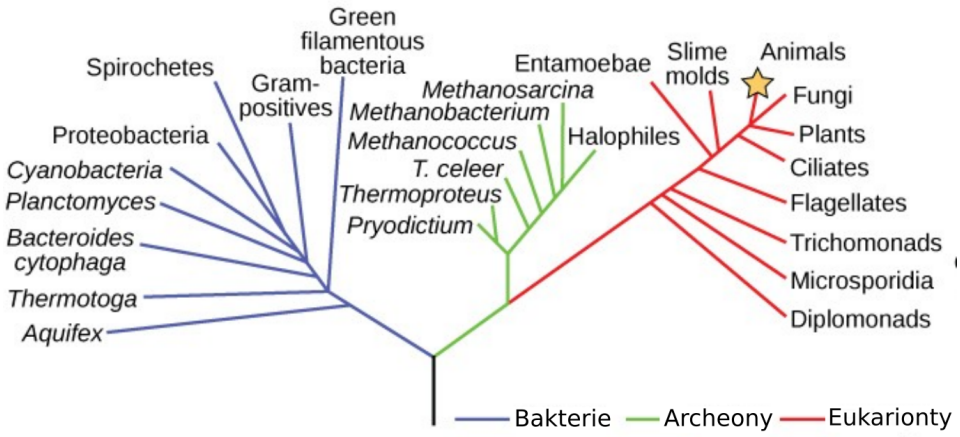
Większość (nawet 90%) RNA w komórce to nie (interesujące nas w badaniu ekspresji genów) mRNA tylko rRNA, czyli RNA rybosomalne. Rybosomy to kompleksy białek i rRNA, w obrębie których odbywa się translacja. Występują w każdej żywej komórce, a sekwencja ich podjednostek RNA jest dosyć dobrze zachowana w ewolucji i nie podlega horyzontalnemu transferowi genów, dlatego rybosomalne RNA jest często używane do identyfikacji gatunków (również w próbkach mikrobiomu) oraz ustalania pokrewieństwa gatunków.

Genom nie składa się jedynie z genów. Stanowią one czasami ledwie ok. 1% sekwencji genomu, czasami większość genomu, tak jak u bakterii, ale również u nich nie cały genom jest transkrybowany. Różne regiony niekodujące mogą pełnić różne funkcje, np. regulują ekspresję sąsiednich lub odległych genów czy pełnią role strukturalne (np. stabilizacja). Pozostaje wciąż wiele regionów, których funkcja wciąż nie jest znana. Sposoby regulacji i stabilizacji genomu różnią się między bakteriami a np. organizmami wyższymi, takimi jak zwierzęta. Chociaż wpływają na transkrypcję, nie są jednak bezpośrednio obiektem badań **transkryptomiki**. Omawiam jednak niektóre z nich przy opisie regulacji transkrypcji w rozdziale 1.1.3.

1.1.1. Miejsce bakterii w systematyce życia na ziemi

Wszystkie organizmy żywe na Ziemi możemy podzielić ze względu na sposób przechowywania informacji genetycznej na dwa nadkrólestwa - **prokarionty** (bezjądrowce) i **eukarionty** (jądrowce) (patrz rysunek 1.1.2). Do eukariontów należą zarówno organizmy jednokomórkowe jak i wielokomórkowe, tworzące królestwa: roślin, zwierząt, grzybów i protistów. Choć komórki organizmów należących do tych królestw różnią się znacznie budową, np. obecnością ściany komórkowej i niektórych organellów, łączy je obecność jądra komórkowego. Do prokariontów zaliczamy natomiast dwie duże domeny jednokomórkowców: bakterie i archeony. W podziale tym nie pojawiają się wirusy, gdyż zwykle nie są w ogóle zaliczane do organizmów żywych.

Zarówno bakterie jak i archeony nie mają oddzielonych błoną organelli komórkowych, w tym jądra komórkowego. Choć pod względem struktury materiału genetycznego i sposobu rozmnażania archeony są podobne do bakterii [Olsen and Woese, 1997], to inne ich cechy (obecność pewnych ścieżek metabolicznych, proces ekspresji genów) sugerują, że są bliżej spokrewnione z eukariontami niż z bakteriami. Ponieważ archeony nie będą obiektem moich badań, w dalszej części pracy skupiam się tylko na omówieniu różnic w organizacji genomu bakterii i eukariontów. Są one istotne, gdyż wpływają na możliwość użycia wielu narzędzi bioinformatycznych, które zwykle są projektowane i optymalizowane pod kątem konkretnej grupy organizmów.



Rysunek 1.1.2: Drzewo filogenetyczne pokazujące pokrewieństwo między 3 domenami życia na ziemi: bakteriami, archeonami i eukariontami.
Rysunek zaadaptowany (przetłumaczony) z materiałów Openstax [Clark et al., 2018c] na podstawie licencji CC BY 4.0.

Zgodnie z Międzynarodowym Kodeksem Nomenklatury Bakterii (ang. *The International Code of Nomenclature of Bacteria*, ICNB) bakterie dzielone są na 5 kategorii taksonomicznych (od najbardziej szczegółowych): gatunki (*species*), rodzaje (*genus*), rodziny (*familia*), rzędy (*ordo*) i klasy (*classis*) [Lapage et al., 1992]. Niezależnie od tej formalnej klasyfikacji wyróżnia się też kilkadziesiąt typów (*phylum*) bakterii [Euzéby, 1997]. Taksonomia, szczególnie u bakterii jest bardzo umowna – podział opiera się zarówno na morfologii, chemotaksonomii, jak i genetyce [Schleifer, 2009]. Nawet jeden z najbardziej podstawowych poziomów taksonomicznych czyli gatunek jest trudny do zdefiniowania. O ile u organizmów rozmnażających się płciowo za jeden gatunek uznaje się zwykle organizmy zdolne w wyniku rozmnażania stworzyć płodne potomstwo, to wśród bakterii (które rozmnażają się bezpłciowo), taka definicja nie jest możliwa. Uznaje się zatem na przykład pewien stopień podobieństwa jednej z podjednostek rybosomu (tzw. podjednostkę 16S), a następnie podobieństwo genomów [Schleifer, 2009].

W obrębie gatunku bakterie mogą również być bardzo zróżnicowane. Poszczególne **populacje bakterii**, czyli grupy komórek jednego gatunku zamieszkujących daną lokalizację, mogą składać się z prawie identycznych linii klonalnych (powstałych poprzez podział komórki, podczas którego materiał genetyczny jest klonowany), jednak z powodu horyzontalnego transferu genów (o którym więcej w 1.1.2) i rekombinacji między komórkami danego gatunku i pokrewnych gatunków obecnych w tym samym miejscu, tworzą bardziej skomplikowaną strukturę genetyczną, która może wyróżniać się od innych populacji. Populacje bakterii danego gatunku, które wy-

kazują wspólne cechy fenotypowe i genetyczne to **szczepy**. Podział na szczepy jest często bardzo istotny z punktu widzenia analizy wpływu danych bakterii na środowisko, ponieważ w obrębie jednego gatunku można znaleźć np. zarówno szczepy patogenne, jak i korzystne dla zdrowia, jak np. *Escherichia coli*, która występuje powszechnie w ludzkich jelitach, ale której pewne szczepy powodują zagrażające życiu zakażenia [Mainil, 2013].

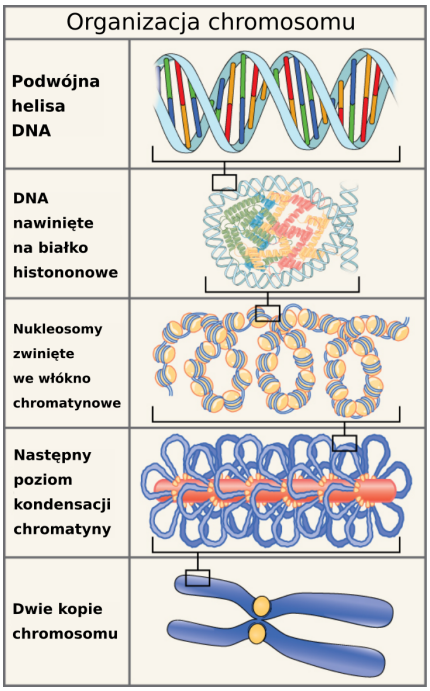
Podziały taksonomiczne są w pewnej mierze uznaniowe i nieścisłe, jednak nadanie grupom organizmów etykiet znacznie ułatwia analizę populacji i jej opis. Przypisanie genomów do szczepów, gatunków, czy też zaklasyfikowanie ich na innych poziomach taksonomicznych jest jednak przydatne do opisania przynajmniej w przybliżeniu spokrewnienia pewnych grup.

1.1.2. Organizacja genomów eukariotycznych i bakteryjnych

Główną cechą różniącą eukarionty i bakterie jest obecność u eukariontów jądra komórkowego, oddzielonego błoną od reszty komórki organellum, w którym mieści się DNA. U prokariontów natomiast DNA znajduje się w cytoplazmie i nie jest oddzielone od reszty komórki błoną. Różna jest także organizacja samego DNA. U bakterii jest to (zwykle) pojedyncza, kolista cząsteczka [Koonin and Wolf, 2008]. Genomy bakteryjne są też zdecydowanie mniejsze (rzędu 100 kb - 10 Mb [Ussery et al., 2009]) oraz bardziej zwarte – ok. 90% genomu bakteryjnego pokryte jest genami. U bakterii mogą pojawiać się też dodatkowe, małe cząsteczki DNA zwane **plazmidami**, które nie zawierają genów niezbędnych do podstawowego funkcjonowania komórki, ale często znajdują się na nich geny np. oporności na antybiotyki. Plazmidy mogą być przekazywane między komórkami (poprzez koniugację) pozwalając na inny rodzaj przekazywania informacji genetycznej niż dziedziczenie, tzw. **horyzontalny transfer genów**. Dodatkowo, bakterie potrafią też pobrać DNA ze swojego otoczenia.

U eukariontów większość DNA znajduje się w jądrze komórkowym. Organella takie jak mitochondria i chloroplasty zawierają własne genomy, pod wieloma względami podobne do prokariotycznych – małe, koliste, zwykle bez intronów. Wynika to z ich pochodzenia – organella te najprawdopodobniej są wynikiem endosymbiozy – wchłonięcia komórek bakteryjnych przez przodka eukariontów. Znacznie większa część materiału genetycznego u eukariontów znajduje się jednak w jądrze. DNA jądrowe jest liniowe i silnie związane z białkami histonowymi. Podwójna helisa DNA jest nawinięta na kompleksy białek histonowych (patrz rysunek 1.1.3), a następnie, w zależności od aktywności danego fragmentu genomu, mniej lub bardziej skompresowana. Genomy eukariotyczne są zwykle dużo większe niż bakteryjne (rzędu 10 Mb - 100 Gb) i pokryte genami jedynie w kilku procentach, stąd fizyczne skompresowanie cząsteczki DNA musi być bardzo efektywne.

DNA u eukariontów jest często podzielone na co najmniej kilka (lub kilkadziesiąt, choć może być ich nawet kilka tysięcy) osobnych łańcuchów – **chromosomów**. Wiele spośród eukariontów jest diploidalnych, czyli posiada dwie kopie każdego chromosomu. To znacznie utrudnia analizę genomu takich organizmów, gdyż kopie jednego chromosomu (chromosomy homologiczne) zawierają podobną informację genetyczną, czyli zawierają (w przybliżeniu) te same geny, jednak warianty tych genów (allele) mogą się różnić. Dzieje się tak np. w wyniku rozmnażania płciowego, podczas którego organizmy rodzicielskie przekazują organizmom potomnym po jednej kopii chromosomu.



Rysunek 1.1.3: Schematyczne przedstawienie organizacji DNA w organizmach eukariotycznych. Dwuniciowa, liniowa cząsteczka DNA jest nawinięta na kompleksy białek histonowych.
Rysunek zaadaptowany (przetłumaczony) z materiałów Openstax [Clark et al., 2018a] na podstawie licencji CC BY 4.0.

Liczba chromosomów i rozmieszczenie genów są w przybliżeniu takie same dla przedstawicieli tego samego gatunku. Dzięki temu można zdefiniować sekwencję genomu reprezentatywną dla całego gatunku, tzw. **genom referencyjny**. Genom referencyjny nie odpowiada zwykle pojedynczej komórce (a tym bardziej konkretnej kopii chromosomu u komórek diploidalnych) tylko zawiera warianty występujące u wielu przedstawicieli gatunku i złożone w jedną (dla każdego chromosomu), możliwie reprezentatywną sekwencję. Pozwala to analizować różne sekwencje biologiczne pobrane od innych przedstawicieli tego gatunku w odniesieniu do genomu referencyjnego, co ułatwia m.in. porównanie wyników uzyskanych dla różnych osobników. Różnice sekwencji genomu mogą dotyczyć pojedynczych pozycji w sekwencji (np. często występujące jednonukleotydowe polimorfizmy, ang. *single-nucleotide polymorphism*, SNP), lub większych fragmentów, które względem genomu referencyjnego mogą nie występować (**delecje**), być nadmiarowe (**insercje**) lub znajdować się w innymi miejscu genomu (**translokacje**).

Zarówno u eukariontów, jak i u prokariontów występują krótsze lub dłuższe powtórzenia fragmentów sekwencji zwane **regionami repetytywnymi**. Są to fragmenty genomu, zwykle długości od kilkudziesięciu nukleotydów do nawet kilku tysięcy, których sekwencja jest identyczna bądź różni się bardzo niewiele. Wystąpienia powtórzeń czasami sąsiadują, a czasami znajdują się w różnych miejscach genomu, nawet na różnych chromosomach. Mogą też znaleźć się na nici komplementarnej. Sekwencje repetytywne są czasem wynikiem duplikacji genów lub większych fragmentów genomu, mogą być wynikiem działania transpozonów. Mają też rolę strukturalną, gdyż tworzą telomery i centromery (u eukariontów), lub szczególnie interesujące ostatnio systemy ochrony przed patogennym DNA – CRISPR (u prokariontów).

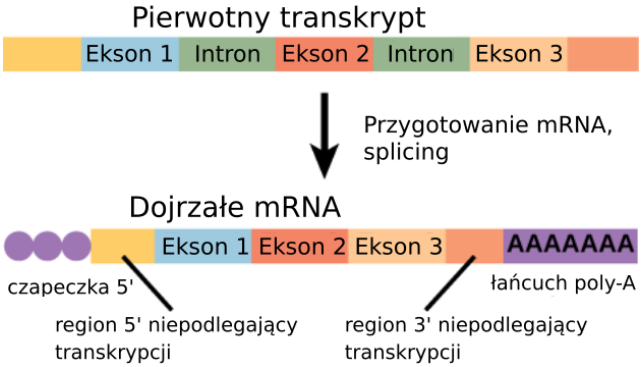
Powtórzone fragmenty w sekwencji genomu sprawiają wiele problemów obliczeniowych gdyż na podstawie krótkiego wycinka z sekwencji genomu czasami nie da się jednoznacznie wskazać miejsca jego wygenerowania w genomie referencyjnym. Jest to dosyć duży problem w przypadku eukariontów, jednak w genomach bakteryjnych tylko 3% sekwencji to większe powtórzenia, a dwa razy tyle to lokalne, krótkie powtórzenia mieszczące się w obrębie 100 bp [Ussery et al., 2009].

1.1.3. Proces transkrypcji u eukariontów i u bakterii

Większość procesów zachodzących w komórce odbywa się przy udziale białek, do których produkcji niezbędne jest odkodowanie sekwencji białka zapisanej w DNA. Wymaga to najpierw transkrypcji, czyli procesu nazywanego czasem ekspresją genu, a później translacji.

Transkrypcja, czyli proces syntezy RNA na podstawie DNA jest przeprowadzany przez polimerazę RNA. Polega on na dołączaniu kolejno nukleotydów komplementarnych do jednej z nici DNA. Transkrypcja rozpoczyna się w regionie chromatyny nazywanym **promotorem**, gdzie pewne białka – tzw. czynniki transkrypcyjne, wspomagają przyłączenie enzymu syntetyzującego RNA – polimerazy RNA.

U eukariontów geny zwykle transkrybowane są osobno, tzn. każdy gen ma własny promotor i miejsce zakończenia transkrypcji, chociaż zdarzają się wyjątki, np. operony u nicienia *C. elegans* [Blumenthal, 2004]. Niedojrzała cząsteczka kodującego mRNA (pre-mRNA) powstała w wyniku transkrypcji zanim posłuży do translacji musi opuścić jądro komórkowe. W tym celu do mRNA pod koniec transkrypcji dołączany jest długi łańcuch składający się z adenin (łańcuch poly-A), który bierze udział w transporcie, stabilizacji i ochronie cząsteczki mRNA przed degradacją. Po opuszczeniu jądra, a przed translacją, mRNA może podlegać jeszcze **splicingowi**, czyli wycięciu **intronów** (fragmentów niekodujących białka) i sklejeniu pozostałych fragmentów (kodujących **eksonów**) w dojrzałe mRNA. Fragment genu nie zawsze jest jednoznacznie przypisany do intronów lub eksonów. W wyniku **alternatywnego splicingu** granice intronów i kolejność eksonów mogą być inaczej określone w różnych okolicznościach, w wyniku czego z jednej sekwencji RNA mogą powstać różne mRNA, co skutkuje powstaniem różnych białek.



Rysunek 1.1.4: Schemat genu eukariotycznego – zawierającego introny.
Rysunek zaadaptowany (przetłumaczony) z materiałów Openstax [Clark et al., 2018e] na podstawie licencji CC BY 4.0.

U prokariontów splicing jest rzadko obserwowany [Belfort et al., 1995], za to powszechne są **mRNA policistronowe**, czyli cząsteczki mRNA, które na jednej nici zawierają informację kolejno o kilku białkach należących do tego samego **operonu**,

tzn. grupy genów wspólnie podlegających regulacji ekspresji. Operon składa się z fragmentu inicjującego transkrypcję – **promotora**, fragmentu regulującego transkrypcję **operatora**, następujących kolejno po sobie genów kodujących białka, oraz sekwencji powodującej zakończenie transkrypcji. Często białka kodowane w jednym operonie należą do jednego szlaku metabolicznego lub są potrzebne w tych samych ściśle określonych warunkach. Tak jest w przypadku operonu laktozowego [Jacob and Monod, 1961] u *Escherichia coli* (patrz rysunek 1.1.5), jednego z pierwszych poznanych mechanizmów kontroli ekspresji genów. Brak laktozy w środowisku oznacza, że białka ją metabolizujące nie muszą być produkowane i operon jest wyciszany poprzez przyłączenie cząsteczki represora. Gen kodujący represor jest zawsze aktywny i nie jest częścią operonu, chociaż jest położony blisko. Obecność laktozy w cytoplazmie powoduje odłączenie represora, który łączy się wtedy z izomerem laktozy. Dzięki temu polimeraza ma dostęp do promotora i może transkrybować 3 położone obok geny kodujące białka związane z metabolizmem laktozy. Poza negatywną regulacją w przypadku braku laktozy, również obecność glukozy wpływa negatywnie na aktywność operonu. Jako że glukoza jest bardziej korzystnym źródłem energii, w jej obecności operon laktozowy jest mało aktywny. Bez glukozy wzrasta poziom cząsteczek cAMP, sygnalizujących „głód", które w połączeniu z odpowiednim czynnikiem transkrypcyjnym łączą się do DNA w okolicy promotora i stymulują przyłączanie polimerazy. Zatem operon jest najbardziej aktywny w obecności laktozy, przy równoczesnym braku wystarczającej ilości glukozy, choć w innych przypadkach białka również są obecne w małej ilości, i również mogą być transkrybowane w małym stopniu.

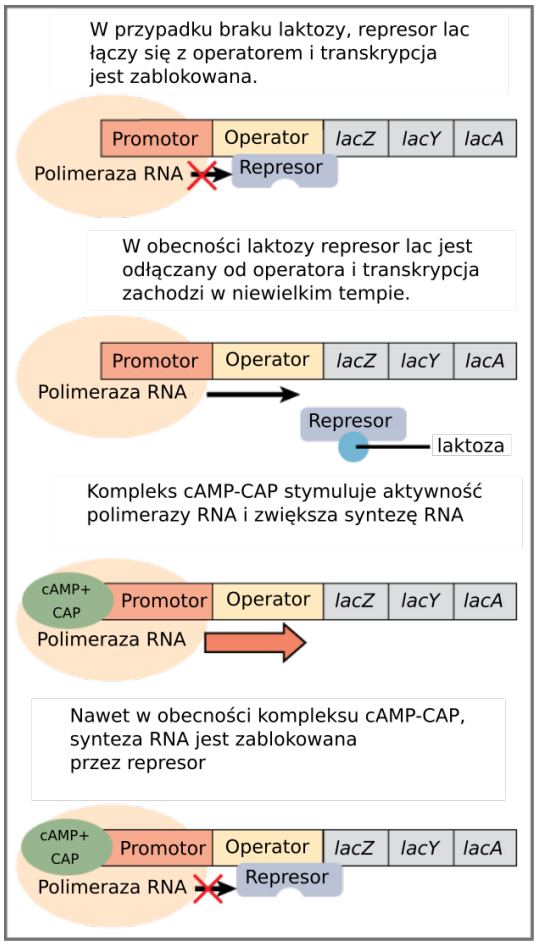
Operony są powszechne w świecie bakteryjnym, jednak trzeba pamiętać, że nie są one jedyną formą regulacji transkrypcji, a wraz z kolejnymi odkryciami w dziedzinie nauk biologicznych możemy spodziewać się odkrycia nowych form [Güell et al., 2011].

Zmiany ekspresji genów to główny sposób, w jaki komórka może reagować na zmiany środowiska, jak również konieczny by np. replikować się i różnicować. Dlatego o ile sekwencja genomu praktycznie nie zmienia się w trakcie życia organizmu, to transkryptom jest bardzo dynamiczny i jego zmiany mogą wiele powiedzieć o zachodzących w niej procesach, w szczególności jakie biochemiczne reakcje może przeprowadzić i jakie substancje może wyprodukować w ich wyniku.

Przy bioinformatycznej analizie transkryptomu trzeba wziąć pod uwagę opisane w tym rozdziale (a podsumowane w tabeli 1.1.1) różnice w strukturze transkryptów między prokariontami a eukariontami, a szczególnie fakt, że z jednej strony, u eukariontów zaobserwowane transkrypty mogą zawierać przerwy względem genomu organizmu, a z drugiej strony u prokariontów możemy zaobserwować na jednym transkrypcie więcej niż jeden gen.

1.2. Mikroorganizmy i ich znaczenie

Mikroorganizmy to bardzo szerokie pojęcie określające organizmy jednokomórkowe: wszystkie bakterie i archeony, a czasem także jednokomórkowe grzyby, protisty, czasem wirusy. Pełnią one bardzo wiele kluczowych funkcji środowiskowych, np. biorą udział w rozkładzie martwej materii organicznej oraz w wiązaniu azotu. Szerokie znaczenie mikroorganizmów w środowisku jest również związane z symbiotycznymi relacjami z innymi organizmami, zarówno korzystnymi dla obu stron (mutualizm, np.



Rysunek 1.1.5: Operon laktozowy – przykład bakteryjnej regulacji ekspresji genów i produkcji policistronowego mRNA w przypadku obecności laktozy. Rysunek zaadaptowany (przetłumaczony) z materiałów Openstax [Clark et al., 2018d] na podstawie licencji CC BY 4.0.

bakterie brodawkowe z roślinami motylkowymi), obojętnymi dla gospodarza (komensalizm, np. niektóre bakterie żyjące na skórze), i niekorzystnymi (pasożytnictwo, np. bakterie chorobotwórcze). Człowiek wykorzystuje mikroorganizmy przemysłowo, np. w produkcji żywności (fermentacja), do oczyszczania ścieków czy w medycynie (m.in. produkcja leków).

1.2.1. Rozmiar i zróżnicowanie mikrobioty człowieka

Do tej pory wykazano, że symbiotyczne mikroorganizmy zamieszkują wiele organizmów wielokomórkowych, zarówno roślinnych i zwierzęcych, w tym właściwie wszystkie regiony ciała człowieka, które mają styczność ze środowiskiem zewnętrznym. Starsze szacunki mówiły, że ludzkie ciało zamieszkuje 10 razy więcej komórek bakterii niż wynosi liczba komórek człowieka [Savage, 1977]. Nowsze szacunki wskazują raczej na proporcje zbliżone 1:1 [Sender et al., 2016b, Sender et al., 2016a], co wciąż oznacza ogromną liczbę $\approx 4 \times 10^{13}$ komórek mikroorganizmów, z których większość znajduje się jelitach. Gwałtowna kolonizacja jelit rozpoczyna się tuż po narodzinach podczas gdy płód uznaje się raczej za sterylny, chociaż pewne ilości bakterii wykrywa się nawet w łożysku [Rodríguez et al., 2015].

	Bakterie (prokarionty)	Eukarionty
Miejsce DNA w komórce	cytoplazma	jądro komórkowe
liczba chromosomów	zwykle 1 (+ ew. plazmidy)	kilka - kilkaset
struktura chromosomów	kolista	liniowa
długość genomu	100 kb - 10 Mb	10 Mb - 100 Gb
białka histonowe	nie	tak
sekwencje kodujące	większość (ok. 90%)	mało (kilka %)
obecność splicingu	nie (zwykle)	tak
Rozmnażanie	bezpłciowe	płciowe lub bezpłciowe

Tablica 1.1.1: Podstawowe różnice w budowie i organizacji genomu między bakteriami (prokariontami) a eukariontami.

Różne rejony ciała człowieka tworzą odrębne nisze środowiskowe, zatem posiadają swoje unikalne zestawy mikroorganizmów czyli mikrobioty – m.in. skóra, jama ustna, poszczególne części układów: rozrodczego, moczowego, czy pokarmowego. W różnych badaniach zidentyfikowano do tej pory ponad 2 tys. gatunków bakterii jelitowych [Thursby and Juge, 2017], wśród nich część jest całkowicie beztlenowa. Zidentyfikowane gatunki należą do 12 typów, w zdecydowanej większości do *Proteobacteria*, *Firmicutes*, *Actinobacteria* i *Bacteroidetes*, ale np. znaleziono jedynie jednego reprezentanta typu *Verrucomicrobia* – *Akkermansia muciniphila*. Skład gatunkowy a nawet proporcje ilościowe wyższych poziomów taksonomicznych mogą się znacznie różnić pomiędzy poszczególnymi osobami [Eckburg et al., 2005]. Szacuje się, że w jelicie jednego człowieka żyją bakterie należące do ok. 1000 gatunków [Lozupone et al., 2012] oraz archeony, i w mniejszej liczbie grzyby czy protisty.

Skład mikrobioty, w szczególności mikrobioty jelitowej, zależy od wielu czynników. Należą do nich oczywiście dieta [David et al., 2014] czy genetyka gospodarza [Goodrich et al., 2014], ale też np. wiek [O'Toole and Jeffery, 2015]. Również pochodzenie etniczne jest związane z mikrobiotą, chociaż nie jest jasne, czy relacja jest związana z wpływem kultury, np. spożywaniem pewnych potraw typowych dla danej kultury, czy genetyki [Deschasaux et al., 2018]. Miejsce zamieszkania (geograficzna lokalizacja) [He et al., 2018] też jest związane ze składem mikrobioty, co może utrudniać przeniesienie wyników badań pomiędzy populacjami mieszkającymi w różnych regionach. Mikrobiota kształtowana jest również przez sam organizm człowieka np. poprzez produkcję kwasów żółciowych i przez układ immunologiczny, w tym poprzez produkcję przeciwciał IgA [Thursby and Juge, 2017].

Właśnie ze względu na tak duże zróżnicowanie mikrobiomu po zsekwencjonowaniu ludzkiego genomu, podjęte zostały również wysiłki aby zsekwencjonować i skatalogować i ludzki mikrobiom. W ramach Human Microbiome Project [Peterson et al., 2009] scharakteryzowano mikrobiomy pobrane (początkowo) od kilkuset osób, a w dalszej fazie (Integrative Human Microbiome Project) skupiono się dodatkowo na poznaniu wzajemnego oddziaływania mikrobiomu i człowieka.

W zależności od metodologii badań udaje się czasami zidentyfikować do ok. 100 gatunków, które są wspólne dla (prawie) wszystkich zdrowych ludzi, jednak nie ma powszechnej zgody co do istnienia i składu gatunkowego „rdzenia" mikrobiomu koniecznego dla zdrowia człowieka. Dużo większą stabilność mikrobiomu można zaobserwować pod względem funkcjonalnym, czyli zapisanego w genach potencjału do pełnienia pewnych funkcji, takich jak produkcja enzymów, elementów ścieżek metabolicznych [Lozupone et al., 2012, Huttenhower et al., 2012].

1.2.2. Wpływ mikrobiomu jelitowego na gospodarza

Bakterie żyjące w jelitach produkują wiele metabolitów, które mogą tak jak składniki pokarmowe być wchłaniane z przewodu pokarmowego, wpływając w ten sposób na cały organizm. Bakterie biorą też udział w fermentowaniu i wchłanianiu niestrawionych węglowodanów, np. w trawieniu skrobi. W wyniku fermentacji błonnika w jelitach powstają krótkołańcuchowe kwasy tłuszczowe (SCFA, ang. *short-chain fatty acids*), np. maślany, które są wchłaniane do żyły wrotnej, którą docierają najpierw do wątroby. SCFA pełnią w organizmie wiele funkcji, włączając w to regulację apetytu. Innym przykładem istotnej roli bakterii na podaż składników pokarmowych są witaminy z grupy B oraz witamina K i PP, które są syntetyzowane przez różne gatunki bakterii jelitowych [Hill, 1997]. Bakterie jelitowe mogą również przekształcać niewchłonięte wcześniej pierwotne kwasy żółciowe do wtórnych kwasów żółciowych, biorących udział w trawieniu. Wszystko to może mieć wpływ na występowanie chorób metabolicznych takich jak syndrom metaboliczny, otyłość i cukrzyca typu 2 [Zhang et al., 2010, Wu et al., 2020].

Bakterie jelitowe są powiązane z odpornością. Mogą produkować metabolity działające przeciwzapalnie lub działać na inne, jeszcze niepoznane sposoby [Thursby and Juge, 2017]. Biorą także udział w formowaniu układu immunologicznego, np. poprzez stymulację produkcji przeciwciał IgA. Wykazano również wpływ bakterii jelitowych m.in. na zaburzenie wytwarzania limfocytów T u myszy pozbawionych mikrobiomu jelitowego [Mazmanian et al., 2005]. Mikrobiota jelitowa ma duży wpływ na formowanie błony śluzowej jelit, np. stymulując produkcję śluzu. Część z bakterii zasiedlających jelita tworzy barierę ściśle przylegającą do nabłonka utrudniając w ten sposób patogenom dostęp do ściany jelita. Dlatego mikrobiom jest wiązany np. z zespołem jelita drażliwego.

Dosyć zaskakujący jest wpływ mikrobiomu na fizjologię, rozwój oraz działanie układu nerwowego [Sekirov et al., 2010, Sgritta et al., 2019]. *Oś jelitowo-mózgowa (ang.* gut-brain axis*)*, czasami nazywana osią mikrobiom-jelita-mózg, to dwukierunkowa biochemiczna komunikacja między przewodem pokarmowym a nerwowym, która jest związana z wieloma procesami zachodzącymi w mózgu, m.in. ze stanami lękowymi i depresją [Foster and Neufeld, 2013, Rhee et al., 2009]. Szczegółowy mechanizm wpływu mikrobiomu na psychikę nie jest znany, jednak wiadomo, że niektóre bakterie jelitowe mogą produkować neuromodulatory, stymulować komórki nabłonka jelit do ich produkcji [Rhee et al., 2009] lub pośrednio wpływać na ich produkcję dostarczając substratów. Do takich związków należy tryptofan, który jest prekursorem serotoniny. Bakterie mogą również aktywować nerw błędny łączący jelita bezpośrednio z mózgiem – są dowody na udział (u myszy) nerwu błędnego w przekazywaniu wpływu mikrobiomu na objawy związane z autyzmem [Sgritta et al., 2019].

Wpływ na rozwój wielu chorób oraz potencjał do ich leczenia jest intensywnie badany i dyskutowany. Dotyczy to wymienionych wcześniej zespołu jelita drażliwego, depresji, autyzmu ale też np. alergii, żeby wymienić tylko niektóre. Duża osobnicza różnorodność oraz pewna zmienność mikrobiomu, horyzontalny transfer genów i bliskie pokrewieństwo niektórych bakterii jelitowych, oraz zależność „zachowań" bakterii od czynników takich jak podaż pewnych składników w diecie utrudniają czy wręcz uniemożliwiają wskazanie pojedynczych gatunków odpowiedzialnych za pewne rodzaje interakcji z gospodarzem, gdyż wpływ ten może zależeć od wspólnego lub wymiennego wpływu różnych bakterii. Dlatego istotne jest badanie nie tylko składu gatunkowego ,ale również aktywności mikrobiomu w różnych warunkach, co można zrobić badając aktywne geny, czyli sekwencjonując metatranskryptom.

1.3. Techniczne aspekty badań genomu i transkryptomu – sekwencjonowanie

Odkrycie w 1953 roku struktury przestrzennej DNA rozpoczęło nową erę badań genomu, dając podstawy do poznania mechanizmu transkrypcji oraz sposobu kodowania w DNA sekwencji białek. Trzeba było jednak czekać kolejne dziesięciolecia aby odczytywanie kolejności nukleotydów w cząsteczkach DNA stało się możliwe na dużą skalę [Heather and Chain, 2016].

Wynikiem **sekwencjonowania** czyli techniki eksperymentalnej umożliwiającej poznanie kolejności nukleotydów w sekwencjach biologicznych jest zbiór **odczytów** – krótkich słów nad alfabetem $\Sigma = \{A, C, T, G\}$ odpowiadającym nukleotydom występującym w DNA. Długość odczytu zależy od użytej technologii sekwencjonowania. W technologiach wysokoprzepustowych (HTS, ang. *High-throughput sequencing*) otrzymuje się dużą liczbę odczytów o długości od kilkudziesięciu do kilkudziesięciu tysięcy znaków czyli **par zasad**, (ang. *base pairs*, w skrócie **bp**). Odczyt odpowiada zwykle skrajnej (początkowej lub końcowej) części pewnej cząsteczki DNA, która znajdowała się w próbce. Cząsteczki te są uzyskiwane podczas przygotowywania próbki, po ekstrakcji DNA (lub RNA) często potrzebna jest fragmentacja tych długich sekwencji np. poprzez cięcie enzymami restrykcyjnymi albo przy użyciu ultradźwięków.

Otrzymanie znacznie (o kilka rzędów wielkości) dłuższych odczytów ułatwiło by sekwencjonowanie bez fragmentacji, jest ono jednak wciąż wyzwaniem, głównie dlatego, że wraz z długością odczytu wzrasta liczba błędów (patrz 1.3.6). Zarówno częstość występowania błędów, jak też ich rodzaj, zależy od użytej metody sekwencjonowania i wpływa na dalszą analizę bioinformatyczną. Ze względu na błędy, konieczność fragmentacji DNA przed sekwencjonowaniem i pewną losowość procesu konieczne jest wygenerowanie większej liczby odczytów niż wskazywałaby na to długość sekwencjonowanej cząsteczki. Zwykle odczytów powinno być co najmniej kilkukrotnie więcej, tak aby oczekiwana liczba odczytów reprezentujących każde miejsce sekwencji wynosiła kilka lub kilkadziesiąt odczytów.

W tym podrozdziale omówię krótko historię i metody sekwencjonowania skupiając się na tym, jak wyglądają typowe wyniki sekwencjonowania oraz jakie możliwości, ale też błędy i obciążenia, niosą ze sobą współczesne technologie sekwencjonowania.

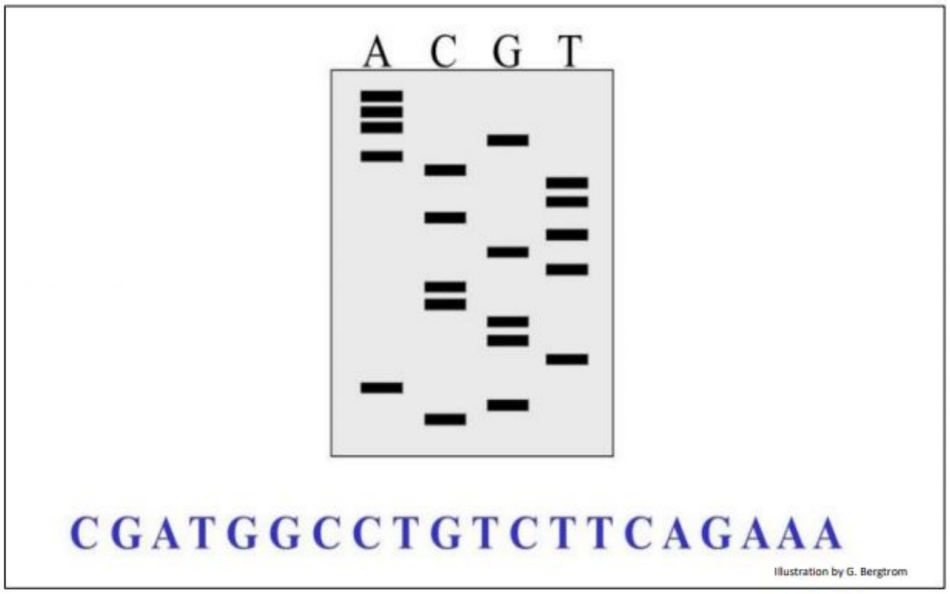
1.3.1. Powielanie DNA – metoda PCR

Do sekwencjonowania potrzebne jest uzyskanie odpowiednio dużej ilości DNA. Często wymaga to powielenia DNA, które znajduje się w próbce. Historycznie wykorzystywano do tego plazmidy bakteryjne, w których umieszczano odpowiedni fragment. Niestety niektóre sekwencje (np. kodujące pewne substancje uniemożliwiające namnożenie bakterii) nie są możliwe do namnożenia w ten sposób. W latach 80-tych Mullis zaprojektował metodę, która zrewolucjonizowała badania genomu – łańcuchową reakcję polimerazy, zwaną PCR (ang. *polymerase chain reaction*) [Mullis et al., 1987]. Jest to cykl reakcji wykorzystujący polimerazę DNA odporną na wysoką temperaturę, a także cykliczną zmianę temperatury mieszaniny reakcyjnej. Dwuniciowe DNA jest rozdzielane na pojedyncze nici w wysokiej temperaturze, następnie w niższej temperaturze do pojedynczych nici DNA dołączają się krótkie oligomery – primery reakcji. Potem temperatura jest zmieniania na optymalną dla elongacji. Dzięki powtarzaniu tych trzech kroków liczba cząsteczek wzrasta wykładniczo, więc

stosunkowo tanio i szybko można uzyskać duże ilości DNA. Aby powielić odpowiednie fragmenty primery muszą być komplementarne do końców badanej sekwencji. Wymaga to znajomości skrajnych fragmentów sekwencji lub użycia losowych primerów.

1.3.2. Sekwencjonowanie metodą Sangera

Początkiem możliwości odczytywania sekwencji DNA na większą skalę była opublikowana w 1977 roku metoda Sangera [Sanger et al., 1977]. Wykorzystuje ona polimerazę DNA do budowy nici komplementarnej do jednoniciowej cząsteczki używanej jako matrycy. Dzięki dodaniu do standardowej mieszaniny nukleotydów zmodyfikowanych nukleotydów blokujących dalszą elongację DNA (brak im reszty hydroksylowej, która tworzy wiązanie z następnym nukleotydem) można w pewnym momencie zablokować dalszą syntezę. Przy zastosowaniu odpowiedniego primera reakcja rozpoczyna się w stałym miejscu sekwencji i trwa dopóki nie zostanie włączony zmodyfikowany nukleotyd. Następnie tak uzyskane cząsteczki jednoniciowego DNA są rozdzielane przy pomocy elektroforezy wg. długości. Początkowo by rozpoznać nukleotydy (A, C, T, G) należało przeprowadzić reakcję dla każdego z nich osobno, aby potem odczytać ich kolejność poprzez występowanie fragmentów o odpowiedniej długości w jednym z 4 pasków po elektroforezie (patrz rys. 1.3.1). Jednym z ulepszeń było wprowadzenie zmodyfikowanych nukleotydów o właściwościach fluorescencyjnych, świecących na różne kolory w zależności od rodzaju nukleotydu. To pozwalało odczytać sekwencję przy użyciu pojedynczego paska elektroforezy.



Rysunek 1.3.1: Schemat sekwencjonowania metodą Sangera. Fragmenty DNA kończące się danym nukleotydem są rozdzielone wg. długości, co pozwala odczytać kolejność liter w sekwencji.
Rysunek wykorzystany na podstawie licencji CC BY 3.0 [lib,]

Metoda Sangera pozwala uzyskać odczyty długości nawet do 1000 bp i wciąż bywa stosowana, np. do potwierdzania innych wyników sekwencjonowania, szczególnie w miejscach gdzie krótsze odczyty nie pozwalają dokładnie odtworzyć sekwencji. Jednak koszt i praco- oraz czasochłonność metody znacząco utrudniają pełne zanalizowanie tą metodą nawet małych genomów bakteryjnych. Wśród wad należy też

wymienić konieczność użycia primerów, a często też wcześniejszej amplifikacji (namnożenia) sekwencji.

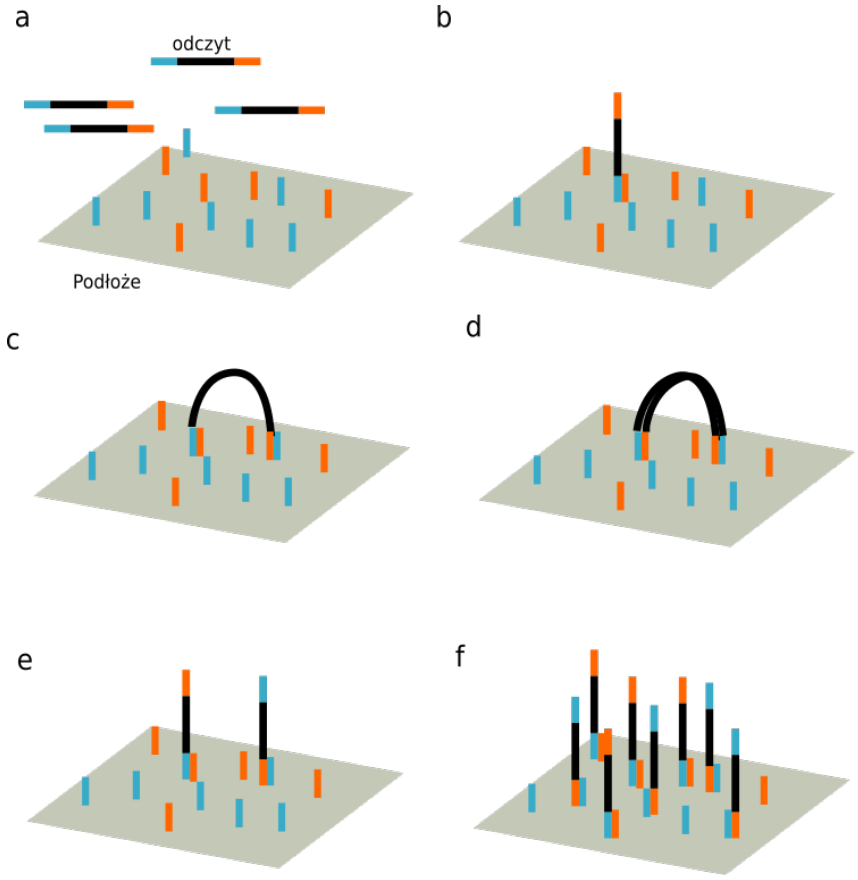
1.3.3. Sekwencjonowanie nowej generacji

Technologie sekwencjonowania nowej generacji (ang. *next-generation sequencing*, NGS) zwane są też wysoko-przepustowymi (ang. *high-throughput sequencing (HTS)*) pozwalają wygenerować jednocześnie nawet miliard odczytów dzięki zrównolegleniu procesu amplifikacji i sekwencjonowania. Pierwsze masowe zrównoleglenie uzyskano podczas pirosekwencjonowania [Ronaghi et al., 1998], i było udostępnione komercyjnie przez firmę 454 Life Sciences (a później Roche). To kolejna metoda bazująca na syntezie, czyli używająca polimerazy DNA. W tym przypadku jednak odczyt sekwencji odbywa się podczas reakcji, dzięki czemu nie trzeba wykonywać elektroforezy.

Pirosekwencjonowanie wykorzystuje pirofosforan uwalniany podczas reakcji przyłączenia nukleotydu przez polimerazę do wygenerowania w ciągu reakcji błysku światła. Matrycowe, jednoniciowe fragmenty DNA zostają pojedynczo dołączone do małych kulek, skopiowane tak, by do jednej kulki było doczepionych wiele kopii danego fragmentu DNA, a następnie umieszczone stabilnie w podłożu (w miarę możliwości jedna kulka na jedno zagłębienie w podłożu). Później kulki omywane są iteracyjnie cząsteczkami nukleotydów wraz z resztą substancji potrzebnych do przeprowadzenia wszystkich reakcji. Świecenie w obrębie danej kulki po dostarczeniu określonego nukleotydu jest rejestrowane jako kolejna pozycja w sekwencji danego fragmentu. Wadą tej metody jest trudność w identyfikacji sekwencji monomerowych – określenie długości ciągów tego samego nukleotydu dłuższych niż 4-5 bp jest bardzo trudne. Technologia ta pozwala uzyskać setki tysięcy odczytów o długości maksymalnie 300-500 bp, przegrała jednak konkurencję ze stworzoną niewiele później tańszą i szybszą platformą sekwencjonowania – Illuminą.

Platforma Illumina (Solexa) [Voelkerding et al., 2009] jest kolejną metodą wysokoprzepustowego sekwencjonowania przy pomocy syntezy. Fragmenty są w niej powielane w tzw. sposób mostkowy (ang. *bridge amplification*, patrz rysunek 1.3.2), ponieważ podczas tworzenia kopii fragment jest połączony z podłożem poprzez oba końce, tworząc łuk. Po zsyntetyzowaniu drugiej nici DNA jest denaturowane i powstają w ten sposób dwie nici – pierwotna i komplementarna do niej, połączone z podłożem jednym końcem. Procedura ta jest powtarzana wielokrotnie i w jej wyniku powstają skupiska fragmentów DNA. Każde skupisko powinno zawierać kopie (oraz kopie odwrotnie komplementarne) jednego wyjściowego fragmentu. Wszystkie skupiska są następnie sekwencjonowane synchronicznie dzięki zmodyfikowanym nukleotydom, które umożliwiają dalszą syntezę dopiero po usunięciu z nich fluoroforu [Turcatti et al., 2008]. Sekwencje wyznakowane odpowiednim fluoroforem są wzbudzane, a kolor ich świecenia rejestrowany jest przez czułe kamery, dzięki czemu można rozpoznać ostatnio dobudowany do sekwencji nukleotyd. Następnie fluorofory są enzymatycznie usuwane i w następnym cyklu kolejny nukleotyd może zostać przyłączony.

Mimo, że odczyty uzyskane technologią Illumina są relatywnie krótkie (początkowo było to do 35 bp, teraz standardowo osiągane jest co najmniej 100 bp), to do jej zalet należy m.in. możliwość sekwencjonowania obu końców fragmentu, tzw. tryb sparowanych końców (ang. *paired-end*, PE). Najpierw sekwencjonowane są końce odpowiadające kopiom pierwotnych sekwencji, zsekwencjonowane fragmenty są usuwane, a następnie sekwencjonowane są komplementarne kopie sekwencji. Dzięki temu, że znana jest oczekiwana długość fragmentu, którego końce są sekwencjonowane,



Rysunek 1.3.2: Schemat działania powielania mostkowego w sekwencjonowaniu w technologii Illumina. Fragment DNA z dołączonymi primerami przyczepia się do podłoża zawierającego komplementarne do primerów sekwencje (a i b). Drugi koniec fragmentu dołącza się do podłoża tworząc mostek (c) i jest kopiowany (d) w wyniku czego obok pierwotnego fragmentu widzimy DNA odwrotnie komplementarne (e). Cykl powtarzany jest wielokrotnie aby uzyskać w pobliżu wiele kopii (f), aby przy sekwencjonowaniu z fluoforami uzyskać wykrywalny poziom świecenia.

dostajemy znacznie więcej informacji niż mając dwa razy więcej odczytów, ale niesparowanych. Informacja o sparowaniu odczytów ułatwia mapowanie, szczególnie na regiony repetytywne, wykrywanie różnego rodzaju mutacji, szczególnie rearanżacji genomu, czy alternatywnego splicingu [Heather and Chain, 2016].

Dodatkową zaletą technologii Illumina jest również zwracana przez nią ocena jakości każdego odczytanego nukleotydu. Wartość ta określa pewność właściwego odczytania danego nukleotydu i pozwala odrzucić lub skrócić odczyty na jej podstawie. Dzięki swoim zaletom platforma Illumina jest teraz bardzo często wykorzystywana zarówno w badaniach naukowych jak i w zastosowaniach komercyjnych. Została również użyta w przypadku danych eksperymentalnych, które analizuję w tej pracy.

1.3.4. Sekwencjonowanie trzeciej generacji

Mianem sekwencjonowania trzeciej generacji określa się zwykle metody, które pozwalają sekwencjonować pojedyncze cząsteczki DNA, unikając tym samym konieczności powielania DNA [Van Dijk et al., 2014]. W tej dziedzinie dominują aktualnie 2 technologie – PacBio i Nanopore, obie pozwalające uzyskać odczyty długości

wielu tysięcy bp, jednak szansa błędnego odczytania pojedynczego nukleotydu wynosi kilka-kilkanaście procent, ale wciąż się zmniejsza, również dzięki nowym algorytmom redukcji błędów [Kono and Arakawa, 2019]. Dzięki nim konsensusowe sekwencje są już dużo bardziej dokładne. Wciąż jednak częstość błędów jest większa niż dla sekwencjonowania nowej (drugiej) generacji, co stawia zupełnie nowe wyzwania przy bioinformatycznej analizie takich danych. Technologie sekwencjonowania trzeciej generacji dają jednak ogromne możliwości, np. sekwencjonowania repetytywnych fragmentów genomu.

Porównanie opisanych technik sekwencjonowania przedstawiam w tabeli 1.3.1.

	Sanger	454	Illumina	3 generacja
długość odczytu [bp]	1000	300-500	2 × 100-300	2000
częstość błędu	0.001% - 1%	1%	0.26-0.8%	1.78-38.2%
liczba odczytów	1	500 mln	50-8000 mln	0.03-0.1 mln

Tablica 1.3.1: Porównanie niektórych technik sekwencjonowania nowej generacji, za [Ambardar et al., 2016]. Szczególnie dla sekwencjonowania trzeciej generacji parametry wciąż się poprawiają, także wyniki te nie są już aktualne, ale mogą oddawać proporcje i z takimi danymi możemy się jeszcze spotkać.

1.3.5. Sekwencjonowanie RNA

Sekwencjonowanie RNA jest możliwe dzięki wykorzystaniu odwrotnej transkryptazy, enzymu który pozwala syntetyzować DNA na podstawie matrycy RNA. Powstaje wtedy tzw. cDNA (ang. *complementary DNA*), czyli DNA komplementarne do RNA. cDNA jest bardziej stabilne i może zostać namnożone w reakcji PCR i sekwencjonowane tak samo jak w omówionych wcześniej technikach sekwencjonowania DNA. Jednak ponieważ mRNA w komórce to jedynie ok. 10% RNA, w celu zbadania transkryptomu trzeba oddzielić mRNA od innego RNA albo przynajmniej usunąć rRNA. Usunięcie rybosomalnego RNA (**deplecja rRNA**, ang. *rRNA depletion*) jest możliwe dzięki temu, że sekwencje rRNA są dobrze zachowane ewolucyjnie. Dzięki temu łatwo skonstruować dosyć uniwersalne sekwencje, które, połączone z magnetycznymi kulkami, po przyłączeniu do komplementarnego im rRNA umożliwiają usunięcie go z próbki.

W technikach sekwencjonowania RNA możliwe jest również w dużej mierze zachowanie informacji o nici pochodzenia sekwencji (protokoły ang. *strand-specific*) poprzez dobudowanie do pierwszej wyprodukowanej nici DNA znacznika w postaci sekwencji nukleotydów C (raczej niespotykanej w transkryptach), dzięki czemu po standardowym namnożeniu fragmentu DNA sekwencja ta wskazuje na pierwotną orientację transkryptu.

1.3.6. Błędy w sekwencjonowaniu nowej generacji

Sekwencjonowanie nowej generacji, jak każda technika eksperymentalna, obarczone jest pewnymi błędami. Mogą one zdarzyć się na każdym etapie: podczas powielania, podczas syntezy czy podczas przetwarzania danych. Powielanie poprzez PCR (używane przed sekwencjonowaniem czy podczas powielania mostkowego) może wygenerować mutacje punktowe w przypadku „pomyłki" polimerazy, szczególnie w pierwszych cyklach, gdy taki błąd zostaje powielony w wielu kopiach. Takie mutacje

są kłopotliwe w późniejszej analizie, jednak ponieważ prawdopodobieństwo ich wystąpienia w każdym miejscu sekwencji jest (mniej więcej) takie samo i jest bardzo małe, to przy odpowiednio dużym pokryciu są możliwe do skorygowania. Większym problemem jest skłonność PCR do pomijania sekwencji o dużej zawartości cytozyny i guaniny w sekwencji [Aird et al., 2011]. Wiązanie G-C jest silniejsze niż wiązanie A-T stąd bogate w nie sekwencje są trudniejsze do zdenaturowania, bez którego reakcja nie zachodzi. Sekwencje palindromowe mogą uniemożliwić dalsze sekwencjonowanie ponieważ jednoniciowe palindromowe DNA może zhybrydyzować samo ze sobą.

Tego typu sekwencjonowanie jest też podatne na błędy niezależne od PCR. O ile błędy punktowe są łatwe do skorygowania w późniejszej obróbce, to delecje czy insercje w technologii Illumina zmniejszają jakość sygnału bo powodują „rozminięcie się" w fazie części fragmentów DNA. Po nawarstwieniu się takich błędów rozpoznanie dołączanego aktualnie nukleotydu jest bardzo utrudnione. Takim sytuacjom sprzyjają sekwencje składające się z kilku takich samych nukleotydów, bo przypadkowo dołączony może zostać więcej niż 1 nukleotyd na cykl. Stąd prawdopodobieństwo, że dany nukleotyd jest błędnie odczytany rośnie wraz z długością [Dohm et al., 2008]. Również szansa na pojawienie się danej substytucji może się różnić nawet 10-krotnie [Dohm et al., 2008]. Dlatego rozróżnienie pomiędzy prawdziwą zmiennością genetyczną w próbce a błędami sekwencjonowania jest dużym wyzwaniem. Sekwencjonując genom diploidalny dostajemy zmieszane odczyty z dwóch bardzo podobnych zestawów sekwencji. Niezgodności w sekwencji powinny występować wtedy w częstościach o stosunku bliskim 1:1. W badaniu metagenomu nie ma w zasadzie możliwości, żeby na podstawie częstości w odczytach odróżnić rzadką mutację od błędu sekwencjonowania.

Ilościowe badanie materiału genetycznego jest również obarczone pewnymi systematycznymi błędami związanymi głównie z reakcją PCR, ale nie tylko. Założenie o równej szansie na wygenerowanie odczytów niezależnie od pozycji w sekwencji referencyjnej często nie jest zatem zgodne z realiami eksperymentów. Wynika to z wielu czynników:

- trudniejszej ekstrakcji niektórych fragmentów DNA i RNA, np. związanych silniej z białkami
- różnej stabilności cząsteczek DNA i RNA zależnej od sekwencji fragmentu
- artefaktów PCR, w tym większej amplifikacji niektórych cząsteczek zależnej od zawartości par GC lub losowego zdarzenia
- artefaktów sekwencjonowania, np. częstości popełnianych błędów zależnych od sekwencji odczytu (m.in. trudności z dokładnym uzyskaniem liczby powtórzeń w dłuższych sekwencjach monomerów). Takie odczyty mogą nie spełnić warunków jakości i zostać odrzucone powodując mniejsze pokrycie danego regionu

Wszystkie wspomniane czynniki wpływają na analizę danych z sekwencjonowania, a w przypadku sekwencjonowania metatranskryptomu powiększają i tak dużą liczbę wyzwań.

1.3.7. Główne wyzwania w badaniu metatranskryptomu

W metatranskryptomice stawiane są dwa wyzwania: identyfikacja transkryptów oraz zwykle również ich kwantyfikacja. W pierwszym przypadku celem jest poznanie sekwencji transkryptów, w drugim, często wynikającym z pierwszego, celem jest

ocena częstości występowania danego transkryptu, czyli pomiar ilościowy. Są dwa podejścia do tego zagadnienia, które omawiam dokładniej w następnym rozdziale – poprzez złożenie sekwencji transkryptów *de novo* lub poprzez mapowanie na znane genomy. Oba te podejścia mają swoje znaczące ograniczenia, ale niektóre wyzwania są wspólne, tak jak polimorfizm genomów.

W pojedynczej próbce znajdują się zapewne różne dalekie filogenetycznie bakterie, ale też komórki bakterii blisko ze sobą spokrewnione i nawet jeśli należą do jednego gatunku, a nawet szczepu, to ich genomy i transkryptomy będą się różnić. Ten **polimorfizm** jest trudny do odróżnienia od błędów sekwencjonowania. Do pewnego stopnia należy się pogodzić z tym, że rekonstruując genomy czy transkrypty dokona się uproszczenia różnorodności genetycznej występującej w próbce i wybierze tylko najczęstsze warianty genetyczne. Przy rozpoznawaniu gatunków na podstawie znanych genomów trudnością jest wyznaczenie granicy między tym, kiedy odczyt jest wystarczająco podobny by przypisać go znanemu genomowi bakterii, a kiedy należałoby go zaklasyfikować do gatunku, którego genom nie został jeszcze poznany. Polimorfizm i błędy sekwencjonowania wpływają też na złożoność obliczeniową zarówno metod rekonstrukcji sekwencji jak i przypisania odczytów genomom.

Badanie metatranskryptomu ma wiele cech wspólnych z badaniem metagenomu. Tak jak w badaniu metagenomu, w odtwarzaniu metatranskryptomu *de novo* na podstawie odczytów skład gatunkowy próbki jest nieznany, a zatem nie wiadomo jak wiele sekwencji należy odtworzyć. W przypadku przypisywania odczytów do znanych genomów może się zdarzyć, że genomu bakterii nie ma w bazie, ale wraz ze wzrostem zsekwencjonowanych genomów wyzwaniem staje się rozstrzygnięcie prawdziwości przypisania gdy odczyt pasuje do więcej niż jednego genomu. Zwykle dotyczy to bardzo blisko spokrewnionych gatunków i pojedynczy błąd sekwencjonowania czy polimorfizm może spowodować niewłaściwe zaklasyfikowanie, a w konsekwencji rozpoznanie bakterii, której w próbce nie ma.

Częstości poszczególnych bakterii w metagenomach są różne, podobnie jak w transkryptomie częstości transkryptów, więc dużo więcej odczytów będzie pochodzić z częstych sekwencji. Rzadko występujące sekwencje mogą nie być nawet całkowicie reprezentowane w odczytach, a więc będą one w wynikowej rekonstrukcji podzielone na wiele fragmentów. Wiele metod analizy wyników sekwencjonowania wykorzystuje estymację częstości sekwencji w próbce. W metatranskryptomie jest ona jeszcze trudniejsza, bo sekwencje są liczne i krótkie.

W analizie różnicowej transkrypcji wyzwaniem staje się wyznaczenie, które zmiany w ekspresji są statystycznie istotne, a które są związane niedokładnością metod eksperymentalnych lub losowymi zmianami w ekspresji. Problemem w eksperymentach z wykorzystaniem sekwencjonowania nowej generacji jest również to, że obserwowane liczby odczytów zależą od głębokości sekwencjonowania, a nie od bezwzględnej liczby transkryptów (np. w pojedynczej komórce albo w jednostce objętości). Częstym podejściem jest normalizacja przez całkowitą liczbę odczytów, ale w tej dziedzinie powstało też wiele modeli statystycznych służących do opisu różnicowej ekspresji szerzej, np. estymujących bazowy poziom ekspresji w próbce. Zakładają one często np. że ekspresja większości genów jest niezmienna, a także niezależność zmian w ekspresji poszczególnych genów. W transkryptomice przynajmniej to pierwsze założenie jest spełnione, gdyż znaczna część genów, np. odpowiadająca za zwykłą homeostazę w komórce, powinna być stale aktywna na podobnym poziomie, w metatranskryptomice raczej takie założenie będzie błędne, chociaż im bardziej kontrolowane warunki eksperymentu, tym bardziej można liczyć na mniej więcej stały

skład bakteryjny, dzięki czemu zmiany aktywności genów całej populacji bakterii będą bardziej podobne do zmian w transkryptomice. Jednak do wskazań o statystycznie istotnie zmienionych genach w badaniach metatranskryptomicznych trzeba podchodzić ostrożnie. Są prace, które wskazują, że metody analizy różnicowej ekspresji stosowane w transkryptomice, w badaniach metagenomicznych nie kontrolują dobrze FDR, przeszacowując istotność zmiany częstości niektórych bakterii [Hawinkel et al., 2019].

Chociaż w metatranskryptomice nie trzeba uwzględniać alternatywnego splicingu, to warto mieć na uwadze policistronowe mRNA. Badane sekwencje mogą zawierać kilka genów, ale możemy obserwować też już mRNA podzielone na geny, zatem tak jak w transkryptomice oczekiwane pokrycie jednej sekwencji odczytami może się skokowo zmieniać. To tylko jedno z wielu wyzwań, które dotyczą badania aktywności bakterii, ale bardzo wiele z trudności jest wspólnych dla analizy różnych eksperymentów wykorzystujących sekwencjonowanie nowej generacji. W następnym rozdziale przedstawiam formalnie podejścia do analizy danych NGS i związane z tym problemy obliczeniowe i opisuję strategie rozwiązań.

Podsumowując, rozwój oraz znaczny spadek kosztów sekwencjonowania wysokoprzepustowego pozwolił nie tylko na zsekwencjonowanie genomu człowieka i powstanie licznych nowych technik eksperymentalnych bazujących na wysokoprzepustowym sekwencjonowaniu. Wpłynął on też znacząco na badania mikrobiomu, umożliwiając sekwencjonowanie **metagenomów** – genomów całych populacji mikroorganizmów z danego środowiska bez konieczności uzyskania dużej ilości materiału genetycznego, np. poprzez wcześniejszą hodowlę poszczególnych gatunków bakterii. Dało to początek nowym dziedzinom biologii – **metagenomice** i **metatranskryptomice**.

ROZDZIAŁ 2

Obliczeniowe metody analizy wyników eksperymentów NGS

W wyniku sekwencjonowania wysokoprzepustowego nowej generacji (ang. *high-throughput next generation sequencing*) otrzymujemy ogromny zbiór krótkich odczytów, powstały z mieszaniny kopii fragmentów analizowanych sekwencji biologicznych, np. fragmentów DNA lub RNA. Pojedyncze, nieuporządkowane odczyty, szczególnie o typowych aktualnie długościach (ok. 100 bp) niosą same w sobie niewiele informacji. Zwykle interesujące jest przyporządkowanie odczytów do większej struktury, jak na przykład genu, który kodują, określonego typu fragmentów genomu, z którego zostały uzyskane, lub w przypadku próbek metagenomicznych genomu organizmu. Najczęstszą procedurą jest mapowanie na zbiór sekwencji referencyjnych (czyli przypisanie odczytom dokładnych pozycji w sekwencjach referencyjnych, z których prawdopodobnie mogły zostać uzyskane), a następnie zliczanie odczytów zmapowanych na wybrane regiony (np. geny) lub wyliczanie pokrycia (dla danej pozycji - ile odczytów zawiera ją w swoim mapowaniu). W przypadku, gdy sekwencje referencyjne nie są znane, mogą być one złożone („zasemblowane", ang. *assembly*) *de novo*.

W poniższym rozdziale przedstawione są metody obliczania nieznanego genomu referencyjnego (i innych sekwencji referencyjnych) na podstawie krótkich odczytów z sekwencjonowania NGS – **asemblację *de novo***, a następnie niektóre metody wykorzystujące genom referencyjny do analizy krótkich odczytów, w szczególności służące do mapowania odczytów na genom. Przedstawione są też metody szacowania liczności sekwencji referencyjnych na podstawie odczytów NGS.

2.1. Asemblacja genomu *de novo*

Asemblacja (ang. *assembly*) *de novo* to rekonstrukcja nieznanej sekwencji referencyjnej (genomu, transkryptomu) na podstawie jej krótszych fragmentów, np. odczytów NGS. W literaturze pojawiają się 2 podstawowe techniki asemblacji - wykorzystujące grafy nałożeń odczytów, oraz grafy *de Bruijna*, jednak ich implementacji jest znacznie więcej. Programy do asemblacji różnią się głównie heurystykami służącymi do rozwiązywania dwóch głównych problemów spotykanych przy asemblacji: problemu z ustalaniem kolejności fragmentów w przypadku występowania w rekonstruowanej sekwencji dłuższych powtórzeń, oraz polimorfizmów (wynikających z błędów sekwencjonowania lub genetyki badanego organizmu czy populacji), które powodują znaczny wzrost trudności obliczeniowej, a nawet uniemożliwiają jednoznaczne rozwiązanie problemu.

2.1.1. Sformułowania problemu asemblacji *de novo*

Odczytem nazwiemy słowo nad alfabetem $\Sigma = \{A, C, G, T\}$ zadane tablicą $s[1..n]$. Długość odczytu s oznaczmy $|s|$. W problemie asemblacji *de novo*, mając dany zbiór odczytów $S = \{s_1, s_2, ...s_n\}$, szukamy słowa (ew. zbioru słów) R – sekwencji referencyjnej (zwykle sekwencji genomu). Na ogół odczyty są tej samej lub bardzo podobnej długości i często są dużo krótsze od szukanego słowa R.

Definicja 2.1.1 (Podsłowo)
Słowo t jest podsłowem słowa s (ozn. $\Subset$) jeśli istnieją takie $i < j$, że $t = s[i..j]$. Wtedy s jest supersłowem słowa t ($s \Supset t$).

Definicja 2.1.2 (Słowo odwrotnie komplementarne)
Niech $\overline{s}$ oznacza słowo odwrotnie komplementarne do s, czyli sekwencję nukleotydów na nici komplementarnej. Dla pojedynczych symboli jest to symbol zgodny z zasadą komplementarności nukleotydów: $\overline{A} = T$, $\overline{C} = G$, $\overline{T} = A$, $\overline{G} = C$.

Dla słowa s o długości $|s|$ $\overline{s}$ to takie słowo o długości $|s|$, dla którego zachodzi:

$$\forall_{1 \leq i \leq |s|} s[i] = \overline{s[|s| - i + 1]}$$

Zauważmy, że $\overline{\overline{s}} = s$, co jest spodziewane biorąc pod uwagę, że tak operacja reprezentuje dwuniciowość DNA.

W kontekście DNA pojęcie palindromu jest zdefiniowane właśnie przy pomocy odwrotnej komplementarności, czyli palindromem jest s spełniające $\overline{s} = s$. Zauważmy, że mogą być one tylko parzystej długości, ponieważ żaden pojedynczy nukleotyd nie jest komplementarny ze sobą. Sekwencje palindromowe pojawiają się w wielu genomach i są istotne ze względu na swoją zdolność do tworzenia struktur o kształcie „szpilek" poprzez stworzenie wiązań pomiędzy nukleotydami na tej samej nici, jak również ze względu na to, że wiele enzymów restrykcyjnych tnie DNA specyficznie w miejscu występowania pewnych sekwencji palindromowych. W kontekście asemblacji są jednak istotne ponieważ takie sekwencje mogą wprowadzać pętle w omawianych później grafowych podejściach do rozwiązania problemu asemblacji.

W swoim najprostszym sformułowaniu problem asemblacji jest problemem SCS – najdłuższego wspólnego supersłowa.

Definicja 2.1.3 (Problem SCS – ang. „Shortest Common Superstring")
Dla danego zbioru słów $S = \{s_1, s_2, \ldots s_n\}$ znajdź najkrótsze słowo R, takie, że

$$\forall_{s_i \in S} s_i \Subset R$$

W podstawowym sformułowaniu problemu rekonstrukcji słowa z jego fragmentów szukane jest najkrótsze słowo, którego podsłowami są słowa z zadanego zbioru („Shortest common superstring"). Problem ten był rozważany jeszcze przed rozwojem sekwencjonowania. Jest on NP-zupełny [Maier and Storer, 1977], a popularnym rozwiązaniem aproksymacyjnym jest np. zachłanny algorytm iteracyjnego łączenia par słów o najdłuższym nałożeniu. Problem SCS jest dużym uproszczeniem problemu asemblacji ponieważ nie bierze pod uwagę:

1. dwuniciowości DNA — powodującej to, że odczyty pochodzą z obu nici komplementarnych, podczas gdy rekonstruowana jest jedna z nich

2. błędów sekwencjonowania (i polimorfizmów DNA) — które w przypadku próby dokładnej reprezentacji każdego odczytu prowadzą do znacznego wzrostu długości wynikowej sekwencji referencyjnej i wprowadzają do niej błędne sekwencje, czego należałoby uniknąć

3. powtórzeń w sekwencji referencyjnej (podsłów występujących więcej niż raz i dłuższych niż długość odczytu) — co prowadzi do nieprawidłowego odtworzenia sekwencji referencyjnej

Wymienione powyżej zagadnienia można uwzględnić w innym sformułowaniu problemu asemblacji, który przedstawię po wprowadzeniu nowych pojęć adresujących powyższe problemy. Jako pierwszą można uwzględnić dwuniciowość DNA rozszerzając definicję podsłowa o ewentualną odwrotną komplementarność.

Definicja 2.1.4 (Podsłowo opcjonalnie odwrotnie komplementarne)
Słowo t jest podsłowem opcjonalnie odwrotnie komplementarnym s (ozn. $t \sqsubseteq s$), jeśli $t \boxdot s$ lub $\overline{t} \boxdot s$ (co jest równoważne z $t \boxdot \overline{s}$).

Podobnie można uwzględnić jeden (bardzo częsty) z błędów sekwencjonowania, czyli substytucję (zamianę jednego nukleotydu na inny, ang. *mismatch*) poprzez rozszerzenie równości słów o maksymalną liczbę lub dopuszczalny odsetek możliwych błędów substytucji.

Definicja 2.1.5 (Dopasowanie z błędami)
Dla ustalonej liczby błędów m: Słowo t jest dopasowane do słowa s (ozn. $t \simeq_m s$) jeśli liczba pozycji, na których s i t mają różne symbole, jest mniejsza lub równa m, czyli

$$\left|\{i : s[i] \neq t[i]\}\right| \leq m$$

Dopasowanie z błędami bywa czasami definiowane dla ustalonego odsetka błędów ϵ: słowo t jest dopasowane do słowa s (ozn. $t \simeq_\epsilon s$) jeśli częstość różnic między s i t jest mniejsza niż epsilon, czyli

$$\left|\{i : s[i] \neq t[i]\}\right| < \epsilon \cdot |s|$$

Ponieważ te sformułowania są równoważne dla odczytów tej samej długości, a poza tym częściej używane jest ustalanie liczby, (a nie odsetka) błędów, w tej pracy używać będę ograniczenia liczby błędów (chyba że zaznaczone inaczej).

Dopuszczalna liczba substytucji (lub odsetek błędów) jest ważnym parametrem. Im więcej dozwolonych błędów w dopasowaniu, tym bardziej kosztowne obliczeniowo są operacje dopasowania odczytów, które wykonywane są zarówno w asemblacji jak i mapowaniu (patrz 2.2). Ponieważ jednak odczyty w technologiach sekwencjonowania nowej generacji mają błąd sekwencjonowania (ang. *error-rate*) na poziomie ok. 0.1-1% to zakładając równe prawdopodobieństwo błędu niezależnie od pozycji w odczycie (i przyjmując optymistyczne założenie o błędzie równym 0.1%), dla odczytów o typowej długości 100bp ok. 9.5% z nich będzie zawierało przynajmniej 1 błąd, jednak mniej niż 0.5% odczytów 2 lub więcej błędów (korzystając z rozkładu Poissona). Dopuszczając zatem 1 błąd, przy powyższych założeniach, dodatkowe 9% odczytów ma szansę być uwzględnionych w analizie.

Definicja 2.1.6 (Podsłowo (komplementarne) z błędami)
Analogicznie słowo t jest podsłowem (komplementarnym) z błędami słowa s (ozn. $t \sqsubseteq_m s$) jeśli istnieją takie $i < j$, że

$$t \simeq_m s[i..j] \vee \bar{t} \simeq_m s[i..j]$$

Wtedy $s \sqsupseteq_m t$.

Definicja 2.1.7 (Asemblacja – wariant SCS)
Dla danego zbioru słów $S = \{s_1, s_2, \ldots s_n\}$ znajdź najkrótsze słowo R, takie, że

$$\forall_{s_i \in S} : s_i \sqsubseteq_m R$$

SCS w rozszerzonej o błędy wersji również jest NP-zupełny [Kececioglu, 1991]. Jednak w przypadku sekwencji biologicznych minimalizacja długości rekonstruowanego słowa nie jest konieczna, a nawet może prowadzić do błędów związanych z usunięciem powtarzających się fragmentów genomu. Aby jednak rozpoznawać powtórzenia sekwencji na podstawie jej fragmentów zdefiniujmy pojęcie *ułożenia* (ang. *layout*) odczytów.

Definicja 2.1.8 (Ułożenie odczytów)
Dla danej sekwencji odczytów $S = [s_1, s_2, \ldots s_n]$ ułożeniem odczytów nazwiemy sekwencję pozycji $P = [(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)]$, $1 \leq a_i, b_i \leq n$, i związane z nimi słowo R_P o długości n w taki sposób, że

$$\forall_{1 \leq j \leq n} \exists_{(a_i, b_i) \in P} \min(a_i, b_i) \leq j \leq \max(a_i, b_i)$$

oraz jeśli $a_i < b_i$ to:

$$s_i \simeq_m R_P[a_i, b_i]$$

w przeciwnym przypadku:

$$s_i \simeq_m \overline{R_P[b_i, a_i]}$$

Ułożenie odczytów opisuje zatem możliwe pochodzenie każdego odczytu, czyli (zakładając, że R_P jest właściwą sekwencją referencyjną) nie tylko pozycję odczytu jako podsłowa w R_P, ale też orientację odczytu, czyli informację, czy odczyt pochodzi z sekwencji R_P bezpośrednio (gdy $a_i < b_i$), czy z sekwencji odwrotnie komplementarnej.

Jeśli $m = 0$, czyli nie dopuszczamy błędów dopasowań, to ułożenie odczytów P definiuje odpowiadającą im sekwencję referencyjną R_P jednoznacznie, ale dana sekwencja może być czasami definiowana przez więcej niż jedno ułożenie. W przypadku $m > 0$ błędy prowadzą do powstania pozycji w R_P, w których możliwy jest wybór różnych liter, są jednak metody ustalenia sekwencji konsensusowej.

Możliwe i w praktyce przydatne jest rozszerzenie pojęcia ułożenia odczytów o możliwość pominięcia odczytu (patrz definicja 2.1.11). Wtedy, jeśli ułożenie pomija i-ty odczyt, można oznaczyć to w sekwencji, np. jako $P[i] = (0, 0)$. Jeśli w ułożeniu występują dwie pozycje i i $i + 1$, dla których nie ma odczytów umiejscowionych na obu tych pozycjach, czyli $\forall_{(a,b) in P} a, b leq i \vee a, b \geq i + 1$, to ułożenie reprezentuje przynajmniej dwie niepołączone sekwencje, co można interpretować np. jako sekwencje różnych chromosomów. Dla czytelności takiego podziału na niezależne sekwencje referencyjne, który nie jest ewidentny bez analizy wszystkich pozycji, można ułożenie odczytów zapisać np. przy pomocy trójek liczb, z których pierwsza symbolizuje numer sekwencji, a pozostałe dwie pozycję w jej obrębie.

Definicja 2.1.9 (Wsparcie i pokrycie sekwencji, rozkład odczytów)
Dla danego ułożenia odczytów P i reprezentowanej przez nie sekwencji R_P, fragment R_P od a do b ($R_P[a..b]$), $a < b$, jest ***wspierany*** *przez $ws(P,a,b)$ odczytów:*

$$ws(P,a,b) = \left|\{P[i] = (a_i, b_i) : a \le a_i \le b \wedge a \le b_i \; leqb\}\right|$$

czyli przez odczyty w ułożeniu P ***zawarte*** *w przedziale (a,b).*

Fragment R_P od a do b jest ***pokryty*** *przez $pok(P,a,b)$ odczytów:*

$$pok(P,a,b) = \left|\{P[i] = (a_i, b_i) : a_i \le a \le b_i \vee a_i \le b \le b_i\}\right|$$

czyli przez odczyty w ułożeniu P ***zawierające*** *chociaż jedną pozycję z przedziału $[a,b]$.*

Rozkład odczytów *do pozycji x to liczba odczytów rozpoczętych przed lub na tej pozycji:*

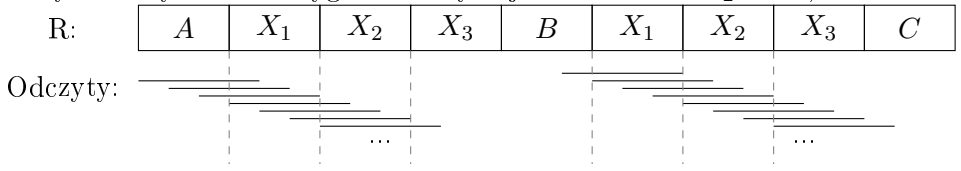
$$D(P,x) = \left|\{(a_i, b_i) \in P : a_i < x\}\right|$$

Definicja 2.1.10 (Asemblacja – wariant równomiernego rozkładu odczytów)
Dla danego zbioru odczytów $S = \{s_1, s_2, \ldots s_n\}$ o długościach $|s_i| = l$ znajdź ułożenie odczytów P (i reprezentującą je sekwencję R_P), dla którego rozkład początków odczytów jest jak najbardziej równomierny:

$$\min_{P} \max_{1 \le i \le |R_P| - l} \left| D(P,i) - |S| \cdot \frac{i}{|R_P| - l} \right|$$

Podobnego sformułowania, w odpowiedzi na ryzyko zbytniej kompresji sekwencji referencyjnej w problemie SCS i w konsekwencji nierównomiernego jej wsparcia, używa Myers w pracy [Myers, 1995].

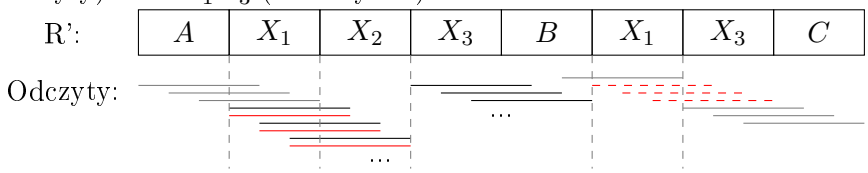
Przykład 2.1.1
Jako przykład wykorzystania wsparcia sekwencji i rozkładu odczytów w asemblacji weźmy genom referencyjny $R = AXBXC$, gdzie powtórzenia X mają długość większą niż długość odczytu l. Dla uproszczenia przyjmijmy $X = X_1X_2X_3$, $|X_1| = |X_2| = |X_3| = |A| = |B| = |C| = l - 1$ oraz że sekwencje pozwalają łatwo rozróżnić podsłowa oznaczone literami, np. poprzez specjalne znaki na końcach. Przyjmijmy, że odczyty nie mają błędów, i zostały równomiernie wygenerowane po jednym z każdej pozycji R (poza ostatnimi l pozycjami), tylko z jednej nici. Wtedy pojedynczy odczyt jest podsłowem AX_1, X_1X_2, X_2X_3, X_3B, BX_1, lub X_3C. Nie ma jednak odczytów o symbolach wygenerowanych jednocześnie z X_2 i z A, B lub C.



Na powyższym przykładzie każdy segment AX_1, X_1X_2 itd. wspierają po 3 odczyty.

Rozważmy zatem słowo $R' = AX_1X_2X_3BX_1X_3C$. Zawiera ono jako podsłowa wszystkie odczyty, ale jest krótsze niż R. Można jednak zauważyć, że pojawia się tam X_1X_3, które nie jest podsłowem R. Spójrzmy na to gdzie mogą znajdować się odczyty, które wspierały oba fragmenty X_1X_2 (na czerwono nowe pozycje, czerwone przerywane linie to poprzednie pozycje, na których nie mogą się już znaleźć) oraz

ile odczytów wspiera w R' równej długości fragmenty X_1X_2 (6 odczytów), X_3B (3 odczyty) oraz X_1X_3 (0 odczytów).



Dla X_1X_2, która występuje w R dwa razy (a w R' raz), rozpoczynające się na każdej z pozycji z X_1, czyli $2l-2$ odczytów. Jest ich dwukrotnie więcej niż dla X_3B (odczyty o początkach w X_3 pochodzącym z X_3B, czyli $l-1$ odczytów). Natomiast nie ma odczytów wspierających X_1X_3, gdyż takiej sekwencji nie było w R.

Powyższy przykład pokazuje, że wariant asemblacji minimalizujący długość rozwiązania (wariant SCS) może dawać w wyniku sekwencję krótszą, niż sekwencja, z której wygenerowano odczyty, np. w przypadku, gdy zawierała ona odpowiednio długie powtórzenia. W sekwencjach biologicznych spodziewamy się wielu powtórzeń, stąd przy asemblacji danych eksperymentalnych wariant SCS raczej nie jest używany. Powyższy przykład pokazuje też jak pominięcie powtórzonego fragmentu wpływa na rozkład początków odczytów dla ułożenia odczytów najbliższego oryginalnym miejscom wygenerowania odczytów w sekwencji.

Rozkład początków bywa przydatną informacją, jednak w praktyce liczba odczytów wspierających dany fragment (czy też rozkład początków odczytów) są używane w asemblacji rzadko, również dlatego, że dane eksperymentalne nie zawsze spełniają założenie o równomiernym rozkładzie odczytów. Średnie wsparcie odczytami różnych fragmentów sekwencji czy inne miary dotyczące liczby odczytów, bywają wykorzystywane w grafowych podejściach do asemblacji do rozwiązywania niektórych problemów, np. eliminacji błędnych odczytów albo problemów z powtórzeniami w sekwencji (reprezentowanymi przez cykle i rozwidlenia w grafie), a częściej wykorzystuje się w takich sytuacjach inne wskazówki np. przy sekwencjonowaniu w trybie *paired-end* informację o sparowaniu odczytów (patrz rozdział 2.1.7).

Trudnością w sformułowaniu problemu asemblacji w wariancie równomiernego rozkładu odczytów (patrz definicja 2.1.10) jest również konieczność uwzględnienia w rozwiązaniu wszystkich odczytów. W danych eksperymentalnych część odczytów trzeba odrzucić, gdyż powstają w wyniku błędów i zawierają więcej błędów niż akceptowalny poziom (lub zawierają błędy innego rodzaju). Często zamiast jednej sekwencji referencyjnej lepiej odczyty opisze kilka sekwencji, szczególnie w sytuacji, gdy powtórzenia uniemożliwiają jednoznaczne ustalenie kolejności tych sekwencji. Dlatego problem asemblacji powinien również uwzględniać takie możliwości:

Definicja 2.1.11 (Asemblacja – wariant używający części odczytów) *Dla danego zbioru słów $S = \{s_1, s_2, \ldots s_n\}$ i akceptowalnej liczby błędów dopasowania m, zbiór r sekwencji $R = R_1, \ldots R_r$, $r \ll n$, dla którego co najwyżej ϵ odczytów nie zostanie zawarte w R:*

$$\left|\{s_i : \exists_{1 \le j \le r} s_i \sqsubseteq_m R_j\}\right| \ge (1-\epsilon) \cdot |S|$$

Jak widać w definicji pojawia się wiele powiązanych ze sobą, czasem w nieoczywisty sposób, parametrów, które mogą być ustalone z góry lub optymalizowane. Od dobrego rozwiązania zwykle oczekiwane jest:

- ϵ bliski zera
- r jak najmniejsze (ale nie mniejsze niż liczba asemblowanych sekwencji, np. liczba gatunków w próbce mikrobiomowej lub liczba chromosomów organizmu)
- równomierny rozkład odczytów w R i związana z tym sumaryczna długość R

Minimalizacja ϵ bez ograniczenia r oraz sumarycznej długości R może prowadzić do otrzymania w wyniku całego zbioru S – niepołączonych odczytów. Z kolei najmniejsze r oraz najmniejszą sumaryczną długość dostaniemy podając jako rozwiązanie zbiór pusty, wtedy jednak $\epsilon = 1$. Podobny problem dotyczy sumarycznej długość wszystkich sekwencji z R. Zezwalając na większy ϵ można znacznie skrócić R, zwracając w wyniku tylko dobrze pokryte odczytami fragmenty.

Najczęściej jednak ani r, ani ϵ nie są ograniczone wprost. Używa się raczej heurystyk biorących pod uwagę wsparcie lub pokrycie fragmentów, np. nie są zwracane fragmenty o mniejszym wsparciu niż ustalony parametr, lub fragmenty o pokryciu znacznie mniejszym niż średnia. To pozwala balansować pomiędzy tymi sprzecznymi wymaganiami, jednak utrudnia formalny opis problemu.

Podejścia do rozwiązania wariantu asemblacji z wykorzystaniem części odczytów, będącego najbliżej realiów eksperymentów NGS, należą zwykle do podejść grafowych, i dzielą się na dwie główne grupy, które umówię w następnych podrozdziałach: podejście Overlap-Layout-Consensus oraz wykorzystanie grafów *de Bruijna*.

2.1.2. Overlap-Layout-Consensus i graf nałożeń (String Graph)

Overlap-layout-consensus to podejście do asemblacji sekwencji DNA wykorzystujące graf zbudowany na podstawie nakładania się na siebie odczytów. Nałożenie (ang. *overlap*) to w tym kontekście taki prefiks lub sufiks jednego odczytu, który jest jednocześnie sufiksem lub prefiksem (komplementarnym) drugiego (czasami dopuszczalne są małe różnice tych sekwencji). Przy tym kroku należy pamiętać, że DNA to cząsteczka dwuniciowa a odczyty mogą pochodzić z dowolnej z nici, więc podczas obliczania dopasowań należy wziąć pod uwagę również sekwencję odwrotnie komplementarną do danego odczytu.

Definicja 2.1.12 (Nałożenie odczytów)
(dla ustalonej minimalnej długości $L \geq 1$ ***i dopuszczalnej liczby błędów*** m***)***
Odczyty s i t ***nakładają się*** *jeśli spełniają jeden z warunków:*

- *zawieranie ($s \Rightarrow t$) — odczyt t jest całkowicie zawarty w s, czyli*

$$t \sqsubseteq_m s$$

  To oznaczenie wprowadzam dla spójności z innymi pracami o grafach nałożeń, np. [Myers, 1995].

- *zwykłe nałożenie ($s \rightarrowtail t$) — koniec odczytu s jest dopasowany do początku odczytu t:*

$$\exists_{l \geq L} s[|s| - l + 1..|s|] \simeq_m s[1..l]$$

  Wtedy również dla odczytów odwrotnie komplementarnych zachodzi $\overline{t} \rightarrowtail \overline{s}$.

- *nałożenie prefiksowe ($s \leftrightarrow t$) — prefiksy odczytów s i t są dopasowane.*

$$\exists_{l \geq L} s[1..l] \simeq_m \overline{t[1..l]}$$

  W tej sytuacji zachodzi też $\overline{t} \rightarrowtail \overline{s}$ oraz $\overline{t} \rightarrowtail s$.

- *nachodzenie sufiksowe ($s \rightarrowtail t$) — sufiksy odczytów s i t są dopasowane.*

$$\exists_{l \geq L} s[|s| - l + 1..|s|] \simeq_m \overline{t[|t| - l + 1...|t|]}$$

W tej sytuacji zachodzi też $\overline{t} \leftrightarrow \overline{s}$ oraz $s \rightarrowtail \overline{t}$.

Zatem wszystkie nałożenia niebędące zawieraniem możemy zapisać przy pomocy zwykłego nakładania ($\rightarrowtail$) korzystając z sekwencji odwrotnie komplementarnych.

Overlap-layout-consensus to trzystopniowe podejście do asemblacji składające się kolejno z:

1. „Overlap" — obliczenia wszystkich nałożeń między parami odczytów.

   Implementując ten krok z użyciem FM-indeksu [Ferragina and Manzini, 2000] można uzyskać liniową złożoność czasową względem sumy długości odczytów.

2. „Layout" — obliczenia ułożenia odczytów wykorzystując nałożenia

3. „Consensus" — ustaleniu sekwencji na podstawie ułożenia odczytów, szczególnie w przypadku, gdy pozwoliliśmy wcześniej na dopasowania z błędami

Najtrudniejszy do rozwiązania jest krok drugi — „layout". Myers [Myers, 1995] do jego rozwiązania proponuje użycie grafu nałożeń odczytów oraz serię redukcji tego grafu znacznie upraszczającą późniejsze obliczenia.

Definicja 2.1.13 (Graf nałożeń (ang. *overlap graph*))
Graf nałożeń odczytów o minimalnym nałożeniu L dla odczytów S to graf skierowany $\mathcal{G}_L = (V, E_L)$, w którym każdemu odczytowi odpowiadają 2 wierzchołki – odpowiadający właściwemu odczytowi oraz odwrotnie komplementarnemu do niego, etykietowane sekwencją: $\forall_{v \in V} seq(v) \in S \cup \{\overline{s_i} : s_i \in S\}$. Krawędzie $E_L = E_{\Rightarrow} \cup E_{\rightarrowtail}$ odpowiadają nałożeniom o minimalnej długości L, $E \ni e = (v, w, t, len) \in V \times V \times \{\Rightarrow, \rightarrowtail\} \times \mathbb{N}$, etykietowane typem i długością nałożenia: $type(e)$, $len(e)$.

Każda krawędź $\rightarrowtail$ łączy koniec jednej sekwencji z początkiem drugiej, dzięki temu w tak zdefiniowanym grafie każda ścieżka $p = (v_1 \overset{e_1}{\rightarrowtail} v_2 \rightarrowtail ... \overset{e_{|p|-1}}{\rightarrowtail} v_{|p|})$ odpowiada pewnej sekwencji q, t.że $\forall_{v_i \in p} seq(v_i) \boxdot_m q$. Jeśli nie dopuścimy błędów, $q = q_n = seq(p)$ jest jednoznacznie wyznaczone przez p rekurencyjnie:

$$q_1 = seq(v_1)$$

$$q_i = q_{i-1} + seq(v_i)\big[(len(e_{i-1}) + 1)..|seq(v_i)|\big]$$

W przypadku dopuszczalnych błędów trzeba ustalić sekwencję nachodzących fragmentów, ale dla uproszczenia przyjmijmy, że tak jak we wzorze powyżej arbitralnie wybieramy sekwencję z pierwszego z odczytów w ścieżce. Wtedy jednak możliwe, że niektóre odczyty będą zawierały więcej niż m błędów. Wybierając najczęstszą literę na danej pozycji też może nastąpić taka sytuacja, jednak jest to mniej prawdopodobne.

Przykład 2.1.2
Dla $s_1 \rightarrowtail s_2 \rightarrowtail s_3$ i $m = 1$:

W powyższym przykładzie zostanie wybrana sekwencja zawierająca dwa a, chociaż wtedy trzeci odczyt, zawierający dwa razy b będzie się różnił o 2 litery. W tym wypadku wybór większościowy, który zwykle jest stosowany, nie spowodowałby błędu, jednak już w przypadku 3 kopii s_1 nie byłoby to wystarczające rozwiązanie. Biorąc jednak pod uwagę, że zwykle m jest bardzo małe, a takie sytuacje są rzadkie, dopuszczamy większą liczbę różnic między odczytem a sekwencją wynikową.

Wiedząc, że każda ścieżka w grafie odpowiada pewnej sekwencji, problem asemblacji przy użyciu grafu nałożeń można sprowadzić do znalezienia odpowiedniej ścieżki (lub ścieżek) w $\mathcal{G}_L$.

Definicja 2.1.14 (Asemblacja przy użyciu grafu nałożeń)
Dla grafu nałożeń $\mathcal{G}_L$ o akceptowalnej liczbie błędów m, znajdź $r \ll n$ ścieżek $p_1, \ldots p_r$, reprezentujących zbiór sekwencji $R = R_1, \ldots R_r$, dla których co najwyżej ϵ odczytów nie zostanie zawarte w R.

$$\left|\{s_i : \exists_{1 \le j \le r} s_i \sqsubseteq_m R_j\}\right| \ge (1 - \epsilon) \cdot |S|$$

Wybór pomiędzy położeniem większego nacisku na minimalizację r kosztem ϵ, czy odwrotnie, zależy m.in. od konkretnych danych, np. ilości błędów, głębokości sekwencjonowania, oraz tego, czy większy nacisk stawiamy na pewność, czy na długość uzyskanej sekwencji.

Aby zmniejszyć trudność wyszukiwania odpowiedniej ścieżki można wykonać serię transformacji grafu $\mathcal{G}_L$, które zmniejszają graf, ale nie powodują zmiany potencjalnych rozwiązań (z dokładnością do błędów), tzn. jeśli sekwencja r była opisywana pewną ścieżką w grafie $\mathcal{G}_L$ to będzie istniała sekwencja $r' \sqsupseteq r$ opisana ścieżką w uproszczonym grafie $\mathcal{G}'_L$.

Definicja 2.1.15 (Transformacje grafu nałożeń)
Transformacje grafu nie zmieniające rozwiązań to:

1. *Usunięcie zawartych odczytów*
2. *Usunięcie krawędzi przechodnich — jeśli krawędzie są wzajemnie spójne, tzn. jeśli $seq(s \rightarrowtail t \rightarrowtail u) = seq(s \rightarrowtail u)$, można bez straty informacji usunąć $s \rightarrowtail u$*
3. *Połączenie jednoznacznych ścieżek — jeśli ścieżka nie zawiera rozwidleń, to jej wierzchołki są łączone w jeden blok (ang.* chunk*):*
   jeśli $s \rightarrowtail t \wedge \nexists_{t' \neq t} s \rightarrowtail t' \wedge \nexists_{s' \neq s} s' \rightarrowtail t$ to zastąp s i t wierzchołkiem u o $seq(u) = seq(s \rightarrowtail t)$, a wszystkie krawędzie $v \rightarrowtail s$ na $v \rightarrowtail u$ oraz $t \rightarrowtail v$ na $u \rightarrowtail v$.

Zaprezentowane powyżej transformacje grafu zmniejszają liczbę wierzchołków i krawędzi, zachowując jednak informacje o sekwencji wszystkich odczytów (z dokładnością do błędów). Jednak kroki 1 i 3 zmieniają liczbę odczytów zawartych w grafie wspierających daną sekwencję. Jeśli informacja o liczbie odczytów tworzących dany

wierzchołek będzie potrzebna, można niewielkim kosztem pamięciowym (stałym dla wierzchołka, czyli liniowym względem $|S|$) dodać do wierzchołków informację ile odczytów one reprezentują, aktualizowaną podczas upraszczania grafu.

Twierdzenie 2.1.1 (Problem asemblacji jako problem ścieżki Hamiltona) *Ścieżka Hamiltona w uproszczonym grafie nałożeń, będzie (jeśli istnieje) reprezentowała sekwencję referencyjną zawierającą wszystkie odczyty (wierzchołek uznamy za odwiedzony jeśli on lub jego odwrotnie komplementarny odpowiednik został odwiedzony). Z powodu błędów sekwencjonowania ścieżka Hamiltona może nie istnieć. Również powtórzenia w rekonstruowanej sekwencji mogą spowodować konieczność odwiedzenia niektórych wierzchołków wielokrotnie.*

Problem znalezienia w grafie ścieżki Hamiltona jest NP-zupełny, zatem w praktyce możliwy do rozwiązania jedynie dla bardzo małych grafów, które być może można uzyskać w przypadku małych genomów, długich odczytów i dokładnego sekwencjonowania dzięki opisanym uproszczeniom grafu. Niestety, nawet wtedy ścieżka Hamiltona może nie istnieć w rozważanym grafie. Dla większych grafów stosowane są kolejne metody zmniejszania rozmiaru grafu oraz heurystyki wyszukiwania ścieżek. Do sposobów zmniejszania grafu oraz rekonstrukcji sekwencji na podstawie grafu odniosę się w podrozdziałach 2.1.5 i 2.1.6, gdyż są one podobne również dla drugiego typu grafów – grafów *de Bruijna*.

2.1.3. Graf *de Bruijna*

O ile podejście overlap-layout-consensus sprawdza się w przypadku asemblacji mniejszych genomów, z niezbyt krótkimi odczytami i małą głębokością sekwencjonowania (tzn. oczekiwaną liczbą odczytów, które mapują się na pojedynczą pozycję w genomie) to przy wielu milionach bardzo krótkich odczytów i próbach asemblacji coraz dłuższych sekwencji zarówno obliczenie dopasowań jest bardziej wymagające pamięciowo i obliczeniowo, jak i graf jest większy, a jego uproszczenie trudniejsze i mniej skuteczne. Stąd pomysł użycia grafów *de Bruijna*, które konstruowane są nie na podstawie całych odczytów, a na podstawie podsłów odczytów długości k – k-merów. Podejście takie zaproponowali m.in. Pevzner et. al. [Pevzner et al., 2001], a zostało zaimplementowane np. w znanym programie do asemblacji `Velvet` [Zerbino and Birney, 2008].

Definicja 2.1.16 (k-mer) *k-mer t to słowo o długości k nad alfabetem Σ.*

Definicja 2.1.17 (Pełny graf *de Bruijna*) *Graf* de Bruijna *rzędu k, dla alfabetu Σ, to graf skierowany $G_k = (V, E)$, w którym wierzchołki odpowiadają wszystkim k-merom nad alfabetem Σ, a krawędzie (ozn. $t \to u$) znajdują się między każdą parą wierzchołków t i u, dla których sufiks t o długości $k-1$ jest równy prefiksowi u. Krawędzie odpowiadają zatem $k+1$-merom:*

$$V = \Sigma^k$$

$$(t, u) \in E \iff t[2..k] = u[1..k-1]$$

Oryginalnie, pełny graf *de Bruijna* został zaproponowany jako metoda rozwiązywania problemu znalezienia słowa *de Bruijna* – najkrótszego cyklicznego słowa zawierającego wszystkie k-mery nad alfabetem Σ [De Bruijn and Erdös, 1948].

Definicja 2.1.18 (Słowo *de Bruijna* rzędu k)
Słowo de Bruijna *rzędu k dla alfabetu Σ, to najkrótsze słowo w, takie że każdy k-mer jest zawarty w cyklicznym rozszerzeniu słowa w, czyli w jego konkatenacji:*

$$\forall_{x:x\in\Sigma^k}\ x \boxdot ww$$

Ponieważ w pełnym grafie *de Bruijna* krawędzie odpowiadają $k+1$-merom, można zaobserwować ciekawą właściwość tego grafu przy przekształceniu krawędziowym:

Definicja 2.1.19 (Krawędziowe przekształcenie (ang. *line graph*))
Krawędziowe przekształcenie grafu $G = (V, E)$ (ozn. $L(G)$) to graf, którego zbiorem wierzchołków jest zbiór krawędzi grafu G, a zbiór krawędzi to zbiór par $((u, w), (w, v))$, $u, v, w \in V$, t.że $\exists_w : u \to w \wedge w \to v$

Obserwacja 2.1.2
Zauważmy, że krawędziowe przekształcenie pełnego grafu de Bruijna *rzędu k jest izomorficzne z pełnym grafem* de Bruijna *rzędu $k+1$ [Zhang and Lin, 1987].*

Twierdzenie 2.1.3
Każdy pełny graf de Bruijna *jest Eulerowski (co wynika wprost ze zrównoważenia wszystkich wierzchołków) i Hamiltonowski (co wynika z obserwacji 2.1.2).*

Fakt 2.1.4
Cykle Hamiltona w pełnym grafie de Bruijna *rzędu k opisują słowa* de Bruijna *rzędu k.*

Uzasadnienie: Ponieważ cykl Hamiltona zawiera każdy wierzchołek, więc reprezentowane przez niego cykliczne słowo zawiera każdy k-mer. Jest to najkrótsze słowo, ponieważ każdy k-mer jest w nim zawarty tylko raz, więc słowo, które byłoby krótsze, musiałoby nie zawierać pewnego k-meru.

Twierdzenie 2.1.5
Cykl Eulera w pełnym grafie de Bruijna *rzędu k opisuje słowa* de Bruijna *rzędu $k+1$.*

Uzasadnienie: Powyższe twierdzenie wynika z obserwacji 2.1.2. Weźmy graf *de Bruijna* rzędu k i jego krawędziowe przekształcenie, będące grafem *de Bruijna* rzędu $k+1$. Cykl Hamiltona w grafie rzędu $k+1$, opisujący słowo *de Bruijna* rzędu $k+1$ można przedstawić w oryginalnym grafie rzędu k jako ścieżkę Eulera w następujący sposób: każdy wierzchołek cyklu Hamiltona to krawędź w oryginalnym grafie. Dwa kolejne wierzchołki cyklu Hamiltona są połączone krawędzią, jeśli krawędzie w oryginalnym grafie przechodziły przez wspólny wierzchołek, zatem dwie kolejno odwiedzane krawędzie w oryginalnym grafie mają wspólny wierzchołek, a więc będą tworzyły cykl. Krawędzi w oryginalnym grafie rzędu k jest dokładnie tyle samo co w przekształceniu, zatem cykl w oryginalnym grafie będzie cyklem Eulera.

Zatem, żeby znaleźć słowa *de Bruijna* rzędu k, najoptymalniej pod względem złożoności obliczeniowej jest użyć grafu rzędu $k-1$ i znaleźć w nim ścieżkę Eulera.

W problemie asemblacji używa się pewnej modyfikacji (będącej podgrafem) pełnego grafu *de Bruijna*. Taki graf ma reprezentować nie wszystkie możliwe k-mery, a jedynie k-mery występujące w odczytach [Pevzner et al., 2001].

Definicja 2.1.20 (Wierzchołkowy graf *de Bruijna* w asemblacji)
Wierzchołkowy graf de Bruijna *dla odczytów S to graf skierowany $G^w_{S,k} = (V, E)$, w którym wierzchołki odpowiadają k-merom $\mathcal{S}_k$ obecnym w odczytach:*

$$\mathcal{S}_k(S) = \left\{\forall_{s_i\in S}\forall_{1\le j\le|s_i|-k+1} : s_i[j..j+k-1]\right\}$$

$$V = \mathcal{S}_k(S)$$

$$(t,u) \in E \iff t[2..k] = u[1..k-1]$$

Inną zmianą w stosunku do problemu znalezienia słowa *de Bruijna* jest to, że poszukiwane słowo (genom) zwykle nie jest cykliczne, więc reprezentowane jest przez ścieżkę, a nie przez cykl.

Definicja 2.1.21 (Problem asemblacji w grafie *de Bruijna*)
Dla odczytów S i ustalonego k znajdź w grafie de Bruijna *ścieżkę (lub ścieżki) reprezentującą słowo (lub słowa) zawierające wszystkie k-mery obecne w odczytach.*

Zauważmy, że o ile w pełnym grafie *de Bruijna* zawsze istnieje cykl Hamiltona reprezentujący szukane słowo, to w przypadku jego podgrafu, np. wierzchołkowego grafu *de Bruijna*, cykl i ścieżka Hamiltona nie muszą istnieć, np. ze względu na powtórzenia (powtórzenia dłuższe niż k wymagają wielokrotnego odwiedzenia pewnych wierzchołków), błędy sekwencjonowania (dodające dodatkowe wierzchołki i krawędzie) i niepełne pokrycie (uniemożliwiające reprezentację sekwencji przez tylko jedną ścieżkę). Aby ominąć te problemy, można w wynikowej ścieżce pominąć część wierzchołków/krawędzi oraz wygenerować więcej niż jedną ścieżkę.

Obserwacja 2.1.6
Ścieżka Hamiltona w wierzchołkowym grafie de Bruijna *rzędu k, jeśli istnieje, reprezentuje sekwencję referencyjną zawierającą dokładnie raz wszystkie k-mery obecne w odczytach.*

Ścieżka, która zawiera każdy wierzchołek przynajmniej raz, reprezentuje sekwencję referencyjną zawierającą wszystkie k-mery obecne w odczytach.

O ile dla małych genomów bakteryjnych podejście wierzchołkowe, czyli poprzez znalezienie ścieżki Hamiltona, mogło być wykonalne obliczeniowo dla odpowiednich danych, to dla większych genomów jest to praktycznie niewykonalne – problem ścieżki Hamiltona jest NP-zupełny, a graf może mieć miliony (4^k, gdzie zwykle $15 \leq k \leq 31$) wierzchołków. Dlatego w eksperymentach NGS stosuje się znacznie prostsze obliczeniowo podejście krawędziowe.

Definicja 2.1.22 (Krawędziowy graf *de Bruijna* w asemblacji)
Krawędziowy graf de Bruijna *rzędu k dla odczytów S to graf skierowany $G^{kr}_{S,k} = (V,E)$, w którym wierzchołki odpowiadają $k-1$-merom, a krawędzie (ozn. $\rightarrow$) k-merom $\mathcal{S}_k(S)$ obecnym w odczytach:*

$$\forall_{t \in \mathcal{S}_k(S)} : t[1..k-1], t[2..k] \in V$$

$$\forall_{t \in \mathcal{S}_k(S)} : t[1..k-1] \rightarrow t[2..k] \in E$$

Zatem każdemu odczytowi długości l odpowiada ścieżka o $l-k+1$ wierzchołkach i $l-k$ krawędziach.

Zauważmy, że krawędziowy graf *de Bruijna* rzędu k jest wynikiem przekształcenia krawędziowego wierzchołkowego grafu *de Bruijna* rzędu k. Pozwala to na zamianę wyszukiwania ścieżki odwiedzającej wszystkie wierzchołki na odwiedzanie wszystkich krawędzi.

Obserwacja 2.1.7
Ścieżka Eulera w krawędziowym grafie de Bruijna *rzędu k (jeśli istnieje), reprezentuje sekwencję referencyjną zawierającą wszystkie k-mery obecne w odczytach.*

Ścieżka, która zawiera każdą krawędź przynajmniej raz, reprezentuje sekwencję referencyjną zawierającą wszystkie k-mery obecne w odczytach.

Parametr k jest głównym parametrem rozwiązań używających grafów *de Bruijna*, jednak bardzo istotnie wpływa na właściwości grafu. Małe k zwiększa spójność grafu przez zwiększenie szansy na współdzielenie pojedynczego k-meru przez więcej niż jeden odczyt. Może się jednak pojawić przez to więcej pętli i rozgałęzień, a w skrajnych przypadkach można nawet otrzymać graf pełny, na podstawie którego nie uda się zrekonstruować sekwencji. Zbyt duże k zwiększa za to wrażliwość grafu na błędy, ponieważ każdy pojedynczy błędny nukleotyd zamienia k k-merów na inne. W szczególności jeśli błędy w odczycie wystąpią częściej niż co k pozycji, odczyt taki może nie mieć wspólnego k-meru z sekwencją referencyjną, co uniemożliwi jego uwzględnienie nawet przy heurystykach wykrywających błędy. Zwiększenie k pomoże jednak uniknąć problemów z powtórzeniami, gdyż mniej będzie powtórzeń dłuższych niż k, a są nieuniknione gdy genom jest dłuższy niż 4^k. Dlatego nie ma jednej rekomendowanej wielkości k, zależy to rekonstruowanego genomu (np. jego wielkości) oraz właściwości konkretnego zbioru odczytów.

Użycie grafu *de Bruijna* do asemblacji znacznie przyśpiesza obliczenia, jednak nie rozwiązuje wszystkich problemów. Nawet gdyby odczyty nie zawierały błędów, wyzwaniem pozostają powtórzenia, a szczególnie te fragmenty, które pojawiają się więcej niż 2 razy.

Przykład 2.1.3
Weźmy powtórzenie podsłowa R w słowie $A_1RA_2RA_3RA_4$ ($|A_i| > k$, $|R| > k$). Poprawna rekonstrukcja kolejności A_2 i A_3 w takim słowie nie jest możliwa za pomocą samych k-merów, ponieważ na ich podstawie nie można rozróżnić słów $A_1RA_2RA_3RA_4$ i $A_1RA_3RA_2RA_4$. Graf dla takich słów zawiera dwie pętle, których kolejności obejścia nie można ustalić przy pomocy samych k-merów, a dla R o długości większej niż długości odczytów ($|R| > \max_{s \in S} |s|$) nawet przy pomocy odczytów.

Takie sytuacje związane z pętlami w grafie dotyczą również grafów nałożeń, i zostaną omówione w następnym podrozdziale.

2.1.4. Asemblacja dwuniciowego DNA

Dwuniciowość DNA powoduje pewne komplikacje przy asemblacji, jednak w przypadku obu podejść grafowych można je rozwiązać. Podczas budowy grafu używane są dodatkowo sekwencje odwrotnie komplementarne do odczytów (tak jak to jest opisane w definicji 2.1.13), dzięki czemu każdy wierzchołek ma odpowiadający mu wierzchołek etykietowany sekwencją odwrotnie komplementarną (nazwijmy go *bliźniaczym*, ozn. $\overline{u}$). Dzięki temu każda krawędź $u \to v$ również ma odpowiadającą jej $\overline{v} \to \overline{u}$. Wierzchołek (krawędź) uznajemy za odwiedzony, jeśli odwiedzony został ich bliźniaczy wierzchołek (krawędź). W ten sposób w rozwiązaniu odczyt lub k-mer znajdzie się wprost lub poprzez sekwencję odwrotnie komplementarną do niego.

Aby uniknąć problemów z palindromami w DNA (to słowa spełniające $s = \overline{s}$), dla których fakt, że wierzchołek jest bliźniaczy sam do siebie ($u = \overline{u}$) może powodować pewne komplikacje, w grafach *de Bruijna* wystarczy ustawić odpowiednie k

tak, aby wierzchołek reprezentował słowo o nieparzystej długości. Ze względu na zasadę komplementarności, słowo o nieparzystej długości nie może być odwrotnie komplementarne do samego siebie, bo środkowy znak nie może być odwrotnie komplementarny do siebie. W przypadku grafów nałożeń zwykle pomija się ten problem ponieważ sytuacje, gdy odczyt jest palindromem są bardzo rzadkie ze względu na większą długość odczytu.

2.1.5. Usuwanie błędów i upraszczanie grafu

Zarówno graf nałożeń, jak i graf *de Bruijna*, może mieć znacznie więcej krawędzi i wierzchołków niż by wynikało z wielkości rekonstruowanej sekwencji wskutek błędów sekwencjonowania i polimorfizmu obecnego w sekwencji (czyli gdy odczyty pochodzą z więcej niż jednego wariantu sekwencji referencyjnej, co jest szczególnie częste przy sekwencjonowaniu populacji bakterii, wynikiem powinien być jednak jeden z wariantów, patrz 1.3.7). W grafie *de Bruijna* pojedyncza błędnie odczytana litera w odczycie zmienia aż do k k-merów wprowadzając potencjalnie nieistniejące w sekwencji k-mery. W grafie nałożeń również powstają dodatkowe rozgałęzienia. Heurystyk do radzenia sobie z takimi sytuacjami na poziomie przygotowania odczytów, budowy grafu, czy „czyszczenia" grafu jest wiele i programy do asemblacji implementują różne podejścia, które jednak opierają się zwykle na usuwaniu dwóch rodzajów błędów w grafie – bąbli i końcówek, oraz rozdzielaniu wierzchołków, które tworzą „splątania".

Definicja 2.1.23 (Bąble w grafie)
Bąblem (ang. bubble*) w grafie nazwiemy dwie rozłączne ścieżki między parą wierzchołków s i t:*

$$p_1 = s \to v_1 \to \dots v_{l_1} \to t$$

$$p_2 = s \to u_1 \to \dots u_{l_2} \to t$$

które opisują podobną sekwencję, tzn. $seq(p_1) \simeq seq(p_2)$. Czasami można użyć innej funkcji podobieństwa sekwencji np. odległości edycyjnej (patrz 2.2).

Bąble w grafie reprezentują polimorfizmy sekwencji (np. u organizmów diploidalnych, wtedy pokrycie obu ścieżek odczytami powinno być podobne), lub błędy sekwencjonowania (szczególnie te, które wystąpiły w środku odczytu, czyli w grafie *de Bruijna* w odległości większej niż k od obu końców, a w grafie nałożeń większej niż L). Ścieżki wynikające z błędów sekwencjonowania zwykle mają małe pokrycie odczytami.

Usuwanie bąbli polega zwykle na usunięciu jednej ze ścieżek (oczywiście bez początkowego i końcowego wierzchołka) lub scaleniu obu ścieżek. Ścieżka scalona w jeden wierzchołek jest etykietowana wspólną dla scalonych ścieżek sekwencją konsensusową, wybieraną zwykle poprzez wybranie litery częściej występującej na danej pozycji.

Definicja 2.1.24 (Końcówka)
Końcówką (ang. tip, dead-end*) w grafie nazwiemy krótką ścieżkę bez rozwidleń, kończącą się lub zaczynającą wierzchołkiem o stopniu 1, czyli takie $p = v_1, ..., v_{l_d}$ spełniające wszystkie warunki:*

- $v_1 \to v_2 \to \dots v_{l_d}$
- $v_i \neq v_j$ *dla* $i \neq j$

- $$\forall_{1\leq i<l_d} deg_{out}(v_i) = 1 \quad \wedge \quad deg_{out}(v_{l_d}) = 0$$
$$\wedge \quad deg_{in}(v_1) > 1 \quad \wedge \quad \forall_{1<i\leq l_d} deg_{in}(v_i) = 1$$

  lub

$$deg_{in}(v_1) = 0 \quad \wedge \quad \forall_{1\leq i<l_d} deg_{in}(v_i) = 1$$
$$\wedge \quad \forall_{1<i\leq l_d} deg_{out}(v_i) = 1 \quad \wedge \quad deg_{out}(v_{l_d}) > 1$$

Końcówki w grafie zwykle wynikają z błędów sekwencjonowania i są usuwane. Minimalna długość końcówek, które są usuwane może być określona przez długość ścieżki l_d lub długość opisywanej przez nią sekwencji $seq(p)$.

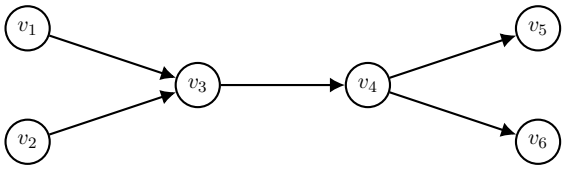
Zarówno bąble jak i końcówki są strukturami w grafie, które występując dla pewnych wierzchołków, tworzą analogiczne struktury dla wierzchołków bliźniaczych (ale w odwrotnej kolejności).

Strukturą pojawiającą się w grafie nie z powodu błędów, ale z powodu powtórzeń, są „splątania", tworzone najczęściej przez pętle:

Definicja 2.1.25 (Splątanie)
Splątaniem (ang. entanglement*) nazwiemy miejsce w grafie, w którym wierzchołki tworzą strukturę w kształcie litery X, czyli istnieją wierzchołki v_3 i v_4 (możliwe $x_3 = x_4$) połączone ścieżką bez rozwidleń, oraz:*

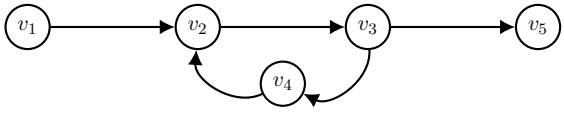
$$deg_{in}(v_3) > 1 \wedge deg_{out}(v_4) > 1$$

Przykładowe „splątanie" prezentuje rysunek 2.1.1.



Rysunek 2.1.1: Przykładowy schemat fragmentu grafu tworzący „splątanie".

Przykład 2.1.4
Rozwiązanie takiej sytuacji łatwo widać w przypadku prostych, niedużych pętli, takich jak na poniższym schemacie 2.1.2:



Rysunek 2.1.2: Przykładowy schemat fragmentu grafu z pętlą tworzącą „splątanie".

W przypadku takich odosobnionych pętli rozwiązanie wydaje się proste, ponieważ nawet wyszukując ścieżkę $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_5$ i cykl $v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_2$ można jednoznacznie rozszerzyć ścieżkę o ten cykl uzyskując ścieżkę odwiedzającą wszystkie wierzchołki i krawędzie przynajmniej raz. Jednak gdyby wierzchołki v_2 i v_3 były zawarte w jeszcze jednym cyklu, dajmy na to z wierzchołkiem v_4' o ułożeniu identycznym do v_4, to ustalenie kolejności odwiedzenia wierzchołków v_4 i v_4' może być niemożliwe bez dodatkowej informacji, która nie jest zawarta w grafie.

W grafach *de Bruijna* wierzchołków i krawędzi reprezentujących powtórzenie, a powodujących „splątanie" ścieżki reprezentującej sekwencję referencyjną, jest wiele, i nie muszą być tak łatwe do zidentyfikowania jak proste pętle, szczególnie lokalnie, patrząc na pojedyncze rozwidlenie, gdy nie widać, czy któreś wierzchołki tworzą pętlę. Rozplątanie takich powtórzeń lokalnie ułatwia wyszukiwanie najlepszej ścieżki, może też czasem umożliwić znalezienie ścieżki bez powtarzania krawędzi czy wierzchołków. Uproszczenie, czyli „rozplątanie" odbywa się poprzez stworzenie kopii niektórych wierzchołków i połączenie ich odpowiednio z pozostałymi – patrz poniższy przykład:

Przykład 2.1.5 (Rozwiązanie „splątania" przez kopiowanie wierzchołków) Lokalnie w przypadku „splątania" widzimy strukturę w kształcie litery X, jak na schemacie 2.1.1 (lub bardziej rozwidloną). Jeśli aktualnie konstruowana ścieżka przechodzi przez v_1, to ustalenie, czy po v_4 powinien nastąpić v_5 czy v_6 bez sprawdzenia czy któraś z wybranych opcji nie zakończy się pętlą zawierającą v_2, nie jest możliwe na podstawie samej struktury grafu. Trzeba wykorzystać dodatkowe informacje, np. odczyty lub pokrycie:

- w grafie *de Bruijna* można wrócić do odczytów, z których pochodziły k-mery, jak proponują to Pevzner et. al w [Pevzner et al., 2001]. Odczyty są w grafie *de Bruijna* reprezentowane przez ścieżki. Jeśli widzimy ścieżki przechodzące przez v_1 i v_5 a nie przez v_6 (lub nie tak często), to należy wybrać taką kolejność odwiedzenia. Nie jest konieczne nawet śledzenie całych ścieżek odczytów, a jedynie obecność k-merów z v_1 i v_5 lub v_6 w jednym odczycie.

- w przypadku grafu nałożeń lub gdy poprzedni punkt nie daje rozwiązania, można wykorzystać pokrycie poszczególnych wierzchołków, korzystając z założenia, że pokrycie sekwencji będzie lokalnie podobne, czyli np. v_1 powinno mieć podobne pokrycie do v_5 lub v_6. Konkretne rozwiązania zależą od rodzaju danych i wyborów dokonanych przez twórców konkretnych implementacji.

Jeśli uda się lokalnie zidentyfikować jak „rozwiązać" splątanie, powtórzona krawędź (lub wierzchołek) jest kopiowany i pozostają w grafie osobne ścieżki, np. dla v_1 mającego połączenie z v_5 otrzymamy dwie rozłączne ścieżki: $v_1 \to v_3 \to v_4 \to v_5$ i $v_2' \to v_3' \to v_4 \to v_6$.

Powyższe metody usuwania błędów i rozwiązywania powtórzeń pozwalają usunąć wiele rozwidleń w grafie, dzięki czemu możliwe jest dalsze zmniejszenie grafu poprzez scalenie jednoznacznych ścieżek (jak w 2.1.15). Tę transformację grafu można w podobny sposób wykonać dla obu rodzajów grafów i zmniejszyć dzięki temu liczbę wierzchołków nie zmieniając liczby możliwych ścieżek ani sekwencji, które one reprezentują.

2.1.6. Rekonstrukcja sekwencji na podstawie grafu

Rekonstrukcja sekwencji po uproszczeniu grafu przy pomocy heurystyk polega na znalezieniu ścieżki Hamiltona lub Eulera (zależnie od rodzaju grafu). Gdy z powodu powtórzeń w rekonstruowanej sekwencji jest to niemożliwe, rekonstrukcja polega na znalezieniu ścieżek zawierających każdy wierzchołek/krawędź przynajmniej raz.

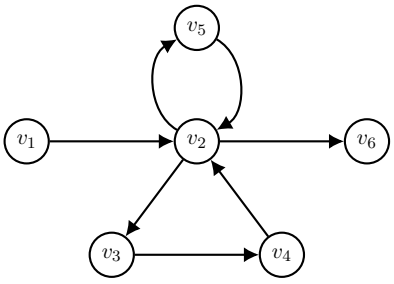
Algorytm znajdowania cyklu Eulera w grafie eulerowskim jest bardzo prosty: zaczynając od dowolnego wierzchołka v podążaj ścieżką aż natrafisz na v (nie powtarzając krawędzi). Jeśli któryś z wierzchołków w ścieżce ma nieodwiedzone krawędzie rozpocznij ww. procedurę w tych wierzchołkach, rozszerzając uzyskany wcześniej

cykl. Analogicznie ścieżkę Eulera znajduje się rozpoczynając w wierzchołku o stopniu wyjściowym większym niż wejściowy, a kończąc w innym wierzchołku, o stopniu wejściowym większym niż wyjściowym.

Obserwacja 2.1.8
Jeśli przy wyznaczaniu ścieżki (analogicznie cyklu) Eulera znajdują się wierzchołki, które mają stopień wyjściowy większy niż 2, to w ścieżce Eulera muszą się rozpoczynać co najmniej 2 dodatkowe cykle, których kolejności obejścia nie można wyznaczyć jedynie na podstawie grafu.

Powyższa obserwacja pokazuje, że nawet jeśli ścieżka Eulera w danym grafie istnieje, to uzyskana sekwencja może być nieprawidłowa. Dlatego w przypadku niejednoznaczności graf jest zwykle rozdzielany i zwracanych jest więcej sekwencji, dla każdego uzyskanego podgrafu, na przykład:

Przykład 2.1.6
Graf zawierający 2 pętle rozpoczynające się w tym samym wierzchołku wymaga rozdzielenia (patrz 2.1.3):

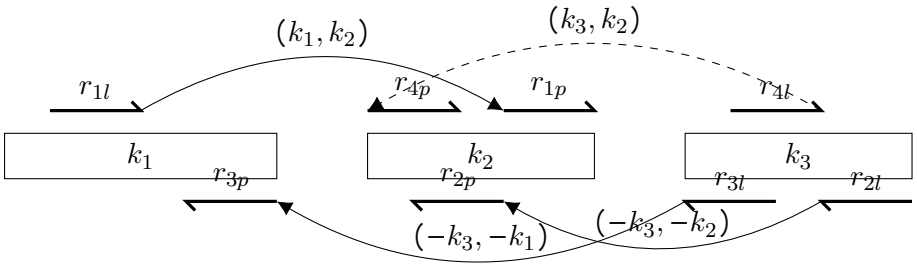


Rysunek 2.1.3: Przykładowy schemat fragmentu grafu zawierający dwie pętle o wspólnym wierzchołku.

Aby zachować informację o wszystkich pętlach i innych ścieżkach graf jest rozcinany np. w wierzchołku powodującym niejednoznaczność, czyli w tym przykładzie w v_2. Otrzymujemy ścieżki: $v_1 \to v_2$, $v_2 \to v_3 \to v_4 \to v_2$, $v_2 \to v_5 \to v_2$, $v_2 \to v_6$.

Jednoznaczne ścieżki, które udaje się wyznaczyć na podstawie grafu, dają pierwsze sekwencje wyjściowe tzw. **„kontigi"** (ang. *contigs*). Następnie kontigi mogą być połączone w dłuższe fragmenty – **skafoldy** (ang. *scaffolds*). Skafoldy składają się jednego lub więcej kontigów ułożonych względem siebie w określonych odległościach, niekoniecznie poprzez nachodzenie. To oznacza, że skafoldy mogą zawierać fragmenty o nieokreślonej sekwencji (ozn. literą N), jeśli są przesłanki by umieścić kontigi w pewnej odległości, jednak nie da się opisać sekwencji pomiędzy kontigami, np. przez wysoką repetytywność, liczbę błędów, albo po prostu brak odczytów. Nieokreślone fragmenty w skafoldach są generowane dla odczytów zsekwencjonowanych technologią sparowanych końców (paired-end). Przy tego typu danych do ustalenia relacji między kontigami używane są pary odczytów. Heurystyki określania relacji kontigów mogą być różne, ale podstawą ich działania są te pary odczytów, które są przyporządkowane do różnych kontigów (np. tworzyły dane kontigi, albo zawierały k-mery tworzące dany kontig). Relacja między kontigami jest podobna do nakładania odczytów, tzn. kontigi są w pewnej konfiguracji (koniec do końca, koniec do początku, itp.) ale zamiast długości nałożenia, możemy mieć szacowaną odległość pomiędzy kontigami, która wynika ze spodziewanej odległości pomiędzy odczytami w parze.

Przykład 2.1.7
Poniższy przykład (patrz 2.1.4) przedstawia możliwe połączenie trzech kontigów k_1, k_2 i k_3 w skafold przy pomocy par odczytów.



Rysunek 2.1.4: Przykład łączenia trzech kontigów k_1, k_2 i k_3 w skafold przy pomocy par odczytów.

Pary odczytów r_1, r_2, r_3, r_4 zostały zaznaczone na kontigach w kierunkach odpowiadających ich sekwencji, nad kontigiem dla sekwencji identycznej, pod dla komplementarnej. Łuki odpowiadają relacjom kontigów, które są generowane przez ustawienia odczytów , przy czym przerywana linia odpowiada relacji, która przy tym ustawieniu kontigów jest niepoprawna, ponieważ wymaga zamiany kolejności k_3 i k_2. Tak jak w przypadku nałożeń sekwencji, trzeba wziąć pod uwagę możliwą odwrotną komplementarność odczytów, np. relacja $(-k_3, -k_1)$ jest tożsama relacji $(-k_1, -k_3)$. W tym przykładzie mamy zatem relacje $(k_1, k_2), (k_2, k_3)$ i (k_1, k_3), które są spełniane przez ustawienie kontigów w takiej kolejności, jak na rysunku.

Relacje między kontigami tworzą graf skierowany, w idealnym przypadku acykliczny, a wtedy do ustalenia kolejności kontigów wystarczyłoby posortować graf. Jednak ponieważ przy takim ustaleniu kolejności kontigów może zdarzyć się ustawienie, w którym odległości między parami odczytów staną się zbyt duże, więc w praktyce raczej łączy się kontigi zachłannie w pary np. wg najsilniejszego wsparcia (tj. liczby par wspierających), a następnie dokłada kolejne kontigi, starając się zachować sugerowane przez pary odczytów odległości między kontigami.

Rekonstrukcja sekwencji na podstawie grafu również pozostawia wiele swobody w doborze heurystyk, zatem nie dziwi mnogość istniejących implementacji mających na celu rozwiązywanie problemu asemblacji.

2.1.7. Przykłady implementacji rozwiązań problemu asemblacji *de novo*

Do tej pory powstało bardzo wiele implementacji rozwiązania problemu asemblacji *de novo*, zarówno przy pomocy grafów nałożeń jak i grafów *de Bruijna*. Programy do asemblacji wykazują dużą różnorodność, dostosowując używane techniki do wariantów problemu, np. asemblacji bardzo dużych genomów, metagenomów, transkryptomu itp. Główne różnice, poza użytym grafem (nałożeń / *de Bruijna*) to:

1. Wstępne przetwarzanie odczytów
2. Heurystyki upraszczania grafu
3. Kryteria wyboru ścieżek w grafie

Graf nałożeń – String Graph Assembler

`SGA` (String Graph Assembler [Simpson and Durbin, 2012, Simpson and Durbin, 2010]) to szybka implementacja asemblacji przy użyciu grafu nałożeń. Podczas gdy standardowe metody obliczania nałożeń odczytów mają złożoność $O(N^2)$ względem sumarycznej długości odczytów N, autorzy `SGA` użyli do obliczenia nałożeń odczytów FM-indeksu [Ferragina and Manzini, 2000] opartego na związanej z tablicą sufiksową transformacie Burrowsa-Wheelera. Nałożenia są identyfikowane poprzez wyszukiwanie przy pomocy FM-indeksu sufiksów o minimalnej ustalonej długości, które są jednocześnie prefiksem jakiegoś odczytu. Innego rodzaju nałożenie (tj. sufiks-sufiks i prefiks-prefiks) można uzyskać przez wyszukiwanie odwróconego odczytu albo przez wyszukiwanie w drugim FM-indeksie, zbudowanym dla odczytów odwróconych. Dzięki temu wszystkie z nałożeń da się znaleźć w czasie liniowym względem sumy długości odczytów oraz liczby wystąpień nałożeń, czyli liczby krawędzi w grafie.

Aby jeszcze przyspieszyć algorytm autorzy zmniejszają liczbę tworzonych krawędzi i eliminują krawędzie przechodnie (patrz 2.1.15) już na etapie obliczania nałożeń odczytów. Autorzy wykorzystali fakt [Simpson and Durbin, 2010], że jeśli w grafie odczyt X nakłada się (w zwykły sposób, czyli sufiksowo-prefiksowy) z odczytami Y oraz Z, a dodatkowo Y nakłada się na Z, czyli otrzymujemy krawędzie $X \rightarrowtail Y$, $X \rightarrowtail Z$ i $Y \rightarrowtail Z$, przy czym to $X \rightarrowtail Z$ jest przechodnia, to część odczytu Z niepokryta przez X (ozn. $tail(X \rightarrowtail Z)$), która skonkatenowana z X daje sekwencję odpowiadającą krawędzi $X \rightarrowtail Z$, jest równa konkatenacji $tail(X \rightarrowtail Y)$ oraz $tail(Y \rightarrowtail Z)$. Pozwala to na wyliczenie jedynie nieredukowalnych, tzn. nieprzechodnich krawędzi, poprzez iteracyjne przedłużanie wszystkich nałożonych na dany odczyt X odczytów o daną sekwencję, i zwrócenie najszybciej zakończonych. Zmniejsza to czas obliczeń i wymaganą przez program pamięć.

`SGA` podzielone jest na kilka osobnych skryptów wykonujących osobno kroki takie jak wstępne przetwarzanie odczytów, konstrukcja FM-indeksu, obliczanie dopasowań i budowa grafu, upraszczanie grafu i obliczenie kontigów. Wstępne przetwarzanie odczytów oferowane opcjonalnie przez `SGA` to, poza standardowym filtrowaniem lub przycinaniem na podstawie jakości sekwencjonowania, także wykrywanie i korekcja odczytów zawierających bardzo rzadko występujące k-mery na często występujące k-mery. Ma to na celu wykrycie substytucji wprowadzonych w odczytach podczas sekwencjonowania, dlatego poprawki wprowadzane są tylko wobec bardzo rzadko pojawiających się liter, by uniknąć skorygowania polimorfizmów sekwencji.

Następnie generowany jest graf nałożeń dla zadanej minimalnej długości dopasowania i poziomu błędu, zawierający lub nie krawędzie przechodnie. Upraszczanie grafu w `SGA` odbywa się iteracyjnie. Początkowo usuwane są tylko krawędzie wierzchołków, które mają bardzo wielu sąsiadów (np. domyślnie 128), co znacznie zmniejsza graf, a regiony, których dotyczą to bardzo zmienne fragmenty i prawdopodobnie niemożliwe do odtworzenia. Krok 3. transformacji grafu (patrz. 2.1.15 oraz praca Myersa [Myers, 1995]), czyli łączenie jednoznacznych ścieżek, jest wykonywane na zmianę z upraszczaniem grafu poprzez usuwanie krótkich „odstających" ścieżek, czyli końcówek (ang. *dead-end, tip*). Usunięcie takich wierzchołków zmniejsza stopień rozgałęzienia jego sąsiadów, co często powoduje, że powstają nowe jednoznaczne ścieżki, które można połączyć, oraz nowe końcówki. Następnie usuwane są, również iteracyjnie, na zmianę z łączeniem jednoznacznych ścieżek, bąble.

`SGA` jako kontigi uznaje sekwencję wierzchołków po uproszczeniu grafu, nie pró-

bując wybierać ścieżek. To konserwatywne podejście pozwala uniknąć niewłaściwego składania kontigów, jednak otrzymane w ten sposób sekwencje są krótsze. Mając do dyspozycji dane *paired-end* można jeszcze połączyć niektóre kontigi używając mapowania na nie odczytów z jednej pary. Na tej podstawie można ustalić wzajemne położenie i wyestymować odległości między kontigami, i zwrócić **skafoldy** zawierające fragmenty sekwencji niepokryte odczytami.

Graf *de Bruijna* – Velvet

`Velvet` [Zerbino and Birney, 2008] to jeden z popularniejszych programów do asemblacji używający grafu *de Bruijna*, implementujący zestaw algorytmów dedykowanych asemblacji dużych genomów z krótkich odczytów. Wyróżniają go m.in. heurystyki usuwania błędów: końcówek, bąbli oraz błędnych krawędzi. Głównym parametrem jest k – nieparzysta liczba oznaczająca długość k-meru w ujęciu wierzchołkowym, chociaż budowa grafu przypomina bardziej ujęcie krawędziowe. Każdy k-mer ma przypisany pierwszy odczyt, w którym się pojawił, a dla każdego odczytu powstaje lista jego k-merów, które pojawią się w następnych odczytach. Budowa grafu różni się zatem od klasycznego podejścia – wierzchołki tworzone są dla tych ciągów k-merów w odczycie, które nie będą przerwane k-merami z następujących po nich odczytach, a po utworzeniu wierzchołków krawędzie są tworzone (lub zwiększany jest ich licznik) dla każdego odczytu. Powstaje zatem od razu graf z uproszczonymi prostymi ścieżkami, a krok z pojedynczymi wierzchołkami odpowiadającymi dokładnie k-merom jest pominięty.

Podobnie jak w przypadku grafu nałożeń, jednoznaczne ścieżki są łączone po każdym etapie usuwania błędów. W pierwszej kolejności usuwane są końcówki, jeśli ich długość jest mniejsza niż $2k$, a wsparcie krawędzi prowadzącej do takiego wierzchołka jest znacznie mniejsze niż alternatywnych krawędzi. Bąble są identyfikowane i usuwane przez algorytm *Tour Bus*. Korzysta on z przeszukiwania grafu wszerz (czyli metodą BFS) w sposób podobny do algorytmu Dijkstry, do wyznaczenia wierzchołków, do których można dojść ze startowego wierzchołka algorytmu różnymi ścieżkami. Znajdując bąbel cofa się do wierzchołka, w którym ścieżki się rozwidlają i wyznacza ich sekwencje. Jeśli sekwencje są wystarczająco podobne mogą zostać połączone – dłuższa ścieżka jest włączana do krótszej zachowując przy tym resztę struktury grafu, czyli np. sąsiadów obu połączonych ścieżek. Ostatnim krokiem usuwania błędów jest usunięcie pozostałych błędnych połączeń, czyli krawędzi o pokryciu odczytami mniejszym niż pewien ustalony próg, z założenia znacznie mniejszy niż średnie pokrycie pozostałych krawędzi.

`Velvet`, inaczej niż `SGA`, nie zakłada, że po uproszczeniu grafu sekwencje wierzchołków będą wystarczająco długie. Kontigi są tworzone przez moduł, którzy autorzy nazwali *Breadcrumb*. Wyznacza on pary wierzchołków A i B o długich sekwencjach połączone przez sparowane odczyty (po jednym odczycie z pary na wierzchołek). By wyznaczyć ścieżkę łączącą A i B zostawia „ślad okruszków" na wierzchołkach, które są połączone sparowanymi odczytami z A lub B. Podążając śladem okruszków rozszerza długie wierzchołki aż do momentu, w którym znajdzie rozwidlenie albo ślad się urwie. W idealnym przypadku długie wierzchołki uda się połączyć pojedynczą ścieżką, w przeciwnym wypadku znajdą się w osobnych kontigach.

`Velvet` to program, który został zoptymalizowany do asemblacji dużych genomów z bardzo krótkich odczytów. Chociaż być może nie jest to wyraźnie widoczne, to część z założeń niekoniecznie będzie działać dobrze przy asemblacji transkryptomów czy

metagenomów, szczególnie:

- równomierne pokrycie zakładane jest przy łączeniu bąbli - dla metagenomów warianty z mniej częstego gatunku mogą zostać zunifikowane z częstszymi
- usuwanie krawędzi wspartych przez zbyt mało odczytów - zarówno w metagenomach jak i transkryptomach może uniemożliwić odtworzenie mniej częstych genomów czy transkryptów
- szacowanie „insert size" może być nieodpowiednie dla transkryptomicznych danych, w których występuje alternatywny splicing

Dlatego wciąż powstają modyfikacje `Velvet` takie jak opisane poniżej `Oases` i `Metavelvet`, które wykorzystują część z jego modułów, ale modyfikują lub dodają inne, np. usuwanie błędów i wybieranie ścieżek.

Velvet dla transkryptomów – Oases

`Oases` [Schulz et al., 2012] jest narzędziem do asemblacji skomplikowanych transkryptomów *de novo*, czyli gdy genom organizmu jest nieznany, a transkryptom może zawierać alternatywny splicing. Do budowy grafu wykorzystuje program `Velvet`, ale zmienia sposób oczyszczania grafu i znajdowania kontigów. Modyfikuje np. algorytm Tour Bus do oczyszczania grafu z bąbli, tak by sprawdzał więcej ścieżek (co było ograniczone w programie `Velvet` ze względu na wydajność). To potrzebne w tym przypadku, ponieważ szczególnie dla genów o dużej częstości w transkryptomie pokrycie jest bardzo duże, a zatem również liczba błędów w takich regionach będzie większa, więc może pojawić się wiele alternatywnych ścieżek. Łączenie wierzchołków w celu stworzenia kontigów jest wykonywane podobnie do programu `Velvet`, ale zarówno przy pomocy odczytów jak i par odczytów łączących kontigi.

Ponadto `Oases` implementuje dodatkową, ciekawą technikę, czyli łączenie transkryptów stworzonych dla różnych wartości k. Transkryptów wygenerowanych przez `Oases` dla różnych wartości k można użyć jako danych wejściowych dla następnej asemblacji wykonywanej przez `Velvet` i `Oases` (ze specyficznymi parametrami do łączenia wcześniej zasemblowanych odczytów).

Velvet dla metagenomów – Metavelvet

`Metavelvet` [Namiki et al., 2012] to rozszerzenie `Velvet` do asemblacji odczytów pochodzących z sekwencjonowania mieszanki genomów o zróżnicowanej częstości w próbce. Wybudowany przy pomocy `Velvet` gotowy graf rozdzielany jest na podgrafy mające odpowiadać genomom kolejnych organizmów. Zakładając, że częstość k-merów dla pojedynczego genomu ma rozkład Poissona, to dla metagenomu jest to suma rozkładów Poissona, w której każdy pik odpowiada jednemu gatunkowi. Podgrafy odpowiadające jednemu gatunkowi są więc oddzielane iteracyjnie zaczynając od najczęstszych genomów na podstawie częstości k-merów. Podczas przypisania wierzchołków do gatunku identyfikowane i rozdzielane (kopiowane) są wierzchołki dzielone przez różne gatunki (częstsze niż wskazuje oczekiwana częstość aktualnie procesowanego gatunku), dzięki czemu po asemblacji jednego gatunku można usunąć wszystkie odpowiadające mu wierzchołki nie ingerując w podgrafy odpowiadające pozostałym gatunkom. Pozostałe kroki asemblacji, takie jak oczyszczanie grafu, generowanie ścieżek itd. wykonywane są przy pomocy Velvet osobno dla każdego z podgrafów.

Graf *de Bruijna* – IDBA-UD

`IDBA-UD` [Peng et al., 2012] to kolejne narzędzie używające grafów *de Bruijna* do asemblacji metagenomów i innych próbek o nierównym pokryciu sekwencji. Wykorzystuje jednak różne wartości parametru k. Wybór właściwej wartości k jest tym trudniejszy w próbkach metagenomicznych (i innych o spodziewanej dużej różnicy w pokryciu sekwencji), że małe k ułatwia odtworzenie fragmentów o małym pokryciu w próbce, a dla bardziej licznych genomów większe k pomaga rozwiązać problem pętli i powtórzeń. `IDBA-UD` startuje z wybranego minimalnego $k = k_{min}$ i dla kolejnych, stopniowo zwiększanych wartości k, buduje graf, upraszcza, wylicza kontigi, a następnie buduje graf dla zwiększonego k używając zarówno odczytów jak i wygenerowanych w tym kroku kontigów.

Do rozwiązania problemu nierównego pokrycia `IDBA-UD` używa zmiennego odcięcia przy filtrowaniu krawędzi i wierzchołków w grafie, a do odtwarzania powtarzających się fragmentów o niskim pokryciu lokalnej asemblacji przy użyciu par odczytów. Poza tym, tak jak pozostałe metody, zmniejsza graf np. poprzez łączenie ścieżek o podobnej sekwencji i o małym pokryciu ze ścieżkami o dużym pokryciu.

Graf *de Bruijna* – MEGAHIT

`MEGAHIT` [Li et al., 2015,Li et al., 2016] to narzędzie dedykowane szybkiej asemblacji metagenomów, wykorzystujące zwięzłe grafy *de Bruijna* (ang. *succinct de Bruijn graph*), czyli skompresowaną formę grafów *de Bruijna*, oraz zrównoleglenie niektórych operacji podczas budowy grafu, np. sortowania k-merów, aby znacznie przyspieszyć asemblację względem wcześniejszych metod.

Poza dużą szybkością atutem `MEGAHIT` jest też wykorzystanie wielu wartości k-merów, w podobny sposób do `IDBA-UD`. `MEGAHIT` dla kolejnych, stopniowo zwiększanych wartości k buduje graf, usuwa końcówki, łączy bąble oraz usuwa mało wsparte krawędzie, a następnie do budowy grafu dla kolejnego k, poza odczytami wykorzystuje kontigi, które w tym przypadku są prostymi ścieżkami w oczyszczonym grafie.

Jedną z nowych heurystyk zaproponowanych przez autorów do pierwszego (czyli jedynie dla k_{min}) oczyszczenia grafu, jest wprowadzenie tzw. *mercy k-mers*, czyli k-merów o bardzo małej liczbie wystąpień, które zostają zachowane w grafie mimo usuwania krawędzi wspieranych przez małą liczbę wystąpień w odczytach. Mało wsparte krawędzie są pozostawione jeśli łączą dwa wierzchołki o dużych wsparciach (czyli wierzchołki mające inną wejściową lub wyjściową krawędź o dużym wsparciu), a które to dwa wierzchołki nie mają innego połączenia (ścieżkami krawędzi o dużych wsparciach). Dzięki temu nie są usuwane mało wparte odczytami krawędzie, które nie mają poza małym wsparciem innych cech błędów (takich jak np. właśnie alternatywna, lepiej wsparta ścieżka). To jest heurystyka szczególnie przydatna w danych metagenomowych, w których może znacznie zwiększyć możliwość odtworzenie genomów tych organizmów, które w danej próbce są mało liczne.

Podsumowując, spośród wielu implementacji różnych sposobów asemblacji sekwencji *de novo*, brak jest programu dedykowanego danym metatranskryptomicznym, jednak niektóre z założeń, takie jak asemblacja sekwencji o różnych pokryciach w próbce, odpowiadają takim danym, co pozwala sądzić, że mogą być odpowiednie w tym przypadku. Pozostałe elementy, którymi różnią się programy, np. szczegóły heurystyk upraszczania grafu w celu eliminacji błędów sekwencjonowania, mają trudny do przewidzenia wpływ na wyniki, gdyż nie jest znany np. stopień podobieństwa

transkryptów w próbce. Dlatego w dalszej części pracy (patrz 3.3) porównane jest działanie tych programów reprezentujących szerokie spektrum podejść do szczegółów asemblacji na rzeczywistych danych eksperymentalnych.

2.2. Mapowanie odczytów na genom i ilościowa analiza odczytów

Gdy genom referencyjny R jest znany, mając dany zbiór odczytów S, celem zwykle staje się ilościowa analiza odczytów, tzn. wskazanie które fragmenty sekwencji referencyjnej są w nich nadreprezentowane. Taka sytuacja występuje np. w eksperymentach z immunoprecypitacją, w których sekwencjonowane są fragmenty DNA połączone z białkami selekcjonowanymi za pomocą przeciwciał specyficznych dla tego białka, ale też w badaniu ekspresji genów czy ilościowych badaniach metagenomów.

Jednym ze sposobów zliczania jest wcześniejsze umiejscowienie odczytów na genomie, czyli **zmapowanie** ich na genom, a później zliczenie zmapowanych odczytów.

2.2.1. Sformułowanie problemu mapowania i zliczania odczytów

Definicja 2.2.1 (Mapowanie odczytów)
Mapowanie to funkcja $m_R : \Sigma^* \to \mathcal{P}(\{1,...|R|\} \times \{1,...|R|\})$*, która odczytom przypisuje wszystkie pozycje (początek, koniec) w słowie* R*, do których są dopasowane.*

Dokładne mapowanie:

Odczyt s_i *mapuje się do* R *na pozycję* (a,b) *(ozn.* $(a,b) \in m_R(s_i)$*) jeśli:*

$$s_i \overset{n}{\cong} R[a..b]$$

Jeśli nie istnieją takie (a,b)*, to* $m_R(s_i) = \emptyset$

W tym sformułowaniu jest to ten sam problem co ***wyszukiwanie wzorca w tekście*** *(ang.* Pattern Matching*), przy czym wyszukiwanych różnych wzorców jest bardzo dużo (tyle, ile odczytów).*

Mapowanie z błędami:

Odczyt s_i *mapuje się do* R *na pozycję* (a,b) *(ozn.* $(a,b) \in m_R(s_i)$*) z co najwyżej* m *błędami, jeśli:*

$$s_i \overset{n}{\cong}_m R[a..b]$$

Tak zdefiniowane mapowanie ma jednak taką wadę, że nie pozwala na dopasowanie odczytu w przypadku błędu insercji/delecji. Nie są to tak częste błędy jak substytucja, jednak mogą istotnie wpłynąć na późniejsze wyniki (jeśli genom referencyjny zawiera tego typu różnice względem faktycznie sekwencjonowanego organizmu), dlatego jeśli to możliwe uwzględnia się je rozszerzając definicję dopasowania sekwencji o jej uliniowienie do genomu, przez co mapowanie bywa też nazywane **uliniowieniem** odczytów do genomu, (ang. *alignment*).

Definicja 2.2.2 (Uliniowienie globalne (ang. *Global alignment*))
Uliniowienie globalne dwóch słów s *i* t *o długościach odpowiednio* n *i* m *to takie słowa* s' *i* t' *nad alfabetem* $\Sigma \cup \{—\}$*, gdzie* $|s'| = |t'|$*, które symbolizują rozmieszczenie względem siebie przerw (ozn. —) i liter obu słów* s *i* t*, takie że po usunięciu znaków —* s' *jest równe* s*, a* t' *równe* t*.*

Uliniowienie można interpretować jako wydarzenia ewolucyjne potrzebne do przekształcenia jednego słowa w drugie, przy czym:

- $s'[i] = t'[i]$ *oznacza zachowaną ewolucyjnie pozycję,*
- $s'[i] = —$ *oznacza insercję w* t*,*
- $t'[i] = —$ *oznacza delecję w* t *(względem* s*),*
- *a* $s'[i] \neq t'[i], s'[i], t'[i] \neq —$ *oznacza substytucję na pozycji* i

Zwykle szukamy najlepszego uliniowienia, zatem zakładamy, że nie zachodzi $s'[i] = — = t'[i]$.

Aby określić najlepsze uliniowienie trzeba zdefiniować **funkcję oceny uliniowienia**. Powinna ona premiować zachowane pozycje, a karać insercje/delecje i substytucje. Najprostsze takie funkcje to odległość edycyjna lub funkcja, która daje +1 pozycjom o takiej samej sekwencji, a 0 o różnej:

$$score(s', t') = \left|\{i : s'[i] = t'[i]\}\right|$$

Jednak funkcja oceny będzie lepiej odwzorowywać proces ewolucji sekwencji, gdy uwzględni zmiany sekwencji w sposób przypominający rzeczywisty mechanizm powstawania mutacji. Usunięcie w jednej z sekwencji bloku x znaków jest dużo bardziej prawdopodobna niż x razy usunięcie pojedynczego znaku, chociaż krótsze delecje są częstsze niż dłuższe. Nie dotyczy to jednak pojedynczych zamian litery, które zachodzą w zasadzie niezależnie. Zwykle zakłada się, że kara za wielkość insercji/delecji rośnie wraz z długością liniowo.

Definicja 2.2.3 (Ocena uliniowienia z liniową karą za przerwę)
Funkcja oceny uliniowienia z liniową karą za przerwę zwiększa karę za przerwę wraz z jej długością (o g_{extend} *za jedną pozycję), ale rozpoczęcie przerwy jest dodatkowo karane (*g_{start}*). Taką funkcję najłatwiej opisać rekurencyjnie:*

$$score(s', t', 0) = 0$$

Jeśli na pozycji i *rozpoczyna się przerwa, czyli* $(s'[i-1] \neq — \wedge s'[i] = —) \vee (t'[i-1] \neq — \wedge t'[i] = —)$*, to:*

$$score(s', t', i) = score(s', t', i-1) - g_{start}$$

jeśli przerwa jest kontynuowana, czyli $(s'[i-1] = s'[i] = —) \vee (t'[i-1] = t'[i] = —)$*:*

$$score(s', t', i) = score(s', t', i-1) - g_{extend}$$

A jeśli nie ma na tej pozycji przerwy, czyli $s'[i] \neq$ *pause* $\wedge\, t'[i] \neq —$*, to:*

$$score(s', t', i) = score(s', t', i-1) + subst(s'[i], t'[i])$$

gdzie funkcja $subst(a, b) : \Sigma \times \Sigma \to \mathbb{N}$*, np.* $subst(a, b) = \begin{cases} 1 & : a = b \\ -1 & : a \neq b \end{cases}$*, oznacza karę za zamianę litery (lub nagrodę za identyczną literę), chociaż substytucja pewnych liter na inne może być też zdefiniowana w bardziej skomplikowany sposób, np. poprzez* ***macierz substytucji****, która każdej parze liter przypisuje pewną liczbę, tak by odzwierciedlić wpływ zmiany na kodowane białko, albo częstość takiej zmiany w znanych sekwencjach o dużym podobieństwie ewolucyjnym.*

Przy takiej funkcji kary możliwe jest rozwiązanie korzystające z programowania dynamicznego, takie jak algorytm Needlemana–Wunscha [Needleman and Wunsch, 1970], który w czasie kwadratowym i przy liniowej pamięci wskazuje wartość funkcji oceny dla najlepszego uliniowienia globalnego (wartość ta opisuje podobieństwo sekwencji), a korzystając z kwadratowej pamięci odtwarza też sekwencję uliniowienia. Myers i Miller [Myers and Miller, 1988] pokazali jednak sposób wykorzystania algorytmu Needlemana–Wunscha do uzyskania sekwencji uliniowienia w liniowej pamięci bez zwiększenia złożoności czasowej wykorzystując technikę "dziel-i-rządź"poprzez obliczenie optymalnych kosztów uliniowienia dla połówek jednej z sekwencji, a następnie rekursywne obliczenie uliniowienia dla krótszych sekwencji (o sumarycznej długości wyjściowych sekwencji).

Definicja 2.2.4 (Uliniowienie lokalne)
Uliniowienie lokalne dwóch słów s i t o długościach odpowiednio n i m to takie słowa s' i t' nad alfabetem $\Sigma \cup \{—\}$, gdzie $|s'| = |t'|$, które symbolizują rozmieszczenie względem siebie przerw (ozn. —) i liter obu słów s i t, takie że po usunięciu znaków — s' jest równe pewnemu spójnemu fragmentowi s, a t' równe spójnemu fragmentowi t.

Uliniowienie lokalne można interpretować jako fragment sekwencji wykazujący np. na ewolucyjne pokrewieństwo tych fragmentów.

Dokładne rozwiązanie problemu lokalnego uliniowienia sekwencji pozwala znaleźć algorytm Smitha–Watermana [Smith et al., 1981], również oparty na programowaniu dynamicznym. Różni się on od algorytmu Needlemana–Wunscha brzegowymi wartościami, tzn. oceną uliniowienia na początku jednej z sekwencji — w lokalnym uliniowieniu przerwa na początku uliniowienia nie jest oczywiście karana, co więcej może być ona dowolnie przedłużona, co jest uwzględnione w regule wypełniania poszczególnych komórek, tj. jeśli kontynuowanie uliniowienia jest niekorzystne (ocena uliniowienia przyjęłaby wartości ujemne), to algorytm *de facto* rozpoczyna uliniowienie od tej pozycji z oceną równą 0. Różnica pojawia się też w sposobie znajdowania optymalnego uliniowienia, które w problemie lokalnym "kończy się"w miejscu tabeli o maksymalnej ocenie, a w problemie globalnym w pozycji tabeli reprezentującej koniec sekwencji.

Łącząc dwa podejścia — uliniowienia globalnego oraz lokalnego — dochodzimy do rozwiązania problemu mapowania krótkich odczytów na genom, w którym ze strony genomu uliniowienie jest lokalne, a z punktu widzenia odczytu globalne.

Definicja 2.2.5 (Pokrycie (odczytami w punkcie))
Pokrycie sekwencji referencyjnej R na pozycji p (ozn. $cov_{m_R,S}(p)$) odczytami S, zmapowanymi funkcją m_R, to liczba odczytów $s_i \in S$ zmapowanych na pozycję p:

$$cov_{m_R,S}(p) = \left|\{s_i \in S : \exists_j p \in m_R(s_i)[j]\}\right|$$

Definicja 2.2.6 (Średnie pokrycie sekwencji odczytami)
Średnie pokrycie sekwencji referencyjnej R (ozn. $cov_{m_R,S}$) odczytami S, zmapowanymi funkcją m_R, to:

$$cov_{m_R,S} = \frac{\sum_{1 \le p \le |R|} cov_{m_R,S}(p)}{|R|}$$

Pokrycie jest używane do oszacowania, które sekwencje oraz jak licznie występują w próbce, np. w transkryptomice wskazuje, które geny są transkrybowane. Jednak mapowanie w przedstawionej wcześniej formie, umożliwiającej dopasowanie odczytu w wielu miejscach, nie pozwala wiarygodnie porównać ze sobą różnych pozycji w sekwencji. Fragmenty występujące wielokrotnie w genomie referencyjnym, np. zduplikowane geny, mogą mieć wielokrotnie większe pokrycie, jeśli odczyty pochodzące z jednego z tych fragmentów są mapowane wielokrotnie, zwiększając pokrycie pozostałych fragmentów. Nie jest to problem przy typowych badaniach porównawczych, gdzie porównujemy mapowania na ten sam fragment genomu w różnych próbkach, ale w innych wypadkach trzeba pamiętać o wyborze pojedynczego, najlepszego pod względem dopasowania sekwencji, miejsca mapowania dla każdego odczytu, a w przypadku wielu takich miejsc o losowym wyborze jednego z nich.

Z możliwością zmapowania jednoznacznie odczytów na dane miejsce w genomie referencyjnym związane jest pojęcie **mapowalności genomu** (ang. *mappability*). Ponieważ często odczyty, których nie można zmapować jednoznacznie (czyli są dopasowane do wielu miejsc), są odrzucane, to pewne fragmenty sekwencji referencyjnej mogą być gorzej reprezentowane w takich mapowaniach, czyli mieć niską „mapowalność". Podobnie na mapowalność na genom może mieć wpływ duża liczba mutacji względem genomu referencyjnego w jakimś obszarze i inne czynniki. W porównawczych badaniach zakładamy jednak, że mapowalność regionów nie zmienia się między próbkami (co jest na pewno prawdziwe w przypadku zależności mapowalności od powtórzeń w genomie referencyjnym). W kontekście odczytów pojęcia **mapowalności** (ang. *mapping rate*) używam jednak jako możliwości zmapowania (zwykle jednoznacznie lub niejednoznacznie) odczytu lub też procentu odczytów, które można zmapować daną metodą.

2.2.2. Implementacje rozwiązań problemu mapowania

Problem mapowania krótkich odczytów na sekwencję referencyjną przedstawia dwa główne wyzwania: problem jest trudny obliczeniowo ze względu na rozmiar danych, a dodatkowo musi uwzględniać potencjalne niedopasowanie odczytów do sekwencji referencyjnej. Z punktu widzenia obliczeń trudność polega na odnalezieniu w genomie referencyjnym (lub innej sekwencji referencyjnej) najbardziej prawdopodobnego miejsca pochodzenia odczytu ponieważ genomy, lub inne rodzaje sekwencji referencyjnych (np. transkryptomy), zwykle są duże - np. genom ludzki ma długość 3 mld nukleotydów, a mapowanie trzeba wykonać dla miliardów odczytów. Jednak dzięki temu, że sekwencja referencyjna jest identyczna dla wszystkich odczytów, a często dla wielu eksperymentów, można przyspieszyć mapowanie pojedynczego odczytu kosztem przetwarzania wstępnego (ang. *preprocessingu*) sekwencji referencyjnej polegającego na przygotowaniu struktury danych ułatwiającego mapowanie, nazywanej zwykle indeksem. Drugi problematyczny aspekt to różnice między odczytami a sekwencją referencyjną. Różnice te wynikają zarówno z błędów podczas sekwencjonowania jak i z tego, że prawie nigdy genom referencyjny nie odpowiada dokładnie genomowi sekwencjonowanego organizmu, zwłaszcza w przypadku mikroorganizmów, które szybko mutują dzięki krótkiemu cyklowi życia. Programy mapujące rozwiązują powyższe problemy w różny sposób, balansując między szybkością działania, a dokładnością mapowania oraz liczbą zmapowanych odczytów.

W tej pracy użyte są dwa programy do mapowania: `bowtie2` [Langmead and Salzberg, 2012] - starsza i bardzo popularna metoda mapowania przeznaczona do mapowania krótkich odczytów do dużych genomów zachowując stosunkowo małe użycie

pamięci, oraz `minimap2` [Li, 2018] - nowsza metoda, która poza krótkimi odczytami pozwala też mapować dłuższe (nawet ponad 1 kbp) odczyty o częstości błędu (*error-rate*) 5-10%. Wykorzystane jest również narzędzie `kallisto` [Bray et al., 2016], czyli pseudomapowanie na transkryptom, które ma na celu przypisanie odczytu do jednej z sekwencji referencyjnych, a nie dokładną identyfikację miejsca jego pochodzenia. Dodatkowo w tym miejscu warto wspomnieć heurystyczne narzędzie `BLAST` [Altschul et al., 1990a], które jest dedykowane raczej wyszukiwaniu spokrewnionych ewolucyjnie sekwencji (np. kodujących białko) w bazie wszystkich znanych sekwencji referencyjnych niż mapowaniu, i w podobnym celu jest wykorzystane w tej pracy, chociaż prezentuje ono podobny schemat działania do powyższych programów mapujących. Wszystkie te metody łączy fakt, że zaczynają od próby znalezienia w sekwencji referencyjnej krótszego fragmentu odczytu (nazywanego ziarno/minimizer/słowo), a później rozszerzyć/połączyć takie dopasowania w celu uzyskania uliniowienia całego odczytu, a różnice w ich działaniu umówione są poniżej.

BLAST

`BLAST` (*Basic Local Alignment Search Tool*) [Altschul et al., 1990a] jest heurystycznym algorytmem do szybkiego znajdowania lokalnego dopasowania sekwencji zapytania (ang. *query*) do bazy sekwencji referencyjnych. Wraz z wykładniczym wzrostem liczby poznanych sekwencji (białek lub DNA), rosła potrzeba, aby nowe sekwencje porównywać z bazą zawierającą sekwencje poznane wcześniej, np. by zidentyfikować funkcję genu czy białka na podstawie znanych podobnych sekwencji, które to zastosowanie jest aktualne do dziś. Dokładne rozwiązania problemu uliniowienia (algorytmy Smitha–Watermana i Neeedlemana–Wunscha) nie były wystarczająco szybkie dla ogromnej bazy sekwencji referencyjnych, zawierającej głównie zupełnie niepodobne do sekwencji zapytania sekwencje, a tym bardziej do całych genomów.

`BLAST` rozpoczyna od stworzenia listy wszystkich k-merów pojawiających się w bazie sekwencji referencyjnych i podobnych (czyli o sumie wartości w macierzy substytucji dla każdego znaku k-meru przekraczającej ustalony próg) do k-merów w zapytaniu. Następnie k-mery z listy są lokalizowane w sekwencjach referencyjnych. Znalezione w jednej sekwencji i blisko siebie dopasowania są łączone w dłuższe słowa, a następnie maksymalnie rozszerzane (bez przerw) dopóki rozszerzenie zwiększa ocenę podobieństwa, tworząc w ten sposób wysoko ocenione pary segmentów zapytania i sekwencji referencyjnej (ang. *High-scoring segment pair, HSP*). Następnie `BLAST` ocenia statystyczną istotność HSP o największym podobieństwie (przekraczającym empirycznie wyliczony przy pomocy losowych sekwencji próg), i dopiero dla tych sekwencji zwraca uliniowienie obliczone algorytmem Smitha-Watermana, razem z oceną statystycznej istotności znalezionego dopasowania.

Bowtie

`Bowtie` [Langmead et al., 2009] to program do mapowania krótkich odczytów na duże genomy wykorzystujący stosunkowo niewielką pamięć, co umożliwia uruchomienie go na większości komputerów osobistych. Druga jego wersja, `bowtie2` [Langmead and Salzberg, 2012], pozwala również na mapowanie z przerwami. `Bowtie2`, tak jak kilka innych programów do mapowania, szybkość i oszczędne wykorzystanie pamięci uzyskuje dzięki użyciu FM-indeksu [Ferragina and Manzini, 2000] (ang. *Full-text Minute-space index*). Indeks ten silnie bazuje na przekształceniu genomu transformatą Burrowsa-Wheelera [Burrows and Wheeler, 1994] (w skrócie BWT,

ozn. $BWT(T)$), pomysłem pokrewnym do tablicy sufiksowej [Manber and Myers, 1993] (ang. *suffix array*, ozn. SA). Transformata Burrowsa-Wheelera przekształca tekst T, w tym przypadku genom, w czasie $O(|T| \log |T|)$, na słowo o tej samej długości, ale często łatwiejsze do skompresowania. To dlatego, że BTW ma tendencję do grupowania liter, ponieważ na pozycji i-tej zawiera literę poprzedzającą i-ty leksykograficznie obrót sekwencji genomu. W językach naturalnych własność ta wynika z często występujących zbitek literowych, w genomie wpływają na to powtórzenia.

Wyszukiwanie wzorca p w tekście przy użyciu BWT odbywa się od końca wzorca. Poza $L(T)$ znana jest również tablica zliczeń każdej z liter w alfabecie (co odpowiada wprost pierwszym literom posortowanych sufiksów, nazwijmy to słowo F). Znajdujemy zakres pozycji w F, w którym są litery odpowiadające $p[|p|]$. Dzięki BWT wiemy jakie litery znajdowały się we wzorcu przed daną literą, tzn. są na odpowiadających mu pozycjach. BWT zachowuje też rangi liter, tzn. i-te wystąpienie danej litery w F odpowiada i-temu wystąpieniu tej litery w L. Zatem w $BWT(T)$ szukamy które z kolei wystąpienia przedostatniej litery wzorca pojawiają się w wybranym już zakresie. Znając te wystąpienia wybieramy zakres w F odpowiadający tym wystąpieniom. Wykonujemy tą procedurę dla kolejnych pozycji we wzorcu aż do znalezienia wzorca (zakres wtedy wskaże liczbę wystąpień) lub do momentu gdy zakres będzie pusty.

FM-indeks, poza BWT i tablicą zliczeń każdej z liter w T, to również tablica zliczeń ile razy każda z liter wystąpiła w BWT do danej pozycji oraz tablica sufiksów. Pozwala to wykonać wyszukanie zakresu w tablicy $BWT(T)$ w czasie stałym. Tablica zliczeń jest zatem potrzebna szybkiego działania, jednak zajmuje dużo pamięci – $O(|T| \times \Sigma)$. Aby zmniejszyć wykorzystanie pamięci, zapisywany jest tylko fragment tablicy, tzn. co któraś pozycja, tak aby reszta była możliwa do obliczenia w stałym czasie. Aby odszukać pozycję wzorca w tekście można wykorzystać procedurę dekodowania BWT, jednak w pesymistycznym przypadku mogłoby to zająć $O(|T|)$. Dzięki użyciu tablicy sufiksowej, a dokładnie jej części, podobnie jak w przypadku tablicy zliczeń, wyszukiwanie wzorca p można wykonać w $O(|p|)$, a jego pozycję znaleźć w czasie sublogarytmicznym względem T.

`Bowtie2` mapowanie odczytu rozpoczyna od wyboru ziaren, czyli niektórych fragmentów odczytu (k-merów w tekście rozpoczynających się na co którejś pozycji). Ziarna i ich odwrotnie komplementarne sekwencje są następnie mapowane z wykorzystaniem FM-indeksu bez przerw i bez błędów lub ewentualnie z jednym dopuszczalnym błędem podstawienia. W wyniku tego mapowania dla każdego ziarna dostajemy dwa zakresy pozycji ziarna w $BWT(T)$, dla sekwencji ziarna i odwrotnie komplementarnej do ziarna. Aby znaleźć pozycje wszystkich ziaren w oryginalnym tekście trzeba wykonać dodatkowy krok dla każdej pozycji z każdego zakresu, dlatego jest to ograniczane poprzez wybór losowo niektórych pozycji, z większym priorytetem dla pozycji z małych zakresów. Wylosowane pozycje w genomie – kandydaci do uliniowienia – są rozszerzane do pełnego uliniowienia z przerwami algorytmem Smitha-Watermana w szybkiej implementacji wspomaganej sprzętowo poprzez równoległe operacje (ang. *single-instruction multiple-data (SIMD)*) [Rognes and Seeberg, 2000].

Jeśli dopuszczamy wyszukiwanie ziaren z błędem, to w sytuacji, gdy podczas wyszukiwania ziarna zakres jest pusty, wyszukiwanie „cofa się" (ang. *backtracking*). Z sufiksu ziarna, którego wyszukiwanie zakończyło się niepowodzeniem wybierana jest pozycja (preferując te o niskiej jakości sekwencjonowania), a następnie wyszukiwanie kontynuowane jest dla jednej z pozostałych w alfabecie liter, przy zmniejszonej do-

puszczalnej liczbie błędów. To podejście zachłanne, zwracające pierwszą napotkaną wersję ziarna, które znajduje się w genomie. Cofanie może być jednak konieczne wiele razy, aby znaleźć odpowiednią pozycję, na której występuje błąd. Aby poradzić sobie z tym problemem, szczególnie kiedy błędy występują zwykle na końcu odczytu, czyli w tym fragmencie, od którego wyszukiwanie w tablicy BWT się zaczyna, autorzy wykorzystują drugi indeks, tzw. lustrzany, dla słowa o odwróconej kolejności liter. Wyszukiwanie w dwóch indeksach polega na tym, że w zwykłym indeksie wyszukiwanie dopuszcza błędy tylko w pierwszej, lewej połowie słowa (czyli w drugiej połowie wyszukiwania), a podobnie jak wyszukiwanie odwróconego słowa w indeksie lustrzanym dopuszcza błędy w drugiej połowie wyszukiwania, czyli w prawej połowie pierwotnego słowa. `Bowtie2` dopuszcza maksymalnie 1 błąd w wyszukiwania ziarna, więc znajdzie się on w jednej z połów ziarna, podczas gdy druga połowa musi znaleźć się bezbłędnie w tekście, zatem do podejście w większości przypadków wystarcza, chociaż aby uniknąć zbyt wielu prób cofania dodatkowo jest ustalona maksymalna liczba takich operacji.

Biorąc pod uwagę np. ograniczenia na cofanie w wyszukiwaniu i losowanie pozycji ziaren, `bowtie2` nie gwarantuje znalezienia najlepszego uliniowienia (zawierającego ziarno o ustalonej długości), ani nawet w ogóle znalezienia ziarna z błędem, nawet jeśli ono istnieje, chociaż w praktyce jest to bardzo mało prawdopodobne, i dotyczy raczej słabo dopasowanych do sekwencji odczytów. Jest to jednak metoda, która wciąż relatywnie szybko pozwala znaleźć wiele dosyć dokładnych mapowań odczytów.

Minimap

Wraz z pojawieniem się technologii sekwencjonowania trzeciej generacji wzrosła potrzeba mapowania odczytów zezwalająca na wiele błędów. `Minimap2` [Li, 2018] poza krótkimi odczytami pozwala też mapować odczyty trzeciej generacji – dłuższe (nawet $\geq$ 1kbp) odczyty zawierające 5-10% błędnych pozycji. Podobnie jak `bowtie2`, wykorzystuje technikę wyboru ziaren i rozszerzenia potencjalnych ziaren na dopasowanie odczytu, jednak dla znacznego przyspieszenia zamiast całych ziaren wykorzystuje ich *minimizery* [Roberts et al., 2004]. (w, k)-minimizery słowa s to zestaw k-merów – reprezentantów wybranych z każdego zbioru w kolejnych k-merów. Wybór reprezentanta ze zbioru k-merów może być różny, np. możliwym porządkiem jest wybranie najmniejszego leksykograficznie, chociaż w przypadku sekwencji DNA raczej wprowadzane są modyfikacje, by nie faworyzować sekwencji składających się z wielokrotnych powtórzeń litery A, które w kontekście DNA są mało informatywne. Reprezentacja słowa s poprzez wybrane minimizery wymaga mniej miejsca niż przechowywanie zbioru wszystkich k-merów słowa, a zapewnia, że jeśli dwa słowa mają wspólny $w + k - 1$-mer, to będą miały przynajmniej jeden wspólny minimizer.

`Minimap2` wykorzystuje tablicę haszującą do przechowania wszystkich minimizerów wraz z ich pozycjami w sekwencji referencyjnej. Dzięki temu wyszukiwanie tzw. kotwic (ang. *anchors*), czyli pozycji minimizera z odczytu w sekwencji referencyjnej jest efektywne. Aby zmapować odczyt konieczne jest wybranie wśród wielu kotwic takiego ich ciągu, który będzie najlepiej opisywał odczyt, tzn. pokrywał największą jego część zachowując odległości i wzajemne położenie kotwic na sekwencji referencyjnej odpowiadające ich położeniu w odczycie. Autorzy `minimap2` wykorzystują do tego programowanie dynamiczne, w którym spośród posortowanych po końcowej pozycji w referencji kotwic wybierają kolejne kotwice na podstawie liczby wspólnych pozycji łączących ją z poprzednią wybraną kotwicą i jej odległości od niej (karząc

za przerwę). Dzięki temu `minimap` w wersji 2 umożliwia też mapowanie odczytów z RNA-seq, ponieważ możliwe jest uwzględnienie przerw w mapowaniu.

2.2.3. Zliczanie poprzez pseudomapowanie — program `kallisto`

W badaniach transkryptomicznych, ale też np. metagenomicznych, szukaną wartością nie jest dokładna pozycja odczytów w obrębie sekwencji referencyjnych tylko liczba kopii sekwencji referencyjnych (dla transkryptomiki transkryptów, a dla metagenomiki liczba kopii genomu będzie mówiła o liczbie komórek bakterii). Właśnie w celu szybkiego szacowania liczby kopii, a dokładniej relatywnej liczby kopii sekwencji w próbce, stworzono narzędzie `kallisto` [Bray et al., 2016]. W celu uzyskania zliczeń `kallisto` dla każdego odczytu wyznacza zbiór kompatybilnych z nim sekwencji referencyjnych na podstawie wspólnych k-merów. Tworzy graf *de Bruijna* dla sekwencji referencyjnych, a następnie wierzchołki ścieżki odpowiadającej każdej sekwencji referencyjnej „koloruje" na inny kolor. To kolorowanie grafu określa dla każdego wierzchołka **zbiór „kompatybilnych" sekwencji referencyjnych** (ang. *k-compatibility class*). Kompatybilne sekwencje referencyjne dla odczytu to przecięcie zbiorów kompatybilnych sekwencji dla k-merów z odczytu.

Aby oszacować częstość poszczególnych sekwencji referencyjnych autorzy używają algorytmu EM (ang. *expectation maximization*), iteracyjnej metody estymacji parametrów modelu, którymi to parametrami w tym przypadku są częstości sekwencji referencyjnych. Odczyty, których zbiory kompatybilnych sekwencji referencyjnych zawierają więcej niż jedną sekwencję mogą zostać w ten sposób przypisane do bardziej prawdopodobnych sekwencji referencyjnych korzystając z uzyskanego w poprzedniej iteracji oszacowania częstości transkryptu.

Jedną z ogromnych zalet `kallisto` jest jego duża szybkość oraz stosunkowo mała podatność na niesklasyfikowanie odczytu przez błędy sekwencjonowania (bądź różnicę między faktyczną sekwencją z badanego organizmu a referencyjną). Jest to osiągane poprzez:

- ignorowanie k-merów z odczytów, które nie występują w grafie — są pomijane i nie powodują niezmapowania odczytu
- klasyfikowanie tylko niektórych k-merów. Jeśli k-mer występuje w grafie, to sprawdzana jest długość ścieżki o tych samych kolorach rozpoczynanej przez ten k-mer. Taka liczba k-merów z odczytu jest pomijana i kolejny klasyfikowany k-mer odpowiada końcowi tej ścieżki (lub końcowi odczytu, jeśli ten następuje wcześniej), aby sprawdzić, czy po drodze nie nastąpił błąd. Tylko w przypadku jego niezgodności pominięte k-mery są sprawdzane, gdy etykieta się zgadza sprawdzany jest kolejny k-mer. Ta metoda często pozwala „przeskoczyć" nad błędami sekwencjonowania bądź polimorfizmami, które mogłyby spowodować niesklasyfikowanie odczytu, jeśli tylko k-mer kończący ścieżkę nie zawiera błędu.

Chociaż do oszacowania częstości sekwencji referencyjnych nie jest konieczne mapowanie, to w wyniku przypisania niektórych k-merów z odczytu do k-merów z sekwencji referencyjnej można uzyskać przybliżone miejsce mapowania – tzw. „pseudomapowanie". Dlatego dla `kallisto` będę używać sformułowania o zmapowaniu odczytu w znaczeniu „przypisania do sekwencji referencyjnej".

Wpływ parametru k na mapowalność odczytów

Parametr k ma duże znaczenie dla działania metody, a zarazem jest w zasadzie jedynym parametrem istotnie wpływającym na działanie algorytmu. Przy częstościach błędu osiąganych w sekwencjonowaniu nowej generacji można założyć, że częstość różnic względem referencyjnego genomu będzie mniejsza niż np. 3% (patrz 1.3.6). Przy odczytach długości 100 bp powoduje, że spodziewamy się ok. 3 błędów na odczyt. Wtedy, niezależnie od pozycji na której wystąpiły błędy, w sekwencji jest minimum 25 kolejnych pozycji, na których błędu nie było, czyli przy $k = 25$ odczyt i źródłowa sekwencja referencyjna powinny dzielić przynajmniej jeden identyczny k-mer. Jest to wskazówka jakiej wielkości powinno być k, jednak to jaka wielkość k jest najlepsza zależy też od grafu. Aby empirycznie określić jak zmienić wartość k warto spojrzeć na wyniki mapowania odczytów.

Na podstawie przecięcia oraz sumy zbiorów kompatybilnych sekwencji referencyjnych dla k-merów dany odczyt można przypisać do jednego ze stanów:

1. zmapowany jednoznacznie (ang. *uniquely*, unikalnie) — przecięcie stanowi tylko jedna sekwencja. Jest to najbardziej pożądana sytuacja, chociaż możliwe są mało pewne mapowania na podstawie pojedynczego identycznego k-meru.

2. zmapowany niejednoznacznie (a.g. *non-uniquely*) - przecięcie zawiera więcej niż jedną sekwencję, co oznacza, że na podstawie sprawdzonych k-merów z tego odczytu nie można zdecydować, która sekwencja referencyjna jest najbardziej prawdopodobnym źródłem odczytu. Jest to rozstrzygane później przez algorytm EM, więc ostateczne przypisanie odczytu jest mniej pewne niż zmapowanie jednoznaczne.

3. niezmapowany ze względu na brak k-merów - zarówno przecięcie jak i suma zbiorów dopasowanych sekwencji są puste. Oznacza to, że w sekwencjach referencyjnych nie występuje żaden z k-merów obecnych w odczycie - być może sekwencje referencyjne są zbyt dalekie od genomu organizmu w próbce. W zmapowaniu takiego odczytu może pomóc zmniejszenie k, lub dodanie odpowiedniej sekwencji referencyjnej. Jednak dodawanie kolejnych sekwencji referencyjnych może skutkować wystąpieniem sytuacji z następnego podpunktu

4. niezmapowany ze względu na konflikt - przecięcie zbiorów kompatybilnych sekwencji referencyjnych jest puste, ale suma nie. Dzieje się tak, gdy poszczególne k-mery są sprzecznie klasyfikowane - do wykluczających się wzajemnie zbiorów kompatybilnych sekwencji referencyjnych. Jest to pożądana sytuacja w wypadku chimer, czyli odczytów, które pochodzą z połączenia w jedną cząsteczkę DNA różnych transkryptów. Jest to artefakt sekwencjonowania i taki odczyt powinien pozostać niezmapowany, jednak jest to rzadka sytuacja. Konflikt w klasyfikacji może być też związany ze stosunkowo dużą liczbą różnic między odczytem a sekwencją referencyjną, przy jednoczesnej przypadkowej identyczności innych k-merów (a nawet jednego k-meru) z inną sekwencją. Zwykle taki odczyt chcielibyśmy jednak sklasyfikować. W tym przypadku zwiększenie k może pomóc uniknąć przypadkowego dopasowania, jednocześnie być może zachowując dopasowanie do właściwej sekwencji referencyjnej.

Manipulując wielkością parametru k można więc zmieniać status klasyfikacji niektórych odczytów balansując między czułością i specyficznością. Zwiększanie k zwięk-

sza specyficzność klasyfikacji — powinno być mniej przypadkowych mapowań wskutek przypadkowej identyczności k-meru w odczycie i w sekwencji referencyjnej. Część odczytów może przestać się jednak mapować (bo nie będzie identycznego k-meru), a część z odczytów niezmapowanych z powodu konfliktów może zacząć się mapować. Zmniejszanie k zwiększa czułość - można zmapować odczyty o większej liczbie różnic względem referencji, jednak mogą pojawić się konflikty. Aby zbadać, które odczyty są niezmapowane z powodu braku k-merów, a które z powodu konfliktów, a dodatkowo ile odczytów zostało sklasyfikowanych niejednoznacznie, użyty został zmodyfikowany kod `kallisto`, wzbogacony o raportowanie o stanie odczytu (opisane w sekcji 3.2.1).

Zastosowanie `kallisto` do danych metatranskryptomicznych

`kallisto` zostało zaprojektowane do kwantyfikowania transkryptomów, w szczególności dużych transkryptomów, (jak np. transkryptom ludzki), które charakteryzują się alternatywnymi wariantami splicingu - alternatywne transkrypty powstają poprzez wycięcie różnych zbiorów intronów, z których większość się powtarza, jednak zmieniają się niektóre końce intronów, niektóre nie zostają wycięte, jak również może zmienić się kolejność pozostałych eksonów. W efekcie w transkryptomie znajdziemy grupy sekwencji, których duże fragmenty są identyczne, niektóre fragmenty jednak znacznie się różnią. Wydaje się, że analogiczną własność posiadają metagenomy (i metatranskryptomy). Co prawda u bakterii raczej nie obserwuje się alternatywnego splicingu, ale spodziewamy się, że genomy, a zatem też transkryptomy blisko spokrewnionych gatunków, będą w wielu miejscach identyczne. Z takiego założenia wyszli również autorzy pracy [Schaeffer et al., 2017], którzy analizowali przy pomocy `kallisto` m.in. odczyty wygenerowane z genomów 100 bakterii, naśladując eksperyment metagenomiczny. Ich obiecujące wyniki pokazują, że `kallisto` może być też przydatne przy analizowaniu danych eksperymentalnych, dlatego jednym z pierwszych etapów tej pracy jest przetestowanie możliwości wykorzystania `kallisto` i porównanie go z innymi metodami mapowania na eksperymentalnych danych metatranskryptomicznych z próbki o nieznanym składzie gatunkowym (patrz rozdział 3.2).

2.2.4. Badanie różnic w ekspresji genów

Mapowanie odczytów na sekwencje referencyjne lub inna forma przypisania odczytów do sekwencji pozwala na przypisanie odczytów do sekwencji referencyjnych. Liczba odczytów przypisanych do fragmentu sekwencji referencyjnej, w której zakodowany jest gen może dać (przybliżoną) informację o tym ile jej kopii znajdowało się w próbce.

Definicja 2.2.7 (Zliczenia odczytów)
Zliczenie to funkcja $c_{m_R,S} : (\{1,...|R|\} \times \{1,...|R|\}) \to \mathbb{N}$*, która zakresom w referencji* R *przypisuje liczbę odczytów ze zbioru* S *zmapowanymi funkcją* m_R*, które są dopasowane do tego fragmentu:*

$$c_{m_R,S}(p,q) = \left|\{s_i \in S : \exists jp \in m_R(s_i)[j] \wedge q \in m_R(s_i)[j]\}\right|$$

W tej definicji odczyt musi być całkowicie zawarty w zakresie genu, ale nie jest to konieczne założenie, można również zliczać odczyty jedynie nachodzące na gen.

Zwykle zliczenia są używane do określenia poziomu ekspresji genu, czyli liczby transkryptów w próbce poddanej sekwencjonowaniu mRNA. Ponieważ jednak na

same zliczenia wpływa wiele czynników, jak np. długość genu, mapowalność do danej sekwencji, głębokość sekwencjonowania, nie można ich traktować jako bezpośredniej informacji o liczbie transkryptów, a tym bardziej porównywać ekspresji pomiędzy różnymi próbkami. W celu porównania ekspresji genów z różnych próbek konieczne jest znormalizowanie zliczeń przez liczbę odczytów w danej próbce ($|S|$) lub użycie bardziej skomplikowanych modeli.

Celem różnicowej analizy ekspresji genów jest wyodrębnienie genów, których ekspresja zmienia się w istotny sposób między różnymi próbkami:

Definicja 2.2.8 (Różnicowa ekspresja genów)
Mając dany zbiór genów $\mathcal{G} = \{g_1, \ldots, g_G\}$ gdzie $g_i \in \mathbb{N} \times (\mathbb{N} \times \mathbb{N})^+$ oznacza pozycję genu w sekwencji referencyjnej, tzn. chromosom oraz listę pozycji eksonów tego genu, znajdź które geny są istotnie zmienione między dwoma zbiorami próbek — traktowanych $\mathcal{T} = \{S_1^t, \ldots, S_{n_t}^t\}$ (ang. treated*) oraz kontrolnych $\mathcal{C} = \{S_1^c, \ldots, S_{n_c}^c\}$ (ang.* control*) na podstawie zliczeń dla każdej z próbek.*

Hipotezą zerową w testowaniu różnicowej ekspresji genów jest taka sama (relatywna) liczba kopii transkryptu genu g_i między zbiorem traktowanym a kontrolnym. Hipoteza zerowa jest odrzucana jeśli na podstawie uzyskanych zliczeń jej p-wartość jest mniejsza niż zadany poziom pewności α. Zwykle testowanych jest wiele genów, więc konieczne jest zastosowane poprawki przy testowaniu wielu hipotez, np. poprawki Bonferroniego, Benjaminiego i Hochberga czy FDR. Geny, które po poprawieniu p-wartości otrzymują wynik mniejszy niż założony poziom istotności α nazwiemy ***różnicowymi****.*

Powiązanie zmiany w liczbie transkryptów ze zmianą w zliczeniach wymaga pewnego doprecyzowania. Najprostszą możliwością jest użycie **krotności zmiany** (ang. *fold change*).

Definicja 2.2.9 (Krotność zmiany)
Krotność zmiany (ang. fold change*) dla i-tego genu to stosunek jego zliczeń odczytów z warunku $\mathcal{T}$ do zliczeń z drugiego warunku:*

$$fc(i) = \frac{\overline{c}_{m_R,\mathcal{T}}(g_i)}{\overline{c}_{m_R,\mathcal{C}}(g_i)}$$

gdzie zliczenia dla zbioru próbek zdefiniujemy jako znormalizowaną sumę zliczeń z próbek $\overline{c}_{m_R,\mathcal{T}}(g_i) = \frac{\sum_j c_{m_R,S_j^t}(g_i)}{\sum_j |S_j^t|}$

Wtedy za istotną zmianę można uznać sytuację gdy krotność zmiany przekracza pewien ustalony próg FC, czyli $fc(i) \geq FC$ (wzbogacenie), lub $fc(i) \leq \frac{1}{FC}$ (zmniejszenie). Jednak ponieważ zmiana krotności jest szczególnie wrażliwa na drobne zaburzenia w zliczeniach dla mało licznych genów (mających mało kopii w próbce, a więc mało odczytów), warto zastosować model, który uwzględnia to oraz może uwzględnić zróżnicowanie między próbkami z tego samego warunku. Do takich modeli należy model stosowany przez autorów `DESeq2` [Love et al., 2014].

DESeq2

Do różnicowej analizy ekspresji genów zostało stworzonych wiele modeli wraz z implementacjami. Jednym z powszechnie stosowanych i rozpoznawanych programów, choć nie jedynym, jest `DESeq2` [Love et al., 2014]. Danymi wejściowymi tego modelu

są zliczenia odczytów K_{ij} dla każdego genu i, dla każdej próbki j, oraz opis próbek (kontrolna lub traktowana).

Model zakłada, że zliczenia K_{ij} pochodzą z rozkładu ujemnego dwumianowego (ang. *negative binomial*) o średniej μ_{ij}, zależącej od nieznanej liczby transkryptów, przeskalowanej przez stałą normalizacyjną (reprezentującą zwykle sumę odczytów w próbce), i dyspersji α_i, która modeluje zmienność w zliczeniach pomiędzy replikatami tego samego warunku. Parametry są ustalane przy pomocy uogólnionego modelu liniowego. Aby ustalić tak dużą liczbę parametrów, szczególnie gdy próbek jest mało, autorzy założyli, że geny o podobnej średniej liczbie zliczeń mają podobną dyspersję, co zwykle jest potwierdzane przez dane eksperymentalne, i zgodnie z tym założeniem poprawiają oszacowanie dyspersji uzyskane dla każdego genu. Przy pomocy wyestymowanych μ_{ij} obliczane są średnie μ_i dla każdego genu i wyestymowane zmiany krotności. P-wartości wyliczane są przy pomocy testu Walda.

Heurystyki wykorzystywane przez `DESeq2` pozwalają uzyskać powtarzalne wyniki, nawet jeśli liczba próbek jest niewielka. `DESeq2` może być też wykorzystany nie tylko do analizy eksperymentów RNA-seq ale również dowolnych innych danych z sekwencjonowania wysokoprzepustowego, w których pojawiają się zliczenia, z danymi metagenomicznymi na czele. Chociaż False discovery rate tej metody może w danych metagenomicznych być przeszacowany, to w porównaniu z konkurencyjnymi metodami jak `edgeR` i `metagenomeSeq` FDR dla `DESeq2` jest mniejsze [Hawinkel et al., 2019]. Biorąc pod uwagę ten fakt należy podejść do wskazań metody ostrożnie — niekoniecznie 95% genów wskazanych jako istotnie zmienione z 5% FDR faktycznie jest istotnie zmieniona.

Uzyskiwane przy pomocy `DESeq2` dla każdego z testowanych genów p-wartości oraz zmiany krotności pozwalają ustalić ranking genów najbardziej zmienionych, dzięki czemu do dalszej analizy można wybrać dowolnie wiele najbardziej obiecujących genów.

ROZDZIAŁ 3

Badanie różnicowej ekspresji genów w mikrobiomie jelitowym myszy na różnych dietach

W tym rozdziale przedstawiona jest analiza różnicowej ekspresji genów w próbkach pochodzących z sekwencjonowania mikrobiomu jelitowego myszy. W sekcji 3.1 opisany jest sposób pozyskania danych eksperymentalnych. W sekcjach 3.2 i 3.3 porównane zostały standardowe metody wykorzystywane w metagenomice: podejścia bazujące na mapowaniu na znane sekwencje referencyjne (uwzględniając różne zbiory tych sekwencji oraz różne metody mapowania) oraz na asemblacji (uwzględniając różne rodzaje asemblacji). Chociaż powyższe metody nie dają w pełni satysfakcjonujących wyników jeśli chodzi o liczbę wykorzystanych odczytów ani liczbę różnicowych sekwencji (3.4), w sekcji 3.5 zaprezentowane jest podejście umożliwiające wyciągnięcie znaczących biologicznie wniosków.

3.1. Pochodzenie próbek — opis eksperymentu

Założeniem eksperymentu było zbadanie zmian w mikrobiomie myszy przy zmianie diety ze standardowej na wysokotłuszczową u myszy o dwóch różnych genotypach: dzikim (ozn. **WT**, ang. *wild-type*) i szczepie trisomicznym Ts(1716)65Dn (ozn. **trisomic**) odpowiadającym ludzkiemu zespołowi Downa. Myszy te mają trisomię fragmentów genomu odpowiadających trisomii ludzkiego chromosomu 21. Chorzy na zespół Downa wykazują niedobory intelektualne ale również częściej niż ogół populacji cierpią na otyłość i późniejsze choroby metaboliczne. Podatność na otyłość pokazano również u myszy Ts65Dn [Fructuoso et al., 2018].

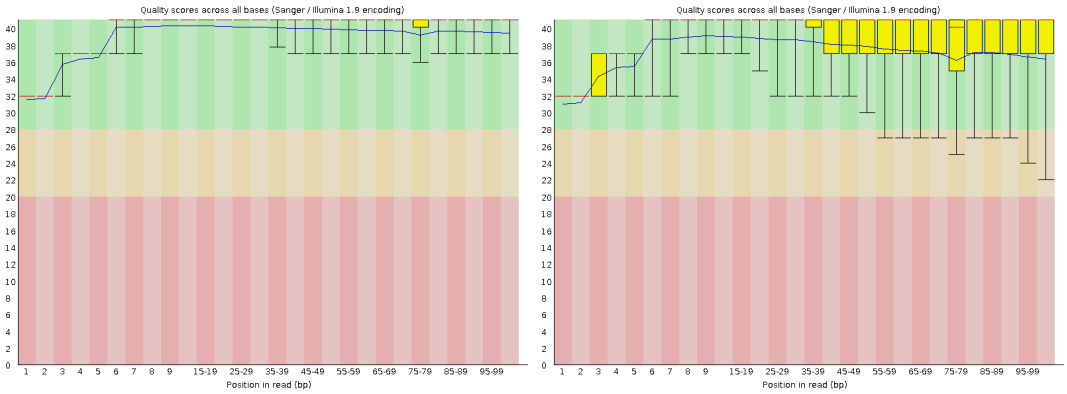
W eksperymencie wykorzystano 3 myszy WT i 2 myszy TS. Myszom podawano standardową karmę (ozn. **SC**, ang. *standard chow*) a następnie w dniu 0 udostępniono myszom do wyboru karmę standardową i wysokotłuszczową (ozn. **HF**, ang.*high-fat*). Próbki w postaci bobków pobrano w dniu 0 i 2 uzyskując próbkę dla SC (połączenie tej próbki wynika z błędu podczas ekstrakcji materiału do sekwencjonowania) oraz po 14 dniach diety wysokotłuszczowej.

W tej pracy analizowane są dane z sekwencjonowania RNA obecnego w mysich odchodach po usunięciu rybosomalnego RNA (patrz 1.3.5), przy pomocy technologii HiSeq Illumina. W wyniku sekwencjonowania otrzymaliśmy dla każdej z próbek od 3 do 12 mln par odczytów o długościach 100 kb (patrz tabela 3.1.1).

Próbka [mysz_data]	genotyp	dieta	odczyty [mln]
6683_16-06-2015	WT	HF	12.066571
6685_04-06-2015	WT	SC	11.557403
6685_16-06-2015	WT	HF	11.876270
6690_04-06-2015	WT	SC	12.888782
6690_16-06-2015	WT	HF	11.275417
6695_04-06-2015	trisomic	SC	7.374023
6695_16-06-2015	trisomic	HF	12.136999
6704_04-06-2015	trisomic	SC	3.018723
6704_16-06-2015	trisomic	HF	8.376929

Tablica 3.1.1: Przegląd zsekwencjonowanych próbek — genotyp myszy i dieta oraz głębokość sekwencjonowania próbek.

Jakość odczytów sprawdzona[1] programem `FastQC` [Andrews et al., 2010] była bardzo dobra. Średnią jakość sekwencjonowania reprezentującą pewność poprawnego odczytania litery w zależności od pozycji w odczycie prezentuje rysunek 3.1.1. W związku z tym, że również pozostałe parametry odczytów były zadowalające, np. nie wykryto primerów, w dalszych analizach odczyty nie były poddane częstym elementom przetwarzania wstępnego, takim jak np. skrócenie odczytów o końcówki o słabej jakości sekwencjonowania.



Rysunek 3.1.1: Wykres jakości sekwencjonowania wzdłuż odczytów pierwszego (z lewej) i drugiego (z prawej) z pary dla jednej z próbek. Zdecydowana większość odczytów na całej długości miała satysfakcjonującą jakość (zielony kolor tła).

Jakość rybodeplecji sprawdzona[2] narzędziem `SortMeRNA` [Kopylova et al., 2012] również była dosyć wysoka. `SortMeRNA` wyszukuje dopasowania odczytów w bazie znanych sekwencji rybosomalnych z różnych gatunków. Przesuwając okno wzdłuż odczytu wyszukuje jego sekwencję w sekwencjach rRNA dopuszczając 1 błąd. Przy dużej liczbie dopasowanych okien odczyt jest uznawany za pochodzący z rybosomów. `SortMeRNA` określił niewielką część odczytów z próbek jako pochodzące rybosomów: od 5.4% do 32.4% odczytów, mediana 9% (patrz tabela 3.1.2).

Ponieważ taka zawartość rRNA nie powinna przeszkadzać w analizie metatran-

[1] Wykonane przez dr Ilonę Grabowicz
[2] Wykonane przez dr Ilonę Grabowicz

Próbka	Odczyty dopasowane do rRNA [%]
6683_16-06-2015	7.55
6685_04-06-2015	9.36
6685_16-06-2015	17.96
6690_04-06-2015	32.42
6690_16-06-2015	25.94
6695_04-06-2015	8.60
6695_16-06-2015	21.85
6704_04-06-2015	8.78
6704_16-06-2015	5.41
mediana	9.36

Tablica 3.1.2: Identyfikacja odczytów rRNA przy użyciu `SortMeRNA`.

skryptomicznej, próbki nie zostały oczyszczone z odczytów zidentyfikowanych jako rybosomalne.

3.2. Mapowanie na genom referencyjny

Mapowanie odczytów na genom referencyjny lub na transkryptom jest domyślną metodą, jeśli sekwencje te są znane. Ponieważ w przypadku próbek metatranskryptomicznych nie jest znany ich skład gatunkowy, więc wybór sekwencji referencyjnych stwarza pewien problem. W tym rozdziale zaprezentowane są wyniki mapowania na różne zbiory genomów — wszystkie dostępne bakteryjne, znane ludzkie bakterie jelitowe, czy genomy bakterii rozpoznanych w próbkach przez program do identyfikacji składu próbki. Porównane są mapowania programami `kallisto`, `minimap2` i `bowtie2`, a w przypadku tego pierwszego pokazane są również powody niezmapowania odczytów. `Kallisto` uruchomione zostało z różnymi wartościami parametru k, `minimap2` z domyślnymi a `bowtie2` w trybie *very-sensitive*, czyli z zestawem parametrów dobranych tak, aby maksymalnie zwiększyć czułość mapowania.

3.2.1. Ocena mapowania

Aby porównać różne sposoby mapowania, zarówno jeśli chodzi o metodę jak i o bazę sekwencji referencyjnych, trzeba wybrać sposób oceny mapowania. Nie znając ani rzeczywistego składu gatunkowego próbki, ani rzeczywistej ekspresji, nie mamy złotego standardu, do którego można porównać wyniki. Jednak do opisania skuteczności mapowania można użyć **mapowalności**, zdefiniowanej jako procent odczytów z próbki zmapowanych (jednoznacznie lub niejednoznacznie). Chcąc uzyskać jak największą mapowalność zwiększamy szansę na błędne mapowania, jednak zakładając, że jedynie mały odsetek odczytów jest zupełnie błędnych, np. jest zanieczyszczeniem lub hybrydą, oczekujemy, że dobry sposób mapowania pozwoli zmapować zdecydowaną większość odczytów. Dodatkowo warto uzyskać możliwie wysoki procent jednoznacznie zmapowanych odczytów, bo ułatwia to dalszą różnicową analizę.

Sprawdzając różne metody mapowania trzeba rozważyć możliwe przyczyny niskiej mapowalności:

- braku (wystarczająco podobnej) sekwencji referencyjnej w bazie sekwencji referencyjnych

- nieoptymalnych parametrów metody (szczególnie w przypadku parametru k dla `kallisto`)
- nieodpowiedniość metody (np. zbyt mała dopuszczalna liczba różnic między odczytem a sekwencją referencyjną)

Oczywiście czynniki te mogą występować jednocześnie, wspólnie z artefaktami sekwencjonowania, zanieczyszczeniami próbki itp. Ta część pracy skupiona jest jednak na wskazaniu możliwie najlepszych rozwiązań dla powyższych 3 potencjalnych problemów z mapowaniem.

Aby lepiej zrozumieć powody niezmapowania odczytów przez `kallisto` potrzebne było zmodyfikowanie źródłowego kodu programu tak, aby w momencie obliczania dla odczytu przecięcia zbiorów kompatybilnych sekwencji referencyjnych jego k-merów raportował w dodatkowym pliku wielkości zarówno przecięcia jak i sumy tych zbiorów, co pozwala później rozdzielić odczyty na niezmapowane z powodu braku wspólnych k-merów (suma zbiorów kompatybilnych sekwencji równa 0) oraz z powodu konfliktu (suma większa od zera, ale iloczyn równy zero).

3.2.2. Genomy bakteryjne z bazy RefSeq

Pierwsza zastosowana baza sekwencji referencyjnych zawiera wszystkie genomy bakteryjne dostępne w 2014 roku w bazie RefSeq [Pruitt et al., 2007] (ozn. **RefSeq2014**), zawierającej wtedy genomy 2785 gatunków bakterii, o sumarycznej długości ponad 9.6mld bp. Na te sekwencje referencyjne zmapowano odczyty używając opisanych wcześniej metod: `kallisto`, `bowtie2` i `minimap2`. Użyto `kalisto` z różną wielkością k: 13, 21, 25 i 31 (31 to wartość maksymalna a także domyślna dla ludzkiego transkryptomu). Niestety dla $k = 17$ nie udało się zbudować indeksu – występował błąd programu, być może spowodowany niewystarczającą ilością RAMu. Przy użyciu `kallisto` udało się zmapować, w najlepszym wypadku czyli dla $k = 21$, średnio 26.9% par odczytów w próbce (21.1% jednoznacznie), podczas gdy `bowtie2` dawał średnią mapowalność 3.64%, a `minimap2` – 8.64% (patrz tabela 3.2.1).

Ponieważ na dalszych etapach używane są tylko zmapowane odczyty, zatem w przypadku każdego z tych programów mapujących, zdecydowana większość odczytów zostałaby niewykorzystana. Nawet pomijając koszt uzyskania odczytów, to przy mapowalności rzędu kilku procent zaufanie do ewentualnie zidentyfikowanych różnicowych genów powinno być mniejsze — wynik może być bardzo zafałszowany przez nierozpoznane geny (które były źródłem większości odczytów), szczególnie jeśli ich częstości różnią się między warunkami eksperymentu. Dlatego chcąc zwiększyć mapowalność sprawdzono możliwe przyczyny niezmapowania odczytu.

Dla `kallisto` sprawdzone zostały różne wartości k. Najlepszą mapowalność przy pomocy `kallisto` osiągnięto dla $k = 21$, zatem poniżej zaprezentowany jest wynik dla tego parametru:

- ponad 30% odczytów nie mapuje się z braku k-merów — dodanie genomów referencyjnych może być dla nich rozwiązaniem.
- 40% odczytów generuje konflikty — te odczyty wymagałyby zwiększenia wielkości k (jednak dla k=25 mapowalność bardzo spada), albo zmodyfikowania `kallisto`.

Modyfikacja `kallisto`, która mogłaby pomóc w przypadku wielu konfliktów to np. klasyfikacja na podstawie najczęściej pojawiających się w zbiorach kompatybilnych

transkryptów, zamiast przypisania do przecięcia tych zbiorów. Jednak implementacja tej modyfikacji może znacznie spowolnić działanie `kallisto`, ponieważ wymagałoby to klasyfikację wszystkich k-merów w odczycie, bez możliwości pomijania niektórych, oraz bez szybkiego zakończenia klasyfikacji odczytu w przypadku konfliktu.

Bakteryjne genomy referencyjne pobrane z bazy RefSeq później (19.11.2018, ozn. **RefSeq2018**) są ponad 4-krotnie liczniejsze niż poprzednio rozważane: 11 455 gatunków, 45 mld bp. Przy pomocy `bowtie` i `minimap` udało się zmapować jednak tylko ok. dwukrotnie więcej odczytów (patrz tabela 3.2.1), czyli powiększenie zbioru sekwencji referencyjnych nie powoduje proporcjonalnego wzrostu mapowalności, a ta uzyskana wciąż jest zdecydowanie za mała.

Wykorzystanie tego zbioru jako genomu referencyjnego dla `kallisto` z optymalnym wcześniej $k = 21$ było zbyt kosztowne pamięciowo (budowa indeksu przekracza dostępne dla nas 800GB RAM). Dla $k = 31$ mapowalność była dwa razy większa niż przy poprzednim zbiorze, ale 10% jednoznacznie zmapowanych odczytów to wciąż zbyt mało, a wymagało to użycia maszyny z ponad 400GB RAM. Co prawda dostępność mocy obliczeniowej i powierzchni dyskowej wciąż wzrasta, jednak liczba zsekwencjonowanych organizmów rośnie nawet szybciej. Przewidywany dalszy wzrost liczby zsekwencjonowanych bakterii może spowodować, że w bazie będzie wiele genomów blisko spokrewnionych gatunków, co dodatkowo utrudni mapowanie (zwiększy liczbę konfliktów i niejednoznacznie zmapowanych odczytów). Dlatego w dalszej części tego rozdziału pokazana jest możliwość mapowania na inne, mniejsze zbiory genomów.

3.2.3. Genomy bakterii jelitowych

W bazie RefSeq znajdują się sekwencje bakterii z różnych środowisk, np. glebowych czy oceanicznych, których nie spodziewamy się w próbce jelitowej. Badania nad ludzkim mikrobiomem, w tym mikrobiomem jelitowym, są na tyle popularne, że powstał już duży zbiór genomów ludzkich bakterii jelitowych. W projekcie *Human Microbiome Project* [Peterson et al., 2009] (ozn. **HMP**) zidentyfikowano (w momencie dostępu) prawie 500 bakterii, których sumaryczna długość dostępnych genomów wyniosła 1.5 mld bp. Dla tych genomów mapowalność przy pomocy `bowtie2` wynosi 4.97%, a 23.52% przy pomocy `kallisto` z $k = 21$. To wciąż bardzo mała część, a ponad połowa odczytów nie ma wspólnego k-meru z tymi sekwencjami. Trzeba jednak pamiętać, że są to bakterie zidentyfikowane w ludzkich jelitach, a nie u myszy, zatem wielu gatunków może brakować a pozostałe mogą być luźno spokrewnione.

Wg mojej wiedzy nie ma podobnie szerokiej bazy dla mysiego mikrobiomu jelitowego — w zestawieniu bakterii jelitowych u myszy (ozn. **miBC**) Lagkouvardos et al. [Lagkouvardos et al., 2016] prezentują jedynie 100 szczepów z 76 gatunków bakterii z 26 rodzin, zidentyfikowanych na podstawie 16SRNA. To stosunkowo niewielki zbiór, a dodatkowo genomów większości z tych szczepów nie ma w bazach danych takich jak RefSeq (stan na 05.2019), jednak dla prawie każdego z wymienionych szczepów udało się znaleźć inny genom z tego samego gatunku lub z najbliżej wg. EzTaxon spokrewnionego gatunku.

Wyniki mapowania `kallisto`, minimap i bowtie na zbiór miBC są porównywalne z mapowaniem na opisane wcześniej zbiory sekwencji referencyjnych. Biorąc pod uwagę bardzo mały rozmiar tego zbioru (jedynie 77 gatunków, 278 mln bp) te mapowalności są dosyć obiecujące, tzn. zidentyfikowanie większej liczby mysich bakterii jelitowych, w różnych szczepach i dla różnych diet, jak również zsekwencjonowanie genomów szczepów wskazanych w miBC, może być dobrą metodą popra-

Genom referencyjny			Mapowanie				
Zbiór sekwencji	Gatunki	Długość [suma bp]	metoda	zmapowane	jednoznacznie	brak k-merów	konflikty
RefSeq2014	2,785	9,632,441,987	k=13	0.19%	0.09%	0	99.81%
			k=21	26.86%	21.06%	32.86%	40.28%
			k=25	11.05%	8.37%	78.12%	10.83%
			k=31	6.6%	4.67%	88.07%	5.32%
			bowtie2	3.64%	0.5%		
			minimap2	8.64%	8.38%		
RefSeq2018	11,455	45,124,432,435	k=21	przekroczone 800GB RAM			
			k=31	12.56%	10.83%	81.05%	6.39%
			bowtie2	9.14%	3.15%		
			minimap2	15.4%	8.58%		
HMP	454	1,548,914,481	k=21	23.52%	15.68%	54.77%	21.71%
			bowtie2	4.97%			
miBC	77	278,408,039	k=21	21.02%	18.37%	68.31%	10.67%
			bowtie2	8.96%			
			minimap2	11.81%	5.98%		
kraken ($\Sigma > 18$)	6,142	34,821,837,885	k=31	12.56%	10.87%	81.06%	6.37%
			bowtie2	9.14%			

Tablica 3.2.1: Średnia mapowalność odczytów z próbki w zależności od metody (`kallisto`, `minimap2`, `bowtie2`), parametrów i użytych sekwencji referencyjnych: RefSeq w latach 2014 i 2018, HMP – Human Microbiome Project, miBC – bakterie jelitowe znalezione u myszy, Kraken18 – sekwencje genomów z RefSeq wg programu `Kraken` posiadające minimum 18 odczytów

wienia mapowania na genomy referencyjne. Jednak żeby uzyskać satysfakcjonujące mapowalności należałoby prawdopodobnie zwiększyć taką bazę kilkakrotnie, co jest kosztowne i trudne, a stworzony w ten sposób zbiór genomów jest bardzo specyficzny dla danego zadania (organizmu – myszy, a także np. rodzaju karmy), nie będzie zatem miał prostego przełożenia na badania nad innymi gatunkami w innych warunkach.

3.2.4. Identyfikacja gatunków przy pomocy programu Kraken

Innym podejściem jest wstępna identyfikacja gatunków obecnych w próbkach na podstawie odczytów, a następnie mapowanie jedynie na część genomów. `Kraken` [Wood and Salzberg, 2014] to narzędzie służące do przypisania odczytów do węzłów w drzewie taksonomicznym na podstawie k-merów obecnych w odczytach. Każdy k-mer przypisany jest do wierzchołka LCA (najniższego wspólnego przodka, ang. *least common ancestor*) gatunków, w których genomach referencyjnych się znajduje. K-mery z odczytu są przypisywane do odpowiednich wierzchołków, a następnie analizowane jest poddrzewo drzewa taksonomicznego indukowane przez wierzchołki z przypisanymi k-merami. W tym poddrzewie obliczane są sumy k-merów przypisanych do wierzchołków na ścieżkach od korzenia do liści, a odczyt jest klasyfikowany do taksonu reprezentowanego przez odczyt, dla którego ta suma jest największa. Klasyfikacja k-merów wykorzystuje minimizery (l-mery), a w drugiej wersji programu [Wood et al., 2019] dla zmniejszenia wymagań pamięciowych dla drzewa przechowywane są jedynie l-mery, a do klasyfikacji dodatkowo wykorzystywana jest funkcja haszująca.

Przy użyciu domyślnych parametrów dla programu `kraken2` ($k = 35$, $l = 31$) wybrano te gatunki (lub szczepy, poziomy S oraz S1-S4 w drzewie taksonomicznym), które średnio miały 2 lub więcej odczytów w próbce (czyli w sumie co najmniej 18 odczytów, ozn. bazę **Kraken18**). Było to 52.3% spośród 7434 obecnych w drzewie (pod)-gatunków, a aż 63.3% z 6142 (pod)-gatunków o dostępnym genomie w RefSeq2018. Wybrane sekwencje miały łącznie 34.8 mld bp (dużo więcej niż baza RefSeq2014).

Dla tak wybranego zbioru genomów znów nie udało się użyć `kallisto` z $k = 21$, ale dla $k = 31$ `kallisto` sklasyfikowało praktycznie tyle samo odczytów co bez ograniczania bazy. Nadal jednak 80% odczytów nie ma z genomami referencyjnymi wspólnego k-meru. Należałoby zatem obniżyć k, co wymaga zapewnienia bardzo dużej pamięci operacyjnej, lub powiększyć zbiór genomów, ale ponieważ wyniki nie były obiecujące dla całej bazy RefSeq2018, to dla innego podzbioru wyniki nie będą satysfakcjonujące. Dalsze zawężenie bazy i użycie $k = 21$ byłoby najbardziej obiecujące, chociaż również obarczone ryzykiem usunięcia genomów bakterii mniej licznych w próbce.

Podsumowując, nie udało się znaleźć zadowalającej metody mapowania na genomy referencyjne, która pozwalałaby sklasyfikować większość odczytów, mimo:

- zastosowania różnych programów mapujących o różnych parametrach
- użycia różnych zbiorów genomów, w tym wszystkich genomów dostępnych w 2018 roku w bazie RefSeq
- ograniczenia zbioru genomów referencyjnych do rozmiaru umożliwiającego użycie `kallisto`

Sposób wyboru gatunków bakterii, których genomów należy użyć jako sekwencji referencyjnych, nie jest oczywisty. Dysponując jedynie danymi RNA-seq z deplecją

rybosomalnego RNA nie można znacząco zawęzić puli gatunków, nawet jeśli dosyć restrykcyjnie wybierane zostaną jedynie te gatunki, które posiadają unikalne k-mery wspólne z naszymi próbkami, a nie te grupy gatunków, których obecność wskazują k-mery dzielone przez więcej niż jeden blisko spokrewniony takson. Wydaje się zatem, że dalszy wzrost liczby dostępnych zsekwencjonowanych genomów bakteryjnych nie rozwiąże problemu, bo powoduje coraz większe problemy obliczeniowe przy nieproporcjonalnie małej poprawie mapowalności. Dlatego jako alternatywę zastosowano w tej pracy inne podejście często stosowane w metagenomice, w którym sekwencje referencyjne są składane (asemblowane) na podstawie odczytów.

3.3. Asemblacja transkryptomu *de novo*

Metod asemblacji *de novo* jest wiele, jednak brakuje programów dedykowanych metatranskryptomom. Wybrano więc spośród dostępnych programów zarówno znane programy dostosowane do asemblacji pojedynczych dużych genomów, jaki i asemblery mikrobiomowe czy transkryptomowe, w większości opisane w podrozdziale 2.1.7.

3.3.1. Sposób asemblacji

Aby uzyskać jednorodny zbiór sekwencji referencyjnych dla wszystkich próbek, który pozwoli porównać ekspresję tych samych sekwencji pomiędzy próbkami, ujednolicając genetyczne zróżnicowanie bakterii w próbkach od różnych myszy i w różnym czasie, sprawdzone zostały ww. programy poprzez asemblację łącznie odczytów ze wszystkich próbek. Minimalna długość kontigu ustalona została na 200 bp, czyli długości minimum 2 odczytów.

Spośród licznych programów do asemblacji wybrany został `Velvet` – dedykowany asemblacji dużych genomów i będący bazą dla innych asemblerów, `Oases` – do asemblacji transkryptomów, `Metavelvet` – do mikrobiomów, `MEGAHIT` i `IDBA-UD` – inne podejścia do asemblacji mikrobiomów, oraz `SGA` – asembler oparty o graf nałożeń.

Program `Velvet`, wykorzystujący graf *de Bruijna*, uruchomiono z maksymalnym parametrem k = 31, oraz z niskim oczekiwanym pokryciem `exp_cov=2`, odległość między odczytami z pary jako `ins_length=200` i pokrycie, poniżej którego dokonywana jest korekcja k-merów: `cov_cutoff=5`.

`Oases` pozwala połączyć wyniki asemblacji otrzymane przez `Velvet` dla rożnych k. W tym wypadku połączono wyniki `Velvet` dla domyślnych parametrów i k = 21 oraz k = 31 używając do łączenia wyników k = 25. `Metavelvet` uruchomiono również dla k = 31 jednak z automatycznym wyliczeniem oczekiwanego pokrycia: `exp_cov=auto`.

`MEGAHIT` uruchomiono z domyślnymi parametrami jeśli chodzi o zakres długości k (od 21 do 119) oraz krok zmiany k w kolejnych iteracjach (10). Również `IDBA-UD` uruchomiono z domyślnymi parametrami, czyli k od 20 do 100 wzrastające co 20. `SGA` to zestaw programów opisany dokładniej we wcześniejszej części pracy – w podrozdziale 2.1.7. W tym wypadku użyto poprawiania błędów przez zamianę rzadkich k-merów (k = 21, domyślnie rzadkie są k-mery i mniej niż 3 wystąpieniach) na częstsze, odfiltrowania zduplikowanych odczytów (ale bez odfiltrowania zawierających rzadkie i niepoprawione wcześniej k-mery) oraz nałożeń odczytów o długości m = 31.

3.3.2. Sekwencje otrzymane w wyniku asemblacji

Zwykle wynik asemblacji *de novo* ocenia się na podstawie długości otrzymanych fragmentów, np. sumarycznej długości kontigów albo wartości N50. N50 to długość najdłuższego kontigu, który wraz z dłuższymi od niego kontigami odpowiada połowie sumarycznej długości złożonych sekwencji. Im dłuższe kontigi i im mniej ich jest, tym lepszy wydaje się wynik, chociaż szansa pojawienia się błędnie połączonych fragmentów w sekwencji jest wtedy większa. Jeśli nie jest znany rozmiar rekonstruowanej sekwencji, jak w przypadku metagenomów i metatranskryptomów, nie można ograniczyć z góry oczekiwanej długości uzyskanej sekwencji, ani też oczekiwanego N50.

Dla transkryptomów maksymalizacja N50 jest szczególnie nieodpowiednia, ponieważ oczekiwane nie jest niewiele długich sekwencji, tak jak w przypadku asemblacji genomu o kilku-kilkudziesięciu chromosomach. Zamiast tego spodziewane jest bardzo wielu (np. tysięcy) krótszych sekwencji o długościach transkryptów, które dla prokariontów przewiduje się, że mają średnio ok. 1 tys. bp [Xu et al., 2006]. Zatem przedstawione w tabeli 3.3.1 wyniki są tylko pewnym podsumowaniem zbiorów kontigów, a nie skuteczności programów.

Program	Liczba kontigów	Sumaryczna długość	N50
Velvet k=31	215,813	144,650,085	926
Oases k=21 i k=31	635,425	591,707,639	1,538
Metavelvet k=31	471,289	217,391,825	514
MEGAHIT, k=21 do 119	237,151	230,704,864	1,430
IDBA-UD	219,352	228,239,757	1,563
SGA k=31	232,576	134,778,668	676

Tablica 3.3.1: Struktura zbioru kontigów otrzymanych przy pomocy różnych programów do asemblacji (po odfiltrowaniu kontigów krótszych niż 200 bp).

Sumarycznie najdłuższe są kontigi otrzymane przez `Oases`, co jest zgodne z założeniem, że duże części sekwencji mogą się powtarzać by reprezentować różne warianty rekonstruowanych transkryptów. N50 pokazuje również, że `Oases`, `MEGAHIT` i `IDBA-UD` rekonstruują najbardziej zwarte kontigi. Użyteczności takich sekwencji nie można jednak ocenić na podstawie wyłącznie tych informacji.

3.3.3. Ocena wyników asemblacji poprzez mapowanie na kontigi

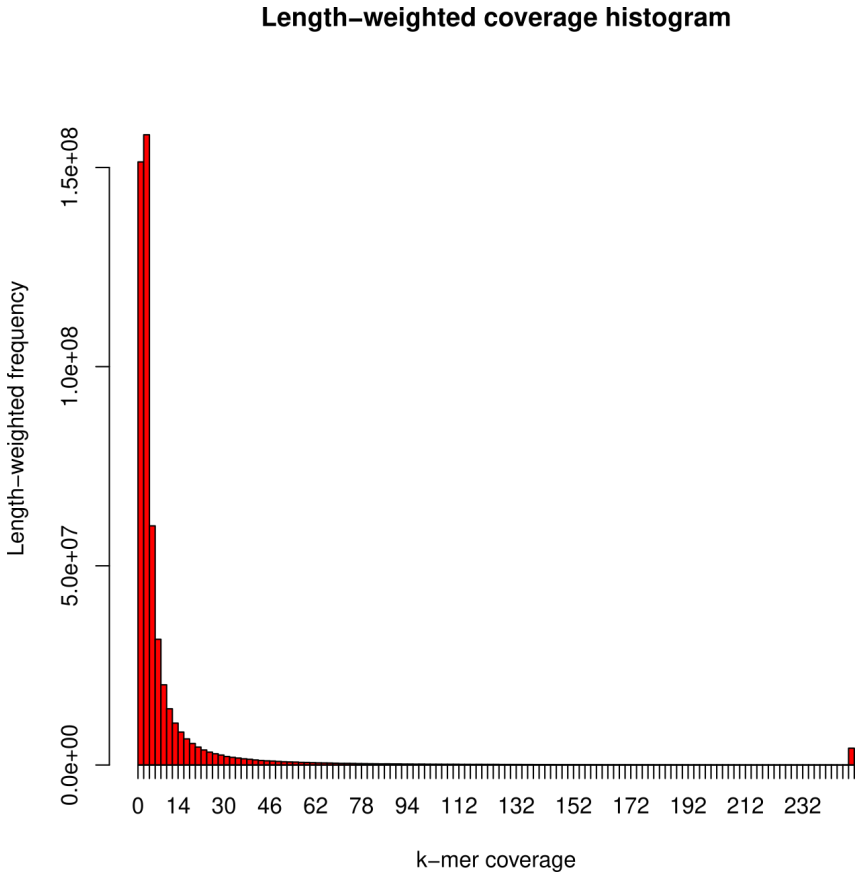
Ponieważ następnym krokiem standardowej analizy różnicowej ekspresji jest obliczenie częstości sekwencji na podstawie mapowań, zbiór sekwencji referencyjnych wybrano na podstawie mapowalności, tak jak jest to opisane w podrozdziale 3.2.1. Podobnie jak wtedy użyto `kallisto` pozostawiając ustalony wcześniej parametr $k = 21$, a porównano średnią z mapowalności wszystkich próbek.

Najgorzej wypada `SGA` — metodzie korzystającej z grafu nałożeń udało się uzyskać sekwencje dające mapowalność jedynie 38.1%. Metody asemblacji *de novo* używające grafu *de Bruijna* wygenerowały sekwencje, do których udaje się zmapować średnio 42% – 83.9% odczytów (patrz tabela 3.3.2). Najlepszy wynik, czyli średnio 83.1% odczytów mapuje się jednoznacznie do sekwencji wygenerowanych przez `MEGAHIT`. Słaby wynik `Metavelvet` wynika z tego, że jego założenia nie są spełnione w naszych danych: w rozkładzie częstości k-merów nie widać pików odpowiadających pojedynczym transkryptom (patrz wykres 3.3.1), podczas gdy `Metavelvet` zakłada, że na podstawie wykresu częstości k-merów można wyestymować częstość

rekonstruowanych sekwencji. Wobec niespełnienia tego założenia działanie programu `Metavelvet` sprowadza się do działania programu `Velvet`, chociaż pewne szczegóły, np. ustawiane parametry, różnią się.

Program	zmapowane (jednoznacznie)	brak k-merów	konflikty
Velvet, k=31	49.5% (48.9%)	22.1%	28.4%
Oases, k=21 i k=31	52.8% (16.9%)	5.8%	41.4%
Metavelvet, k=31	42.5% (42.0%)	28.7%	28.8%
MEGAHIT, k=21 do 119	83.9% (83.1%)	4.4%	11.7%
IDBA-UD	77.5% (77.2%)	4.8%	17.7%
SGA k=31	38.1% (37.3%)	35.1%	26.8%

Tablica 3.3.2: Mapowanie przy pomocy `kallisto` z k = 21 na kontigi uzyskane różnymi narzędziami do asemblacji sekwencji.

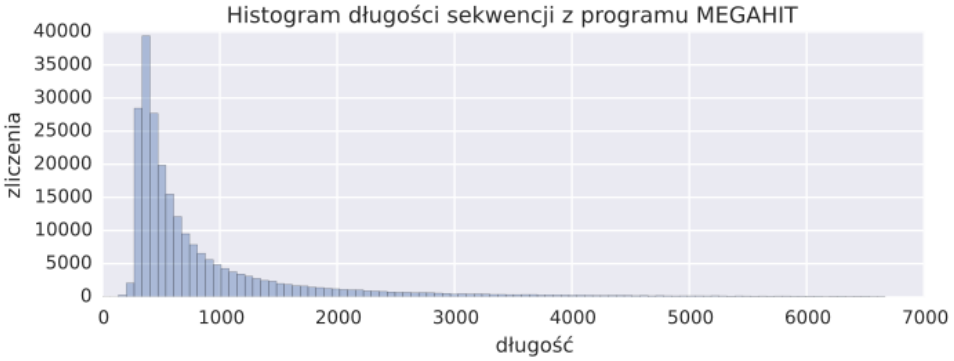


Rysunek 3.3.1: Histogram częstości występowania 31-merów w odczytach.

Wszystkie z użytych metod poprawiają liczbę odczytów, których można użyć w dalszej analizie. Zaskakujące jednak jest to, że dla wielu z ww. metod nie udaje się zmapować ponad połowy z odczytów, na podstawie których kontigi zostały stworzone. Te odczyty mogły np. tworzyć zbyt krótkie kontigi (mniej niż 200 bp), które zostały odfiltrowane. W wyniku heurystyk upraszczających graf i naprawiających błędy odczyty mogły również przestać pasować do konsensusowej sekwencji lub wybór ścieżek reprezentujących kontigi w grafie spowodował powstanie konfliktów. Ostatnia

możliwość to prawdopodobnie częsta sytuacja w przypadku `Oases`, gdzie aż 41% odczytów tworzy konflikty.

Najwięcej odczytów pozwalają zmapować kontigi z `MEGAHIT`, jednak wiele z nich jest dłuższych niż spodziewana długość genu bakteryjnego, czyli 1 tys. bp (patrz wykres 3.3.2). Może to wynikać z błędnego połączenia kilku genów podczas asemblacji (kontig hybrydowy), ale przyczyna może być również biologiczna. Ekspresja genów u bakterii często odbywa się poprzez transkrypcję całego operonu, wobec czego kilka genów kodujących białka tego samego operonu może się znajdować na tym samym transkrypcie, i być wspólnie regulowane. W analizie różnicowej ekspresji genów użyto zatem kontigów z `MEGAHIT`.



Rysunek 3.3.2: Histogram długości sekwencji uzyskanych przy pomocy programu `MEGAHIT`.

3.4. Identyfikacja różnicowych kontigów

Transkrypty utworzone przy pomocy programu `MEGAHIT` i wyestymowane przy pomocy `kallisto` zliczenia odczytów wykorzystano do analizy różnicowej ekspresji przy pomocy m.in. programu `DESeq2` [Love et al., 2014]. Ponieważ liczba sekwencji uzyskanych przez `MEGAHIT` jest bardzo duża, (większa niż standardowo badana liczba genów człowieka, tj. ok. 30tys.), usunięto z badanego zbioru znaczną część sekwencji, bazując na długości i liczbach zliczeń. Testowanie bardzo wielu hipotez jednocześnie powoduje, że stosując poprawkę *False Discovery Rate* (poprawka Benjaminiego-Hochberga [Benjamini and Hochberg, 1995]), bardzo trudno uzyskać istotne wyniki, dlatego warto nie uwzględniać pewnych hipotez w testowanym zbiorze.

Ponieważ transkrypty mające zliczenia tylko w nielicznych próbkach nie są wiarygodnymi transkryptami pokazującymi różnicę między warunkami, a zaburzają model, pozostawiono jedynie sekwencje, które miały po minimum 5 zliczeń w minimum 3 próbkach. Dodatkowo pozostawiono tylko sekwencje mające oczekiwaną dla bakterii długość genu, czyli między 500 a 1500 bp. Odrzucono w ten sposób 90% sekwencji, pozostawiając 22417 sekwencji z prawie 12 mln odczytów (15% wszystkich zmapowanych odczytów).

Do oceny statystycznej istotności różnic w zliczeniach transkryptów między próbkami od myszy na diecie standardowej i wysokotłuszczowej użyto kilku testów zaimplementowanych w pakietach `R`. Pierwszym jest test Manna-Whitneya-Wilcoxona

[Mann and Whitney, 1947] porównujący identyczność dwóch rozkładów nie zakładając, że są one rozkładem normalnym. W `R` test ten dostępny jest jako `wilcoxon.test` i zastosowany został do zliczeń dla każdej sekwencji niezależnie. Następnie skorzystano z dwóch pakietów do analizy różnicowej ekspresji genów: `edgeR` [Robinson et al., 2010] i `DESeq` [Anders and Huber, 2012] oraz `DESeq2` [Love et al., 2014] (każdy w 2 wersjach: z parametrycznym i z lokalnym dopasowaniem dyspersji). Wykorzystano również pakiet dedykowany badaniu różnicowej ekspresji w metagenomach – `metagenomeSeq` [Paulson et al., 2013], który oferuje dwa sposoby modelowania zliczeń odczytów: mieszane rozkłady log-normalny oraz normalny, wzbogacone o wartości zerowe (ang.*Zero-inflated Log-Normal mixture mode* oraz *zero-inflated Gaussian (**ZIG**) mixture model*).

Metoda	Liczba sekwencji o *pval* < 0.05	
	bez poprawki	po poprawce FDR
Mann-Whitney-Wilcoxon	451	0
edgeR	511	0
DESeq local	269	0
DESeq parametric	nieudana estymacja dyspersji	
DESeq2 local	1705	5
DESeq2 parametric	nieudana estymacja dyspersji	
metagenomeSeq LogNorm	9	0
metagenomeSeq ZIG	379	0

Tablica 3.4.1: Porównanie liczby statystycznie istotnych sekwencji dla różnych metod badania różnicowej ekspresji genów zastosowanych do wybranych sekwencji zasemblowanych przy pomocy `MEGAHIT`.

Żadna z metod nie była w stanie zidentyfikować istotnych (*pval* < 0.05 po poprawce FDR Benjaminiego-Hochberga) transkryptów, nie licząc `DESeq2` (patrz tabela 3.4.1). Warto zauważyć, że dyspersji w zliczeniach odczytów nie udaje się wyestymować parametrycznym wzorem proponowanych przez obie wersje `DESeqa` ($y = a/x+b$), i musi być estymowana lokalnie. Przy użyciu lokalnej estymacji i `DESeq2` liczba istotnie zmienionych sekwencji również jest bardzo niewielka (patrz tabela 3.4.2), a p-wartości niewiele mniejsze niż punkt odcięcia 0.05.

Sekwencja	średnia liczba odczytów (SD)	(HF)	log2foldChange	p-wartość (po FDR) dla DESeq2 local
17823_len=802	9.5	1340.6	-7.14	0.035
40722_len=569	5	857.4	-7.42	0.021
129783_len=884	1.75	197.8	-6.82	0.021
182675_len=1284	6.5	1338.6	-7.69	0.021
234995_len=1287	3.75	713.6	-7.57	0.024

Tablica 3.4.2: Wszystkie sekwencje o p-wartości < 0.05 wg. `DESeq2` z lokalnym dopasowaniem dyspersji.

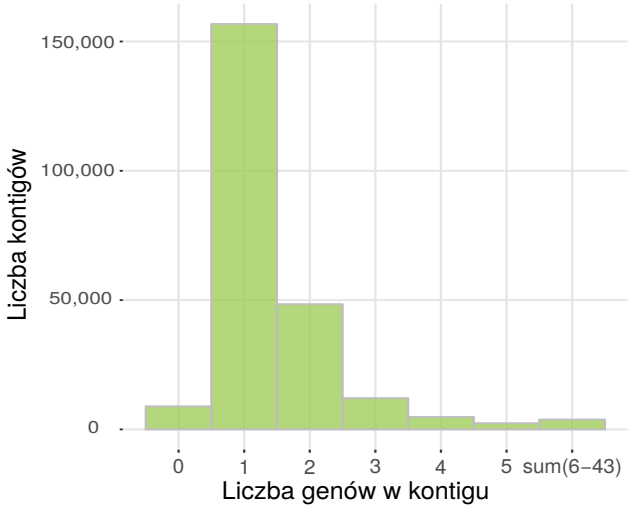
W związku z trudnością w identyfikacji kontigów o zmienionej ekspresji tą metodą, w następnym rozdziale przedstawiona jest procedura zaproponowana przez dr Ilonę Grabowicz, a zrealizowana wspólnie, która pozwoliła zidentyfikować funkcjonalne różnice metatranskryptomu po zmianie diety posiłkując się sekwencjami złożonymi przy pomocy `MEGAHIT`.

3.5. Identyfikacja różnicowo eksprymowanych genów

W związku z dużą długością uzyskanych przez `MEGAHIT` kontigów oraz możliwością kodowania przez wiele bakterii genów o podobnej funkcji (np. wariantów enzymu), dr Ilona Grabowicz zaproponowała procedurę funkcjonalnej analizy zasemblowanych sekwencji, która pozwoliła zidentyfikować kilka genów o różnicowej ekspresji między dietami, oraz ponad setkę genów rozróżniających mysie genotypy.

3.5.1. Identyfikacja eksprymowanych genów na podstawie kontigów

Do identyfikacji sekwencji kodujących w obrębie zrekonstruowanych sekwencji użyłyśmy[3] programu `MetaGeneMark` [Zhu et al., 2010], identyfikującego całe kodujące sekwencje bądź ich fragmenty używając ukrytego modelu Markova (HMM) trenowanego przy użyciu znanych genomów bakteryjnych by rozpoznawać kodujące fragmenty na podstawie częstości nukleotydów. `MetaGeneMark` może wskazać zarówno czy dana sekwencja jest kodująca, jak i wskazać w dłuższej sekwencji jej kodujące fragmenty. Zwykle na jeden kontig z `MEGAHIT` przypadała jedna sekwencja kodująca, jednak wiele miało 2-3 znalezione geny, a rekordowy transkrypt aż 43 (patrz 3.5.1[4]).



Rysunek 3.5.1: Histogram liczby sekwencji kodujących wskazanych w danym kontigu przez MetaGeneMark

Wybierając jedynie sekwencje kodujące wg. `MetaGeneMark` utworzyłyśmy zbiór referencyjny o 353 729 sekwencjach, ponad 205 mln bp, o medianie długości 429. Zmapowałyśmy odczyty wykorzystując `kallisto` z $k = 21$ dopasowując w ten sposób jednoznacznie 65.8% odczytów (patrz tabela 3.5.1[5]).

Sekwencje wybrane przez `MetaGeneMark` zostały dodatkowo zaanotowane funkcjonalnie, żeby zagregować zliczenia dla ortologów – genów powstałych w wyniku

[3] Wykonane przez dr Ilonę Grabowicz
[4] Wykonane przez dr Ilonę Grabowicz
[5] Wykonane przez dr Ilonę Grabowicz

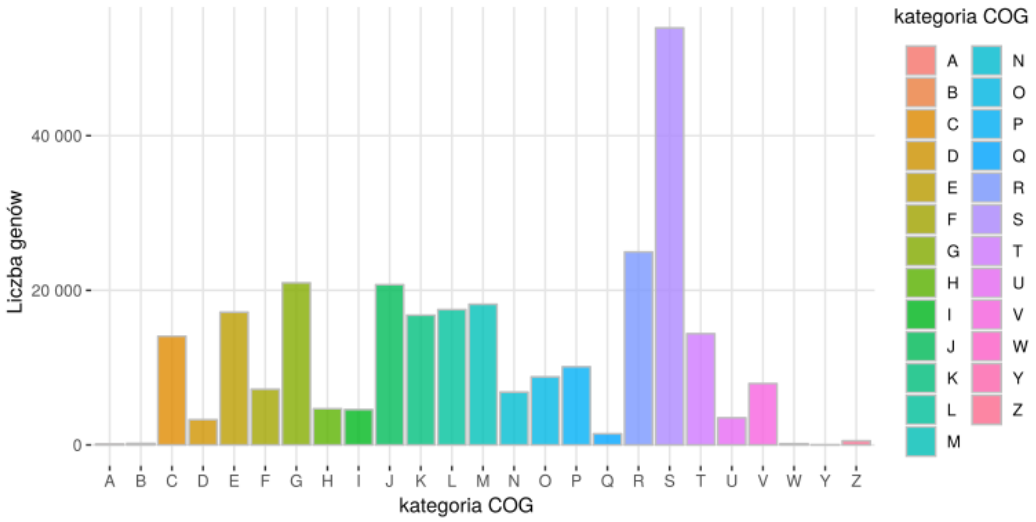
Mapowanie	zmapowane (jednoznacznie)	brak k-merów	konflikty
kallisto k=21	67.1 % (65.8 %)	19.9 %	13.0 %

Tablica 3.5.1: Mapowanie `kallisto` z k = 21 na sekwencje kodujące wg MetaGeneMark.

specjacji, często o tej samej funkcji, ale znajdujących się w genomach różnych organizmów. `EggNOGmapper` [Huerta-Cepas et al., 2017] dopasowuje każdą sekwencję z zapytania do jednej z sekwencji ze swojej bazy używając m.in. ukrytych modeli Markova (HMM) reprezentujących klastry bliskich ortologów, ale również wyszukując sekwencje genów w bazie białek. Funkcja sekwencji przypisywana jest na podstawie podobnej sekwencji z najlepiej dopasowanego klastra i filogenetycznie najbliższych jej ortologów o znanej anotacji funkcjonalnej, np. nazwie kodowanego białka.

Największą ekspresję wykazał gen *fliC* kodujący flagelinę, białko tworzące filamenty w wici, a w związku z tym związane z mobilnością i wirulencją bakterii. Kolejne geny o wysokiej liczbie odczytów to geny biorące udział w replikacji DNA (*dnaK*, *groL*, *clpB*), dojrzewaniu i degradacji białek (*hsp*), translacji (*tuf*), glukoneogenezie (*pckA*), cyklu trójkarboksylowym (*mdh*) i metabolizmie pirogronianu (*ppdK*).

Zidentyfikowane geny należały wg `EggNOGmapper` do 24 spośród 26 funkcjonalnych kategorii, w które zgrupowano klastry grup ortologicznych (COG-sów, z ang. *Clusters of Orthologous Groups* [Galperin et al., 2014]). Najliczniej reprezentowana była kategoria 'S' (patrz rysunek 3.5.2), zawierająca białka o nieznanej funkcji, a następnie kategoria białek o jedynie ogólnie zdefiniowanej funkcji. Kolejne były kategorie: 'G' - metabolizm i transport węglowodanów i 'J' - translacja i struktura rybosomalna.



Rysunek 3.5.2: Histogram liczby zidentyfikowanych genów przypisanych do funkcjonalnych kategorii

3.5.2. Porównanie procedury identyfikacji różnicowych genów przy użyciu kontigów i odczytów

Przybliżony poziom transkrypcji genów o określonej funkcji określa suma zliczeń odczytów przypisanych do sekwencji o określonej funkcji, niezależnie od gatunku, z którego sekwencja pochodzi. Analogicznie, jako rodzaj kontroli przedstawionego powyżej podejścia, można przybliżyć poziom transkrypcji genów o określonej funkcji poprzez określenie funkcji dla każdego odczytu, co jest również możliwe przy pomocy `MetaGeneMark` i `EggNOGmapper`, i zsumowanie liczby przypisań do danej funkcji.

Spośród 20 najczęstszych genów znalezionych przy użyciu kontigów, 12 znalazło się wśród 20 najczęstszych w metodzie kontrolnej (patrz tabela 3.5.2). 4407 genów zostało zidentyfikowanych przez oba podejścia, co stanowi 46% spośród genów znalezionych przy pomocy co najmniej jednej z metod, 814 tylko przy pomocy kontigów, a 4279 tylko przy użyciu odczytów. Spośród nich 95% miało po mniej niż 100 przypisanych odczytów (patrz rysunek 3.5.3[6]).

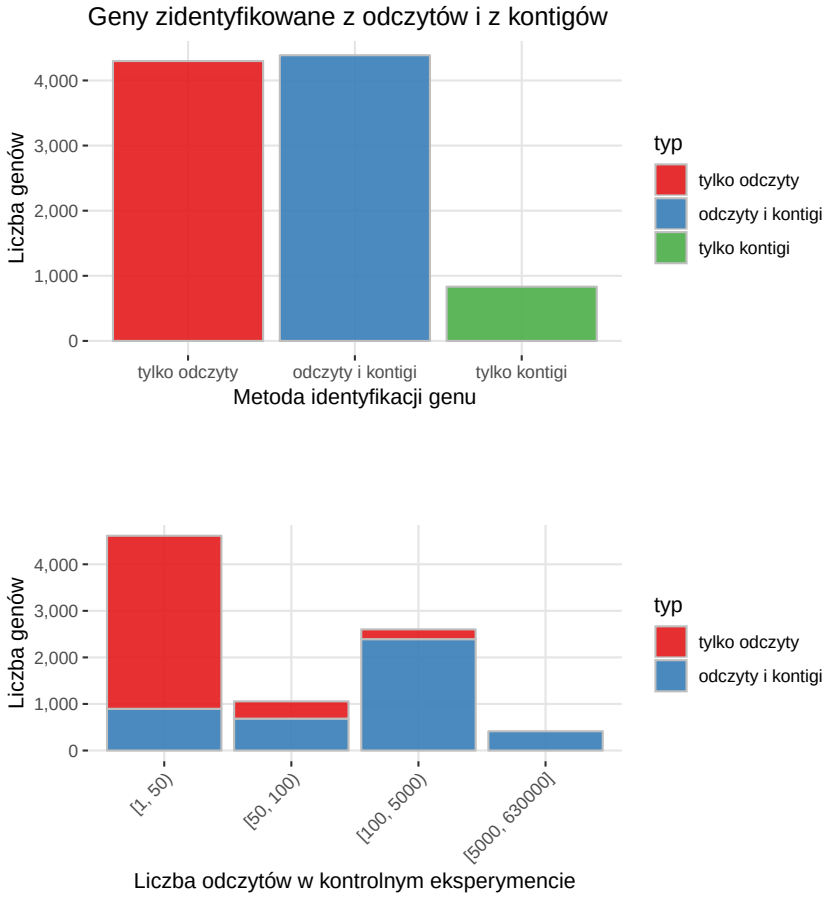
Nazwa genu	Pozycja w rankingu		Liczba odczytów	
	wg kontigów	wg odczytów	kontigi	odczyty
FLIC	1	1	2392259	629853
DNAK	2	2	590624	618575
GROL	3	3	543280	428633
HTRA	4	75	387582	30272
CLPB	5	9	351813	268179
TUF	6	4	326111	417798
PCKA	7	5	269168	339675
HSP	8	42	262838	48174
MDH	9	14	259689	149975
PPDK	10	8	245001	273079
GAPA	11	7	244865	282700
FLID	12	808	240074	1709
NIFJ	13	10	219253	237748
GSIA	14	6713	207437	17
HTPG	15	17	204307	105897
FRC	16	169	203357	15405
GAP	17	6	202582	298039
PG0694	18	–	196516	–
OXC	19	697	171783	2161
SODB	20	60	169933	36012

Tablica 3.5.2: Najczęstsze ortologi.

3.5.3. Geny o zmienionej ekspresji przy zmianie diety

Korzystając z metody `DESeq2` na zagregowanych zliczeniach udało się zidentyfikować 4 istotnie (p-wartość< 0.05, po poprawce FDR) zmienione geny: *ltaS*, *dhbF*, *proWX* oraz *OCAR_4032*, gen pochodzący z bakterii *Oligotropha carboxidovorans*,

[6]Wykonane przez dr Ilonę Grabowicz



Rysunek 3.5.3: Geny zidentyfikowane przez `MetaGeneMark` i `EggNOGmapper` na podstawie odczytów oraz odpowiadające im zliczenia wyliczone na podstawie odczytów.

ale nieznanej funkcji. Najistotniejszy był gen *ltaS*, kodujący syntazę kwasu lipotejchojowego. Kwas lipotejchojowy jest składnikiem ścian komórkowych u bakterii Gram-dodatnich i jest wymagany do wzrostu i podziału komórek bakteryjnych [Gründling and Schneewind, 2007]. Kolejnym istotnie silniej eksprymowanym genem po zmianie diety jest *dhbF*, kodujący nierybosomową syntazę peptydów, która jest częścią ścieżki biosyntezy bacillibaktyny – sideroforu, czyli związku chelatującego jony żelaza. Związek ten pomaga komórkom rosnącym w warunkach ubogich w żelazo wchłaniać ten pierwiastek, więc jego obecność wskazuje na niedobór żelaza w środowisku [Lee et al., 2011].

Ekspresja *proWX* zmniejszyła się istotnie po zmianie diety. Jest to gen kodujący białko systemu transportowego pozwalającego na wchłanianie ze środowiska proliny lub betainy glicynowej (ang. *glycine betaine/proline ABC transporter*). Te substancje są wchłaniane w warunkach wysokiego stresu osmotycznego, czyli w wysokim stężeniu substancji osmotycznie czynnych i ułatwiają im przeżycie [Csonka, 1989].

Ponieważ istotnie zmienionych genów było niewiele, przeanalizowałyśmy kategorie do których można przypisać 100 genów o najmniejszych p-wartościach (< 0.033 bez poprawki) – patrz tabela 3.5.3[7].

Zwiększona obecność genów chroniących przed stresem osmotycznym przy stan-

[7] Wykonane przez dr Ilonę Grabowicz

Kategoria genów	Większa ekspresja na zwykłej diecie SC	Większa ekspresja na diecie wysokotłuszczowej HF
lipopolisacharydy błonowe, biosynteza lipidów	lapB	**ltaS**, lcfA, lpcA, tagA, mapA, dgkA, ugpQ
wirulencja	nanB, cpdA, cdr	iga, yhcA, mecR, SCLAV_0170, sle1, apf1, mapA, agrB, traF2
transpozazy	RV1313C	LPG0144, MPOP_0102
degradacja białek	iadA, sspP	
metabolizm aminokwasów	hgdB, kamA, kamE, yphF, rbsC-2, kce, guaA, kamD, kdd	astA, glxB, tdc, glnN
metabolizm cukrów	SP_2141, nanB, porB, sugC, mngB, mtlD, nagE, fucI, rbsK	pdhA, pdhB, pdhC, bgaC, lacC, ganA, xfp
osmoprotekcja	proWX, opuCA, sugC	
transport membranowy i obrót białek membranowych (ang. *turnover*)	**proWX**, SCLAV_3941, nirC, SP_1328, oppF3, xth, cdr, yhaH, ygjE	cssR, cssS, amt, yhcA, tupA, nagE, opcP, ygiW
regulatory transkrypcyjne nierybosomowa syntaza peptydów	SP_1821, CLS_01740	cytR **dhbF**

Tablica 3.5.3: Kategorie funkcjonalne 100 najbardziej zmienionych genów pomiędzy dietami.

dardowej diecie może sugerować, że dieta wysokotłuszczowa jest mniejszym stresorem dla pewnych bakterii jelitowych. 5 genów zmniejszonych przy tej diecie katalizuje degradację lizyny, skąd można wnioskować, że dieta wysokotłuszczowa prowadzi do zmniejszenia fermentacji lizyny u bakterii. Spośród genów zaangażowanych w wirulencję 9 spośród 12 ma większą ekspresję przy diecie wysokotłuszczowej. Białka kodowane przez te geny ułatwiają bakteriom przetrwanie i kolonizację gospodarza, a niektóre nawet dają odporność na antybiotyki. Mniej zaskakującym wynikiem jest zwiększenie ekspresji genów biosyntezy lipopolisacharydów błonowych i tłuszczów na diecie wysokotłuszczowej, dzięki której po trawieniu tłuszczów w jelicie dostępne stają się dla bakterii glicerol i kwasy tłuszczowe.

Część genów związanych z metabolizmem cukrów jest bardziej aktywnych na diecie wysokotłuszczowej, a część mniej. Wśród bardziej aktywnych genów znajdujemy kilka genów należących do operonu *pdh* (dehydrogenazy pirogronianowej), zwykle transkrybowanego gdy cukier jest czynnikiem limitującym wzrost bakterii. Myszy na diecie wysokotłuszczowej spożywają mniej zwykłej karmy, bogatej w węglowodany, więc ekspresja tego operonu na tej diecie nie dziwi. Ponadto po diecie wysokotłuszczowej aktywowane zostały oba geny z systemu regulacyjnego CssR-CssS, który służy do wykrywania wadliwie zwiniętych białek eksportowanych na zewnątrz komórki i jest związany ze stresem związanym z wydzielaniem (ang. *secretion stress*) [Hyyryläinen et al., 2001].

3.5.4. Geny o różnej ekspresji pomiędzy genetycznymi populacjami

Genotyp osobnika ma znaczący wpływ na jego mikrobiom. Porównując próbki pochodzące od myszy dzikich do próbek od myszy trisomicznych (bez względu na dietę), udało się zidentyfikować aż 137 statystycznie istotnie różnie eksprymowanych genów (p-wartość < 0.05 po poprawce FDR). Najliczniej reprezentowane przez te geny kategorie funkcjonalne to regulacja transkrypcji, transport i wymiana (ang. *turnover*) błon komórkowych, biogeneza ściany i otoczki komórkowej oraz metabolizm węglowodanów (patrz tabela 3.5.4).

Kategoria genów	Większa ekspresja u myszy WT	Większa ekspresja u DS
metabolizm hipoksantyny		hbaC, ygeT, xdhC, allD, ylbA
wirulencja/interakcje z komórkami gospodarza	spaA, quiP, nodI, sdrC, phnB, exoT, tetM, VSAL_I0172, enhC	
transpozazy	CYCMA_1561, DSY0812, THEXY_0761, DACET_0688, RPE_0249, SSQG_02712, BPR_I0156, DALK_0900	CPHY_1803, CALNI_0146
regulacja transkrypcji	ATP_00026, csm3, treR, LP_2003, LLO_1103, puuR, csiR, sigI, SP_1221, agaR	XYLR-1, ocaR_4148, rpiR2, ygeV, gcvR, GYY_09435, yazB, RHOM_13735, nfi, araC8
transport przez błony/obrót białek błonowych	nodI, BL01808, sov, agaF, fecR2, potC, epsK, lktB3, rhaQ	xth, BMUL_4061, licB, yhaI, clvE, ptcC, yjbH
biogeneza ścian i otoczek komórkowych	spaA, ykvP, VSAL_I0172, murE	cap5L, cap5H, wchF, ylbE, mgt, BMUL_2968
metabolizm węglowodanów	cbpJ, bgaA, nanM, rsgI, wcfF	pck, bglA, xylBch, amy, aepX, yegV, agp

Tablica 3.5.4: Kategorie funkcjonalne 137 statystycznie istotnie zmienionych genów pomiędzy genotypami myszy.

Spośród genów różnicujących oba genotypy geny metabolizmu hipoksantyny były bardziej aktywne u myszy trisomicznych, a geny odpowiadające za wirulencję i interakcje z komórkami gospodarza, jak również geny kodujące transpozazy były bardziej aktywne u myszy WT. Ksantyna i hipoksantyna są produktami metabolizmu puryn u ludzi. Mogą być konwertowane do kwasu moczowego, a dalej do alantoiny (w warunkach dostępnych wolnych rodników). U myszy trisomicznych zwiększoną aktywność wykazywały geny *xdhC, hbaC, ygeT*, biorące udział w konwersji hipoksantyny do kwasu moczowego.

U myszy WT spośród 9 silniej eksprymowanych genów wirulencji dwa z nich, *spaA* i *sdrC*, kodują białka ułatwiające bakteriom adhezję do komórek gospodarza,

a geny *tetM* i *enhC* przyczyniają się do odporności na antybiotyki. Gen *nodI* jest genem bakterii kolonizujących korzenie roślin, więc być może przypisanie jest błędne i w próbce znajduje się jakiś ortolog tego genu z innych bakterii.

3.5.5. Podsumowanie

Dzięki zastosowaniu wyszukiwania kodujących fragmentów z transkryptów oraz agregacji wariantów genów udało się zidentyfikować geny różnicujące zarówno diety, jak i genotypy. Odejście od patrzenia na mikrobiom jako na zbiór obecnych gatunków w stronę spojrzenia na ogół funkcjonalnych możliwości mikrobiomu może pozwolić lepiej zrozumieć sposób, w jaki całość mikrobiomu oraz organizm gospodarza oddziałują na siebie wzajemnie. Warto wziąć to podejście pod uwagę w przyszłych analizach, chociaż liczba istotnie zmienionych genów nie jest w tym przypadku duża, a liczba kroków, które mogą być obarczone błędami, spora.

Jednym z problemów jest podwójne użycie odczytów – najpierw do asemblacji, a następnie do mapowania na uzyskane kontigi. Są to kroki wymagające obliczeniowo, których można uniknąć. Słabością tego podejścia jest również uwzględnienie w analizie jedynie znanych genów, nawet jeśli kodują białka o nieznanej funkcji. To ogranicza możliwość odkrycia nowych transkryptów, niepodobnych do genów pochodzących z dotychczas zsekwencjonowanych organizmów. Dlatego w następnym rozdziale rozważona jest możliwość i sposoby, a także zaprezentowane wyniki jednoczesnej asemblacji i identyfikacji różnicowych transkryptów.

ROZDZIAŁ 4

Asemblacja z jednoczesną identyfikacją różnicowych sekwencji

W tym rozdziale przedstawiony jest problem jednoczesnej rekonstrukcji (asemblacji) sekwencji oraz identyfikacji tych sekwencji, które różnicują dwie grupy próbek. Sekcja 4.1 zawiera formalną definicję problemu i omawia możliwość wykorzystania dwóch metod asemblacji – grafu *de Bruijna* oraz grafu nałożeń, sekcja 4.2 przedstawia zastosowanie grafu nałożeń do danych eksperymentalnych, a sekcja 4.3 prezentuje heurystyczne podejście do asemblacji różnicowych sekwencji.

4.1. Problem asemblacji różnicowych kontigów

4.1.1. Definicja problemu

Zdefiniowanie problemu asemblacji różnicowych kontigów jest skomplikowanym zagadnieniem z wielu względów. Pierwszym jest wybór kryterium „różnicowości". Najogólniejszą i najprostszą tego miarą jest krotność zmiany (*fold-change*) pomiędzy (znormalizowanymi przez liczbę odczytów z danej próbki lub z warunku) zliczeniami z jednego i drugiego warunku. Nie wymaga użycia skomplikowanych modeli biorących np. pod uwagę rozkład częstości zliczeń dla różnych długości sekwencji czy zależności różnicowości od całkowitej liczby zliczeń danej sekwencji, jest zatem odpowiednia do opisanych później proponowanych rozwiązań.

Ponadto, w klasycznym problemie asemblacji chcemy maksymalizować liczbę reprezentowanych przez odtworzoną sekwencję odczytów. Jednak gdy zasemblujemy jedynie sekwencje różnicowe, liczba reprezentowanych odczytów będzie znacznie mniejsza, i być może nie powinna być główną miarą sukcesu. Jednocześnie chcemy również minimalizować długość zasemblowanej sekwencji, tak jak w klasycznym problemie asemblacji, z drugiej strony chcemy zidentyfikować jak najwięcej sekwencji różnicowych. Główne założenie tego problemu przedstawia poniższa definicja:

Definicja 4.1.1 (Problem asemblacji *FC*-różnicowych kontigów)
Dane są dwa zbiory próbek zwyczajowo nazywane — traktowane $\mathcal{T} = \{S_1^t, \ldots, S_{n_t}^t\}$ *(ang.* treated*) oraz kontrolne* $\mathcal{C} = \{S_1^c, \ldots, S_{n_c}^c\}$ *(ang.* control*), oraz* FC *– wartość oczekiwanej krotności zmiany.*

Znajdź zbiór sekwencji $\mathcal{R} = \{r_1, r_2, \ldots r_{|\mathcal{R}|}\}$ *oraz mapowanie* $\mathcal{M}_\mathcal{R}$ *takie, że*

- *każda sekwencja $r_i \in \mathcal{R}$ jest FC-różnicowa:*

$$fc(i) = \frac{\overline{c}_{\mathcal{M}_\mathcal{R},\mathcal{T}}(r_i)}{\overline{c}_{\mathcal{M}_\mathcal{R},\mathcal{C}}(r_i)} \geq FC \quad lub \quad fc(i) \leq \frac{1}{FC} \quad lub \quad \overline{c}_{\mathcal{M}_\mathcal{R},\mathcal{C}}(r_i) = 0$$

  gdzie zliczenia dla zbioru próbek zdefiniujemy jako znormalizowaną sumę zliczeń z próbek $\overline{c}_{m_R,\mathcal{T}}(g_i) = \frac{\sum_j c_{m_R,S_j^t}(g_i)}{\sum_j |S_j^t|}$

- *$\mathcal{R}$ jest wynikiem asemblacji pewnego podzbioru sekwencji S, ozn. $\mathcal{A}$, czyli*

$$\mathcal{A} = \{s_i \in \mathcal{C} \cup \mathcal{T}\} : \exists_{1 \leq j \leq |G|} s_i \sqsubseteq_m G_j$$

Szukając optymalnego rozwiązania problemu asemblacji FC-różnicowych kontigów należy wybrać kryterium, a częściej kryteria oceny rozwiązania. Cechy, których można oczekiwać od optymalnego rozwiązania to:

1. jak największa liczba uwzględnionych odczytów, czyli maksymalizacja $|\mathcal{A}|$
2. jak największa sumaryczna długość rozwiązania ($\sum_{r_i \in \mathcal{R}} |r_i|$) przy minimalnej liczbie zrekonstruowanych sekwencji $|\mathcal{R}|$
3. jak największa liczba zrekonstruowanych sekwencji $|\mathcal{R}|$
4. maksymalne średnie pokrycie sekwencji, czyli $cov_{m_R,S}$
5. jak największa „różnicowość" rekonstruowanych sekwencji, np. maksymalne $fc(i)$ lub największe średnie $fc(i)$ (biorąc $\frac{1}{fc(i)}$ jeśli $fc(i) \leq 1$)

Niestety niektóre z powyższych kryteriów są wzajemnie przeciwstawne, a optymalizacja niektórych z nich prowadzi do niepożądanych rozwiązań. Na przykład warunek (1), maksymalizację $|\mathcal{A}|$, można spełnić biorąc $\mathcal{R} = \mathcal{C} \cup \mathcal{T}$, spełniając również warunek (3), ale wtedy minimalizowany w warunku (2) $|\mathcal{R}|$ jest maksymalny, a średnie pokrycie (warunek (4)) jest najmniejsze możliwe: $cov_{m_R,S} = 1$. Maksymalizując warunek (2) można przedłużać sekwencje i w wyniku tego zmniejszać średnią różnicowość (warunek (5)) a być może też średnie pokrycie (warunek (4)). Maksymalizować warunek (5) można np. poprzez osobną asemblację zbiorów $\mathcal{C}$ i $\mathcal{T}$, jednak jest to sprzeczne z warunkiem (2) oraz być może (4).

Jednym ze sposobów uproszczenia definicji problemu jest ograniczenie do wyszukania jednej różnicowej sekwencji (ozn. R), która spełnia warunki:

- R jest wynikiem asemblacji pewnego podzbioru odczytów
- R jest różnicowa
- R jest maksymalna, tzn. nie istnieje sekwencja R' taka, że $R \sqsubset R'$, R' różnicowa oraz mapuje się na nią więcej odczytów: $c_{m'_R,S}(1, |R'|) > c_{m_R,S}(1, |R|)$

Szukając kolejnych takich R na zbiorach odczytów pomniejszonych o odczyty użyte w asemblacji poprzedniej sekwencji otrzymujemy zachłanny algorytm wyboru pewnego zbioru FC-różnicowych sekwencji.

Dodatkowym uproszczeniem, którego można użyć w przypadku rekonstrukcji transkryptów z odczytów NGS pochodzących z sekwencjonowania „strand-specific"

jest brak konieczności rozważania odwrotnej komplementarności, ponieważ przy tym protokole wszystkie odczyty z transkryptu powinny mieć tę samą orientację.

Asemblacja FC-różnicowych kontigów przy użyciu podejść grafowych, zarówno grafu nałożeń, jak i grafu *de Bruijna* sprowadza się do wyznaczenia zbioru ścieżek w grafie. Różnicowa ścieżka $Q = (q_1 \rightarrowtail ... \rightarrowtail q_p)$ w grafie powinna spełniać zatem warunki:

- Q jest FC-różnicowa, czyli suma odczytów z $\mathcal{T}$ odpowiadających tej ścieżce jest FC razy większa lub mniejsza niż suma odczytów z $\mathcal{C}$: $\frac{cov_{Q,\mathcal{T}}}{cov_{Q.\mathcal{C}}}$
- Q jest maksymalna, tzn. nie istnieje ścieżka H taka że $Q' = Q \rightarrowtail H$ lub $Q' = H \rightarrowtail Q$ i Q' również jest różnicowa

4.1.2. Różnicowe kontigi w asemblacji z użyciem grafu *de Bruijna*

W grafie *de Bruijna* odczyty są reprezentowane przez ścieżkę w grafie, jednak mogą i zwykle istnieją ścieżki nieodpowiadające odczytom, a jedynie zawierające ich fragmenty. Liczba odczytów z S odpowiadających danej ścieżce Q, czyli pokrycie Q: $cov_{Q,S}$, konieczna do obliczenia krotności zmiany, nie jest jednoznacznie określona. Liczbę odczytów dla takiej sekwencji może więc określać:

- (a) liczba odczytów, które przecinają się z daną ścieżką, czyli mają wspólny wierzchołek (k-mer). To podejście przypomina podejście `kallisto` do mapowania, gdzie jeden k-mer z odczytu dopasowany do k-meru z transkryptu wystarcza do włączenia transkryptu do potencjalnych miejsc mapowania.
- (b) liczba odczytów całkowicie zawartych w ścieżce (każdy wierzchołek ze ścieżki odczytu powinien być zawarty w ścieżce aby zaliczyć odczyt do $cov_{Q,S}$. To podejście jest z kolei jest bliższe uliniowieniu odczytu.

Dla obu tych definicji jednak obliczenie pokrycia sekwencji jest trudne obliczeniowo, bo wymaga w przypadku (a) obliczenia sumy zbiorów odczytów odpowiadającym każdemu wierzchołkowi ze ścieżki, albo w przypadku (b) sprawdzenia dla każdego odczytu, czy całkowicie pokrywa się ze ścieżką. Bardziej intuicyjne i prostsze jest jednak podejście (a), i tak też zdefiniujmy zliczenia odczytów w grafie *de Bruijna*.

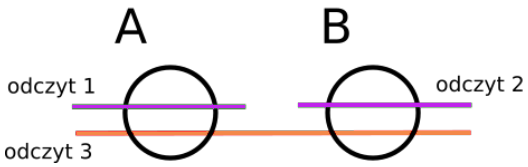
Dodatkową trudnością w asemblacji różnicowych kontigów przy pomocy grafu *de Bruijna* jest nieprzewidywalność funkcji fc. czyli to, że przedłużanie ścieżki w tym grafie może dawać różne zmiany wartości funkcji fc:

Obserwacja 4.1.1
Krotność zmiany połączonych ścieżek A i B $fc(A, B)$ może dawać wartości większe lub mniejsze od $fc(A)$ i $fc(B)$.

Wynika to z nieaddytywności zliczeń odczytów w ścieżkach w grafie *de Bruijna*. Przykład nieprzewidywalności funkcji fc występuje np. w przypadku grafu przedstawionego na rysunku 4.1.1.

W ogólności obserwujemy, że zachodzi:

Obserwacja 4.1.2
Dla dowolnego poziomu odcięcia $t > 1$ i wielkości k-meru k, istnieje graf de Bruijna *gdzie każdy wierzchołek ma $fc < t$, ale istnieje ścieżka o $fc \geq t$.*



Rysunek 4.1.1: Przykład grafu *de Bruijna*, w którym dla $FC = 2$ każdy z wierzchołków (A, B) ma $fc < FC$ a ścieżka $A \rightarrowtail B$ ma $fc \geq FC$ (odczyty fioletowe – 1 i 2 są z jednego warunku, odczyt pomarańczowy – 3 z drugiego): $fc(A) = fc(B) = \frac{1\cdot\text{fioletowy}}{1\cdot\text{pomaranczowy}} = 1$ a $fc(A + B) = \frac{2\cdot\text{fioletowy}}{1\cdot\text{pomaranczowy}} = 2$

Uzasadnienie: Skonstruujmy graf zawierający ścieżkę długości n, której każdy z wierzchołków pokryty jest a odczytami w warunku 1 (każdy z wierzchołków innym zbiorem a odczytów, a b odczytami w warunku 2 (tym samym zbiorem b odczytów. Załóżmy, że graf nie jest możliwy do uproszczenia, bo odczyty z warunku 1 generują rozgałęzienia. Dobierzmy takie a i b, że dla ścieżek długości $m < n$ $\frac{am}{b} < t$ a $\frac{an}{b} \geq t$. W szczególności niech $\frac{a(n-1)}{b} < t$. Wystarczy zatem dobrać naturalne a, b i n takie, że:

$$\frac{bt}{n} \leq a < \frac{bt}{n-1}$$

Z obserwacji 4.1.2 możemy wysnuć wniosek, że każda ze ścieżek w grafie *de Bruijna* (których może być nawet 4^k w grafie pełnym) może być różnicowa i nie można różnicowości stwierdzić jedynie na podstawie fc poszczególnych wierzchołków. Stąd następujący wniosek, że:

Obserwacja 4.1.3
Aby znaleźć ścieżkę różnicową w pesymistycznym przypadku konieczne jest obliczenie fc dla wszystkich ścieżek.

Z powodu 4.1.3 graf *de Bruijna* nie jest użyteczny w przypadku asemblacji FC-różnicowych kontigów.

4.1.3. Różnicowe kontigi w asemblacji z użyciem grafu nałożeń

W grafie nałożeń każdy odczyt jest przypisany tylko do jednego wierzchołka, zatem pokrycie wierzchołka jest łatwo obliczyć – przed uproszczeniem jest równe 1 – odczytowi, który buduje wierzchołek, a po uproszczeniu grafu sumie pokryć wierzchołków, które zostały uproszczone. To znacznie upraszcza obliczanie pokrycia dla ścieżek, gdyż pokrycie ścieżki $Q = q_1, q_2, \ldots q_l$ jest sumą pokryć wierzchołków:

$$cov(Q, S) = \sum_{q_i \in Q} c(q_i, S)$$

Graf nałożeń ma dzięki temu wygodną własność:

Obserwacja 4.1.4
Gdy połączymy ścieżki P i Q, to wartość funkcji fc będzie ograniczona przez jej wartości dla pojedynczych ścieżek:

$$\min(fc(P), fc(Q)) \leq fc(P + Q) \leq \max(fc(P), fc(Q))$$

Z powyższej (4.1.4) własności łączenia ścieżek, w tym ścieżek jednowierzchołkowych, wynika kolejna obserwacja:

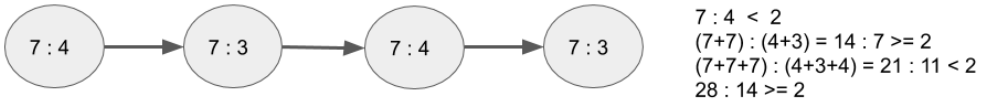
Obserwacja 4.1.5
Ścieżką maksymalizującą funkcję krotności zmiany fc jest pewna ścieżką składającą się z jednego wierzchołka.

Uzasadnienie:

Ponieważ z obserwacji 4.1.4 wynika, że przedłużanie ścieżki nie zwiększa maksymalnego fc dwóch ścieżek (a zwykle zmniejsza), to wierzchołek o największym fc maksymalizuje fc.

Chociaż mogą istnieć dłuższe ścieżki o równym fc, to w praktycznych zastosowaniach definicja problemu asemblacji różnicowych kontigów zakładająca jedynie szukanie ścieżki, która maksymalizuje funkcję fc, będzie faworyzowała krótkie sekwencje. Zatem maksymalizacja fc nie powinna być jedynym celem w asemblacji z użyciem grafu nałożeń.

W zidentyfikowaniu najdłuższych sekwencji spełniających warunek minimalnej różnicowości $fc \geq FC$ kłopot mogą sprawiać grafy zawierające ścieżki, w których wierzchołki na zmianę przekraczają próg istotności. Przykład takiej ścieżki przedstawiony jest na rysunku 4.1.2. W przypadku takich ścieżek widać, że nie można przerwać wydłużania ścieżki przed sprawdzeniem wszystkich możliwości.



Rysunek 4.1.2: Przykład ścieżki o takich pokryciach (zaprezentowanych w wierzchołkach w formie *zliczenia w warunku pierwszym : w warunku drugim*), dla których trzeba sprawdzać wszystkie podścieżki żeby znaleźć optymalną.

Takich ścieżek może być więcej, i mogą się przeplatać. Przykładem to obrazującym może być graf zawierający ciąg wierzchołków na zmianę o $fc \geq FC$ i $fc < FC$, przy czym kolejne wierzchołki $fc < FC$ są połączone z dwoma wierzchołkami o $fc \geq FC$ (podobnie jak na przykładzie 4.1.2 jednak zamiast jednego wierzchołka 7:3 są 2, tworzące dwie równoległe ścieżki łączące wierzchołki 7:4). Wynika z tego następująca obserwacja:

Obserwacja 4.1.6
Istnieją grafy nałożeń, które wymagają sprawdzania różnicowości wszystkich ścieżek w grafie.

Zatem, chociaż z obserwacji 4.1.3 i 4.1.6 wynika, że teoretycznie graf nałożeń może być lepszym narzędziem do asemblacji FC-różnicowych kontigów niż graf *de Bruijna*. Graf *de Bruijna* nie tylko wymaga obliczenia krotności zmiany dla wszystkich ścieżek, obliczenie tego jest również bardziej złożone obliczeniowo. W grafie nałożeń obserwacja 4.1.5 powoduje, że trzeba wybrać definicję problemu polegającą raczej na maksymalizacji długości ścieżki, a nie jej krotności zmiany, ale również z 4.1.6 wynika konieczność przeszukiwania potencjalnie wszystkich ścieżek w grafie. To powoduje, że zamiast szukać odpowiedniego sformułowania oraz dokładnego rozwiązania problemu, warto wykorzystać algorytmy heurystyczne.

Własności grafu nałożeń, a w szczególności addytywność zliczeń po połączeniu ścieżek, pozwalają szukać różnicowych ścieżek np. w sposób zachłanny. Mimo, że tym sposobem nie da się zagwarantować maksymalnej długości wybranej ścieżki, gdyż zachłanny wybór sposobu rozszerzenia ścieżki może prowadzić do wyboru lokalnego minimum, to jest to stosunkowo proste obliczeniowo rozwiązanie, a zatem warte sprawdzenia działania w praktyce. Aby dopasować rozwiązanie do grafu nałożeń dla eksperymentalnych danych metatranskryptomicznych, trzeba poznać własności takiego grafu. W następnym rozdziale opisana jest analiza parametrów grafu nałożeń dla metatranskryptomicznych danych opisanych w rozdziale 3.1.

4.2. Graf nałożeń dla przykładowych danych transkryptomicznych

Użycie grafu nałożeń do asemblacji różnicowych transkryptów jest obiecującym sposobem rozwiązania tego problemu, jednak jego użyteczność dla danych eksperymentalnych zawierających błędy może być ograniczona. W tym podrozdziale przedstawiona jest struktura grafu nałożeń wygenerowanego przy pomocy programu `SGA` (String Graph Assembler, opisanego wcześniej w sekcji 2.1.7) oraz jego działania upraszczające graf.

4.2.1. Działanie `SGA`

Przy standardowej asemblacji z użyciem `SGA` na uzyskane kontigi mapowało się niewiele odczytów, jedynie niecałe 40%. Szczęśliwie `SGA` to zestaw programów, a poszczególne etapy wstępnego przetwarzania danych, budowy grafu, a następnie asemblacji są osobnymi skryptami, dzięki czemu można łatwo modyfikować niektóre części z całej procedury oraz analizować dokładniej poszczególne etapy, opierając się nie tylko na informacjach wypisywanych przez program, ale również analizując pośrednie struktury danych zapisywane przez `SGA` do plików.

Programy składające się na `SGA` to:

1. `preprocessing` – usuwanie odczytów zawierających literę 'N' oraz opcjonalnie filtrowanie wg jakości odczytu
2. `index` – budowa FM-indeksu (musi być wywoływana kilkakrotnie, po każdej zmianie zestawu odczytów)
3. `merge` – łączenie zbiorów próbek. Domyślnie usuwa duplikaty
4. `correct` – korekcja odczytów na podstawie częstości k-merów lub nałożeń — k-mery rzadko występujące w odczytach (domyślnie mniej niż 3 razy) mogą zostać skorygowane na k-mer występujący częściej a różniący się jedną literą. Operacja powtarzana jest iteracyjnie
5. `filter` – filtrowanie odczytów na podstawie częstości k-merów — odczyty zawierające rzadkie k-mery zostaną usunięte
6. `rmdup` – rozdzielenie odczytów na zbiór zawierający pierwsze wystąpienia danej sekwencji i na powtórzenia

7. `overlap` – obliczenie nałożeń odczytów. Dwa odczyty mogą mieć > 1 krawędź między sobą (np. przez różne długości nałożenia), co powoduje powstanie zduplikowanych krawędzi (ang. *„duplicated edges”*). `Overlap` domyślnie nie zwraca przechodnich krawędzi (ang. *„transitive edges”*)

8. `assemble` – wczytuje graf na podstawie obliczonych nałożeń i iteracyjnie upraszcza tworząc w ten sposób kontigi

9. `scaffold` i `scaffold2fasta` – łączą utworzone wcześniej kontigi na podstawie mapowania na nie par odczytów

Z punktu widzenia analizy grafu nałożeń najciekawsze są kroki wykonywane przez program `assemble`. Przydatne w rozumieniu tego procesu są następujące pojęcia używane przez `SGA`:

Definicja 4.2.1 (Wierzchołek super-repetytywny)
Wierzchołek, który ma więcej niż `MAX_EDGES` (domyślnie 128) krawędzi wyjściowych lub więcej niż `MAX_EDGES` krawędzi wejściowych. Krawędzie super-repetytywne to wszystkie krawędzie super-repetytywnego wierzchołka. Domyślnie usuwane są wszystkie krawędzie super-repetytywne.

Definicja 4.2.2 (Krawędź zduplikowana)
Krawędź $v \rightarrow w$ między wierzchołkami v i w, dla których istnieje inna krawędź $v \rightarrow' w$, reprezentująca nałożenie o większej długości.

Definicja 4.2.3 (Wyspa)
Wyspa to wierzchołek bez sąsiadów. Wyspy o długości mniejszej niż minimalna długość kontigu są usuwane.

Definicja 4.2.4 (Końcówka)
Końcówka to wierzchołek o zerowym stopniu wyjściowym lub wejściowym (niebędący jednocześnie wyspą). Jej długość to długość sekwencji „nienałożonej” na inne wierzchołki.

Działanie programu `assemble`:

1. wczytanie grafu (pominięcie zduplikowanych i super-repetytywnych krawędzi)

2. usunięcie odczytów zawartych w innych odczytach (domyślnie robi to `overlap`)

3. iteracyjne (domyślnie 10 razy) upraszczanie grafu:

   (a) łączenie prostych ścieżek,

   (b) usuwanie końcówek o długości mniejszej niż `trimLength=300`,

   (c) usuwanie małych wysp (mniejszych niż minimalna długość kontigu)

4. Wygładzanie grafu (ang. *„Variation smoothing”*), czyli usuwanie bąbli (również iteracyjnie)

Działanie programu `scaffold`:

1. estymacja odległości między kontigami na podstawie sparowanych odczytów zmapowanych na nie (przy pomocy programu `bwa`)

2. opcjonalnie: wyszukanie ścieżek grafie między kontigami o wyestymowanej odległości

3. w przypadku wielu takich ścieżek wybór

   - żadnej
   - dowolnej o długości najbliższej wyestymowanej
   - długości najbliższej wyestymowanej jeśli jest tylko jedna o tej długości

4. opcjonalnie: jeśli nie ma odpowiedniej ścieżki możliwe jest ponowne wykorzystanie nałożeń i użycie nałożenia z `maxErrorRate=0.05`

5. opcjonalnie: zwrócenie kontigów na które nie zmapował się żaden (sparowany) odczyt.

W analizie struktury grafu nałożeń wykorzystałam `SGA` do wykonania części z kroków, takich jak filtrowanie odczytów, obliczenie nałożeń i budowa grafu, a zaimplementowałam w sposób analogiczny do `SGA` upraszczanie i oczyszczanie grafu tak, aby wykonując te kroki śledzić różne parametry grafu oraz liczby odczytów odpowiadające każdemu wierzchołkowi, wliczając w to duplikaty odczytów odfiltrowane na wcześniejszych etapach. Pozwala to sprawdzić na których etapach „gubione" są odczyty, a potem użyć zapamiętanych pokryć do wyszukiwania różnicowych kontigów.

4.2.2. Właściwości grafu dla rzeczywistych danych metatranskryptomicznych

Dzięki temu, że `SGA` to zbiór programów, możliwe jest dopasowanie poszczególnych jego elementów do indywidualnych potrzeb i połączenie ich w bardziej skomplikowaną procedurę analizy danych. Uruchomienie SGA wiąże się np. z usunięciem zduplikowanych odczytów, jednak w kolejnych krokach asemblacji różnicowych transkryptów potrzebna jest informacja o liczbie wystąpienia każdego odczytu, czyli również o usuniętych duplikatach odczytów, wraz z informacją z którego warunku one pochodzą. Opisana poniżej procedura pozwala zachować w pobocznych plikach takie informacje oraz połączyć wiele próbek ostatecznie w jeden graf nałożeń.

Aby uzyskać graf nałożeń dla dwóch zbiorów próbek przy użyciu `SGA` oraz informację o pochodzeniu zduplikowanych odczytów wykonano następujące kroki:

- każda próbka została osobno wstępnie obrobiona przez `preprocess` (w trybie paired-end), z usunięciem pozycji w odczytach o jakości < 5
- każda próbka została poprawiona z `sga correct` przy długości k-meru $k = 21$.
- próbki zostały odfiltrowane z użyciem `sga filter` bez usuwania odczytów z rzadkimi k-merami oraz bez domyślnego usuwania zduplikowanych odczytów.
- następnie próbki zostały połączone w 2 pliki (po jednym dla każdego warunku) poprzez zapisanie odczytów w jednym pliku
- przy użyciu `sga rmdup` usunięte zostały duplikaty dla każdego z warunków
- odczyty z obu warunków zostały połączone przez `sga merge`, co powoduje, że usunięte zostały duplikaty (odczyty występujące w obu warunkach)

- aby odtworzyć, które odczyty są duplikatami, używam `sga overlap` z opcją `target-file`. Pozwala ona policzyć nałożenia odczytów z jednego pliku na odczyty z drugiego
- graf nałożeń dla połączonych odczytów ze wszystkich próbek (z usuniętymi duplikatami) został zbudowany z długością nałożenia $m = 31$ przy użyciu `sga overlap`

W dalszej części pracy analizowany jest graf nałożeń uzyskany przy użyciu `SGA` w powyższej konfiguracji i przy opisanych powyżej parametrach. Dzięki użyciu plików wynikowych z tej procedury we własnej implementacji oczyszczania grafu śledzone są liczby odczytów usuwanych w kolejnych krokach, a później możliwa jest różnicowa asemblacja.

Pełen graf nałożeń zbudowany przez `SGA` dla zbioru odczytów ze wszystkich próbek zawiera ponad 29 mln wierzchołków (odpowiadających unikalnym odczytom) oraz ponad 219 mln krawędzi (patrz tabela 4.2.2. 9% spośród nich to krawędzie o orientacji zbieżnej lub rozbieżnej (wynikające z nałożeń prefiksowych: $\leftrightarrow$ lub sufiksowych: $\rightarrowtail\!\!\!\!\leftarrow$). Takie krawędzie nie powinny się pojawiać w wyniku sekwencjonowania mRNA z wykorzystaniem protokołu „strand-specific", zatem uznawane są za błędne i usuwane z grafu.

	Wierzchołki:	Krawędzie:	% odczytów:
Początkowo	29 343 365	219 236 275	81.15%
w tym:			
wyspy	548 897		
krawędzie prefiksowe/sufiksowe		19 782 441	
super-repetytywne ($deg > 128$)	242 613	283 381 335	

Tablica 4.2.1: Parametry grafu początkowego.

Rozmiar nieoczyszczonego grafu nałożeń jest bardzo duży, co jest powodowane w dużej części przez krawędzie super-repetytywne. Jest ich w grafie bardzo wiele i są prawdopodobnie wynikiem sekwencjonowania regionów repetytywnych lub innych regionów o bardzo dużej zmienności genetycznej. Usunięcie wszystkich krawędzi wychodzących z wierzchołków repetytywnych bardzo zmniejsza rozmiar grafu – pozostaje tylko 14% z początkowej liczby krawędzi, mimo że niecały 1% wierzchołków jest super-repetytywnych.

Wierzchołki w prawie połowie przypadków zawierają tylko 1 odczyt, maksymalną wartością jest 145386 odczytów, a wierzchołki mające więcej niż 50 odczytów zawierają 8.6% odczytów. Pokrycie w obu grupach próbek ma 17.2% wierzchołków.

Początkowy graf zawiera 630 295 spójnych składowych (pomijając pojedyncze wierzchołki), 189 226 składowych o dwóch wierzchołkach i 25 354 składowe o co najmniej 100 wierzchołkach. Takie rozbicie grafu nie jest pożądane dla danych genomicznych, jednak w przypadku danych metatranskryptomicznych jest spodziewane.

4.2.3. Upraszczanie grafu

Upraszczanie grafu służy nie tylko zmniejszeniu rozmiaru ale także usunięciu ścieżek prawdopodobnie wynikających z błędów sekwencjonowania. Domyślnie w `SGA` 10 razy wykonywany jest cykl upraszczania – usuwanie końcówek, łączenie prostych ścieżek, i usuwanie małych wysp. W tabeli 4.2.2 zaprezentowane są wyniki własnej

implementacji tylko jednego cyklu oczyszczania (zaczynając od łączenia ścieżek przed usunięciem końcówek), aby zachować jak najwięcej odczytów. Porównane jest to do wyników całej procedury `SGA` (10 iteracji i usuwanie bąbli), uruchomionej na grafie uzyskanym w domyślny sposób, a także uruchomionej na grafie po usunięciu krawędzi prefiksowych i sufiksowych (dwa ostatnie wiersze tabeli).

	Wierzchołki:	Krawędzie:	Wyspy:	% odczytów:
Początkowo:	29 343 365	32 184 873	2 090 441	81.15%
- proste ścieżki		22 146 563		
(liczba ścieżek)	2 170 948			
- małe wyspy	2 601 781			
Po uproszczeniu:	4 595 021	10 038 310	96 100	75.02%
- końcówki	1 897 472			
- proste ścieżki (l. ścieżek)	315 075	518 514		
- małe wyspy	59 310			
Końcowo:	2 119 725	4 538 739	115 916	64.04%
SGA (10 iteracji)	1 423 561	7 749 584	59 528	
SGA (10 it., bez $\leftrightarrow$ i $\asymp$)	1 090 001	4 956 786	62,889	

Tablica 4.2.2: Rozmiary grafu nałożeń w kolejnych krokach heurystycznego upraszczania grafu: początkowo, czyli po wczytaniu (bez super-repetytywnych krawędzi), po uproszczeniu prostych ścieżek i po 1 cyklu usuwania końcówek, oraz rozmiary grafu po domyślnej procedurze `SGA` (10 iteracji i usuwanie bąbli), uruchomionej na zwykłem grafie i na grafie po usunięciu krawędzi prefiksowych i sufiksowych.

Po 1 iteracji oczyszczania grafu w grafie pozostaje ponad 2 tys. wierzchołków reprezentujących 64% liczby wszystkich odczytów we wszystkich próbkach. Kolejne iteracje oczyszczania pozwalają zmniejszyć graf jednak różnica nie jest już tak znaczna, więc dalsza asemblacja różnicowych kontigów wykonywana jest na grafie po jednej iteracji upraszczania.

4.3. Identyfikacja różnicowych kontigów w grafie nałożeń - heurystyki

Jak pokazane zostało we wcześniejszych częściach pracy, rozwiązanie problemu asemblacji różnicowych kontigów nawet w grafie nałożeń jest trudne. Dlatego w tym podrozdziale zaprezentowane i porównane są 3 różne heurystyczne podejścia do wyszukiwania różnicowych kontigów.

4.3.1. Opis heurystyk wyszukiwania różnicowych kontigów

Mając dany uproszczony graf nałożeń, celem jest rekonstrukcja w nim kontigów (reprezentowanych przez ścieżki w grafie), których zmiana krotności wynosi co najmniej FC. Poniżej zaproponowane są 3 proste „zachłanne" heurystyki wyszukiwania ścieżek, mające na celu znalezienie takich FC-różnicowych kontigów:

1. Maksymalizującą długość (ozn. `longest`): dopóki są niewykorzystane wierzchołki:

- wybieram najdłuższy (pod względem reprezentowanej sekwencji) wierzchołek jako początek ścieżki
- dopóki to możliwe rozszerzam taką ścieżkę z dowolnego jej końca o najdłuższy wierzchołek
- wyjściowe ścieżki muszą zostać odfiltrowane by uzyskać tylko te zmienione

2. Maksymalizująca długość, ale różnicowa (ozn. `longestfc`): dopóki są niewykorzystane wierzchołki:
   - wybieram najdłuższy (pod względem reprezentowanej sekwencji) wierzchołek o $fc \geq FC$
   - dopóki to możliwe i wynikowa ścieżka ma $fc \geq FC$ rozszerzam taką ścieżkę z dowolnego jej końca o najdłuższy wierzchołek
3. Maksymalizująca krotność zmiany (ozn. `bestfc`): dopóki są niewykorzystane wierzchołki:
   - wybieram wierzchołek o największym $fc \geq FC$
   - dopóki to możliwe i wynikowa ścieżka ma $fc \geq FC$ rozszerzam taką ścieżkę z dowolnego jej końca o wierzchołek, który najmniej zmniejsza fold-change.

Sekwencje otrzymane przy pomocy heurystyki `longest` wymagają późniejszego odfiltrowania tych o $fc < FC$.

Powyższe heurystyki porównane są w dalszej części rozdziału z różnicowymi sekwencjami uzyskanymi przez `SGA` uruchomionym na tym samym uproszczonym grafie. Do porównania wybrane są sekwencje zbudowane przez `SGA scaffold` ponieważ `SGA assemble` praktycznie nie używa struktury grafu do budowy ścieżek, więc wynikiem są właściwie tylko sekwencje wierzchołków. Kolejne porównanie dotyczy wybranej heurystyki z kontigami z `SGA` oraz z `MEGAHIT`, i wykorzystuje do tego m.in. mapowania z pomocą `kallisto`.

4.3.2. Wyniki asemblacji przy pomocy heurystyk

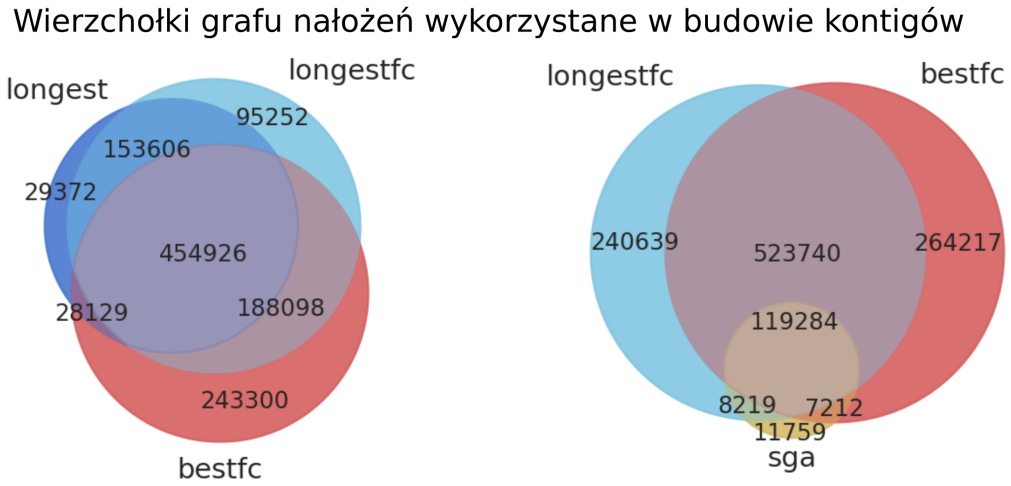
Heurystyki zaimplementowane w języku Python uruchomione zostały na oczyszczonym grafie nałożeń dla progu różnicowości $FC = 2$. Tabela 4.3.1 pokazuje porównanie m.in. liczby odczytów składających się na końcowe kontigi. Zarówno liczba jak i długości oraz zawartość odczytów różnią się pomiędzy metodami, chociaż nieznacznie. Najlepsze wyniki pod względem długości osiąga `bestfc`, a pod względem liczby odczytów `longestfc`.

W dalszej części przedstawione jest porównanie wierzchołków grafu wykorzystanych przez heurystyki, więc dla zwiększenia przejrzystości wyłączone z niej są kontigi otrzymane na podstawie pojedynczego wierzchołka, które powtarzały się we wszystkich tych zbiorach. Rysunek 4.3.1 pokazuje, że wierzchołki wykorzystane przez heurystyki w dużej części się pokrywają, a heurystyki używają większości wierzchołków użytych przez `SGA`, które w sumie używa znacznie mniejszej liczby wierzchołków.

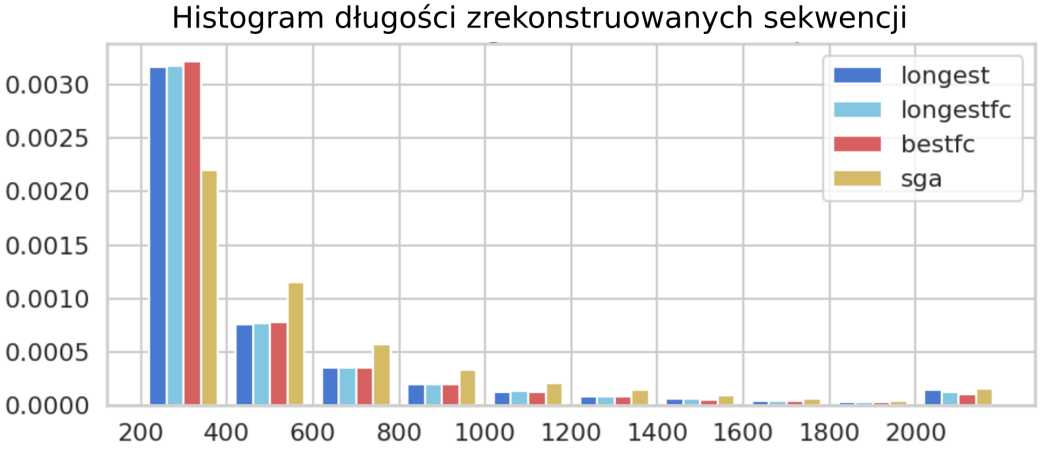
Długości kontigów uzyskanych różnymi heurystykami są podobne niezależnie od tego, czy długość kontigu była brana pod uwagę przy wydłużaniu sekwencji (patrz rysunek 4.3.2). Sekwencje z `SGA` są nieznacznie dłuższe, jednak te bardzo długie ($\geq$ 2000 bp) stanowią porównywalny odsetek. To znaczy, że niezależnie od tego, czy priorytetem jest maksymalizacja długości kontigu, czy krotność zmiany, wyniki pod względem długości są podobne.

Heurystyka	Liczba kontigów	sumaryczna długość (bp)	zliczenia	zliczenia (% odczytów)
longest ($fc \geq FC$)	182 973	85 289 444	42 317 834	23.3%
longest fc	213 507	99 725 607	49 295 380	27.2%
bestfc	220 888	100 319 295	46 086 786	25.4%
SGA	157 276	79 111 002	32 587 074	18.0%
SGA ($fc \geq FC$)	126 648	63 012 306	22 896 390	12.6%

Tablica 4.3.1: Porównanie kontigów otrzymanych przy pomocy różnych heurystyk



Rysunek 4.3.1: Diagramy Venna wierzchołków użytych do konstrukcji 2-różnicowych kontigów przez heurystyki oraz `SGA`.



Rysunek 4.3.2: Histogram długości 2-różnicowych kontigów uzyskanych różnymi metodami.

4.3.3. Porównanie kontigów `longestfc` oraz `MEGAHIT` i `SGA` poprzez mapowanie

Wykonane powyżej porównanie heurystyk nie jest możliwe do powtórzenia z sekwencjami uzyskanymi przez `MEGAHIT`, gdyż nie wykorzystuje on tego rodzaju grafu.

Ponieważ `MEGAHIT` buduje kontigi przy pomocy grafu *de Bruijna* nie udostępnia on też informacji, który odczyt jest zawarty w którym kontigu. Co więcej, odczyt może nie odpowiadać fragmentowi sekwencji żadnego kontigu (ale jego fragmenty mogły być użyte do utworzenia więcej niż jednego kontigu). Dlatego do oceny pokrycia kontigów wybrane zostało mapowanie odczytów, czyli zliczenia obliczone przy pomocy `kallisto`, (w trybie single-end, bo heurystyki nie korzystają z informacji o sparowaniu odczytów), oraz `bowtie2`. W tej części dla uproszczenia umówiona jest tylko jedna wybrana heurystyka – `longestfc`, gdyż pod tym względem pozostałe heurystyki wypadały dość podobnie.

Pokrycia wyliczone na podstawie mapowań do kontigów o długości ≥200 bp przedstawione są w tabeli 4.3.2. Pokrycie sekwencji z heurystyki `longestfc` powinno być mniejsze, ponieważ jej wyniki zawierają jedynie kontigi 2-różnicowe, jednak mimo to pokrycia są większe dla mapowania na sekwencje z `SGA` przy pomocy `bowtie2` i `minimap2`.

		`Kallisto` k=21			`bowtie`	`minimap`
	rozmiar [bp]	zmapowane (jednoznacznie)	konflikty	brak k-merów		
`longestfc`	99 725 607	46.2% (43.9%)	19.5%	34.3%	69.1%	74.5%
`SGA`	78 880 002	68.9% (65.5%)	12.9%	18.1%	45.2%	63.3%
`MEGAHIT`	230 704 864	89.1% (84.3%)	5.4%	5.6%	91.1%	92.6%

Tablica 4.3.2: Pokrycia wyliczone na podstawie mapowań do kontigów o długości ≥ 200 bp.

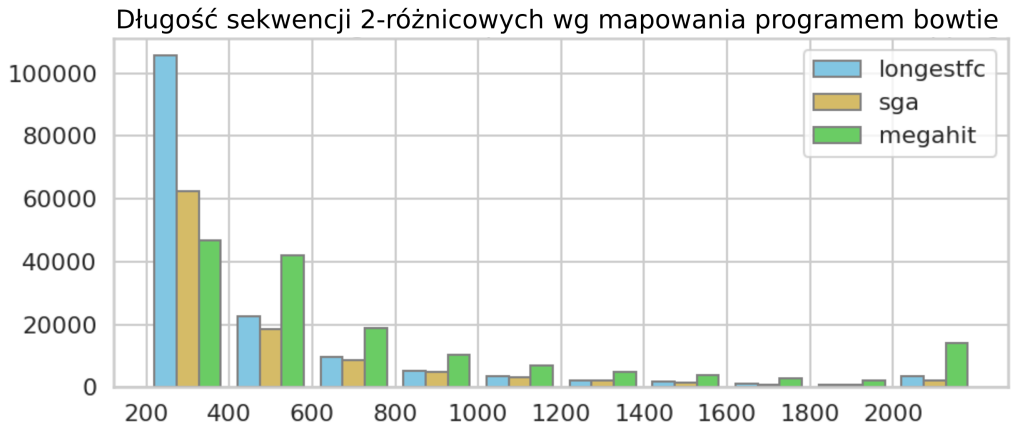
Po odfiltrowaniu sekwencji nieróżnicowych wg. każdej z 3 metod mapowania największą sumaryczną długość istotnych sekwencji otrzymujemy jednak przy użyciu programu `MEGAHIT` (patrz tabela 4.3.3. Liczba kontigów dla `MEGAHIT` oraz heurystyki `longestfc` jest podobna, jednak ich długość jest większa w przypadku `MEGAHIT` (patrz rysunek 4.3.3), tak samo jak procent odczytów zmapowanych. Dla `SGA` i `longestfc` widać duże zmiany w zliczeniach odczytów w zależności od użytej metody mapowania, przy czym przy użyciu `kallisto` mapuje się zdecydowanie mniej odczytów, co przeczy intuicji, że `kallisto` najłatwiej zmapować najwięcej odczytów, choć większa szansa, że przypisanie jest błędne.

Mimo, że dla sekwencji z heurystyki `longestfc` odczyty mapowane są już na kontigi, które powinny być 2-różnicowe, ok. $\frac{1}{4}$ z nich nie spełnia warunku 2-różnicowości podczas filtrowania na podstawie zmapowanych odczytów. Łatwo o taką sytuację dla kontigów o krotności zmiany bliskiej progu odcięcia. Skoro niektóre kontigi odpadają z tego powodu, zapewne istnieje też wiele kontigów, które mogłyby być 2-różnicowe, gdyby nie zostały wcześniej odfiltrowane. Dlatego warto przy budowie kontigów użyć mniejszego progu odcięcia niż docelowy.

Porównanie sekwencji stworzonych metodami korzystającymi z innych podejść, a nawet z różnych grafów, jest trudniejsze niż porównanie metod działających na tym samym grafie, gdyż nie można wykorzystać informacji o użytych wierzchołkach. Dokładny pomiar podobieństwa dwóch zbiorów sekwencji musiałby uwzględnić w inny sposób identyczność ich fragmentów, zawieranie itd. Byłby zatem kosztowny obliczeniowo (przez uliniowienia par sekwencji) oraz wymagałby ustalenia pewnych parametrów. Jednak porównanie występowania identycznych sekwencji również pozwala wyłapać podobieństwa między różnymi podejściami. Różnicowe sekwencje stworzone przez różne heurystyki często powtarzają się pomiędzy podejściami, ale całkowicie

Kallisto k=21				
METODA	liczba kontigów	sumaryczna długość	zliczenia	zliczenia (% odczytów)
longestfc	156 385	73 638 169	59 820 097	33.02%
SGA	102 231	51 937 925	25 667 934	14.17%
MEGAHIT	148 475	139 477 791	102 960 602	56.83%
bowtie				
METODA	liczba kontigów	sumaryczna długość	zliczenia	zliczenia (% odczytów)
longestfc	156 217	73 974 373	86 792 736	47.91%
SGA	104 549	52 976 914	49 683 447	27.42%
MEGAHIT	152 794	142 277 349	105 721 187	58.36%
minimap				
METODA	liczba kontigów	sumaryczna długość	zliczenia	zliczenia (% odczytów)
longestfc	152 267	72 773 928	93 443 350	51.58%
SGA	101 778	51 892 087	68 950 876	38.06%
MEGAHIT	151 208	141 341 748	107 177 460	59.16%

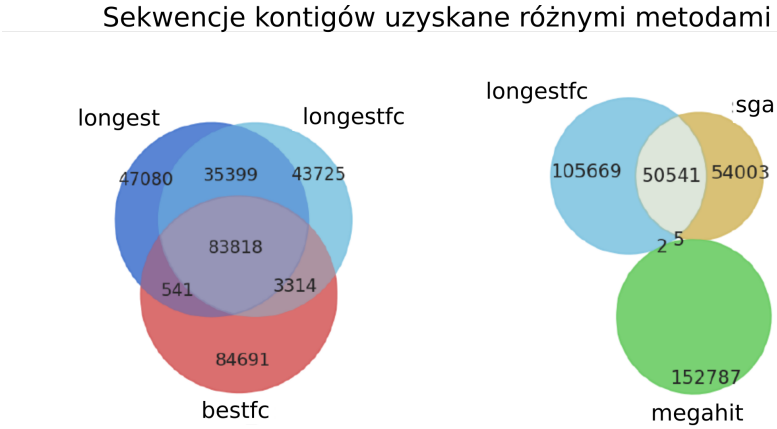
Tablica 4.3.3: Liczba i pokrycie kontigów o $FC \geq 2$ na podstawie mapowania `kallisto`, `bowtie2` i `minimap2`.



Rysunek 4.3.3: Histogram długości 2-różnicowych (wg. `kallisto`) kontigów. `MEGAHIT` zwraca dużo długich sekwencji.

różnią się od sekwencji zwróconych przez `SGA` i `MEGAHIT` (patrz rysunek 4.3.4). Biorąc pod uwagę zupełnie inny sposób budowy grafu oraz usuwania błędów spodziewane jest to, że sekwencje będą się znacznie różnić, jednak praktycznie całkowity brak sekwencji identycznych pomiędzy `MEGAHIT` i sekwencjami zasemblowanymi z użyciem grafu nałożeń jest nieco zaskakujący.

Następna część pracy podejmuje próbę wyjaśnienia powodów, dla których sekwencje z `MEGAHIT` są dłuższe i mapuje się na nie więcej odczytów, a także odpowiedzi, czy to oznacza, że ten zbiór sekwencji jest lepszy – tzn. czy sekwencje są poprawne i zbiór jest pełniejszy.



Rysunek 4.3.4: Współwystępowanie sekwencji 2-różnicowych (wg. `kallisto`) w zbiorach sekwencji uzyskanych różnymi metodami.

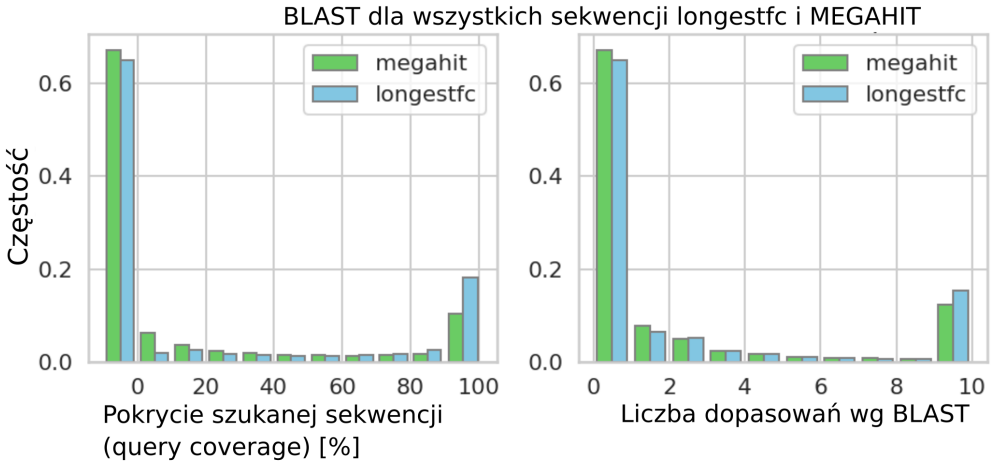
4.3.4. Wiarygodność kontigów z MEGAHIT

Bardzo trudno jest ustalić prawdziwość zidentyfikowanego w rzeczywistych danych kontigu, czyli realną obecność cząsteczki o danej sekwencji (z dokładnością do ustalonej liczby błędów), oraz pełność jej sekwencji (a nie tylko fragmentaryczne odczytanie rzeczywistej cząsteczki). Wymagałoby to być może użycia innej techniki sekwencjonowania dla tej samej próbki. Jednak na wiarygodność uzyskanych kontigów mogłoby wskazać znalezienie podobnej sekwencji w bazie referencyjnej. Nie można wykluczyć w ten sposób poprawności sekwencji kontigu, bo transkrypt może nie być jeszcze obecny w bazie danych, ale rozsądne jest oczekiwać, że pewna część kontigów będzie odpowiadała znanym sekwencjom biologicznym. Do sprawdzenia tego wykorzystano program `BLAST` [Altschul et al., 1990b], który przeszukuje znaczną bazę sekwencji referencyjnych (w tym wypadku nukleotydowych – baza `nt`) zwracając nawet dosyć odległe dopasowania, również jeśli tylko fragment zapytania jest podobny do fragmentu sekwencji z bazy referencyjnej.

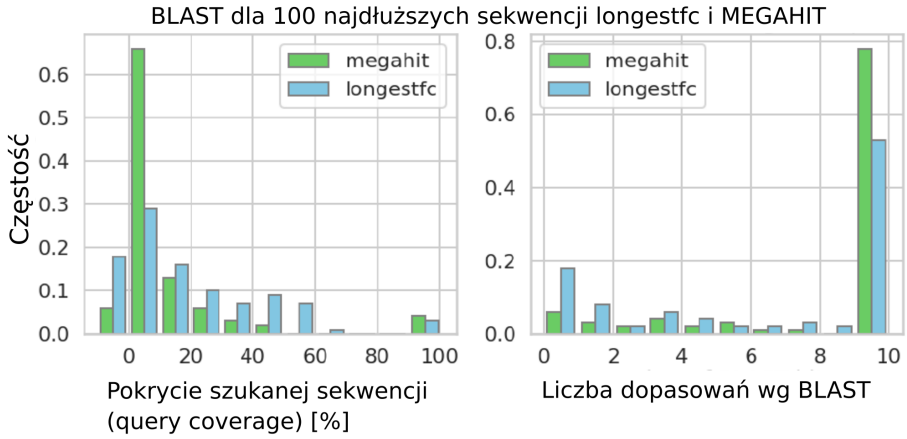
Większość z sekwencji asemblowanych przez `MEGAHIT` jak i przez heurystykę `longestfc` nie ma analogów w bazie `BLAST` (patrz rysunek 4.3.5, który pokazuje częstość znajdowania dopasowania fragmentu całego kontigu o tej długości spośród 1000 losowych kontigów zasemblowanych przez `MEGAHIT` i przez heurystykę `longestfc`, wyłączając identyczne sekwencje). W obu zbiorach sekwencji ponad 60% z nich nie ma żadnego dopasowania (lewy słupek). Kilka % spośród kontigów ma znaczące fragmentaryczne podobieństwo do znanych sekwencji, ale dla 10-20% znalezione dopasowanie odpowiada całej długości kontigu. Nieznacznie gorzej niż heurystyka `longestfc` wypada pod tym względem `MEGAHIT`.

Sekwencji zawartych w bazie w całości jest mniej wśród sekwencji z `MEGAHIT`, a więcej sekwencji ma tylko częściowe dopasowanie wg. `BLAST`, co sugeruje, że te kontigi mogą być hybrydowe — „sklejone” z kilku poprawnych sekwencji. Widać to szczególnie wśród 100 najdłuższych sekwencji (patrz rysunek 4.3.6), dla których najlepsze dopasowanie w bazie znanych sekwencji nukleotydowych obejmuje tylko 0-60% długości tych sekwencji, chociaż dla sekwencji z heurystyki `longestfc` pokrycia te są trochę większe.

Aby odpowiedzieć na pytanie, czy 2-różnicowe sekwencje z `MEGAHIT` są przedłużonymi lub sklejonymi z fragmentów sekwencjami uzyskanymi heurystyką `longestfc`



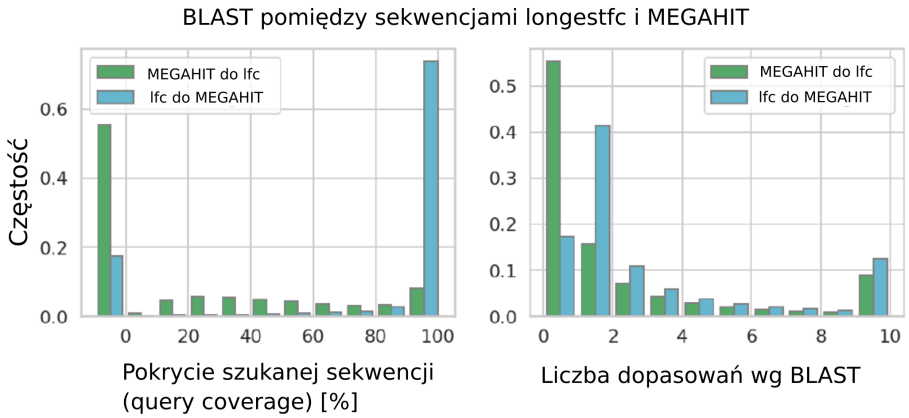
Rysunek 4.3.5: Częstość znalezienia dopasowania o takim pokryciu przy pomocy programu `BLAST` dla losowych 1000 kontigów uzyskanych przez `MEGAHIT` oraz heurystykę `longestfc`.



Rysunek 4.3.6: Częstość znalezienia dopasowania o takim pokryciu przy pomocy programu `BLAST` dla 100 najdłuższych kontigów uzyskanych przez `MEGAHIT` oraz 2-różnicowych sekwencji uzyskanych przez heurystykę `longestfc`.

sekwencje jednego ze zbiorów zostały dopasowane używając programu `BLAST` wykorzystującego drugi ze zbiorów jako bazę referencyjną (patrz rysunek 4.3.7). `BLAST` sekwencji z `MEGAHIT` na sekwencje z heurystyki `longestfc` (i vice versa) pokazuje że prawie wszystkie sekwencje z `longestfc` są zawarte w wynikach `MEGAHIT`, natomiast większość z `MEGAHIT` nie ma odpowiednika w `longestfc`, a pozostałe z nich są jedynie częściowo zawarte w `longestfc`.

Podsumowując, mimo że wyniki mapowalności i długość uzyskanych sekwencji dają przewagę `MEGAHIT`, to mapowanie przy użyciu `BLAST` pokazuje, że część z sekwencji może być hybrydowa, tzn. może wynikać z błędnego scalenia rzeczywistych transkryptów o wspólnym k-merze, a ich odsetek jest większy dla `MEGAHIT` niż dla heurystyki `longestfc`. W takim przypadku może to wpływać na analizę różnicowej



Rysunek 4.3.7: `BLAST` pomiędzy 2-różnicowymi sekwencjami z `MEGAHIT` a sekwencjami uzyskanymi heurystyką `longestfc`.

ekspresji, w szczególności może być niemożliwe odkrycie niektórych różnicowych kontigów. W zależności od oczekiwań, ale też planów użycia w dalszej analizie innych narzędzi, np. do rozpoznawania genów, wybór użycia `MEGAHIT` zamiast heurystyki `longestfc` może być uzasadniony – szerszą reprezentację sekwencji w próbce da `MEGAHIT`, jednak wiarygodność sekwencji jest wyższa dla heurystyki `longestfc`.

4.3.5. Implementacja

Implementacja upraszczania grafu nałożeń oraz heurystyk wyszukiwania różnicowych kontigów, nazwana `diffcog`, została wykonana w języku `Python` z wykorzystaniem implementacji grafu skierowanego z pakietu `networkx`. Jest ona dostępna w repozytorium: `https://github.com/juliahi/diffcog`, razem z kodem umożliwiającym uruchomienie `SGA` dla dwóch zbiorów próbek w sposób umożliwiający śledzenie liczby odczytów.

W tabeli 4.3.4 przedstawione są wyniki działania `diffcog` oraz konkurencyjnych rozwiązań `MEGAHIT` oraz `SGA` na testowym zbiorze danych (dostępnym w ww. repozytorium) zawierającym 296 226 odczyty spośród analizowanych w tej pracy danych eksperymentalnych. Czas przedstawiony w tabeli dotyczy asemblacji sekwencji, a aby uzyskać sekwencje 2-różnicowe dla `MEGAHIT` oraz `SGA` należałoby jeszcze wykonać mapowanie na kontigi i zliczenie odczytów.

Program	real	user	sys
`diffcog`	2m2.146s	2m59.416s	0m3.536s
`MEGAHIT`	0m44.555s	1m11.320s	0m2.016s
`SGA`	0m38.372s	0m59.008s	0m1.004s

Tablica 4.3.4: Czas działania programów na testowym zbiorze danych dla maszyny o 600GB RAM, z wykorzystaniem 2 rdzeni.

ROZDZIAŁ 5

Podsumowanie

Mikroorganizmy, w szczególności bakterie jelitowe, mają ogromny wpływ na życie człowieka. Poznanie ich aktywności, w tym produkowanych przez nie enzymów i możliwych do wyprodukowania przy ich pomocy metabolitów jest ważnym wyzwaniem współczesnej biologii. Przydatne w tym są badania metatranskryptomu przy użyciu wysokoprzepustowego sekwencjonowania nowej generacji, które wymagają do analizy wyników użycia metod obliczeniowych. Dostępne aktualnie metody nie są jednak dostosowane do tak trudnych zbiorów danych, jakimi są krótkie odczyty z sekwencjonowania metatranskryptomu.

W rozdziale 3 przedstawiłam analizę odczytów pochodzących z sekwencjonowania mRNA z odchodów myszy na diecie standardowej oraz wysokotłuszczowej. Zarówno podejścia polegające na mapowaniu odczytów na zbiór genomów referencyjnych, jak i asemblacja transkryptomu nie dały zadowalającego rezultatu. Wykorzystanie zmniejszonych zbiorów sekwencji referencyjnych pozwalało zmniejszyć i tak duże wymagania sprzętowe (konieczny do obliczeń RAM), ale nie poprawiało mapowalności wystarczająco. Wzrost liczby zsekwencjonowanych genomów nie poprawi raczej możliwości identyfikacji w ten sposób różnicowych transkryptów, ale zwiększy na pewno wyzwania obliczeniowe. Czekanie na powiększenie bazy nie jest zatem rozwiązaniem.

Mapowanie na bazę sekwencji genomów bakteryjnych pozwoliło przypisać jedynie ok. 27% z odczytów, a tylko 21% jednoznacznie, czyli ponad 70% odczytów nie byłoby użyte w dalszej analizie, przy czym jest to wynik pseudomapowania przy użyciu `kallisto`. Pokazuje to, że zgodnie z wcześniejszą publikacją autorów tego narzędzia, mimo że zostało ono zaprojektowane do obliczania częstości transkryptów podlegających alternatywnemu splicingowi, to możliwe jest jego użycie do danych metagenomicznych oraz metatranskryptomicznych.

Spośród metod asemblacji pozytywnie wybija się `MEGAHIT`, metoda korzystająca z grafu *de Bruijna* i zaprojektowana do asemblacji metagenomów. Mapowało się na jej kontigi niemal 84% odczytów, jednak część z tych kontigów wydaje się być zbyt długa aby reprezentować pojedyncze geny bakteryjne. To, w niektórych zastosowaniach, szczególnie w badaniu ilościowym transkrypcji, może być duża wada tej metody. Jest to spodziewana obserwacja dla metody wykorzystującej graf *de Bruijna*, gdyż jego zastosowanie ułatwia stworzenie sekwencji hybrydowych. Pod względem minimalizacji szansy na błędną asemblację tworzącą sekwencje hybrydowe bezpieczniejsze są zatem metody wykorzystujące graf nałożeń odczytów.

Do uzyskanych przy pomocy grafu *de Bruijna* sekwencji trzeba zatem podejść krytycznie i zastosować kolejne kroki aby uzyskać bardziej wiarygodne sekwencje pojedynczych genów. Opisany w rozdziale 3.5, zaprojektowany z dr Iloną Grabowicz framework przewiduje w obrębie kontigów geny, następnie anotuje je funkcjonalnie.

Po połączeniu zliczeń dla ortologów udało nam się zidentyfikować niewielką liczbę różnicowych genów.

Zdefiniowanie problemu asemblacji różnicowej ekspresji jest nietrywialne w przypadku ekspresji genów pojedynczego organizmu, a dla populacji bakterii wymaga rozważenia jeszcze innych czynników. W rozdziale 4 zaproponowałam definicję tego problemu i pokazałam, że rozwiązanie go przy użyciu grafu *de Bruijna* jest bardzo trudne, przez nieaddytywność zliczeń odczytów w ścieżce. W grafie nałożeń ten problem nie występuje, jest natomiast potencjalnie wykładnicza liczba rozwiązań optymalnych. Dlatego zaproponowałam trzy warianty zachłannego algorytmu heurystycznego do znajdowania różnicowych transkryptów bezpośrednio w grafie nałożeń. Uzyskane przy ich pomocy sekwencje o krotności zmiany co najmniej 2 wykorzystują aż ok. 25% odczytów i uzyskane sekwencje różnią się między sobą.

Porównanie kontigów uzyskanych przy użyciu heurystyk z sekwencjami otrzymanymi przy pomocy `MEGAHIT` i `SGA` nie daje jednoznacznej odpowiedzi, która metoda jest lepsza. Wyniki mapowalności i długość uzyskanych sekwencji są lepsze dla `MEGAHIT`, chociaż mapowanie przy użyciu `BLAST` sugeruje, że większy odsetek z nich nie jest sekwencjami pochodzącymi z pojedynczego gatunku, zatem prawdopodobnie są to sekwencje wynikające z błędnego połączenia dwóch lub więcej rzeczywistych sekwencji. Dodatkowo, porównanie sekwencji przy pomocy mapowania wykorzystującego k-mery (`kallisto`) może być obciążone w kierunku metod asemblacji wykorzystujących k-mery.

Wobec braku znajomości złotego standardu dla tych danych eksperymentalnych trudno rozsądzić o wyższości którejś z metod. Ich zastosowanie powinno zatem zależeć od oczekiwań w stosunku do wyniku – większą pełność jeśli chodzi o pokrycie obecnych w próbce sekwencji da `MEGAHIT`, i jest to też odpowiedni wybór przy dalszej obróbce sekwencji, np. metodami identyfikacji genów przez wyszukanie fragmentów sekwencji w bazie znanych sekwencji białkowych. W sytuacji jednak gdy istotna jest identyfikacja zbioru różnicowych transkryptów, co do których prawdziwości i faktycznej zmiany ekspresji chcemy być pewniejsi, to lepszym wyborem będzie raczej metoda heurystyczna, nawet kosztem pominięcia części ze spełniających te warunki transkryptów znajdujących się w próbce.

Podsumowując, pokazałam, że popularne metody badania eksperymentalnych danych metagenomicznych i transkryptomicznych (z użyciem znanych programów do mapowania i asemblacji) nie są wystarczająco dostosowane do danych metatranskryptomicznych i w tym przypadku dały niesatysfakcjonujące wyniki. Mimo, że procedura składająca się z kilku kroków używających dostępnych narzędzi (asemblacja, mapowanie, wybór genów i anotacja funkcjonalna, analiza różnicowej ekspresji) jest możliwa do wykorzystania w danych metatranskryptomicznych, zawiera ona wiele redundantnych kroków, a wyniki nie są spektakularne. Zaproponowałam zatem sformułowanie problemu asemblacji różnicowych kontigów i pokazałam, że do jego rozwiązania dużo lepszym narzędziem niż graf *de Bruijna* jest graf nałożeń. Zaproponowane heurystyki wyszukiwania różnicowych kontigów w grafie nałożeń pozwoliły zaproponować zbiory sekwencji o pewnej krotności zmiany, jednak wciąż pozostaje wiele obszarów, w których można zaproponować nowe rozwiązania tego problemu, np. dla innych definicji „różnicowości" lub z wykorzystaniem innych algorytmów heurystycznych. Moja praca wskazuje jednak jasno, które kierunki dalszego rozwoju są najbardziej obiecujące, tzn. metody wykorzystujące graf nałożeń.

Wkład współpracowników oraz dostępność wyników

W rozdziałach 3 oraz 4 wykorzystałam dane eksperymentalne pozyskane we współpracy dr Ilony Grabowicz z grupą prof. Mary Dierssen z Center for Genomic Regulation w Barcelonie. Eksperymentalna część pozyskania próbek, tzn. hodowla myszy i pobranie próbek zostały wykonane w Barcelonie przez dr Martę Fructuoso, a sekwencjonowanie w Genomed S.A.

Projekt eksperymentu i wstępna analiza jakości sekwencjonowania z wykorzystaniem `FastQC` i analizy zanieczyszczenia rybosomalnym RNA (opisane w 3.1) zostały wykonane przez dr Ilonę Grabowicz i stanowią jedną z części jej rozprawy doktorskiej dotyczącej zmian ekspresji genów w mózgu i zmian w mikrobiomie myszy pod wpływem diet wywołujących zmiany w zachowaniu. W jej rozprawie została zawarta również część moich wyników, opisanych poniżej.

W 3.2 w celu analizy ww. danych metatranskryptomicznych wykonałam przegląd narzędzi do mapowania – zainstalowałam (niektóre z nich, np. `kallisto`, uruchomiłam i dobrałam parametry, oraz zebrałam różne zbiory genomów referencyjnych, tj. pobrałam genomy z bazy `RefSeq`, `HMP`, oraz skryptami i przy pomocy narzędzia `Kraken` dokonałam selekcji genomów aby utworzyć zbiory `miBC` i `Kraken18`. Do oceny mapowalności dla różnych zbiorów genomów oraz parametrów `kallisto` użyłam własnego skryptu w języku `Python` oraz samodzielnie zmodyfikowanej wersji programu `kallisto`. Podobnie w 3.3 wybór, instalację i uruchomienie oraz porównanie wyników wszystkich programów do asemblacji wykonałam ręcznie lub przy użyciu własnych skryptów. Analizę różnicowej ekspresji w 3.4 wykonałam własnym skryptem w języku `R` uruchamiającym z odpowiednimi parametrami dostępne pakiety do analizy różnicowej ekspresji genów, na danych odfiltrowanych skryptem w języku `Python`.

Na podstawie wybranych przeze mnie kontigów złożonych przy użyciu `MEGAHIT` dr Ilona Grabowicz zaproponowała i przeprowadziła identyfikację genów przy pomocy `MetaGeneMark` oraz analizę funkcjonalną przy pomocy `eggNOGmapper` oraz sumowanie odczytów opisane w podrozdziale 3.5. Dla uzyskanych zliczeń obliczyłam różnicową ekspresję przy pomocy `DESeq2` oraz przygotowałam listy zidentyfikowanych genów, które wspólnie z dr Grabowicz przypisałyśmy do kategorii korzystając z literatury oraz wspierając się bazą `Uniprot`. Z racji jej większych kompetencji w zakresie nauk biologicznych interpretację biologiczną (tj. przegląd literatury oraz opisy roli danych genów w szlakach metabolicznych w komórkach bakteryjnych), wykonała dr Ilona Grabowicz.

Złożoność wspólnie wypracowanej w rozdziale 3 procedury zaowocowała postawieniem w rozmowach z promotorem problemu asemblacji różnicowych kontigów, który następnie sformalizowałam. Przeprowadziłam później rozważania teoretyczne dotyczące możliwości użycia różnych rodzajów grafów do asemblacji opisane w ro-

dziale 4. Po wybraniu grafu nałożeń jako lepszego rozwiązania oraz przeglądzie możliwości wykorzystania istniejących narzędzi w celu budowy grafu nałożeń i zmodyfikowania ich do asemblacji różnicowych kontigów zaprojektowałam oraz zaimplementowałam procedurę użycia w celu budowy grafu nałożeń podprogramów asemblera `SGA`. Następnie przy pomocy napisanego przeze mnie zbioru skryptów wykorzystujących `SGA` uzyskałam graf nałożeń dla eksperymentalnych danych. W języku `Python` napisałam skrypty wykonujące modyfikacje grafu nałożeń w sposób analogiczny do tych wykonywanych przez `SGA` ale zachowujące zliczenia odczytów budujących wierzchołki i przy jego pomocy zanalizowałam rozmiar grafu podczas różnych etapów upraszczania. Na tak przygotowanym grafie wykonałam asemblację różnicowych kontigów przy pomocy własnej implementacji algorytmów heurystycznych. Przy decyzjach dotyczących heurystyk oraz w części teoretycznej rozważań o grafach służących do asemblacji korzystałam ze wsparcia merytorycznego promotora tej rozprawy w postaci dyskusji i sugestii co do kierunku badań. Samodzielnie wykonałam dobór parametrów oczyszczania grafu, obliczenia z użyciem heurystyk, oraz analizę uzyskanych sekwencji z wykorzystaniem m.in. programu `BLAST`.

Analiza opisana w 3.5 oraz część wyników z 3.2 3.3 zostały opisane w nieopublikowanym jeszcze artykule: *Metatranscriptomic changes upon high-fat diet in a Down syndrome mouse model* autorstwa: Ilona E. Grabowicz, Julia Herman-Iżycka, Marta Fructuoso, Mara Dierssen, Bartek Wilczyński. Preprint pracy dostępny jest po adresem `https://doi.org/10.1101/2021.12.16.472765`, a dotyczący tej części pracy kod jest opublikowany w repozytorium: `https://github.com/juliahi/metatranscriptomics_in_DS`

Kod w `C++` programu `kallisto` z moimi niedużymi modyfikacjami dostępny jest w repozytorium: `https://github.com/juliahi/kallisto`.

Implementacja upraszczania grafu nałożeń oraz heurystyk wyszukiwania różnicowych kontigów opisana w 4.2, razem z potrzebnym do tego kodem umożliwiającym uruchomienie `SGA` dla dwóch zbiorów próbek w sposób umożliwiający śledzenie liczby odczytów, jest dostępna w repozytorium: `https://github.com/juliahi/diffcog`. Planowany jest krótki artykuł (*application note*) na temat tej metody.

Bibliografia

[lib,] Dna sequencing. *University of Wisconsin-Milwaukee*. `https://bio.libretexts.org/@go/page/16508`.

[Aird et al., 2011] Aird, D., Ross, M. G., Chen, W.-S., Danielsson, M., Fennell, T., Russ, C., Jaffe, D. B., Nusbaum, C., and Gnirke, A. (2011). Analyzing and minimizing pcr amplification bias in illumina sequencing libraries. *Genome biology*, 12(2):R18.

[Altschul et al., 1990a] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990a). Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410.

[Altschul et al., 1990b] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990b). Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410.

[Ambardar et al., 2016] Ambardar, S., Gupta, R., Trakroo, D., Lal, R., and Vakhlu, J. (2016). High throughput sequencing: an overview of sequencing chemistry. *Indian journal of microbiology*, 56(4):394–404.

[Anders and Huber, 2012] Anders, S. and Huber, W. (2012). Differential expression of rna-seq data at the gene level–the deseq package. *Heidelberg, Germany: European Molecular Biology Laboratory (EMBL)*, 10:f1000research.

[Andrews et al., 2010] Andrews, S. et al. (2010). Fastqc: a quality control tool for high throughput sequence data.

[Belfort et al., 1995] Belfort, M., Reaban, M. E., Coetzee, T., and Dalgaard, J. Z. (1995). Prokaryotic introns and inteins: a panoply of form and function. *Journal of bacteriology*, 177(14):3897.

[Benjamini and Hochberg, 1995] Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300.

[Blumenthal, 2004] Blumenthal, T. (2004). Operons in eukaryotes. *Briefings in Functional Genomics*, 3(3):199–211.

[Bray et al., 2016] Bray, N. L., Pimentel, H., Melsted, P., and Pachter, L. (2016). Near-optimal probabilistic rna-seq quantification. *Nature biotechnology*, 34(5):525.

[Burrows and Wheeler, 1994] Burrows, M. and Wheeler, D. J. (1994). A block-sorting lossless data compression algorithm.

[Clark et al., 2018a] Clark, M. A., Choi, J., and Douglas, M. (2018a). Dna structure and sequencing. *Biology 2e*. https://openstax.org/books/biology-2e/pages/14-2-dna-structure-and-sequencing.

[Clark et al., 2018b] Clark, M. A., Choi, J., and Douglas, M. (2018b). Nucleic acids. *Biology 2e*. https://openstax.org/books/biology-2e/pages/3-5-nucleic-acids.

[Clark et al., 2018c] Clark, M. A., Choi, J., and Douglas, M. (2018c). Organizing life on earth. *Biology 2e*. https://openstax.org/books/biology-2e/pages/20-1-organizing-life-on-earth.

[Clark et al., 2018d] Clark, M. A., Choi, J., and Douglas, M. (2018d). Prokaryotic gene regulation. *Biology 2e*. https://openstax.org/books/biology-2e/pages/16-2-prokaryotic-gene-regulation.

[Clark et al., 2018e] Clark, M. A., Choi, J., and Douglas, M. (2018e). Rna processing in eukaryotes. *Biology 2e*. https://openstax.org/books/biology-2e/pages/15-4-rna-processing-in-eukaryotes.

[Csonka, 1989] Csonka, L. N. (1989). Physiological and genetic responses of bacteria to osmotic stress. *Microbiology and Molecular Biology Reviews*, 53(1):121–147.

[David et al., 2014] David, L. A., Maurice, C. F., Carmody, R. N., Gootenberg, D. B., Button, J. E., Wolfe, B. E., Ling, A. V., Devlin, A. S., Varma, Y., Fischbach, M. A., et al. (2014). Diet rapidly and reproducibly alters the human gut microbiome. *Nature*, 505(7484):559–563.

[De Bruijn and Erdös, 1948] De Bruijn, N. G. and Erdös, P. (1948). On a combinatorial problem. *Proceedings of the Section of Sciences of the Koninklijke Nederlandse Akademie van Wetenschappen te Amsterdam*, 51(10):1277–1279.

[Deschasaux et al., 2018] Deschasaux, M., Bouter, K. E., Prodan, A., Levin, E., Groen, A. K., Herrema, H., Tremaroli, V., Bakker, G. J., Attaye, I., Pinto-Sietsma, S.-J., et al. (2018). Depicting the composition of gut microbiota in a population with varied ethnic origins but shared geography. *Nature medicine*, 24(10):1526–1531.

[Dohm et al., 2008] Dohm, J. C., Lottaz, C., Borodina, T., and Himmelbauer, H. (2008). Substantial biases in ultra-short read data sets from high-throughput dna sequencing. *Nucleic acids research*, 36(16):e105.

[Eckburg et al., 2005] Eckburg, P. B., Bik, E. M., Bernstein, C. N., Purdom, E., Dethlefsen, L., Sargent, M., Gill, S. R., Nelson, K. E., and Relman, D. A. (2005). Diversity of the human intestinal microbial flora. *science*, 308(5728):1635–1638.

[Euzéby, 1997] Euzéby, J. P. (1997). List of bacterial names with standing in nomenclature: a folder available on the internet. *International Journal of Systematic and Evolutionary Microbiology*, 47(2):590–592.

[Ferragina and Manzini, 2000] Ferragina, P. and Manzini, G. (2000). Opportunistic data structures with applications. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 390–398. IEEE.

[Foster and Neufeld, 2013] Foster, J. A. and Neufeld, K.-A. M. (2013). Gut–brain axis: how the microbiome influences anxiety and depression. *Trends in neurosciences*, 36(5):305–312.

[Fructuoso et al., 2018] Fructuoso, M., Rachdi, L., Philippe, E., Denis, R., Magnan, C., Le Stunff, H., Janel, N., and Dierssen, M. (2018). Increased levels of inflammatory plasma markers and obesity risk in a mouse model of down syndrome. *Free Radical Biology and Medicine*, 114:122–130.

[Galperin et al., 2014] Galperin, M. Y., Makarova, K. S., Wolf, Y. I., and Koonin, E. V. (2014). Expanded microbial genome coverage and improved protein family annotation in the COG database. *Nucleic Acids Research*, 43(D1):D261–D269.

[Gerstein et al., 2007] Gerstein, M. B., Bruce, C., Rozowsky, J. S., Zheng, D., Du, J., Korbel, J. O., Emanuelsson, O., Zhang, Z. D., Weissman, S., and Snyder, M. (2007). What is a gene, post-encode? history and updated definition. *Genome research*, 17(6):669–681.

[Goodrich et al., 2014] Goodrich, J. K., Waters, J. L., Poole, A. C., Sutter, J. L., Koren, O., Blekhman, R., Beaumont, M., Van Treuren, W., Knight, R., Bell, J. T., et al. (2014). Human genetics shape the gut microbiome. *Cell*, 159(4):789–799.

[Gründling and Schneewind, 2007] Gründling, A. and Schneewind, O. (2007). Synthesis of glycerol phosphate lipoteichoic acid in staphylococcus aureus. *Proceedings of the National Academy of Sciences*, 104(20):8478–8483.

[Güell et al., 2011] Güell, M., Yus, E., Lluch-Senar, M., and Serrano, L. (2011). Bacterial transcriptomics: what is beyond the rna horiz-ome? *Nature Reviews Microbiology*, 9(9):658.

[Hawinkel et al., 2019] Hawinkel, S., Mattiello, F., Bijnens, L., and Thas, O. (2019). A broken promise: microbiome differential abundance methods do not control the false discovery rate. *Briefings in bioinformatics*, 20(1):210–221.

[He et al., 2018] He, Y., Wu, W., Zheng, H.-M., Li, P., McDonald, D., Sheng, H.-F., Chen, M.-X., Chen, Z.-H., Ji, G.-Y., Mujagond, P., et al. (2018). Regional variation limits applications of healthy gut microbiome reference ranges and disease models. *Nature medicine*, 24(10):1532–1535.

[Heather and Chain, 2016] Heather, J. M. and Chain, B. (2016). The sequence of sequencers: The history of sequencing dna. *Genomics*, 107(1):1–8.

[Hill, 1997] Hill, M. (1997). Intestinal flora and endogenous vitamin synthesis. *European journal of cancer prevention: the official journal of the European Cancer Prevention Organisation (ECP)*, 6:S43–5.

[Huerta-Cepas et al., 2017] Huerta-Cepas, J., Forslund, K., Coelho, L. P., Szklarczyk, D., Jensen, L. J., Von Mering, C., and Bork, P. (2017). Fast genome-wide functional annotation through orthology assignment by eggnog-mapper. *Molecular biology and evolution*, 34(8):2115–2122.

[Huttenhower et al., 2012] Huttenhower, C., Gevers, D., Knight, R., Abubucker, S., Badger, J. H., Chinwalla, A. T., Creasy, H. H., Earl, A. M., FitzGerald, M. G.,

Fulton, R. S., et al. (2012). Structure, function and diversity of the healthy human microbiome. *nature*, 486(7402):207.

[Hyyryläinen et al., 2001] Hyyryläinen, H.-L., Bolhuis, A., Darmon, E., Muukkonen, L., Koski, P., Vitikainen, M., Sarvas, M., Prágai, Z., Bron, S., Van Dijl, J. M., et al. (2001). A novel two-component regulatory system in bacillus subtilis for the survival of severe secretion stress. *Molecular microbiology*, 41(5):1159–1172.

[Jacob and Monod, 1961] Jacob, F. and Monod, J. (1961). On the regulation of gene activity. In *Cold Spring Harbor Symposia on Quantitative Biology*, volume 26, pages 193–211. Cold Spring Harbor Laboratory Press.

[Kececioglu, 1991] Kececioglu, J. (1991). Exact and approximation algorithms for dna sequence reconstruction /.

[Kono and Arakawa, 2019] Kono, N. and Arakawa, K. (2019). Nanopore sequencing: review of potential applications in functional genomics. *Development, growth & differentiation*, 61(5):316–326.

[Koonin and Wolf, 2008] Koonin, E. V. and Wolf, Y. I. (2008). Genomics of bacteria and archaea: the emerging dynamic view of the prokaryotic world. *Nucleic Acids Research*, 36(21):6688–6719.

[Kopylova et al., 2012] Kopylova, E., Noé, L., and Touzet, H. (2012). Sortmerna: fast and accurate filtering of ribosomal rnas in metatranscriptomic data. *Bioinformatics*, 28(24):3211–3217.

[Lagkouvardos et al., 2016] Lagkouvardos, I., Pukall, R., Abt, B., Foesel, B. U., Meier-Kolthoff, J. P., Kumar, N., Bresciani, A., Martínez, I., Just, S., Ziegler, C., et al. (2016). The mouse intestinal bacterial collection (mibc) provides host-specific insight into cultured diversity and functional potential of the gut microbiota. *Nature microbiology*, 1(10):16131.

[Langmead and Salzberg, 2012] Langmead, B. and Salzberg, S. L. (2012). Fast gapped-read alignment with bowtie 2. *Nature methods*, 9(4):357.

[Langmead et al., 2009] Langmead, B., Trapnell, C., Pop, M., and Salzberg, S. L. (2009). Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome biology*, 10(3):R25.

[Lapage et al., 1992] Lapage, S., Sneath, P., Lessel, E., Skerman, V., Seeliger, H., and Clark, W. (1992). *International code of nomenclature of bacteria: bacteriological code, 1990 revision*. ASM Press.

[Lee et al., 2011] Lee, J. Y., Passalacqua, K. D., Hanna, P. C., and Sherman, D. H. (2011). Regulation of petrobactin and bacillibactin biosynthesis in bacillus anthracis under iron and oxygen variation. *PLoS One*, 6(6):e20777.

[Li et al., 2015] Li, D., Liu, C.-M., Luo, R., Sadakane, K., and Lam, T.-W. (2015). Megahit: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de bruijn graph. *Bioinformatics*, 31(10):1674–1676.

[Li et al., 2016] Li, D., Luo, R., Liu, C.-M., Leung, C.-M., Ting, H.-F., Sadakane, K., Yamashita, H., and Lam, T.-W. (2016). Megahit v1. 0: a fast and scalable metagenome assembler driven by advanced methodologies and community practices. *Methods*, 102:3–11.

[Li, 2018] Li, H. (2018). Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 34(18):3094–3100.

[Love et al., 2014] Love, M. I., Huber, W., and Anders, S. (2014). Moderated estimation of fold change and dispersion for rna-seq data with deseq2. *Genome biology*, 15(12):550.

[Lozupone et al., 2012] Lozupone, C. A., Stombaugh, J. I., Gordon, J. I., Jansson, J. K., and Knight, R. (2012). Diversity, stability and resilience of the human gut microbiota. *Nature*, 489(7415):220.

[Maier and Storer, 1977] Maier, D. and Storer, J. A. (1977). A note on the complexity of the superstring problem. *Computer Science Laboratory: Princeton University*.

[Mainil, 2013] Mainil, J. (2013). Escherichia coli virulence factors. *Veterinary immunology and immunopathology*, 152(1-2):2–12.

[Manber and Myers, 1993] Manber, U. and Myers, G. (1993). Suffix arrays: a new method for on-line string searches. *siam Journal on Computing*, 22(5):935–948.

[Mann and Whitney, 1947] Mann, H. B. and Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60.

[Mazmanian et al., 2005] Mazmanian, S. K., Liu, C. H., Tzianabos, A. O., and Kasper, D. L. (2005). An immunomodulatory molecule of symbiotic bacteria directs maturation of the host immune system. *Cell*, 122(1):107–118.

[Mullis et al., 1987] Mullis, K. B., Erlich, H. A., Arnheim, N., Horn, G. T., Saiki, R. K., and Scharf, S. J. (1987). Process for amplifying, detecting, and/or-cloning nucleic acid sequences. US Patent 4,683,195.

[Myers, 1995] Myers, E. W. (1995). Toward simplifying and accurately formulating fragment assembly. *Journal of Computational Biology*, 2(2):275–290.

[Myers and Miller, 1988] Myers, E. W. and Miller, W. (1988). Optimal alignments in linear space. *Bioinformatics*, 4(1):11–17.

[Namiki et al., 2012] Namiki, T., Hachiya, T., Tanaka, H., and Sakakibara, Y. (2012). Metavelvet: an extension of velvet assembler to de novo metagenome assembly from short sequence reads. *Nucleic acids research*, 40(20):e155–e155.

[Needleman and Wunsch, 1970] Needleman, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453.

[Olsen and Woese, 1997] Olsen, G. J. and Woese, C. R. (1997). Archaeal genomics: an overview. *Cell*, 89(7):991–994.

[O'Toole and Jeffery, 2015] O'Toole, P. W. and Jeffery, I. B. (2015). Gut microbiota and aging. *Science*, 350(6265):1214–1215.

[Paulson et al., 2013] Paulson, J. N., Stine, O. C., Bravo, H. C., and Pop, M. (2013). Differential abundance analysis for microbial marker-gene surveys. *Nature methods*, 10(12):1200–1202.

[Peng et al., 2012] Peng, Y., Leung, H. C., Yiu, S.-M., and Chin, F. Y. (2012). Idba-ud: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. *Bioinformatics*, 28(11):1420–1428.

[Peterson et al., 2009] Peterson, J., Garges, S., Giovanni, M., McInnes, P., Wang, L., Schloss, J. A., Bonazzi, V., McEwen, J. E., Wetterstrand, K. A., Deal, C., et al. (2009). The nih human microbiome project. *Genome research*, 19(12):2317–2323.

[Pevzner et al., 2001] Pevzner, P. A., Tang, H., and Waterman, M. S. (2001). An eulerian path approach to dna fragment assembly. *Proceedings of the national academy of sciences*, 98(17):9748–9753.

[Pruitt et al., 2007] Pruitt, K. D., Tatusova, T., and Maglott, D. R. (2007). Ncbi reference sequences (refseq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic acids research*, 35(suppl_1):D61–D65.

[Rhee et al., 2009] Rhee, S. H., Pothoulakis, C., and Mayer, E. A. (2009). Principles and clinical implications of the brain–gut–enteric microbiota axis. *Nature reviews Gastroenterology & hepatology*, 6(5):306.

[Roberts et al., 2004] Roberts, M., Hayes, W., Hunt, B. R., Mount, S. M., and Yorke, J. A. (2004). Reducing storage requirements for biological sequence comparison. *Bioinformatics*, 20(18):3363–3369.

[Robinson et al., 2010] Robinson, M. D., McCarthy, D. J., and Smyth, G. K. (2010). edger: a bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1):139–140.

[Rodríguez et al., 2015] Rodríguez, J. M., Murphy, K., Stanton, C., Ross, R. P., Kober, O. I., Juge, N., Avershina, E., Rudi, K., Narbad, A., Jenmalm, M. C., et al. (2015). The composition of the gut microbiota throughout life, with an emphasis on early life. *Microbial ecology in health and disease*, 26(1):26050.

[Rognes and Seeberg, 2000] Rognes, T. and Seeberg, E. (2000). Six-fold speed-up of smith–waterman sequence database searches using parallel processing on common microprocessors. *Bioinformatics*, 16(8):699–706.

[Ronaghi et al., 1998] Ronaghi, M., Uhlén, M., and Nyrén, P. (1998). A sequencing method based on real-time pyrophosphate. *Science*, 281(5375):363–365.

[Sanger et al., 1977] Sanger, F., Nicklen, S., and Coulson, A. R. (1977). Dna sequencing with chain-terminating inhibitors. *Proceedings of the national academy of sciences*, 74(12):5463–5467.

[Savage, 1977] Savage, D. C. (1977). Microbial ecology of the gastrointestinal tract. *Annual review of microbiology*, 31(1):107–133.

[Schaeffer et al., 2017] Schaeffer, L., Pimentel, H., Bray, N., Melsted, P., and Pachter, L. (2017). Pseudoalignment for metagenomic read assignment. *Bioinformatics*, 33(14):2082–2088.

[Schleifer, 2009] Schleifer, K. H. (2009). Classification of bacteria and archaea: past, present and future. *Systematic and applied microbiology*, 32(8):533–542.

[Schulz et al., 2012] Schulz, M. H., Zerbino, D. R., Vingron, M., and Birney, E. (2012). Oases: robust de novo rna-seq assembly across the dynamic range of expression levels. *Bioinformatics*, 28(8):1086–1092.

[Sekirov et al., 2010] Sekirov, I., Russell, S. L., Antunes, L. C. M., and Finlay, B. B. (2010). Gut microbiota in health and disease. *Physiological reviews*, 90(3):859–904.

[Sender et al., 2016a] Sender, R., Fuchs, S., and Milo, R. (2016a). Are we really vastly outnumbered? revisiting the ratio of bacterial to host cells in humans. *Cell*, 164(3):337–340.

[Sender et al., 2016b] Sender, R., Fuchs, S., and Milo, R. (2016b). Revised estimates for the number of human and bacteria cells in the body. *PLoS biology*, 14(8):e1002533.

[Sgritta et al., 2019] Sgritta, M., Dooling, S. W., Buffington, S. A., Momin, E. N., Francis, M. B., Britton, R. A., and Costa-Mattioli, M. (2019). Mechanisms underlying microbial-mediated changes in social behavior in mouse models of autism spectrum disorder. *Neuron*, 101(2):246–259.

[Simpson and Durbin, 2010] Simpson, J. T. and Durbin, R. (2010). Efficient construction of an assembly string graph using the fm-index. *Bioinformatics*, 26(12):i367–i373.

[Simpson and Durbin, 2012] Simpson, J. T. and Durbin, R. (2012). Efficient de novo assembly of large genomes using compressed data structures. *Genome research*, 22(3):549–556.

[Smith et al., 1981] Smith, T. F., Waterman, M. S., et al. (1981). Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197.

[Thursby and Juge, 2017] Thursby, E. and Juge, N. (2017). Introduction to the human gut microbiota. *Biochemical Journal*, 474(11):1823–1836.

[Turcatti et al., 2008] Turcatti, G., Romieu, A., Fedurco, M., and Tairi, A.-P. (2008). A new class of cleavable fluorescent nucleotides: synthesis and optimization as reversible terminators for dna sequencing by synthesis. *Nucleic acids research*, 36(4):e25–e25.

[Ussery et al., 2009] Ussery, D. W., Wassenaar, T. M., and Borini, S. (2009). *Computing for comparative microbial genomics: bioinformatics for microbiologists*, volume 8. Springer Science & Business Media.

[Van Dijk et al., 2014] Van Dijk, E. L., Auger, H., Jaszczyszyn, Y., and Thermes, C. (2014). Ten years of next-generation sequencing technology. *Trends in genetics*, 30(9):418–426.

[Voelkerding et al., 2009] Voelkerding, K. V., Dames, S. A., and Durtschi, J. D. (2009). Next-generation sequencing: from basic research to diagnostics. *Clinical chemistry*, 55(4):641–658.

[Wood et al., 2019] Wood, D. E., Lu, J., and Langmead, B. (2019). Improved metagenomic analysis with kraken 2. *Genome biology*, 20(1):257.

[Wood and Salzberg, 2014] Wood, D. E. and Salzberg, S. L. (2014). Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome biology*, 15(3):1–12.

[Wu et al., 2020] Wu, J., Wang, K., Wang, X., Pang, Y., and Jiang, C. (2020). The role of the gut microbiome and its metabolites in metabolic diseases. *Protein & Cell*, pages 1–14.

[Xu et al., 2006] Xu, L., Chen, H., Hu, X., Zhang, R., Zhang, Z., and Luo, Z. W. (2006). Average Gene Length Is Highly Conserved in Prokaryotes and Eukaryotes and Diverges Only Between the Two Kingdoms. *Molecular Biology and Evolution*, 23(6):1107–1108.

[Zerbino and Birney, 2008] Zerbino, D. R. and Birney, E. (2008). Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome research*, 18(5):821–829.

[Zhang et al., 2010] Zhang, C., Zhang, M., Wang, S., Han, R., Cao, Y., Hua, W., Mao, Y., Zhang, X., Pang, X., Wei, C., et al. (2010). Interactions between gut microbiota, host genetics and diet relevant to development of metabolic syndromes in mice. *The ISME journal*, 4(2):232.

[Zhang and Lin, 1987] Zhang, F. J. and Lin, G. N. (1987). On the de bruijn-good graphs. *Acta Math. Sinica*, 30(2):195–205.

[Zhu et al., 2010] Zhu, W., Lomsadze, A., and Borodovsky, M. (2010). Ab initio gene identification in metagenomic sequences. *Nucleic acids research*, 38(12):e132–e132.