Michał Borowski 406105 - Raport do projektu

Implementacja

Do poprawnego działania programu wymagana jest obecność wszystkich dostarczonych do zadania plików w folderze z programem. Na pliku hmm musi być wykonana funkcja hmmpress jeszcze przed uruchomieniem programu. Dodatkowo wymagane jest zainstalowanie bibliotek z 'requirements.txt' oraz oczywiście posiadanie lokalnie zainstalowanych programów blast i hmmer.

W mojej implementacji znajdują się funkcje o nazwach 'blast_work', 'hmm_work', 'gene_ontology_work' i 'fisher_bonferroni_work', odnoszące się do kolejnych punktów zadania. Funkcje w razie potrzeby zwracają wyniki potrzebne w kolejnych punktach zadania. Są wywoływane kolejno w mainie i zapisują wyniki do plików o nazwach 'answer1.txt', 'answer2.txt' i 'answer3.txt'. Wyniki punktu czwartego są wypisywane na standardowe wyjście programu.

Poniżej krótki opis implementacji każdego z punktów zadania:

1. Implementacja była tworzona na podstawie dostarczonego w treści zadania tutoriala. Nie wymaga bibliotek wychodzących poza biopython. Tworzona jest baza z parametrem dbtype równym 'nucl', a wyszukiwanie jest realizowane za pomocą polecenia 'tblastn'. Następnie dla każdego z fragmentów odczytywany jest najlepiej pasujący gen. Wyniki są zapisywane w 'answer1.txt' w formacie: w każdej linii id fragmentu i id najlepszego genu, oddzielone spacją. Dodatkowo rekordy odpowiadające znalezionym genom można znaleźć w pliku 'bests.fasta'.

2. Polecenie hmmscan jest wykonywane systemowo, ale z poziomu pythona. Niezmodyfikowany output można znaleźć w pliku 'hmm_output.txt'. Plik ten jest czytany z pomocą HmmerIO z biopythona. Dla każdego genu znajdowane jest najlepsze trafienie i zapisywane jest w 'answer2.txt' w formacie: w każdej linii id genu i nazwa domeny Pfam, oddzielone spacją.

3. Odczytanie pliku .gaf jest realizowane przy pomocy biblioteki goatools [1], a odczytanie pliku .obo za pomocą biblioteki obonet[2] we współpracy z biblioteką networkx. Terminy ogólniejsze są znajdowane przy pomocy funkcji descendants z networkx. Wyniki są zapisywane w pliku 'answer3.txt' w formacie: w każdych dwóch kolejnych liniach, w pierwszej id genu, a w drugiej terminy oddzielone przecinkami.

4. Test Fishera wykonywany przy pomocy biblioteki scipy, a poprawka Bonferroniego z pomocą biblioteki statsmodels. Większość kodu to wczytywanie wyników poprzednich punktów zadania. Wyniki tej części są wypisywane na standardowe wyjście, najpierw następuje nazwa testu, a potem

[1] https://github.com/tanghaibao/goatools/blob/main/notebooks/annotations_gaf_file.ipynb

[2] https://pypi.org/project/obonet/

nadprezentowane terminy, przy czym tylko w jednym przypadku wynik jest niepusty.

W kodzie znajdują się też zakomentowane fragmenty kodu, które pomagały w analizowaniu wyników. Są one oznaczane poprzez komentarze w cudzysłowach.

Omówienie wyników

1. Tutaj warto odnotować, że w większości wypadków drugie najlepsze trafienie BLASTem ma znacznie wyższe e-value niż pierwsze, co zwiększa wiarygodność wyników. Jedynym przypadkiem w którym obie wartości są tego samego rzędu jest przypadek fragmentu o id 'groupA_47', gdzie obie wartości są rzędu 10^{-83} i wskazują na geny o identyfikatorach 'tufB' i 'tufA'. Są to oczywiście geny o bardzo podobnych sekwencjach. Najprawdopodobniej wybranie wybranie genu o gorszym e-value nie zmieniłoby dalszych wyników. Kolejne trafienia dla tego fragmentu znacząco odbiegają od dwóch pierwszych.

   Warto też dodać, że najlepsze trafienia nie mają e-value większego rzędu niż 10^{-28}, co zdecydowanie nie przekracza standardowo przyjmowanych thresholdów i wskazuje na faktyczne i nieprzypadkowe przyporządkowanie fragmentów do genów.

2. Stosując analogiczną analizę dla tych wyników, dochodzimy do bardzo podobnych wniosków. Jest tutaj więcej przypadków w których evalue dwóch pierwszych trafień jest porównywalne. Dotyczy to genów fumB, cbpA, carA i prfC. Warto zauważyć, że nie dotyczy to genu wspomnianego w poprzednim punkcie. Wydaje mi się, że domeny RF3_C oraz GTP_EFTU, odpowiadające najlepszym trafieniom dla prfC, są dosyć odległe i może to wskazywać na niesłuszne przyporządkowanie danego genu do domeny białkowej.

   Tym razem mamy też przypadek w którym e-value najlepszego trafienia jest bardzo duże, bo wynoszące 0.024 dla genu tdcR, ale chyba wciąż mieszczące się w standardowych thresholdach.

3. Moja analiza w tym przypadku ogranicza się do zauważenia, że różnych terminów jest dokładnie 999 i ogromna większość z nich nie występuje we więcej niż 10 genach. Kilka terminów występuje we wszystkich genach. Wydaje mi się, że sensowne wnioski odnośnie terminów GO to raczej domena punktu 4.

4. W przypadku domen Pfam nie zostało wykryte żadne wzbogacanie, niezależne od przyjętego poziomu istotności. Wydaje się, że ciężko też próbować doszukiwać się takowego wzbogacania gdybyśmy zamiast jednej domeny przyporządkowali ich więcej, ze względu na to, że z wyłączeniem pojedynczych przypadków, wynik o najmniejszym e-value mocno odstaje od pozostałych.

Podobnie jest ze wzbogacaniem w przypadku terminów GO. Jedynym wykrytym, nadprezentowanym termem jest "protein-containing complex" w grupie A. Jest on natomiast przynajmniej wykrywany przy standardowym poziomie istotności równym 0.05. Powiększenie tego poziomu nie zmienia wyników.

Zmiana testu Fishera na dwustronny też oczywiście nie dodaje nic do tej analizy. Nie umieszczałem tego testu w kodzie.