Tekst do prezentacji dotyczącej projektu zaliczeniowego

1. Slajd wprowadzający

2. Plan prezentacji

- analizowane dane,
- obrany cel,
- hipoteza badawcza,
- wykorzystane programy i metody,
- wyniki i wnioski - analiza 1,
- wyniki, wnioski, paralelizacja - analiza 2,
- podsumowanie

3. Analizowane dane

Analizowany przez nas zbiór danych zawiera informacje na temat krów należących do rasy holsztyńsko-fryzyjskiej.
Do dyspozycji dostaliśmy dane genetyczne dotyczące krów zdrowych oraz krów chorych, po 12 samic z każdej grupy. Wszystkie z nich są półsiostrami - pochodzą od jednego buhaja.
Pełny kariotyp krowy składa się z 60 chromosomów - 29 par autosomów i 1 pary allosomów.
OBRAZ: Krowa rasy holsztyńsko-fryzyjskiej.
OBRAZ: Kariotyp krowy: 29,XX.

4. Analizowane dane

Do naszej dyspozycji otrzymaliśmy 4 pliki danych w formacie *.vcf*. To pliki tekstowe zawierające informacje na temat wariantów określonych sekwencji genów. Są to mutacje typu SNP - polimorfizmu pojedynczego nukleotydu *(Single Nucleotide Polymorphism)* polegającego na zmianie pojedynczego nukleotydu (*A*, *T*, *C* lub *G*) pomiędzy osobnikami danego gatunku.
Są to wyniki uzyskane na podstawie genotypowania całego genomu metodą WGS *(Whole Genome Sequencing)*. Dwa z nich zawierały pełną informację na temat znalezionych mutacji u osobników zdrowych oraz chorych, a dwa kolejne obejmowały wyłącznie kolumny dotyczące chromosomów, lokalizacji mutacji oraz genotypów u badanych osobników (w liczbie 12).
Analizowane przez nas pliki zawierające tylko niezbędne dane z genotypami, miały ponad 14,3 mln rekordów dla osobników chorych oraz ponad 13,8 mln rekordów dla osobników zdrowych.
Otrzymane dane dotyczą chromosomów (1-29 oraz płci), jednak zawierają także sekwencje kontigowe o bliżej niezidentyfikowanej lokalizacji.
OBRAZ: Screen po wczytaniu naszych danych.
OBRAZ: Mutacja SNP.

5. Cel badań

Podczas naszych badań postawiliśmy sobie 2 różne cele.
Analiza liczby mutacji typu SNP w chromosomach osobników krów rasy holsztyńsko-fryzyjskiej.
Znalezienie mutacji typu SNP mogących mieć podłoże biologiczne dla rozwoju choroby i wyznaczenie ich zależności z wybranymi parametrami dla każdego chromosomu.

6. Hipotezy badawcze

Podczas pierwszej analizy postawiliśmy sobie hipotezę badawczą, czy zawartość mutacji SNP zależy od długości chromosomu. W tym przypadku badamy osobniki zdrowe i chore osobno - nie porównujemy otrzymywanych wartości pomiędzy nimi.
OBRAZ: Badanie niezależne.
Druga analiza opiera się na znalezieniu wspólnych SNP pomiędzy grupami. Osobniki zdrowe i chore badamy razem. Chcemy przekonać się czy liczba SNP istotnych biologicznie zależy od długości chromosomu oraz czy zależy od liczby znalezionych SNP dla każdego chromosomu (wspólnych SNP dla osobników zdrowych i chorych).
OBRAZ: Badanie zależne.

7. Wykorzystane programy i metody

Nasze analizy przeprowadziliśmy używając dwóch języków programowania - Python oraz R. W obu korzystaliśmy z ich najnowszych dostępnych wersji, tj. Python 3.10.2 oraz R 4.1.2. Nasze skrypty pisaliśmy w programie PyCharm Community Edition wersja 2021.3.2 oraz RStudio wersja 2021.09.2.
Oprócz nich skorzystaliśmy również z bazy danych dotyczących genomów w NCBI, a podczas prostych wyliczeń korzystaliśmy z powszechnie znanych arkuszy kalkulacyjnych Microsoft Excel w pakiecie Office365 wersja 16.0.11425.2022 oraz Numbers w wersji 11.2 (na komputery MacOS).
W Pythonie skorzystaliśmy z paczek:

- *itertools* - moduł pozwalający na działania na zestawie szybkich, wydajnych pamięciowo narzędzi,
- *os* - moduł ten jest zależny od systemu, z którego korzysta użytkownik, zapewnia on funkcje dostępne w systemie do języka Python,
- *pandas* - narzędziem do analizy danych oraz ich manipulacji (dotyczy tabeli), pozwala również na partycjonowanie danych, wczytywanie ich częściami oraz zapis fragmentaryczny na dysku, co pozwala na odciążenie pamięci RAM,
- *scipy* - biblioteka do pracy z procedurami numerycznymi oraz ich optymalizacji,
- *numpy* - biblioteka dodająca możliwość pracy na wielowymiarowych macierzach oraz tablicach, które są dużo szybsze niż wbudowane w język listy, zapewnia również szereg narzędzi do operowania na danych wraz z zapewnieniem dużej szybkości i możliwością współpracy z bibliotekami do akceleracji danych przez GPU,
- *multiprocessing* - paczka pozwalająca na współbieżne efektywne działanie lokalne, jak i zdalne.

OBRAZ: Python i jego paczki.
Natomiast w R były to:

- *dplyr* - manipulacja ramkami danych w intuicyjny i przyjazny sposób, używany w celu przekształcenia istniejących zestawów danych do formatu lepiej dopasowanego celem analizy lub wizualizacji danych,
- *microbenchmark* - celem odpowiedniego wyliczania trwających przez nas funkcji,
- *parallel* - paczka zawiera dużo funkcji umożliwiającej paralellizację naszych danych, skracamy pracę wykonując obliczenia równoległe.

OBRAZ: R i jego paczki.

8. Wykorzystane programy i metody

Podczas naszych analiz użyliśmy wielu ciekawych funkcji zaimplementowanych do języków programowania lub też zawartych we wspomnianych wcześniej paczkach. Najciekawszymi wykorzystanymi przez nas funkcjami w Pythonie są:

- *intertools.zip_longest* - tworzy iterator agregujący elementy z każdej iteracji, wykorzystany w celu działania na częściach danych przy ich wczytywaniu w celu połączenia danych w sposób uporządkowany.
- *multiprocessing.Pool()* - "tworzymy" pulę pracowników, czyli w praktyce ile wątków chcemy przydzielić do pracy,
- *map()* - dzięki tej metodzie przekazujemy nasze dane *(input)* do puli zdefiniowanej przy pomocy *Pool()*, gdzie przypisywane są jako oddzielne zadania,
- *pd.groupby()* - grupujemy nasze dane, po kolumnie '*Chr1*', w celu wykonywania obliczeń na określonych grupach,
- *pd.merge()* - łączenie obiektów serii DataFrame w stylu znanym nam z SQL'a, nasze dane są łączone względem klucza (dla nas kluczem łączenia jest kolumna 'X182*'*),
- *chi2_contingency()* - funkcja testu chi-kwadrat dla hipotezy o niezależności obserwowanych danych.

Natomiast w języku R warto wspomnieć o funkcjach:

- *filter()* z paczki *dplyr* - pozwala łatwo i szybko filtrować dane względem poszczególnych wartości chromosomów lub niepotrzebnych wartości genotypów,
- *inner_join()* - łączy tabele zdrowych i chorych zgodnie z przyjętym założeniem (lokalizacją SNP w odpowiedniej kolumnie),
- *rowSums()* - zlicza ilość określonych genotypów w kolumnach,
- *rbind()* - tworzy tabelę wiersz po wierszu na podstawie podanych wektorów, która stanowi dane wejściowe do testu chi-kwadrat,
- *chisq.test()* - wylicza wartość testu statystycznego chi-kwadrat niezależności dla analizowanych rekordów.

9. Przygotowanie danych

Przygotowanie danych do pierwszej analizy było dziecinnie proste. Wykorzystaliśmy w niej dane pobrane z bazy NCBI.
Napisany przez nas skrypt operował na podstawowych operacjach matematycznych. W oparciu o jego działanie obliczyliśmy całkowitą zawartość SNP w badanych chromosomach.
OBRAZ: Znaleziony organizm w bazie danych.
OBRAZ: Informacje dotyczące chromosomów.

10. Przygotowanie danych

Drugie zadanie było znacznie bardziej skomplikowane i wymagało odpowiedniego przekształcenia danych i znalezienia odpowiadających sobie wartości.
Ponieważ analizowaliśmy bardzo duży zbiór danych (wielkości ponad 1 Gb każdy), dlatego operowanie na jednej ogromnej tabeli jest wysoko nieefektywne i w dużym stopniu obciąża pamięć. Zdecydowaliśmy się podzielić nasze wyniki na tabele z informacjami dotyczącymi poszczególnych chromosomów dla badanych grup organizmów. W naszych analizach nie uwzględniliśmy danych dotyczących kontigów, które nie były przez nas pożądanymi.
Następnie musieliśmy porównać tabele odpowiadających sobie chromosomów względem badanej mutacji, aby znaleźć te, które powtarzają się w obu przypadkach.
Ostatnim punktem przygotowań była filtracja niepotrzebnych danych. W kolumnach zawierających genotypy dla poszczególnych osobników mogliśmy spotkać bardzo rzadko występujące wartości dotyczące pozostałych nukleotydów, ponieważ mutacja typu SNP nie zawsze wiąże się tylko z 2 nukleotydami - referencyjnym i alternatywnym.
Chcielibyśmy zaznaczyć różnice pomiędzy postępowaniem w obu językach.
OBRAZ: R - przygotowanie danych (niebieska tabela - zdrowi, czerwona tabela -

chorzy, zielona tabela - dane połączone).
W pakiecie R postępowanie było dużo prostsze. Za pomocą polecenia *fread* wczytaliśmy obie ogromne tabele danych, które podzieliliśmy na mniejsze poprzez przefiltrowanie wartości kolumny z chromosomami - otrzymaliśmy 30 tabel (każda dla 1 chromosomu) dla każdego zestawu danych (łącznie 60). Następnie, względem tej samej lokalizacji SNP, łączyliśmy tabele osobników zdrowych i chorych odpowiadających sobie chromosomów, odrzucając jednocześnie niepowtarzające się mutacje. Mamy teraz zatem 30 tabel - każda dla odpowiadających sobie SNP w tych samych chromosomach. Filtracja danych zachodziła już w obrębie każdej tabeli osobno. Podobnie jak wykonane dalej obliczenia.
Cały czas pracujemy na rozdzielnych tabelach.

11. Przygotowanie danych

OBRAZ: Python - przygotowanie danych (niebieska tabela - zdrowi, czerwona tabela - chorzy, zielona tabela - dane połączone).
W Pythonie zastosowaliśmy bardziej wyrafinowany sposób przygotowania danych. Najpierw, stosując gotowe funkcje pochodzące z biblioteki Pandas, dzieliliśmy obie tabele z danymi na mniejsze fragmenty. W wykorzystanej przez nas pętli zapisywaliśmy je w pamięci i następnie wczytywaliśmy. Po tym łączyliśmy nasze tabele zgodnie z odpowiednią lokalizacją SNP, zachowując ich właściwe przywiązanie do określonego chromosomu. Zapisywaliśmy otrzymane w ten sposób dane do następnego pliku i usuwaliśmy z pamięci niepotrzebne fragmenty 2 tabeli będących danymi wejściowymi. Na sam koniec połączyliśmy otrzymane tabele w jedną i zastosowaliśmy na niej filtrację danych. Same wyliczenia wykonywaliśmy już niezależnie względem każdego chromosomu.
Najpierw dzielimy nasze dane, aby je potem połączyć - gwarantuje to szybszą pracę, która mniej obciąża pamięć.

12. Wyniki i wnioski

13. Wyniki - analiza 1

Najpierw przeanalizujemy wyniki dla grupy osobników zdrowych. Tabela ukazuje długość oraz liczbę SNP znalezioną dla każdego chromosomu oraz ich wzajemny stosunek.
Wyniki ukazują silną korelację pomiędzy liczbą znalezionych SNP oraz wielkością chromosomu u zdrowych osobników. Widoczny obok wykres został wykonany za pomocą pakietu Seaborn (podobnie jak następne).
Wartość współczynnika korelacji wynosi 0,8964. Obliczyliśmy go za pomocą funkcji *pearsonr* z pakietu SciPy (podobnie jak następne).
W przypadku osobników zdrowych największym stosunkiem liczby mutacji do rozmiaru charakteryzuje się chromosom 23., natomiast najmniejszym współczynnikiem chromosom X oraz chromosom 8.
TABELA: Wyniki.
OBRAZ: Wykres naszych wyników.

14. Wyniki - analiza 1

Następnie identyczną analizę wykonaliśmy dla osobników chorych.
Wyniki ukazują silną korelację pomiędzy liczbą znalezionych SNP oraz wielkością chromosomu u chorych osobników.
Wartość współczynnika korelacji wynosi 0,8983.
W przypadku osobników chorych największym stosunkiem liczby mutacji do rozmiaru

charakteryzuje się chromosom 23., natomiast najmniejszym współczynnikiem chromosom X oraz chromosom 8. Wynik jest identyczny, jak w przypadku osobników zdrowych.

TABELA: Wyniki.

OBRAZ: Wykres naszych wyników.

15. Wnioski - analiza 1

Dla obydwu grup dostaliśmy bardzo podobne wyniki. Współczynniki korelacji wynosiły prawie 0,9. Świadczy to o dużej zależności pomiędzy liczbą znalezionych SNP a wielkością chromosomu. Zatem możemy stwierdzić, że im dłuższy chromosom analizujemy, tym więcej odnajdziemy SNP.

Należy jednak zauważyć interesujący przypadek - wartość, która w znacznym stopniu odstaje od naszej prostej regresji. Dotyczy ona jedynego, „specjalnego" chromosomu - chromosomu płci. Znajduje się na nim znacznie mniej badanych mutacji niż na pozostałych chromosomach. Możemy zatem wyciągnąć kilka hipotez:

- mutuje on wolniej, zatem zawiera mniej mutacji (w tym SNP),
- mutacje zawarte na chromosomie X są znacznie groźniejsze w skutkach niż na autosomach (choroby związane z płcią),
- geny zawarte na chromosomie X biorą udział w różnicowaniu płci, przez co wiele mutacji może skutkować bezpłodnością osobnika,
- występuje proces inaktywacji chromosomu X, na którym znajduje się większa liczba mutacji,
- naprawa wszelkich mutacji i błędów w genach podczas mechanizmu replikacji spowodowana obecnością drugiego chromosomu X, który pełni rolę zapasową, odbywa się za pomocą ich niezmutowanych wersji.

OBRAZY: Wykresy naszych wyników.

TABELE: Analizowane zależności.

16. Wnioski - analiza 1

Zatem odpowiadając na pytanie zawarte w hipotezie badawczej - tak, liczba mutacji typu SNP zależy od długości chromosomu. Na podstawie długości chromosomu możemy oszacować zawartą w jego nici DNA liczbę mutacji typu SNP.

17. Wyniki - analiza 2

Przejdźmy teraz do wyników znacznie dłuższej i trudniejszej analizy do wykonania. Naszym drugim zadaniem było znalezienie miejsc SNP, które mogą mieć znaczenie biologiczne w rozwoju lub pojawieniu się choroby. Po znalezieniu wspólnych mutacji nukleotydowych dla obu badanych grup zwierząt, dla każdej z nich wykonywaliśmy test chi-kwadrat zawarty w pakietach SciPy (funkcja *chi2_contingency()*) lub też wbudowany w R (funkcja *chisq.test()*). W związku z bardzo małą liczebnością naszej próby w badaniach zastosowaliśmy poprawkę Yatesa. Za interesujący nas poziom istotności przyjęliśmy standardową wartość 0,05.

Najwięcej miejsc SNP, które odznaczają się innym stosunkiem genotypów, posiada chromosom 4, a najmniej chromosom 26.

Na koniec zsumowaliśmy otrzymane wyniki dla każdego z chromosomów osobno. Chciałbym również poruszyć czas wykonanych obliczeń - w pakiecie R trwały one zaledwie 267 s; co daje zaledwie ok. 4,5 min. W przypadku obliczeń w Pythonie czas był znacznie dłuższy, bo wynosił aż 4537 s, więc ok. godziny i piętnastu minut. Należy wspomnieć, że obliczenia w pakiecie R obejmowały 8 wątków, podczas gdy w Pythonie było ich aż 16.

TABELA: Wyniki.
OBRAZ: Wykres naszych wyników.

18. Wyniki - analiza 2

Najpierw chcieliśmy przeanalizować czy odsetek znalezionych mutacji typu SNP jest identyczny względem ich początkowej badanej liczby dla każdego chromosomu. Porównywaliśmy ich ilość względem liczby SNP wspólnych dla zdrowych i chorych organizmów (dla każdego chromosomu). Wykonaliśmy test korelacji r-Pearsona celem zbadania zależności obu współczynników.
Otrzymany wynik testu wynosi 0,8015. Obliczyliśmy go za pomocą funkcji *pearsonr* z pakietu SciPy (podobnie jak następne). Widoczny obok wykres został wykonany za pomocą pakietu Seaborn (podobnie jak następne).
OBRAZ: Wykres naszych wyników.

19. Wyniki - analiza 2

Następna analiza dotyczyła zależności między liczbą znalezionych mutacji typu SNP a długością chromosomu. Ponownie wykonaliśmy test korelacji r-Pearsona celem zbadania zależności obu współczynników.
Otrzymany wynik testu wynosi 0,7119.
OBRAZ: Wykres naszych wyników.

20. Wnioski - analiza 2

Na podstawie widocznych obok wykresów możemy stwierdzić niewielkie różnice w zawartości ogólnej liczby SNP i SNP znaczących biologicznie na naszych chromosomach. Na podstawie ukazanych obok wykresów widzimy różnice pomiędzy wysokościami naszych słupków dla poszczególnych chromosomów.
Tabela ułatwia nam zrozumienie wyników. Widzimy inny rozkład naszych danych w obydwu przypadkach. Jest to ogólna obserwacja, niepoparta żadnymi testami statystycznymi. O stosunku wielkości chromosomu do ilości mutacji powiemy na kolejnym slajdzie.
TABELA: Odsetek mutacji znaczących względem wspólnej ich liczby.
OBRAZ: Wykres zależności ilości SNP istotnych biologicznie dla poszczególnych chromosomów.
OBRAZ: Wykres zależności całkowitej ilości SNP dla poszczególnych chromosomów.

21. Wnioski - analiza 2

Dla obu zależności otrzymaliśmy silne korelacje dodatnie. Testy parametryczne r-Pearsona wynosiły dla zależności znaczących biologicznie SNP odpowiednio: 0,8 i 0,71 dla liczby SNP wspólnych i wielkości chromosomu.
WYKRES: Zależność liczby SNP znaczących od całkowitej liczby SNP.
WYKRES: Zależność liczby SNP znaczących od wielkości chromosomu.
Chciałbym jednak rozwinąć temat ukazanych na poprzednim slajdzie wykresów
WYKRES: Zależność całkowitej liczby SNP od wielkości chromosomu.
Tak jak już stwierdziliśmy, w obu przypadkach widzimy zależności. Jednak w przypadku SNP mogących mieć znaczenie biologiczne jest ona wyraźnie słabsza niż dla wszystkich znalezionych przypadków. Wykonaliśmy dodatkowy wykres, który przedstawia całkowitą liczbę wspólnych SNP w zależności od wielkości chromosomu. Różnica pomiędzy wartościami współczynnika korelacji dla obu przypadków wynosi prawie 0,2. Możemy zatem wnioskować, że wielkość chromosomu jest lepszym predyktorem ogólnej liczby SNP niż liczby SNP, w których obserwujemy różnice genotypowe pomiędzy grupami chorych i zdrowych.

22. Wnioski - analiza 2

Pozostało nam odpowiedzieć na pytania zawarte w hipotezach badawczych w sposób dwukrotnie twierdzący - tak, liczba SNP istotnych biologicznie jest zależna zarówno od długości chromosomu, ale także od całkowitej liczby SNP dla każdego chromosomu.
Dlatego znając ich liczbę możemy oszacować długość chromosomu oraz całkowitą liczbę SNP dla chromosomu.

23. Paralelizacja - analiza 2

Paralelizacja w Pythonie dotyczy całej naszej funkcji. Za pomocą metody *multiprocessing.Pool()* do zmiennej pool przypisujemy „pracowników". Liczba tych pracowników wynosi ilość naszych procesorów (wątków) określanych przez *cpu_count()* z biblioteki *os*). W naszym przypadku określiliśmy liczbę wątków jako 16. Metoda *map* zastosowana na obiekcie *pool* przyjmuje 2 argumenty - funkcję której działanie chcemy zrównoleglić oraz dane wejściowe (nazwy plików z mutacjami). Metoda ta zwraca nam dane uzyskane z każdorazowego wywołania funkcji.
Na wykresie czasu w zależności od ilości danych możemy zobaczyć zdecydowaną przewagę wykorzystania wielu rdzeni nad pracą sekwencyjną. Nasze wyniki otrzymujemy średnio w czasie ponad 6 razy krótszym.
OBRAZ: Screen z opisaną metodą obliczeń równoległych.
TABELA: Wyniki czasu w zależności od ilości danych.
WYKRES: Długość pracy zależna od liczby wprowadzonych danych.

24. Paralelizacja - analiza 2

Obliczenia równoległe w R zostały wykonane w prostszy sposób. Dotyczyły one pętli, która jest iterowana po wszystkich chromosomach. Następuje w niej znalezienie identycznych mutacji SNP dla osobników zdrowych i osobników chorych oraz obliczenia statystyczne dla każdego rekordu.
Używaliśmy do tego klastra obliczeniowego za pomocą funkcji *makeCluster* oraz *clusterApply*. Jej argumentem jest liczba wykorzystywanych przez nas wątków, która była zmienna. Po utworzeniu klastra (funkcja *makeCluster*) za pomocą funkcji *clusterExport* importowaliśmy niezbędne dane, które mają być użyte w obliczeniach równoległych. Naszą funkcję zawarliśmy w komendzie *clusterApply*, której to czas mierzyliśmy dla różnej liczby wątków. Możemy zauważyć, że czas wykonania funkcji spada dwukrotnie pomiędzy pracą na 2 i 4 rdzeniami. Następnie poziom trendu dla liczby rdzeni 6 lub 8 znacznie zwalnia.
TABELA: Wyniki czasu w zależności od liczby wątków.
WYKRES: Długość pracy zależna od liczby wątków.
W tym miejscu, na koniec prezentacji chciałbym się również odnieść do szybkości działania obu naszych kodów. Skrypt napisany w R był aż 17 razy szybszy niż ten z Pythona. Należy zauważyć różnicę, że w przypadku pakietu R pracujemy na danych wektorowych, a w Pythonie są to działania na obiektach *data.frame* (tabele), które to zabierają więcej czasu i mocy obliczeniowej. Możemy wysnuć wniosek, że pakiet R lepiej nadaje się (jest szybszy i prostszy) do analizy danych niż język Python.

25. Podsumowanie

Badaliśmy zawartość mutacji typu SNP w genomach krów z rasy holsztyńsko-fryzyjskiej otrzymanych z sekwencjonowania WGS.
Nasze analizy prowadziliśmy w dwóch języka programowania Python oraz R.
Potwierdziliśmy istnienie wszystkich badanych zależności - wszystkie nasze hipotezy badawcze okazały się poprawne.

Obliczenia równoległe bazujące na pracy wielowątkowej gwarantują otrzymanie identycznych wyników w znacznie krótszym okresie czasu.
Praca w pakiecie R okazała się szybsza niż w języku Python - być może powinniśmy spróbować przeprowadzić nasze analizy wyłącznie w oparciu o pakiet NumPy.

26. Slajd zakończeniowy