SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Pronalaženje varijanti gena iz podataka dobivenih sekvenciranjem

Ivan Inkret, Mia Jurdana

Voditelj: *Krešimir Križanović*

Zagreb, lipanj, 2023.

Sadržaj

1. Uvod

Sekvenciranjem su dobiveni uzorci određenog gena za 47 različitih jedinki jelena te 48 jedinki divokoza. Za svaku jedinku sekvenciranje je izvedeno mnogo puta zbog moguće greške kod očitanja, a uzorak svake jedinke sadrži više varijanti istog gena. Očekivana greška sekvenciranja iznosi oko 3% na svakoj pojedinoj sekvenci pa trebamo mnogo očitanja kako bismo došli do koncenzusa. Gene svake jedinke grupirali smo po sličnosti te izdvojili predstavnike svake grupe kako bismo mogli izdvojiti sve varijante gena koje se nalaze u uzorku. Zanima nas koje varijante gena se nalaze u više uzoraka, što bi potvrdilo ili povećalo vjerojatnost, da je pronađena varijanta gena u svojem izvornom obliku.

2. Opis programa

Za izvođenje programa korišteno je nekoliko vanjskih biblioteka čija je instalacija uključena u dostavljenu bash skriptu. Biblioteke potrebne za izvođenje koda su:

- BioPython – pruža različite alate i funkcionalnosti za analizu i manipulaciju bioloških podataka kao što su sekvence DNA, RNA i proteina. Iz ove biblioteke isprobali smo implementacije algoritma k-means i k-medoida.
- Sickit-learn – biblioteka je nešto šireg spektra od BioPython-a te se koristi za različite vrste problema strojnog učenja, od klasifikacije, regresije do grupiranja. Koristili smo implementacije algoritma k-means i gausovih mješavina kako bismo mogli usporediti rezultate.
- Numpy – za brzu i učinkovitu manipulaciju numeričkim podacima.
- Matplotlib – biblioteka za vizualizaciju podataka koju smo u našem slučaju koristili da bi grafički prikazali kretanje pogreške za različit broj grupa.
- Json – rezultati dobiveni pokretanjem grupiranja za broj grupa od 2 do 10 spremljeni su u JSON formatu kako bi ih lakše mogli analizirati. Za svaki uzorak izgrađena je jedna datoteka u kojoj se može vidjeti u koju grupu spada svaka od sekvenci te kolika je pogreška.
- Pandas i Openpyxl – ove biblioteke poslužile su za export konačnih rezultata u .xlsx datoteku. DataFrame struktura podataka iz biblioteke Pandas korištena je za izgradnju dvodimenzionalne tablice podataka, a Openpyxl biblioteka nam je omogućila da zapišemo te podatke u Excel.

Kako bismo izgradili listu svih pronađenih varijanti gena, provodimo sljedeći postupak na svakom uzorku:

2.1. Parsiranje

Dobivene sekvence svake jedinke dane su u .fastq formatu. Sekvence su različitih duljina te za mnoge od njih samo na temelju duljine možemo pretpostaviti da je došlo do većih pogrešaka kod očitanja i možemo ih odbaciti. Budući da je za grupiranje potrebno da sve sekvence budu jednake duljine, iz datoteke su odabrane samo one koje su najzastupljenije po duljini, što u prosjeku ispada blizu 300 parova baza. Time je broj dobivenih sekvenci iz svakog uzorka drastično smanjen, ali pokazuje se da ih ostaje dovoljno za potrebe grupiranja.

2.2. Grupiranje (eng. Clustering)

Listu sekvenci dobivenu parsiranjem prosljeđujemo u algoritam grupiranja. Neki od algoritama koje smo isprobali su k-medians, gauss mixtures i implementacija k-meansa iz scikit-learn biblioteke. Odlučili smo se za algoritam k-means, tj. njegovu implementaciju u biblioteci BioPython. Ovdje se odmah javlja problem jer k-means algoritam radi s brojevima, a sekvence koje imamo su u tekstualnom zapisu. Potrebno je provesti neki postupak konverzije iz string formata u brojčanu

reprezentaciju, a pokazalo se da su najbolji rezultati dobiveni ako se svakom paru baza u sekvenci dodijeli njegova ascii vrijednost. Možemo također pokušati izračunati edit distance od svih sekvenci do neke proizvoljne, ili izgraditi matricu udaljenosti svih sekvenci. K-means također kao argument prima očekivani broj grupa kojega unaprijed ne znamo. Odlučili smo svaki uzorak grupirati u 6 grupa, te zadržati samo one koje su dovoljno velike, a male odbaciti.

2.3. Biranje predstavnika

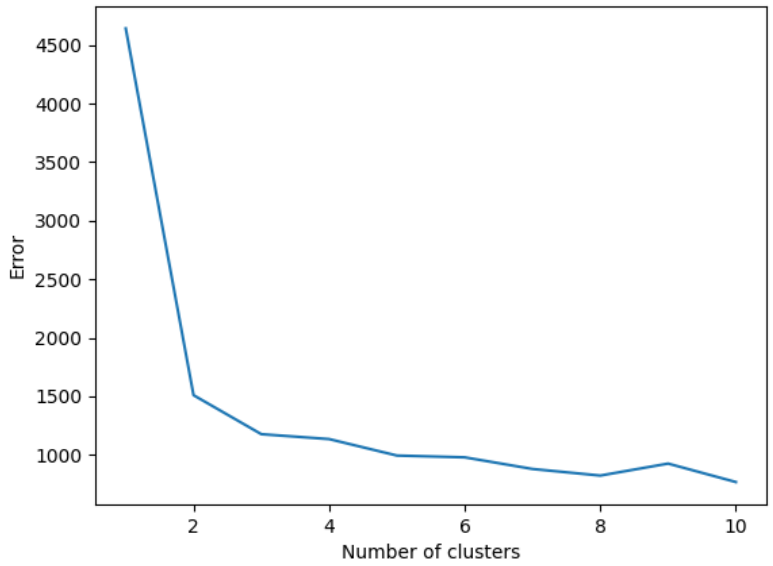
Formirane grupe još uvijek ne sadrže identične sekvence, samo vrlo slične. Ipak, potrebno je izdvojiti jednu koja grupu najbolje predstavlja (poznatu kao centroid clustera, tj. grupe). Odabrali smo opciju gdje je predstavnik sekvenca koja se najviše puta pojavljuje u određenoj grupi, a bila je razmotrena i varijanta gdje na svakoj poziciji biramo par baza koji je najzastupljeniji na toj poziciji i tako izgradimo predstavnika grupe korak po korak.

Nakon ova 3 koraka, izlaz uzorka svake jedinke jest lista od 2-6 predstavnika koji predstavljaju varijante gena u toj grupi. Te predstavnike možemo dodati u globalnu listu svih pronađenih predstavnika te prijeći na sljedeći uzorak. Ova lista biti će korisna kod izgradnje tablice gdje možemo vidjeti koji predstavnici se pojavljuju u najviše uzoraka.

Analiza vremena izvođenja programa za sve podatke pokazuje da je potrebno oko 40s (inter core i5 10300H, 16gb RAM), odnosno ugrubo oko pola sekunde po uzorku. Naravno, većina posla je kmeans algoritam. Ovdje treba primjetiti da se za svaki uzorak kmeans poziva 10 puta za različitu vrijednost broja grupa, a koristimo samo rezultate gdje je broj grupa 6. Ovo je ostavljeno zbog fleksibilnosti, pa se uz minimalne izmjene programa može definirati drugačiji broj clustera.

3. Primjer programa na uzorku j30

U datoteci J30_B_CE_IonXpress_006.fastq nalazi se 3818 sekvenci. Ako zadržimo samo one čija duljina je najzastupljenija, dobivamo 1144 sekvence duljine 296 parova baza. Te sekvence pripremimo u format prikladan za k-means algoritam i pokušamo ih grupirati u 6 grupa. Veličine dobivenih grupa redom ispadaju 5, 471, 626, 12, 16, 14. Odbacujemo sve grupe sa manje od 25 sekvenci. Ovdje valja napomenuti da algoritam k-means nije deterministički te se može dogoditi da, iako su u ovom pokretanju dobivene 2 tražene grupe, kod nekog drugog pokretanja algoritma bi ih moglo biti 3. Odabran je predstavnik svake od 2 grupe na način da je uzeta sekvenca koja se u njima najviše puta pojavljuje, a rezultati su zabilježeni u fasta datoteku. Uzorak J30 korišten je kao primjer jer su za njega bili dostupni kontrolni podaci. Kod usporedbe dobivenih i kontrolnih predstavnika potrebno je obaviti lokalno poravnanje jer su iz kontrolnih podataka izrezani introni s početka i kraja sekvence. Usporedba pokazuje da se predstavnici 2 najveća dobivena clustera savršeno ili jako približno podudaraju s kontrolnim podacima. Slični rezultati dobiveni su i za uzorak J29, za kojega su također bili dostupni kontrolni podaci.



Slika 1. pogreške kod grupiranja za J30

Na slici možemo vidjeti pogreške kod grupiranja za svaki broj grupa od 1 do 10. Analizom ovih podataka metodom koljena moglo bi se zaključiti da je idealni broj grupa za uzorak J30 2. Grupe koje bismo dobili da smo kao parametar algoritmu k-means proslijedili 2 nisu identične onime koje dobijemo kada proslijedimo 6 i zadržimo 2 dovoljno velike, ali su vrlo slične i daju iste predstavnike.

4. Rezultati

Ukoliko želimo dobiti predstavnike za sve prikupljene uzorke, program je potrebno pokrenuti iz naredbenog retka s zastavicom *--all.* Nakon što su podaci obrađeni, u direktoriju *results* biti će svi potrebni rezultati uključujući sljedeće:

- Direktorij *errors* – sadrži fotografije grafova pogreške algoritma k-means za raspon grupa od 2 do 10 za svaki pojedinačni uzorak.
- Direktorij *json* – sadrži listu brojeva grupe u koju u koju su klasificirane sekvence za raspon grupa od 2 do 10 za svaki pojedinačni uzorak. Ovaj direktorij olakšao nam je daljnju analizu, no nije previše važan u smislu konačnih rezultata.
- Direktorij *representatives* – sadrži datoteke u fasta formatu za svaki pojedinačni uzorak u kojima se mogu vidjeti predstavnici tog uzorka. Primjer jedne takve datoteke je na Slici 2.

```
1   >1 representative_1
2   GATCCTCTCTCTGCAGCACATTTCCTGGAGCATCTTAAGGCCGAGTGTCATTTCTTCAAC
3   GGGACGGAGCGGATGCAGTTCCTGGCGAGATACTTCTATAACGGAGAAGAGTACGCGCGC
4   TTCGACAGCGACGTGGGCGAGTTCCGGGCGGTGACCGAGCTGGGGCGGCCGGACGCCAAG
5   TACTGGAACAGCCAGAAGGAGATCCTGGAGCAGCACGGGGCAGAGGTGGACAGGTACTGC
6   AGACACAACTACGGGGTCGGTGAGAGTTTCACTGTGCAGCGGCGAGGTGACGCGAA
7   >2 representative_2
8   GATCCTCTCTCTGCAGCACATTTCCTGGAGTATGCTAAGAGCGAGTGTCATTTCTCCAAC
9   GGGACGCAGCGGGTGCGGTTCCTGGACAGATACTTCTATAACCGGGAAGAGTACGTGCGC
10  TTCGACAGCGACTGGGGCGAGTTCCGGGCGGTGACCGAGCTGGGGCGGCCGTCCGCCAAG
11  TACTGGAACAGCCAGAAGGATTTCATGGAGCAGAAGCGGGCCGAGGTGGACACGGTGTGC
12  AGACACAACTACGGGGTTATTGAGAGTTTCACTGTGCAGCGGCGAGGTGACGCGAA
```

Slika 2. Primjer datoteke J30_B_CE_IonXpress_006_representatives.fasta

- *Representative-table.xslx* – dokument u kojem su konačni rezultati exportirani u Excel tablicu. U prvom stupcu navedeni su svi pronađeni predstavnici. U prvom retku navedeni su svi uzorci. Za svakog pojedinačnog predstavnika ispod uzorka stavljen je znak „+“ ukoliko se taj predstavnik našao u tom uzorku, odnosno, ostavljena je prazna ćelija ukoliko nije. U posljednjem stupcu naveden je ukupan broj pojavljivanja tog predstavnika u svim uzorcima. Četiri najčešća predstavnika navedena su u Tablici 1.

Predstavnik	Uzorci	Broj pojavljivanja
ATCCTCTCTCTGCAGCACATTCTCTGG AGTATAGTAAGAGCGAGTGTCATTTCT TCAACGGGACCGAGCGGGTGCGGTTC CTGGACAGGATACTTCTATAATGGAGA AGAGTACGCGCGCTTCAACAGCGACT GGGGCGAGTACCGGGCGGAGGCCGA GCTGGGGCGGCGGGACGCCGAGCAC TGGAACAGCCAGAAGGAGATTCGTGA CAGACGCGGGCCGCGGTGGACACGT ACTGCAGACACAACTACGGGGTCGGT GAGAGTTTCACTGTGCAGCGGCGA	LME-1193, LME-1194, LME-1215, LME-443, LME-444, LME-445, LME-472, LME-474, LME-475, LME-477, LME-498, LME-541, LME-551, LME-567, LME-567, LME-569, LME-571, LME-575, LME-616, LME-680, LME-870	21
GATCCTCTCTCTGCAGCACATTTCCTG GAGTATGCTAAGAGCGAGTGTCATTTC TCCAACGGGACGCAGCGGGTGCGGTT CCTGGACAGATACTTCTATAACCGGGA AGAGTACGTGCGCTTCGACAGCGACTG GGGCGAGTTCCGGGCGGTGACCGAGC TGGGGCGGCCGTCCGCCAAGTACTGG AACAGCCAGAAGGATTTCATGGAGCAG AAGCGGGCCGAGGTGGACACGGTGTG CAGACACAACTACGGGGTTATTGAGAG TTTCACTGTGCAGCGGCGAGGTGACGC GAA	J1, J10, J11, J17, J18, J2, J20, J21, J23, J25, J27, J3, J30, J32, J34, J5, J6, J9, J7	19
ATCCTCTCTCTGCAGCACATTTCCTGGA GTATCGTAAGAGCGAGTGTCATTTCTTC AACGGGACCGAGCGGGTGCGGTTCCTG GACAGATACTTCCATAATGGAGAAGAGT TGGTGCGCTTCGACAGCGACTGGGGCG AGTTCCGGGCGGTGGCCGAGCTGGGGC GGCCGGACGCCGAGTACTGGAACAGCC AGAAGGAGATTCTGGAGCGGAAGCGGG	LME-1187, LME-1194, LME-1212, LME-1215, LME-367, LME-480, LME-449, LME-506, LME-522, LME-524, LME-528, LME-530, LME-532, LME-551, LME-558, LME-569, LME-571, LME-870	18

CCGCGGTGGACACGTACTGCAGACACA ACTACGGGGTCGTTGAGAGTTTCACTGT GCAGCGGCGA		
ATCCTCTCTCTGCAGCACATTTCCTGGAG TATCATAAGAGCGAGTGTCATTTCTTCAAC GGGACCGAGCGGGTGCGGTTCCTGGACA GATACTTCCATAATGGAGAAGAGTTCGTG CGCTTCAACAGCGACTGGGGCGAGTACC GGGCGGTGGCCGAGCTGGGGCGGCCGG CCGCCGAGCACTGGAACAGCCAGAAGGA GATTCTGGAGCAGAGGCGGGCCGAGGTG GACACGGTGTGCAGACACAACTACGGGG TCGTTGAGAGTTTCACTGTGCAGCGGCGA	LME-1212, LME-367, LME-435, LME-437, LME-480, LME-481, LME-499, LME-522, LME-524, LME-526, LME-530, LME-551, LME-569, LME-571, LME-803, LME-870, LME-885	17

Tablica 1. Popis najčešćih predstavnika i uzoraka u kojima se pojavljuju

5. Zaključak

Klasteriranje bioloških sekvenci je važan zadatak u polju bioinformatike, a pronaći optimalnu i preciznu tehniku može biti izazovno. S obzirom da većina algoritama grupiranja radi s brojčanim reprezentacijama, kao prvi problem nameće se pitanje kako sekvence transformirati u oblik prigodan za ulaz u algoritam. Moguće je računati različite matrične reprezentacije sličnosti ili udaljenosti sekvenci, no takve metode često imaju visok računalni trošak ukoliko rade s velikim brojem uzoraka. S druge strane, sekvence je moguće pretvoriti u brojeve tako da, primjerice, svaki nukleotid (A, C, G, T) bude predstavljen kao binarni vektor ili transformiran u pripadajuću ascii reprezentaciju. U tom slučaju, zadatak postaje vremenski manje zahtjevan, ali će takva pretvorba dovesti do djelomičnog gubitka informacije s obzirom da sekvence gube inherentno biološko značenje i kontekst povezan s nukleotidima. Nakon pretvorbe podataka, nameće se sljedeće pitanje o izboru samog algoritma. Neki od češće korištenih algoritama su k-means, hijerarhijsko grupiranje, samoorganizirajuće mape, model gaussovih mješavina i slično. Za potrebe ovog rada, odabran je algoritam k-means koji je relativno jednostavan za implementaciju, vremenski nije složen te je intuitivno razumljiv i primjenjiv na ovaj problem. Algoritam je pokretan nekoliko puta te je uočeno da je vrlo osjetljiv na početnu poziciju centroida što je dovelo do toga da isti uzorci ponekad rezultiraju različitim grupama. Algoritam se dosta mučio s pronalaskom manjih grupa te se tu najviše očitovalo da dolazi do nedeterminizma u rješenjima. Također, k-means zahtjeva unaprijed poznat broj klastera što u stvarnosti može biti vrlo teško i nepraktično odrediti. Ipak, one najveće grupe pronađene su svaki put te su dva najznačajnija predstavnika za kontrolne uzorke J29 i J30 uspješno detektirana. Rješenje koje smo implementirali proizvelo je dobre rezultate, no svakako ima prostora za poboljšanje.

6. Literatura

[1] Aleb, N., & Labidi, N. (2015, September). An improved K-means algorithm for DNA sequence clustering. In *2015 26th International Workshop on Database and Expert Systems Applications (DEXA)* (pp. 39-42). IEEE.

[2] Zhong, W., Altun, G., Harrison, R., Tai, P. C., & Pan, Y. (2005). Improved K-means clustering algorithm for exploring local protein sequence motifs representing common structural property. IEEE transactions on Nanobioscience, 4(3), 255-265.

[3] Šikić, M., Domazet-Lošo, M. (2013). Skripta iz predmeta bioinformatika.