Detekcija novog alternativnog izrezivanja

Jurčević, Jurica

Sveučilište Josipa Jurja Strossmayera u Osijeku
Fakultet Primjenjene Matematike i Informatike

Sveučilišni preddiplomski studij Matematika i Računarstvo

Detekcija novog alternativnog izrezivanja

Završni rad

Mentor:
izv. prof. dr. sc. Domagoj Matijevič

Komentor:
dr. sc. Luka Borozan

Kandidat:
Jurica Jurčević

Osijek, 2024

Sadržaj

1 | Uvod

Potpuni projekt nalazi se na github repozitoriju

Bioinformatika je interdisciplinarna nauka koja se bavi primjenom metoda računalne znanosti za analizu bioloških podataka, sa posebnim naglaskom na molekularnu biologiju. U eri sekvenciranja genoma i masovnog generiranja bioloških podataka, bioinformatika postaje ključno sredstvo za razumijevanje kompleksnih bioloških sistema. Jedno od najvažnijih područja bioinformatike je analiza gena i transkripta, a proces alternativnog izrezivanja (engl. alternative splicing) ima važnu ulogu u tom području.

Alternativno izrezivanje je posttranskripcijski mehanizam koji omogućava da jedan gen kodira više različitih proteina, što značajno doprinosi genetičkoj i funkcionalnoj raznolikosti organizama. Ključan dio ovog procesa su egzoni, dijelovi DNA koji sadrže kodirajuću poruku i prenose ju ostalim dijelovima stanice. Alternativno izrezivanje privuklo je veliku pažnju znanstvenika kao ključni dio regulacije genskog i proteinskog sastava unutar stanice. Ovaj proces može pružiti uvid u složenost proteinskog sastava kroz evoluciju, dok nepravilnosti u procesu izrezivanja često dovode do razvoja ozbiljnih bolesti poput raka.

Tijekom vremena razvijen je velik broj alata za analizu ovog procesa, a među njima se ističu STAR i AStalavista. Kroz detaljniju analizu alternativnog izrezivanja, biolozi mogu učinkovitije istraživati genome i pokušavati otkriti veze među tim rezultatima i različitim bolestima.

U ovom radu ćemo istražiti alternativno izrezivanje i njegovu detekciju u genomskim podacima pacijenata s autizmom. Najprije ćemo obraditi teorijsku osnovu alternativnog izrezivanja i ključne bioinformatičke koncepte. Zatim ćemo opisati funkcionalnosti alata STAR i AStalavista. Na kraju, predstavit ćemo našu metodologiju za analizu alternativnog izrezivanja i vizualizirati rezultate koje smo dobili.

2 | Teorija

Prije nego što objasnimo proces detekcije novog alternativnog izrezivanja, korisno je najprije objasniti osnovne biološke pojmove koji će biti važni u narednim poglavljima.

2.1 Alternativno izrezivanje

Deoksiribonukleinska kiselina (**DNA**) osnovna je molekula koja nosi genetske upute ključne za rast, razvoj i funkcioniranje svih živih organizama. Ima strukturu dvostrukog heliksa, s dvije duge niti nukleotida uvrnute oko jedna druge. Svaki nukleotid se sastoji od deoksiriboze, fosfatne skupine i jedne od četiri dušične baze: adenin (A), timin (T), gvanin (G) ili citozin (C). Niti su komplementarne, pri čemu se adenin paruje s timinom, a gvanin s citozinom, povezane vodikovim vezama. Ova sekvenca baza kodira genetske informacije koje se transkribiraju u RNA, a zatim prevode u proteine koji obavljaju različite funkcije unutar stanice. DNA se također replicira kako bi se osiguralo da genetske informacije budu točno prenesene na nove stanice tijekom diobe.

Ribonukleinska kiselina (**RNA**) je nukleinska kiselina koja je izravno uključena u sintezu proteina. Ribonukleinska kiselina je važan nukleotid s dugim lancima nukleinske kiseline prisutnim u svim živim stanicama. Njezina glavna uloga je djelovati kao posrednik koji prenosi upute iz DNA-a za kontrolu sinteze proteina. RNA i DNA dijele dušične baze A, G i C. Timin je obično prisutan samo u DNA-u, dok je uracil obično prisutan samo u RNA-u.

Oblik RNA koji se unutar jezgre stvara prepisivanjem poznat je kao **pre-mRNA**. Ona uključuje i **egzone** (kodirajuće regije) i **introne** (nekodirajuće regije). Pre-mRNA prolazi kroz nekoliko koraka obrade prije nego što postane zrela **mRNA**, a jedan od tih koraka je **RNA izrezivanje** (RNA splicing). Ono uključuje uklanjanje introna i spajanje preostalih egzona kako bi se stvorila zrela mRNA. Ovaj ključni proces omogućuju brojni enzimi koji olakšavaju formiranje funkcionalne mRNA, koja zatim putuje do ribosoma, gdje se nastavlja sinteza proteina. Mjesto na kojem se odvija izrezivanje naziva se splice junction. Na kraju, zrela mRNA sastoji se isključivo od egzona.

Alternativno izrezivanje omogućuje da se RNA molekuli obrađuju na različite načine. To znači da jedan gen može generirati različite verzije mRNA, koje potom kodiraju različite proteine. Ovaj proces doprinosi raznolikosti proteina unutar organizma, čime se povećava prilagodljivost i varijabilnost između i unutar vrsta.

Tijekom alternativnog izrezivanja mogu se dogoditi različiti procesi. Najčešći su **Exon Skipping**, **Alternative Acceptor** i **Alternative Donor**. Exon Skipping uključuje preskakanje određenih egzona, često onih koji su neispravni ili neskladni u genetskom kodu. Alternative Acceptor nastaje kada se završni kraj introna zamijeni početnim krajem introna na granici nizvodnog egzona. S druge strane, Alternative Donor se događa kada se početni kraj introna zamijeni završnim krajem introna na granici nizvodnog egzona.

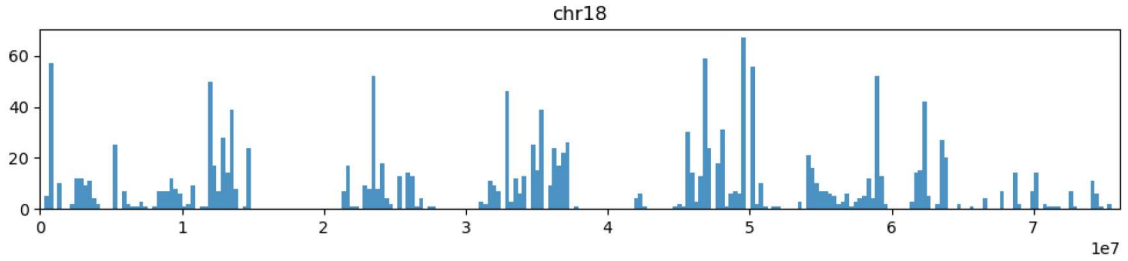
Genomska anotacija je postupak kojim se prepoznaju i bilježe funkcionalni dijelovi genoma, poput gena, regija koje kodiraju proteine, regulatornih elemenata, ne-kodirajućih RNA sekvenci i drugih ključnih sekvenci. Cilj genomske anotacije je prepoznati koje dijelove genoma obavljaju specifične biološke funkcije.

3 | AStalavista

Računalni program **AStalavista** izvlači sve događaje alternativnog izrezivanja iz zadane genomske anotacije koordinata ekson-intron gena. Uspoređujući sve zadane transkripte (mRNA), AStalavista detektira varijacije u njihovoj strukturi izrezivanja i identificira sve AS događaje (exon skipping, alternative donor, itd.) dodjeljujući svakom od njih AS kod. Rezultat je dostavljen u specifičnom **GTF** formatu.

GTF je format odvojen tabulatorima, pri čemu svaki redak opisuje određenu značajku (npr. egzon, intron, start/stop kodon, itd.).

Primjer vizualne reprezentacije podataka danih AStalavistom:



Slika 3.1: Prikaz događaja na kromosomu 18

Prethodna slika prikazuje događaje koje je AStalavista prepoznala da su se dogodili tijekom alternativnog izrezivanja na kromosomu 18.

4 | STAR

Razni algoritmi za poravnanje sekvenci nedavno su razvijeni kako bi se suočili s izazovima alternativnog izrezivanja. Međutim, primjena većine tih algoritama zahtijeva kompromise u područjima točnosti poravnanja (osjetljivost i preciznost) i računalnih resursa (memorija potrebna za izvođenje). S trenutnim napretkom u tehnologijama sekvenciranja, računalna komponenta sve više postaje ograničavajući čimbenik u protoku podataka.

Brza obrada poravnanja ključna je za opsežne projekte, te je s tim na umu dizajniran **STAR** (Spliced Transcripts Alignment to a Reference), algoritam koji je razvijen s ciljem rješavanja brojnih izazova u mapiranju RNA-seq podataka te primjenjuje inovativnu strategiju za poravnanje izrezanih sekvenci. Mnogi ranije razvijeni RNA-seq alati za poravnanje temeljili su se na proširenju postojećih alata za poravnanje kratkih (DNA) očitanja, gdje su se kratka očitanja poravnavala s bazom podataka o spojnim mjestima ili su se dijelovi split-read očitanja (očitanja koja imaju više poravnanja s referentnom sekvencom iz jedinstvenog dijela očitanja.) kontinuirano poravnavali s referentnim genomom, ili kombinacijom tih metoda. Za razliku od tih pristupa, STAR je osmišljen za izravno poravnanje nekontinuiranih sekvenci s referentnim genomom.

STAR algoritam obuhvaća dvije glavne faze: **fazu indeksiranja** i **fazu mapiranja**.

4.1 Indeksiranje genoma

U ovom koraku potrebno je STAR programu dostaviti referentne sekvence genoma (FASTA datoteke) i anotacije (GTF datoteke), na temelju kojih STAR generira genomske indekse. Ti se indeksi pohranjuju na disk i potrebno ih je generirati samo jednom za svaku specifičnu kombinaciju genoma i anotacija. Njihova glavna funkcija je suziti područje pretraživanja sekvenci unutar genoma, čime se štedi vrijeme i računalni resursi. Ovaj je korak važan jer stvara transformiranu verziju genoma koja omogućava STAR-u da učinkovito izvrši sljedeći korak – mapiranje sekvenci na referentni genom.

Naredba koja se koristi za izvršavanje prvog koraka je:

```
c:\User\STAR --runThreadN 128 --runMode genomeGenerate --genomeDir
    index
--genomeFastaFiles sample.fa --sjdbGTFfile sample.rg.gtf --
    sjdbOverhang 150
--sjdbGTFfeatureExon subexon --genomeSAindexNbases 13
```

–runThreadN - opcija definira broj niti koje će se koristiti za generiranje genoma, treba postaviti na broj dostupnih jezgri na poslužiteljskom čvoru

–runMode genomeGenerate - opcija upućuje STAR da pokrene posao generiranja indeksa genoma

–genomeDir - specificira put do direktorija gdje su pohranjeni indeksi genoma
Ovaj direktorij mora biti kreiran prije pokretanja STAR-a i mora imati dozvole za pisanje. Sustav datoteka mora imati najmanje 100 GB diskovnog prostora dostupnog za tipični genom sisavaca. Preporučuje se uklanjanje svih datoteka iz direktorija genoma prije pokretanja koraka generiranja genoma. Ovaj put direktorija morat će se navesti u koraku mapiranja kako bi se identificirao referentni genom.

–genomeFastaFiles - navodi jednu ili više FASTA datoteka s referentnim sekvencama genoma
Višestruke referentne sekvence dopuštene su za svaku FASTA datoteku.

–sjdbGTFfile - navodi put do datoteke s označenim transkriptima u standardnom GTF formatu
STAR će izdvojiti splice junction-e iz ove datoteke i koristiti ih za značajno poboljšanje točnosti mapiranja. Iako je ovo izborno i STAR se može pokrenuti bez anotacija, korištenje anotacija se toplo preporučuje kad god su dostupne.

–sjdbOverhang - specificira duljinu genomske sekvence oko označenog splice junction-a koji će se koristiti u izradi baze podataka splice junction-a
Idealno, ova duljina bi trebala biti jednaka ReadLength-1, gdje je ReadLength duljina read-a. U slučaju read-a različite duljine, idealna vrijednost je max(ReadLength)-1.

4.2 Mapiranje genoma

U ovom koraku je potrebno STAR-u dobaviti datoteke genoma generirane u prvom koraku, kao i RNA-sekvence u obliku FASTA ili FASTQ datoteka. STAR mapira read-ove sekvenci na genom i piše nekoliko izlaznih datoteka, kao što su poravnanja, sažeta statistika mapiranja, splice junction-e, nemapirani read-ovi, itd. Naredba koju smo koristili za izvršavanje drugog koraka je

```
1 c:\User\STAR --runThreadN 128 --genomeDir index --readFilesIn
    sample.fq
```

–genomeDir - navodi put do direktorija genoma gdje su generirani indeksi genoma

–readFilesIn - ime(na) (s putanjom) datoteka koje sadrže sekvence koje treba mapirati (npr. RNA-sekvenca FASTQ datoteke)
STAR može obraditi i FASTA i FASTQ datoteke.

4.3 Output

U konačnici, kao output programa STAR dobijemo nekoliko izlaznih datoteka kao što je prethodno navedeno, ali bitna nam je samo jedna, a to je SAM datoteka.

5 | Provjera podudaranja

Nakon što smo objasnili alternativno izrezivaje, te programe koji se koriste za analiziranje događaja tog procesa, slijedeći zadatak nam je dizajnirati algoritam koji će uspoređivati read-ove iz STAR-a s događajima iz AStalaviste. Nakon toga ćemo podudaranja pohraniti u .txt datoteku, a iz tih podataka i postojeće GTF datoteke izvući ćemo ključne informacije o učestalosti događaja na kromosomima. Na kraju ćemo grafički prikazati te frekvencije, koristeći kromosom 18 kao primjer.

5.1 Implementacija

5.1.1 Generiranje TSV datoteke

Prvi dio algoritma je generiranje TSV (Tab-Separated Values) datoteke koja će biti ključna za proces provjere podudaranja. Ta datoteka se sastoji od koordinata događaja koje smo dobili u GTF datoteci i koristit će se u glavnom programu koji provjerava podudaranja događaja i read-ova.

```
1     with open('outputX.tsv', 'wt') as out_file:
2     tsv_writer = csv.writer(out_file, delimiter='\t')
3     Event_read = open("asta.gtf")
4     for row in Event_read:
5             arr = word_tokenize(row)
6             if (arr[3], arr[4]) not in dict[arr[0]] and arr[0] == "
    chrX":
7                 tsv_writer.writerow([arr[0], arr[3], arr[4]])
8                 dict[arr[0]].append((arr[3], arr[4]))
```

Prva koordinata u TSV datoteci označava zadnju koordinatu egzona prije introna, a druga koordinata označava prvu koordinatu egzona nakon introna. TSV datoteka tada izgleda na sljedeći način:

```
1 chr18 21444409  21451023
2 chr18 21451023  21452083
3 chr18 21508279  21508591
```

5.1.2 Podudaranje read-ova i događaja

Nakon toga, bilo je potrebno koristiti skriptu koja će uspoređivati SAM datoteku koja sadrži read-ove i generiranu TSV datoteku. Koristili smo C++ i tokenizer koji nam je omogućio kretanje po linijama neke datoteke riječ po riječ.

Prvo moramo definirati funkciju TraverseString koja služi za pristupanje svim elementima niza jedan za drugim korištenjem indeksa. Ona nam je ključna za glavni dio algoritma. Definirana je na sljedeći način:

```
1     void TraverseString(string &str, int &Start_Coord, int &
    End_Coord)
2 {
3     // Find length of given variable
4       int n = str.length();
5       // Create an empty string
6       string word = "";
7
8       // Iterate over the string character by character using
9       // For loop
10      for (int i = 0; i < n; i++) {
11
12          if (str[i] == 'D' or str[i] == 'H'){
13              word ="";
14
15          }
16          else if (str[i] == 'M' or str[i] == 'S' ) {
17
18              // Print word
19              Start_Coord += stoi(word);
20              word = "";
21          }
22          else if (str[i] == 'N'){
23              End_Coord = (Start_Coord + stoi(word)+ 1);
24              return;
25
26          }
27
28
29          else {
30              word += str[i];
31          }
32      }
33    return;
34  }
```

U sljedećem dijelu koda koristili smo Tokenizer, c++ klasu koja služi za pretvaranje linija SAM filea u tokene, po kojima možemo iterirati u kodu. Zatim prolazi kroz stringove i traži podudaranja na temelju imena, te početnih i završnih koordinata događaja alternativnog izrezivanja. Podudarni podaci se zatim spremaju u .txt datoteku.

```
    int main(){
    cout<<"started script ";
    Tokenizer Read("chr18_AS.sam");
    ofstream MyFile("Mapped.txt");
    while(Read.nextLine())
    {
        string Read_Name = Read.getToken();
        string a2 = Read.getToken();
        string Read_Chromosome = Read.getToken();
        int Read_SCoord = stoi(Read.getToken());
        Read.getToken();
        int Read_ECoord = 0;
        string Read_details = Read.getToken();
        TraverseString(Read_details, Read_SCoord, Read_ECoord);
        Tokenizer Event("output18.tsv");

        if (a2 =="16" || a2 =="0")
        {
            while(Event.nextLine())
            {
                string Event_Chromosome = Event.getToken();
                int  Event_SCoord = stoi(Event.getToken());
                int  Event_ECoord = stoi(Event.getToken());
                if (Event_Chromosome == Read_Chromosome &&
                    Event_SCoord == Read_SCoord &&
                    Event_ECoord == Read_ECoord)
                    {
                    MyFile <<Event_Chromosome<<"
                          "<<Event_SCoord<<"
                          "<<Event_ECoord<<"\n";
                    }
            }
        }

    }
    cout<<"done\n";
    MyFile.close();
}
```

5.1.3 Grafički prikaz

Za grafički prikaz podataka koristimo se matplotlib bibliotekom u Pythonu. Slje-

deći korak je iterirati kroz sve linije GTF datoteke kako bismo pregledali događaje na kromosomu, kao i kroz linije datoteke generirane u C++-u, gdje su zabilježena podudaranja read-ova s događajima. Iz ovih datoteka izdvojit ćemo podatke potrebne za grafički prikaz broja podudaranja između događaja i read-ova.

Sljedeći kod prolazi kroz linije GTF datoteke i uzima podatke koji nas zanimaju. Svaku liniju dijelimo na riječi pomoću "word_tokenize()" funkcije. Vodimo računa i o tipu događaja koji može biti, a to su Alternative Acceptor, Alternative Donor ili Exon Skipping.

```
1     Event_read = open("asta.gtf")
2     event_type_array = []
3     chromosomearray = []
4     for row in Event_read:
5      arr = word_tokenize(row)
6      chromosomearray.append([arr[0]] + arr[3:5])
7         for elements, index in zip(arr, range(len(arr))) :
8             if elements == "structure":
9                 event_type_array.append(arr[index+2])
```

Isto radimo i na datoteci koju smo generirali kroz C++ kod i koju smo nazvali "Mapped.txt".

```
1     Mapped_read = open("Mapped18.txt")
2     mapped_array = []
3     for row in Mapped_read:
4         arr = word_tokenize(row)
5         mapped_array.append(int(arr[1]))
```

Nakon toga razvrstavamo događaje prema njihovim vrstama, koje ćemo kasnije prikazati grafički.

```
1     event_names = []
2     event_frequency = [0,0,0]
3     for structure_type in event_type_array:
4         elif structure_type == "1-,2-":
5             event_frequency[0] += 1
6         elif structure_type == "1^,2^":
7             event_frequency[1] += 1
8         elif structure_type == "0,1-2^" or structure_type ==
    "1-2^,0":
9             event_frequency[3] += 1
10
11    print(event_frequency)
```

```
12      event_names = ["Acceptor", "Donor", "Exon"]
```

Radi lakšeg pokretanja algoritma za pojedini kromosom, napravimo trodimenzionalno polje koja sprema potrebne podatke o događajima ovisno o imenu kromosoma.

```
1
2 chromosome_names = []
3     for i in range(0,22):
4         chromosome_names.append("chr" + str(i+1))
5     chromosome_names.append("chrX")
6     chromosome_names.append("chrY")
7
8     chromosome3D = []
9     for name in chromosome_names:
10        temp_array = []
11        for i in range(len(chromosomearray)):
12            if chromosomearray[i][0] == name:
13                event_instance = chromosomearray[i]
14                temp_array.append(event_instance[1:3])
15        chromosome3D.append(temp_array)
```

Nakon toga, odabiremo kromosom za koji želimo prikazati grafički prikaz događaja i relevantnih read-ova u obliku histograma. Također, postavljamo razinu preciznosti za taj histogram.

```
1     chr_id = 18
2     histogram_precision = 250
```

Prije iscrtavanja histograma potrebno je definirati konstante koje označavaju duljine kromosoma.

```
1     plt.rcParams['figure.figsize'] = [6, 4]
2     CHROMOSOME_LENGTHS = [247249719, 242951149, 199501827,
3                           191273063, 180857866, 170899992,
4                           158821424, 146274826, 140273252,
5                           135374737, 134452384, 132349534,
6                           114142980, 106368585, 100338915,
7                           88827254, 78774742, 76117153,
8                           63811651, 62435964, 46944323,
9                           49691432, 154913754, 57772954]
```

Na kraju izrađujemo histogram grafove za kromosome. Evente i readove ćemo iscrtati na istom grafu radi lakše analize podudaranja.

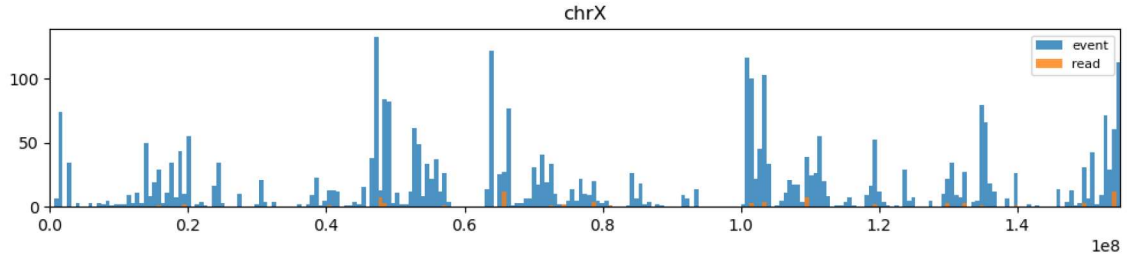
```
    histdata = []
    for events in chromosome3D[chr_id - 1]:
        histdata.append(int(events[0]))

    ys =[]
    for i in range(histogram_precision + 1):
        ys.append(CHROMOSOME_LENGTHS[chr_id - 1] /
                  (histogram_precision + 1)* (i + 1))

    plt.hist(histdata,ys,alpha = 0.8, label = "event")
    plt.hist(mapped_array,ys, alpha = 0.8, label = "read")
    plt.title(chromosome_names[chr_id-1] + "")
    plt.xlim(xmin=0, xmax =CHROMOSOME_LENGTHS[chr_id - 1])
```

S prethodnim kodom naš algoritam dolazi do završetka, a za generiranje histograma za drugi kromosom potrebno je samo proces ponoviti s adektavnim datotekama.

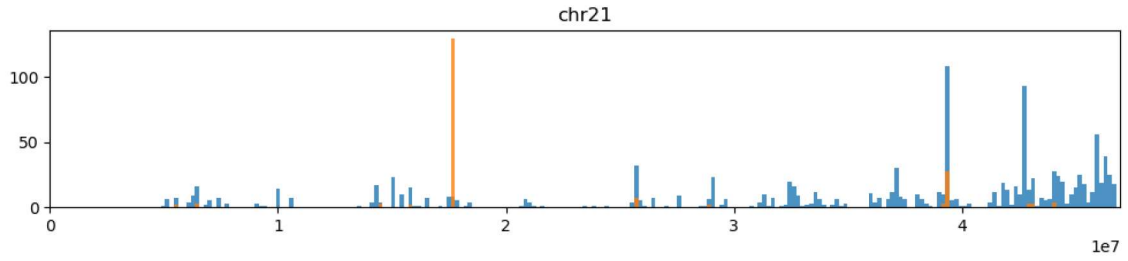
5.2 Rezultati

Slijedeći grafovi su prikaz rezultata prethodno opisanog algoritma.

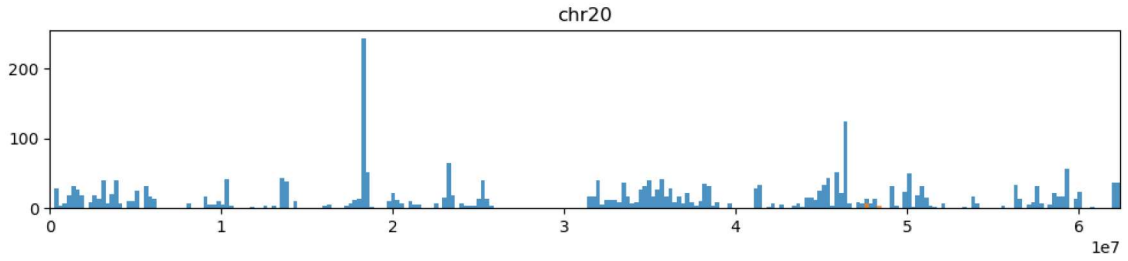
5.2.1 Grafovi mapiranih podataka



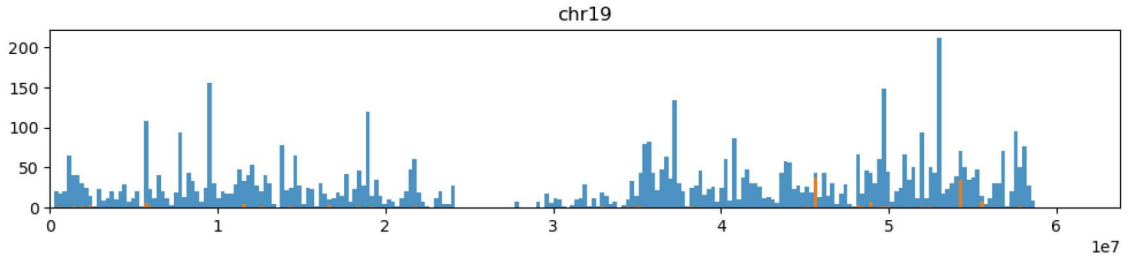
Slika 5.1: Prikaz događaja i mapiranih read-ova na kromosomu X



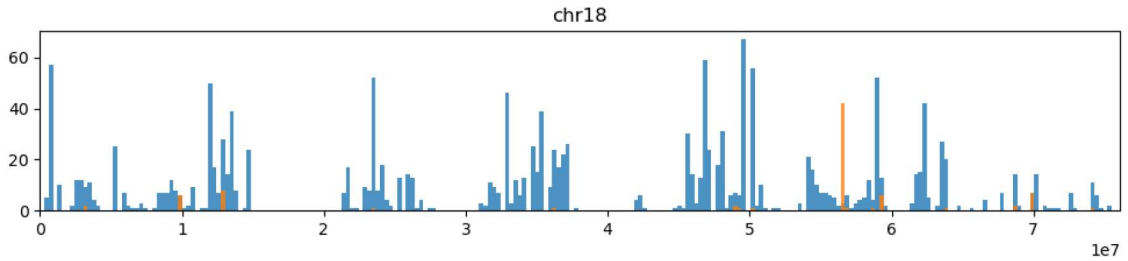
Slika 5.2: Prikaz događaja i mapiranih read-ova na kromosomu 21



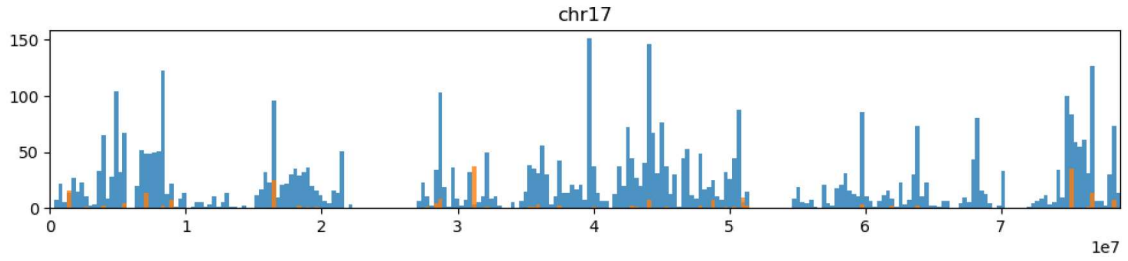
Slika 5.3: Prikaz događaja i mapiranih read-ova na kromosomu 20



Slika 5.4: Prikaz događaja i mapiranih read-ova na kromosomu 19



Slika 5.5: Prikaz događaja i mapiranih read-ova na kromosomu 18



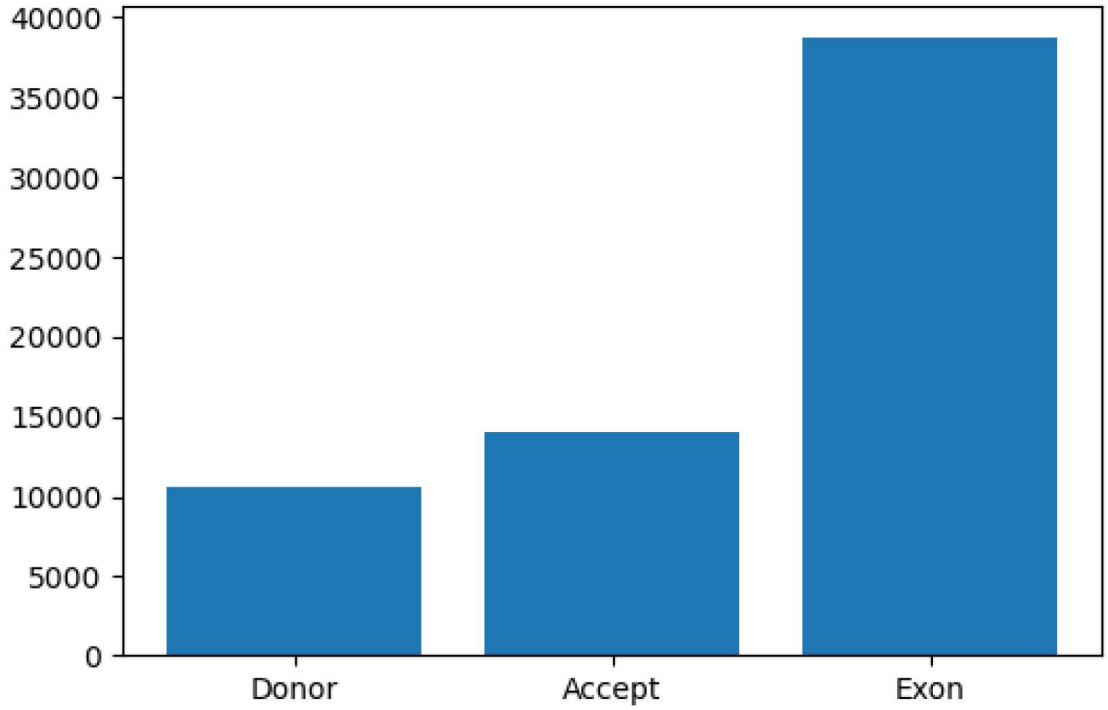
Slika 5.6: Prikaz događaja i mapiranih read-ova na kromosomu 17

Prikazali smo grafove kromosoma X, 21, 20, 19, 18, 17. Grafovi su u omjeru 1 : 10 000 000, s izuzetkom kromosoma X koji je u omjeru 1 : 100 000 000. Y os pokazuje gustoću događaja i readova na određenim dijelovima kromosoma. X os prikazuje duljinu određenog kromosoma.

Grafovi prikazuju na kojim se dijelovima kromosoma prepoznaje najviše događaja (plava boja). Nadalje, grafovi prikazuju gustoću readova (narančasta boja) koje je prethodno opisan algoritam mapirao na AS događaje. Vizualnom analizom grafova zaključujemo da se na kromosomu 19 pojavljuje najviše događaja, ali je malo njih mapirano. To ukazuje na velik broj različitih proteinskih sinteza, s potencijalnim implikacijama na pojavu genskih mutacija i bolesti. S druge strane, kromosom 21 ima značajnu gustoću mapiranih readova oko koordinate 17M. Najveći kromosom koji smo obradili je kromosom X, a najmanji kromosom 21.

5.2.2 Tipovi događaja

Prethodno smo spomenuli različite tipove AS događaja, a sljedeći graf će prikazati statistiku istih u našem genomu.



Slika 5.7: Tipovi događaja na kromosomima

Primjećujemo da se najčešće pojavljuje Exon Skipping tip događaja, a u manjoj proporciji su Alternative Acceptor i Alternative Donor.

6 | Zaključak

U ovom radu prikazan je značaj alternativnog izrezivanja kao ključnog mehanizma za povećanje genetske i proteinske raznolikosti organizama. Kroz primjenu bioinformatičkih alata poput STAR-a i AStalaviste, omogućena je detaljna analiza alternativnog izrezivanja na temelju genetskih podataka. Implementirane metode omogućile su detekciju i vizualizaciju splicing događaja, čime je pružen uvid u složene procese koji se odvijaju unutar genoma. Rezultati ovog istraživanja ukazuju na mogućnosti daljnjih analiza koje bi mogle povezati anomalije u splicingu s razvojem određenih bolesti, poput autizma. Konkretne zaključke ovih podataka donose stručnjaci područja bioinformatike i genetike kroz daljnja istraživanja.

Daljnjim istraživanjem i optimizacijom, proces mapiranja podataka mogao bi se značajno ubrzati i unaprijediti, što bi omogućilo stručnjacima da brže i preciznije donose zaključke na temelju dobivenih podataka. Time bi se otvorile dodatne mogućnosti za otkrivanje novih bioloških uvida, što bi doprinijelo boljim razumijevanjem alternativnog izrezivanja i njegovih uloga u zdravlju i bolesti.

Literatura

[1] Alexander Dobin, "STAR Manual", 2023.

[2] Dobin A, Davis CA, Schlesinger F, Drenkow J, Zaleski C, Jha S, Batut P, Chaisson M, Gingeras TR. STAR: ultrafast universal RNA-seq aligner. Bioinformatics. 2013 Jan 1;29(1):15-21.

[3] Sylvain Foissac, Michael Sammeth, "ASTALAVISTA: dynamic and flexible analysis of alternative splicing events in custom gene datasets", *Nucleic Acids Research*, Vol. 35, April 14, 2007

[4] Jaganathan, K., Panagiotopoulou, S. K., McRae, J. F., Darbandi, S. F., Knowles, D., Li, Y. I., Kosmicki, J. A., Arbelaez, J., Cui, W., Schwartz, G. B., Chow, E. D., Kanterakis, E., Gao, H., Kia, A., Batzoglou, S., Sanders, S. J., and Farh, K. K., "Predicting splicing from primary sequence with deep learning", *Cell*, 2009., 176(3), pp. 535–548.

[5] Jill Roughan, "Your Essential Guide to Different File Formats in Bioinformatics", *Form Bio*, September 27, 2022

[6] Bird, Steven, Edward Loper and Ewan Klein, "Natural Language Processing with Python", *O'Reilly Media Inc*, 2009.

[7] J. D. Hunter, "Matplotlib: A 2D Graphics Environment", *Computing in Science and Engineering, vol. 9*, no. 3, pp. 90-95, 2007.

Sažetak: Ovaj rad se fokusira na detekciju alternativnog izrezivanja pomoću bioinformatičkih alata kao što su STAR i AStalavista. Pruža teoretsku pozadinu o alternativnom izrezivanju i objašnjava kako se ovi alati mogu koristiti za analizu genetskih podataka. Rad također uključuje implementaciju prilagođenih skripti za usporedbu poravnavanja očitavanja s događajima spajanja i vizualizaciju rezultata putem histogramima.

Ključne riječi: alternativno izrezivanje, STAR, AStalavista, Python, C++

Detection of alternative splicing

Abstract: This paper focuses on the detection of alternative splicing using bioinformatics tools such as STAR and AStalavista. It provides a theoretical background on alternative splicing and explains how these tools can be used for analyzing genetic data. The paper also includes the implementation of custom scripts for comparing read alignments with splicing events and visualizing the results through histograms.

Keywords: alternative splicing, STAR, AStalavista, Python, C++