Detekcija alternativnog izrezivanja

Karapetrić, Matej

Sveučilište Josipa Jurja Strossmayera u Osijeku
Fakultet primijenjene matematike i informatike
Sveučilišni prijediplomski studij Matematika i računarstvo

Matej Karapetrić

Detekcija alternativnog izrezivanja

Završni rad

Osijek, 2023.

Sveučilište Josipa Jurja Strossmayera u Osijeku
Fakultet primijenjene matematike i informatike
Sveučilišni prijediplomski studij Matematika i računarstvo

Matej Karapetrić

Detekcija alternativnog izrezivanja

Završni rad

Mentor: izv. prof. dr. sc. Domagoj Matijević
Komentor: dr. sc. Luka Borozan

Osijek, 2023.

Sažetak: U ovom ćemo radu pokazati kako se izvršava proces detekcije alternativnog izrezivanja kroz programe STAR, AStalavista i dodane prikladne kodove te ga detaljno objasniti. Najprije ćemo pojasniti ideju alternativnog izrezivanja. Nakon toga pokazat ćemo sve korake kako proučavati alternativno izrezivanje kroz bioinformatiku te ih komentirati i obrazložiti.

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

Detection of alternative splicing

Abstract: In this paper, we will show how the alternative splicing process is performed through the programs STAR, AStalavista and the added appropriate codes, and explain it in detail. First, we'll explain the idea of alternative splicing. After that, we will show all the steps on how to study alternative splicing through bioinformatics and comment and explain them.

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

Sadržaj

1. Uvod

Bioinformatika je grana znanosti koja usko povezuje biologiju i računarstvo, a ubrzano se razvija zadnja dva desetljeća. Koristi se za računanje, analizu, prikupljanje i tumačenje bioloških podataka. Uvelike pomaže i omogućuje rad na biološkim procesima koji zahtjevaju obradu velikih skupova podataka, kao što je alternativno izrezivanje.

Alternativno izrezivanje (alternative splicing) je proces u kojem se egzoni spajaju u različite kombinacije koje rezultiraju različitim izražajem gena. Egzoni su dijelovi DNA koji sadrže kodirajuću poruku i prenose ju ostalim dijelovima stanice. Alternativno izrezivanje zaokupilo je pozornost znanstvenicima tog područja kao važan mehanizam za modulaciju sadržaja gena i proteina u stanici. Potencijalno bi moglo objasniti složenost proteinskog repertoara tijekom evolucije, a greške u izrezivanju odgovorne su za složene bolesti poput raka.

Iz tog razloga postoje STAR i AStalavista pomoću kojih poravnavamo i analiziramo alternativna izrezivanja i njihove rezultate. Proučavajući dobivene rezultate biolozi mogu istraživati genome pobliže i pokušati pronaći poveznicu rezultata s bolestima.

U ovom radu ćemo proučiti proces alternativnog izrezivanja i njegove detekcije nad genomskim podacima pacijenta s autizmom. Najprije ćemo pojasniti pozadinsku teoriju iza alternativnog izrezivanja, ali i bioinformatičkih pojmova. Nakon toga ćemo približiti funkcionalnost programa STAR i AStalaviste. U konačnici ćemo pokazati naš pristup alternativnom izrezivanju i grafički ćemo prikazati rezultate.

2. Teorija

2.1. RNA i alternativno izrezivanje

DNA je molekula osnovna tvar svake stanice živog bića. Nalazi se u jezgrici, odvojena od citoplazme i drugih organela te je zbog svoje iznimne važnosti okružena dvostrukom jezgrinom ovojnicom. Zbog elektične nabijenosti skupina koje ju čine, a i same veličine molekule, DNA nikad ne može napustiti jezgricu. Specifične je građe dvostruke uzvojnice komplementarno spojenih parova nukleotida koje čine molekule šećera deoksiriboze, fosfatne skupine i purinske ili pirimidinske jedinice. Purini i pirimidini specijalizirane su proteinske molekule specifične sposobnosti stvaranja komplementarnih parova. Komplementarnost se izražava spajanjem dušičnih baza adenina i timina te gvanina i citozina (podvrste purina i pirimidina). Očuvanje pravilne forme DNA nužno je kako bi organizmi funkcionirali u cijelosti; redoslijed pojavljivanja komplementarnih parova dušičnih baza u DNA određuje gene koji su najvažniji proteini gradivne i funkcionalne uloge u organizmu.

RNA je jednolančana vrsta nukleinske kiseline lokalizirana u više odjeljaka stanice te služi različitim funkcijama koje djeluju prema istom cilju; sinteza bjelančevina. RNA nastaje unutar jezgre prepisivanjem kodirajućeg lanca DNA na osnovi komplementarnog sparivanja dušičnih baza. Unutarjezgreni oblik RNA koji nastaje prepisivanjem naziva se pre-mRNA. Najvažnija je osobina mRNA sposobnost izlaska iz jezgre u citoplazmu, što čini inicijalni korak izlaska genskog zapisa od DNA prema ostalim dijelovima stanice koji sudjeluju u sintezi proteina.

Prije izlaska iz jezgre, pre-mRNA mora proći proces sazrijevanja koji nazivamo izrezivanje. RNA izrezivanje (RNA splicing) je proces izrezivanja introna (ne kodirajućih dijelova DNA) i spajanja preostalih egzona (kodirajućih dijelova DNA) u zrelu mRNA. U ovom ključnom procesu sudjeluju brojni enzimi koji omogućuju formiranje mRNA koja odlazi do ribosoma gdje se nastavlja sinteza proteina.

Specifična građa svakog introna ono je što omogućuje vezanje enzimskih kompleksa koji će svojim djelovanjem odsjeći krajeve introna i tako ih razdvojiti od okolnih egzona. Područje na kojem se događa izrezivanje zove se splice junction. Konačno zrela mRNA građena je samo od egzona.

Opisani proces centralna je teorija molekularne biologije koja govori o lokalizaciji gena i načinu njihove ekspresije u proteine. RNA izrezivanje proces je nužan za ekspresiju gena, no uz njega vežemo i pojam alternativnog izrezivanja (alternative splicing) koji rezultira stvaranjem više oblika mRNA (izoformi) nastalih različitim kombinacijama izrezivanja introna i spajanja egzona. Rezultat alternativnog izrezivanja su mRNA koje kodiraju različite proteine, a porijeklom su od istog gena. Alternativno je izrezivanje normalan proces stanica eukariota, a konačna mu je svrha postizanje bioraznolikosti samog organizma i vrsta.

Postoje razni događaji koji se mogu odvijati tijekom alternativnog izrezivanja. Najčešći su Exon Skipping, Alternative Donor i Alternative Acceptor. Exon Skipping je oblik izrezivanja RNA koji uzrokuje preskakanje preko neispravnih ili neusklađenih dijelova genetskog koda, odnosno egzona. Alternative Donor je događaj pri kojem početni kraj introna zamjenjuje završni kraj introna uz granicu uzvodnog egzona. Alternative Acceptor je događaj pri kojem završni kraj introna zamjenjuje početni kraj introna uz granicu nizvodnog egzona.

2.2. Često korišteni formati u bioinformatici

Kako su biološki podaci postali digitalni, s ogromnim količinama sekvenci podataka, pojavilo se nekoliko različitih vrsta datoteka i formata. U nadolezećem tekstu spominjat ćemo više vrsta datoteka i formata koji se koriste u bioinformatici koje bismo prvotno trebali objasniti.

FASTA format - najjednostavniji je način predstavljanja nukleinskih kiselina proteinskih sekvenci pomoću jednoslovnih kodova za nukleotide. Može sadražavati jednu ili više sekvenci proteina

GTF (gene transfer format) format - opisuje različite elemente sekvence koji čine gen i standardni je način označavanja genoma. Definira isključivo značajke gena, uključujući transkripte, regulatorne regije, neprevedene regije, egzone, introne i kodirajuće sekvence.

SAM (sequence alignment/MAP) format - tekstualne datoteke odvojene tabulatorima koje se koriste za pohranjivanje podataka o usklađivanju sekvence. SAM datoteke sadrže zaglavlje i tijelo. Zaglavlje pohranjuje informacije o sekvencama, ispred kojih stoji simbol "@". Tijelo sadrži informacije o tome kako se svaka sekvenca slaže s određenom referentnom sekvencom.

3. STAR

Točno usklađivanje izrezanih transkripta s referentnim genomom kritičan je zadatak u modernoj genomici. Razumijevanje organizacije gena i njihovih alternativnih spojenih izoformi bitno je za razotkrivanje složenosti regulacije i funkcije gena. Pojava visokoučinkovitih tehnologija sekvenciranja revolucionirala je istraživanje genomike omogućivši sveobuhvatnu karakterizaciju transkriptoma, uključujući identifikaciju događaja kod alternativnih izrezivanja.

Jedna od takvih tehnologija je STAR (Spliced Transcripts Alignment to a Reference). STAR je najsuvremeniji alat za usklađivanje RNA sekvenci. RNA sekvence su postale osnova modernih genomskih istraživanja, omogućujući sveobuhvatno detaljno proučavanje gena. Za isčitavanje podataka skrivenih u RNA sekvencama bitno je točno usklađivanje read-ova sekvenciranja s referentnim genomom ili transkriptom. STAR je u tom području vrlo učinkovit i točan.

STAR koristi proces od dva koraka: indeksiranje genoma i mapiranje genoma.

3.1. Indeksiranje genoma

STAR konstruira indeks genoma koji optimalno predstavlja značajke referentnog genoma, uključujući splice junction-e, granice egzona i kontekst sekvence. Ovaj korak poboljšava točnost poravnanja učinkovitim lociranjem potencijalnih mjesta spajanja. Mjesta spajanja su specifične sekvence DNA koje se nalaze na granicama egzona i introna u genu. Bitna su za uklanjanje introna i spajanje egzona kako bi se stvorila zrela molekula mRNA koja se može prevesti u protein.

U ovom koraku je potrebno STAR-u dobaviti referentne sekvence genoma (FASTA datoteke) i anotacije (GTF datoteke), iz kojih STAR generira indekse genoma. Indeksi genoma spremaju se na disk i potrebno ih je generirati samo jednom za svaku kombinaciju genom/anotacija. Oni omogućuju da se suzi područje pretraživanja sekvence unutar genoma, štedeći vrijeme i memoriju. Ovaj korak je potreban jer će generirati transformiranu verziju genoma koja omogućuje STAR-u da učinkovito napravi drugi korak, odnosno mapira sekvence na njega.

Naredba koju smo koristili 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.

3.2. Mapiranje genoma

Tijekom poravnanja, STAR učinkovito poravnava read-ove sekvenciranja uzimajući u obzir poznate splice junction-e i omogućuje otkrivanje novih junction-a. Pristup mapiranju genoma sastoji se od dva koraka. Najprije se usklađuju read-ovi s genomom, a zatim se pojašnjavaju poravnanja preslikavajući ih na indeks transkriptoma.

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

```
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.

3.3. Output

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

Poravnani read-ovi se pohranjuju u SAM formatu ili u njegovom ekvivalentnom binarnom BAM formatu. Takve datoteke sadrže detaljne informacije o svakom poravnanju read-a s referentnim genomom ili transkriptomom. Datoteka sadrži podatke kao što su sekvence

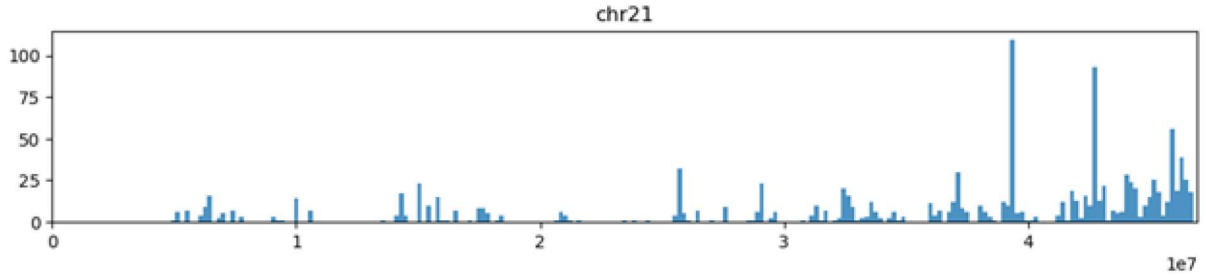
mapiranih read-ova, njihove pozicije u genomu, kvaliteta mapiranja i opcionalne oznake za dodatne informacije.

4. AStalavista

Točna identifikacija i analiza splice junction-a ključna je za razumijevanje složenosti gena. Program AStalavista (Alternative Splicing Traversing Analysis) je bioinformatički alat koji olakšava okrivanje i karakterizaciju splice junction-a u podacima RNA sekvenci. Uz rješavanje složenih analiza splice junction-a, pruža istraživačima vrijedne uvide u događaje alternativnih izrezivanja.

Radi tako da uspoređujući sve dane transkripte, otkriva varijacije u njihovoj strukturi izrezivanja i identificira sve događaje alternativnog izrezivanja (kao što su Exon Skipping, Alternative Donor, itd.) dodjeljivanjem svakom od njih kod. Izlazni dokument je u posebnom GTF formatu.

Primjer vizualne reprezentacije podataka danih AStalavistom:



Slika 1: Prikaz događaja na kromosomu 21

Prethodna slika prikazuje događaje koje je AStalavista prepoznala da su se dogodili tijekom alternativnog izrezivanja na kromosomu 21.

5. Eksperiment

Nakon što smo se upoznali sa teorijom i programima koji su se pokretali tijekom procesa, trebamo objasniti kod koji smo implementirali kako bismo dobili rezultate te ih prikazati.

Cilj nam je prvo implementirati kod koji će provjeriti podudaranje read-ova (generiranih STAR-om) i događaja (generiranih AStalavistom) koje se događa ukoliko se read mapira preko introna koji određuje događaj. Zatim ta podudaranja zapisati u datoteku. Iz tog rješenja i već dobivene GTF datoteke uzimamo bitne podatke za frekvencije određenih događaja na kromosomima. Konačno, želimo grafički prikazati navedene frekvencije. Cijeli proces ćemo prikazati kako radi na kromosomu 21.

5.1. Implementacija

5.1.1. Generiranje TSV datoteke

Prvotno smo napravili kod u Pythonu koji generira TSV (Tab Separated Values) datoteku koju ćemo koristiti pri provjeri podudaranja read-ova i događaja. U ovu TSV datoteku stavljamo samo događaje za neki određeni kromosom.

```
with open('output21.tsv', 'wt') as out_file:
tsv_writer = csv.writer(out_file, delimiter='\t')
Event_read = open("asta.gtf")
for row in Event_read:
        arr = word_tokenize(row)
        if (arr[3], arr[4]) not in dict[arr[0]] and arr[0] == "chr21":
            tsv_writer.writerow([arr[0], arr[3], arr[4]])
            dict[arr[0]].append((arr[3], arr[4]))
```

U prethodnom dijelu koda uzimamo koordinate događaja koje smo dobili u GTF datoteci. 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:

```
chr21        5033408  5034575
chr21        5057143  5061713
chr21        5060445  5063360
```

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.

Tokenizer smo koristili u sljedećem dijelu koda kako bismo SAM i TSV datoteku rastavili

na nama bitne podatke koje trebamo usporediti. Dodatno, sljedeći dio koda pravi zaključak usporedbe ovisno o imenu, početnoj i završnoj koordinati na kromosomu. Podatke koji se podudaraju sprema u .txt datoteku.

```
int main(){
cout<<"started script ";
Tokenizer Read("chr21_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("output21.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();
}
```

U prethodnom kodu koristimo funkciju "TraverseString" koja služi za pristupanje svim elementima niza jedan za drugim korištenjem indeksa i glasi:

```
    void TraverseString(string &str, int &Start_Coord, int &End_Coord)
{
    // Find length of given variable
      int n = str.length();
      // Create an empty string
      string word = "";

      // Iterate over the string character by character using
      // For loop
      for (int i = 0; i < n; i++) {

          if (str[i] == 'D' or str[i] == 'H'){
              word ="";

          }
          else if (str[i] == 'M' or str[i] == 'S' ) {

              // Print word
              Start_Coord += stoi(word);
              word = "";
          }
          else if (str[i] == 'N'){
              End_Coord = (Start_Coord + stoi(word)+ 1);
              return;

          }


          else {
              word += str[i];
          }
      }
    return;
  }
```

5.1.3. Grafički prikaz

Ponovno koristimo Python za nastavak implementacije. Sljedeći korak nam je proći kroz sve linije GTF datoteke, odnosno kroz sve događaje na kromosomu te kroz sve linije datoteke koju smo generirali u C++, odnosno kroz sva podudaranja read-ova s događajima na kromosomu. Iz tih datoteka uzimamo podatke koji nam služe da grafički prikažemo same datoteke, to jest prikažemo graf događaja i read-ova koji se podudaraju s nekim događajem u određenoj količini.

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.

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

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

```
Mapped_read = open("Mapped21.txt")
mapped_array = []
for row in Mapped_read:
    arr = word_tokenize(row)
    mapped_array.append(int(arr[1]))
```

Sada pravimo statistiku frekvencija određenih tipova događaja na kromosomima, kao što smo naveli prije te pripremamo za grafički prikaz.

```
event_names = []
event_frequency = [0,0,0]
for structure_type in event_type_array:
    elif structure_type == "1-,2-":
        event_frequency[0] += 1
    elif structure_type == "1^,2^":
        event_frequency[1] += 1
    elif structure_type == "0,1-2^" or structure_type == "1-2^,0":
        event_frequency[3] += 1

print(event_frequency)
event_names = ["Acceptor", "Donor", "Exon"]
```

Kada smo pripremili, jednostavno na sljedeći način prikazujemo rezultate.

```
plt.bar(event_names, event_frequency)
```

Nakon toga, dajemo svakom kromosomu pripadno ime kako bismo se mogli pozivati na njega pri pokretanju koda za određeni kromosom.

```
chromosome_names = []
```

```
for i in range(0,22):
    chromosome_names.append("chr" + str(i+1))

chromosome_names.append("chrX")
chromosome_names.append("chrY")
```

Radi lakšeg snalaženja u podacima, napravimo 3D matricu koja sprema potrebne podatke o događajima ovisno o imenu kromosoma.

```
chromosome3D = []
for name in chromosome_names:
    temp_array = []
    for i in range(len(chromosomearray)):
        if chromosomearray[i][0] == name:
            event_instance = chromosomearray[i]
            temp_array.append(event_instance[1:3])
    chromosome3D.append(temp_array)
```

Onda biramo kromosom čiji grafički prikaz događaja i relevantnih read-ova želimo prikazati u obliku histograma. Također, izabiremo preciznost istog histograma.

```
chr_id = 21
histogram_precision = 250
```

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

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

Napokon, kreiramo grafove kromosoma u obliku histograma. Svaki kromosom ćemo iscrtati na istoj osi kako bismo ih mogli vizualno uspoređivati.

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

ys =[]
```

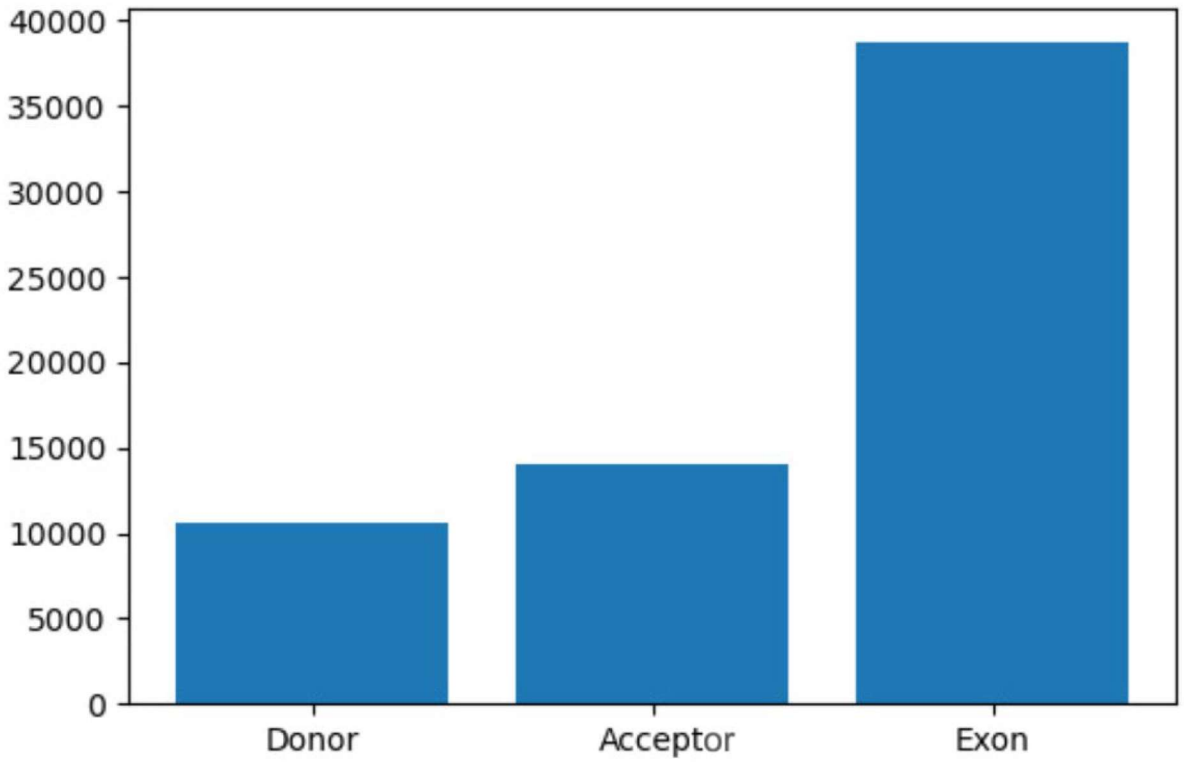
```
for i in range(histogram_precision + 1):
    ys.append(CHROMOSOME_LENGTHS[chr_id - 1] /
              (histogram_precision + 1)* (i + 1))

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])
```

Prethodni kod je završni dio koda koji koristimo za grafički prikaz te i krajnji rezultat našeg eksperimenta.

5.2. Rezultati

Rezultati su sadržani u obliku grafova. Prvi graf u obliku barplota nam prikazuje podijeljenost događaja među kromosomima koje smo proučavali. Ostali grafovi, koji su ujedno i konačno rješenje cijelog procesa, su u obliku histograma. Sadrže podatke o frekvencijama svih događaja na kromosomima koje smo odabrali te podatke o frekvencijama read-ova koji su se podudarili s navedenim događajima.

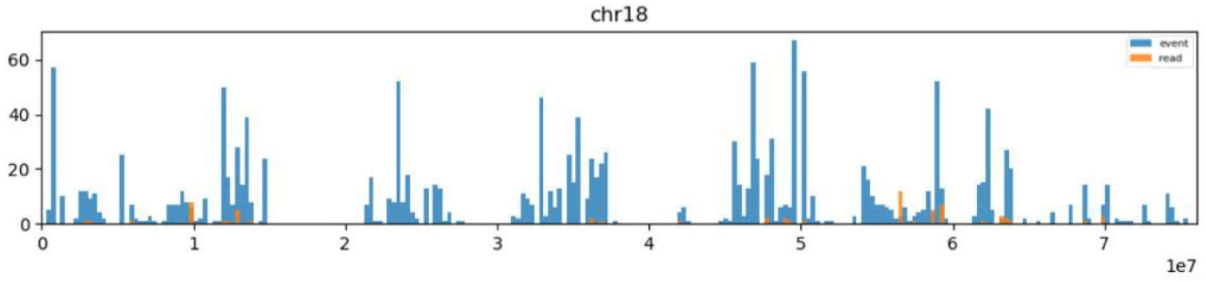
5.2.1. Tipovi događaja



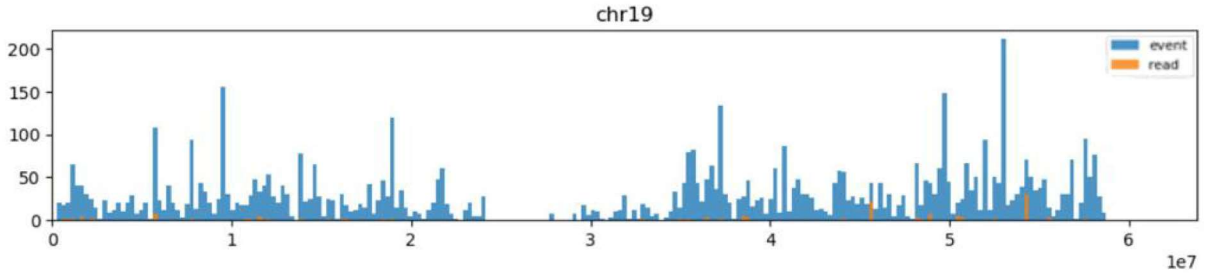
Slika 2: Tipovi događaja na kromosomima

Prema grafu 2 možemo zaključiti da su u velikoj većini detektirani Exon Skipping događaji. Alternative Acceptor i Alternative Donor događaji su reprezentirani u manjoj i sličnijoj mjeri. Jednako tako, broj read-ova se uveliko više podudara s Exon Skipping događajima nego sa ostala dva.

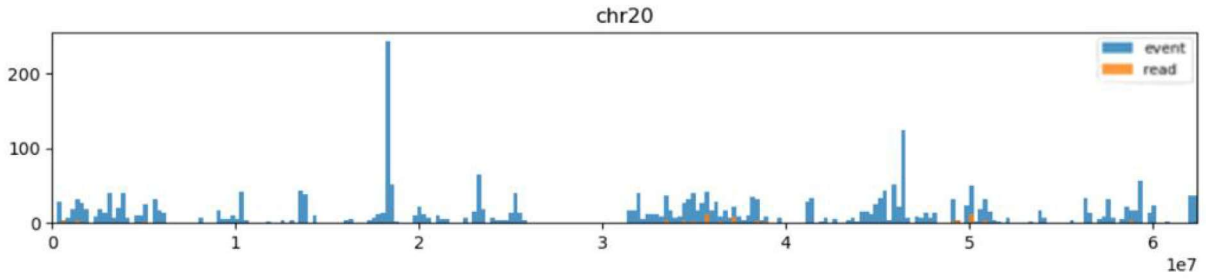
5.2.2. Grafovi događaja i read-ova



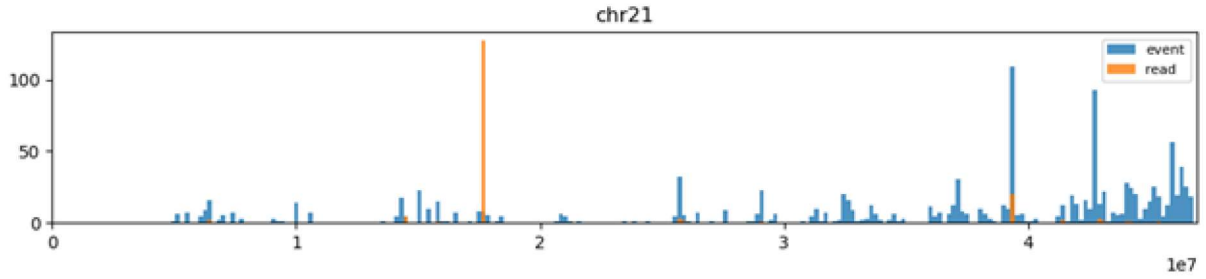
Slika 3: Prikaz događaja i pripadnih read-ova na kromosomu 18



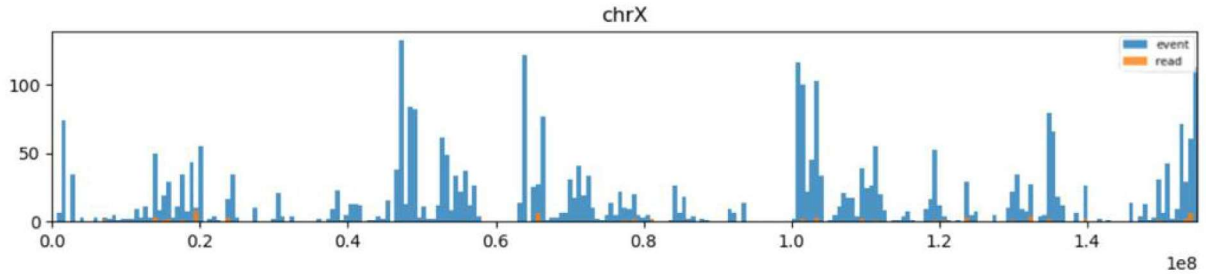
Slika 4: Prikaz događaja i pripadnih read-ova na kromosomu 19



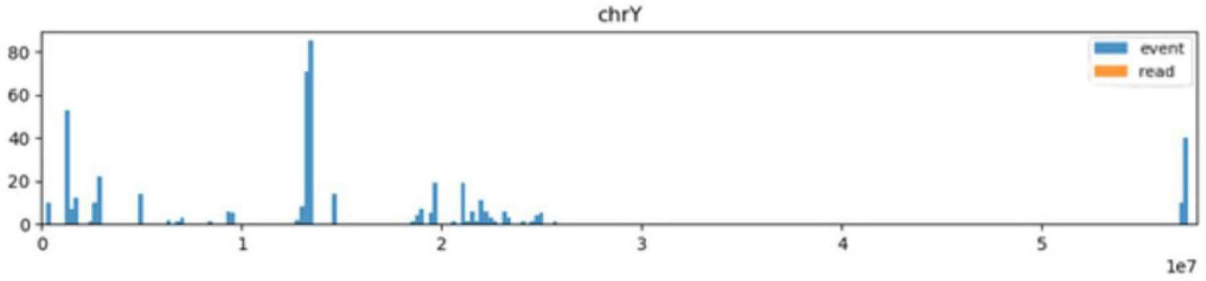
Slika 5: Prikaz događaja i pripadnih read-ova na kromosomu 20



Slika 6: Prikaz događaja i pripadnih read-ova na kromosomu 21



Slika 7: Prikaz događaja i pripadnih read-ova na kromosomu X



Slika 8: Prikaz događaja i pripadnih read-ova na kromosomu Y

Prikazali smo grafove za kromosome 18, 19, 20, 21, X i Y. X os grafa prikazuje duljinu kromosoma u omjeru 1 : 10 milijuna, a y os grafa prikazuje broj događaja ili read-ova. Graf pokazuje točno na kojem području kromosoma se odvija koliki broj događaja ili read-ova.

Na slikama se nalaze prikazi frekvencije događaja (plava boja) i kako su raspoređeni na kromosomu. Također, nalaze se i read-ovi koji su poravnati s tim događajima (narančasta boja). Možemo primijetiti kako je kromosom 18 na slici 3 najduži, ali da kromosom 19 na slici 4 i kromosom X na slici 7 imaju puno gušća područja događaja. Na slici 5 vidimo da kromosom 20 sadrži najviše događaja na jednom području, dok smo na kromosomu 21 koji je prikazan na slici 6 pronašli najviše read-ova koje smo poravnali s pripadnim događajima.

Glavni cilj prikaza ovih podataka pomoću grafova je prikaz podudarnih read-ova i njihova usporedba s događajima na istom području. Read-ovi koji se ističu smatraju se anomalijama i mogu ukazivati na razne biološke procese. Istraživanja i zaključke vezane uz grafove koje smo prikazali donose stručni istraživači tog znanstvenog područja.

6. Zaključak

Pokazali smo cijeli proces detekcije alternativnog izrezivanja koristeći programe STAR i AStalavista te naš kod implementiran u programima Python i C++. Predstavili smo i teoriju iza alternativnog izrezivanja i bioinformatičkih datoteka za bolje razumijevanje procesa. Cijeli proces smo provodili na kromosomima pacijenta s autizmom te prikazali na određenom setu kromosoma.

Prepoznajemo nekoliko novih smjerova kojima bismo mogli krenuti u budućim istraživanjima ove teme. Python implementacija može biti optimalnija ili čak skroz reimplementirana u C++ kod što bi značajno ubrzalo proces prolaženja kroz podatke kromosoma.

Literatura

[1] Geoffrey M. Cooper, "The Cell: A Molecular Approach 8^{th} Edition", *Oxford University Press*, 2019.

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

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

[5] 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.

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