Sveučilište J. J. Strossmayera u Osijeku

Fakultet primijenjene matematike i informatike

Sveučilišni preddiplomski studij Matematike i računarstva

Detekcija novog alternativnog izrezivanja

Jurica Jurčević, Matej Karapetrić

Mentor: Luka Borozan

Osijek, 2023.

Sadržaj

UVOD

Bioinformatika je grana znanosti koja usko povezuje biologiju i računarstvo, a ubrzano se razvija zadnja dva desetljeća. Uvelike se koristi u poljima koja zahtjevaju obrade velikih skupova podataka, kao što je i u našem slučaju. Koristi se i upravo kod proučavanja biološkog procesa alternativnog izrezivanja čime će se naš završni projekt baviti. **Alternativno izrezivanje** (alternative splicing) je proces u kojem se egzoni spajaju u različite kombinacije koje rezultiraju različitim izražajem gena. Cilj našeg projekta je analizirati takva izrezivanja na stvarnim uzorcima pacijenata s autizmom. Analizu provodimo uspoređivanjem kratkih lanaca nukleotida dobivenih tehnikom sekvenciranja nove generacije (next generation sequencing) koristeći alate STAR i Astalavista te algoritam koji smo napravili u programskom jeziku Python.

U sadržaju ćemo proći kroz proces našeg projekta i objasniti svaki korak detaljno kako bismo ga što više približili čitatelju.

POJMOVNIK

Nukleotid – gradivna jedinica DNA; sastoji se od šećera riboze ili deoksiriboze, fosforne kiseline i dušične baze

Egzoni – lanci nukleotida u DNA ili RNA koji su sačuvani u izrezivanju

Introni – lanci nukleotida u DNA ili RNA koji ne kodiraju proteine i uklanjaju se tijekom izrezivanja

Genom – sveukupna DNA nekog organizma; obuhvaća sve gene, ali i ostale nizove nukleotida

Poravnanja - način za usporedbu srodnih sekvenci DNA ili proteina

FASTA datoteka - tekstualni format za predstavljanje nukleotidnih nizova ili proteinskih sekvenci, u kojem su nukleotidi ili proteini predstavljeni pomoću kodova od jednog slova

FASTQ datoteka - tekstualni format za pohranjivanje biološke sekvence i njezinih odgovarajućih ocjena kvalitete

GTF datoteka – anotacija; format datoteke koji se koristi za čuvanje informacija o strukturi gena

Jezgra – mozak CPU-a; prima upute i izvodi izračune ili operacije kako bi zadovoljio te upute

Niti – virtualne komponente ili kodovi koji dijele fizičku jezgru CPU-a u više virtualnih jezgri; jedna CPU jezgra može imati do 2 niti po jezgri

Spojevi (**'Splice junctions'**) – spojevi egzona i introna na kojima se odvija izrezivanje

Read - izvedeni niz parova baza koji odgovara cijelom ili dijelu jednog fragmenta DNK-a

SAM datoteka – tekstualni format koji sadrži informacije o poravnanju različitih sekvenci koje su mapirane u odnosu na referentne sekvence

BAM datoteka - sadrži iste informacije kao SAM datoteka, osim što je u binarnom formatu

STAR

STAR (kratko za 'Spliced Transcripts Alignment to a Reference') je algoritam koji je osmišljen kako bi se posebno bavio mnogim izazovima mapiranja podataka sekvenciranih proteina. Nama najbitniji izazov koji STAR savladava je mapiranje proteinskih sekvenci pune duljine.

Postavljanje STAR-a:

STAR se može instalirati i pokretati preko Linux ili MAC operativnog sustava. Mi smo koristili Linux i Ubuntu okruženje za Linux. STAR se može instalirati s GitHub-a: https://github.com/alexdobin/STAR/releases ili preko glavnog repozitorija https://github.com/alexdobin/STAR. Također, instalacija kako smo ju mi napravili može biti na sljedeći način preko terminala :

- wget https://github.com/alexdobin/STAR/archive/2.7.11a.tar.gz
- tar -xzf 2.7.11a.tar.gz
- cd STAR-2.7.11a

Korištenje STAR-a:

Nakon instalacije, s podacima u STAR-u smo radili uz pomoć STAR manuala (https://github.com/alexdobin/STAR/blob/master/doc/STARmanual.pdf).

Prvo je potrebno instalirati gcc okruženje za rad u STAR-u, ovisno o Linux okruženju.

STAR naredbe imaju sljedeći format:

STAR --opcija1-ime opcija1-vrijednost(i)--opcija2-ime opcija2-vrijednost(i) ...

Tijek rada u STAR-u se općenito sastoji od dva koraka:

1. **Generiranje datoteka s indeksima genoma.**
   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.

   Naredbe koje smo koristili u ovom koraku su:
   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 spojeve ('Splice junctions') 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 spoja koji će se koristiti u izradi baze podataka spojeva. 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.

2. **<u>Mapiranje genoma.</u>**
   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, spajanje spojeva, nemapirani read-ovi, itd.

   Naredbe koje smo koristili u ovom koraku su:
   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.

STAR proizvodi više izlaznih datoteka:

- Log datoteke
- SAM datoteke
- Nerazvrstane i razvrstane po koordinatama BAM datoteke
- Nemapirani read-ovi
- Spojevi

U našem projektu koristimo jedino SAM datoteke iz outputa, ali pojasnit ćemo sve izlazne datoteke za bolje razumijevanje programa.

Log datoteke:

Log.out: glavna datoteka dnevnika s puno detaljnih informacija o izvršenoj radnji. Ova je datoteka najkorisnija za rješavanje problema i otklanjanje pogrešaka.

Log.progress.out: izvješćuje statistiku napretka posla, kao što je broj obrađenih čitanja, postotak mapiranih čitanja, itd. Ažurira se u intervalima od 1 minute.

Log.final.out: sažetak statistike mapiranja nakon što je posao mapiranja završen, vrlo koristan za kontrolu kvalitete. Statistika se izračunava za svako očitavanje i zatim zbraja ili uzima prosjek za sva očitavanja.

SAM datoteke:

Aligned.out.sam: poravnanja u standardnom SAM formatu.

Nerazvrstane i razvrstane po koordinatama BAM datoteke:

STAR može ispisati poravnanja izravno u binarnom BAM formatu, čime se štedi vrijeme na pretvaranju SAM datoteka u BAM. Također, može sortirati BAM datoteke po koordinatama.

Nemapirani read-ovi:

Read-ovi koji nisu uspjeli biti mapirani. Razlog neuspjelog mapiranja može biti jedan od idućih.

Spojevi:

SJ.out.tab je nama najpotrebnija datoteka koja sadrži sažete spojeve visoke pouzdanosti u formatu u kojem tab razdjeljuje informacije. Treba imati na umu da STAR definira početak/kraj spoja kao introničke baze, dok ih mnogi drugi softveri definiraju kao egzoničke baze. Stupci (informacije) odvojeni tab-ovima imaju sljedeća značenja:

stupac 1: kromosom

stupac 2: prva baza introna (na bazi 1)

stupac 3: zadnja baza introna (na bazi 1)

stupac 4: nit (0: nedefiniran, 1: +, 2: -)

stupac 5: motiv introna: 0: nekanonski; 1: GT/AG, 2: CT/AC, 3: GC/AG, 4: CT/GC, 5: AT/AC, 6: GT/AT

stupac 6: 0: neoznačeno, 1: označeno u bazi podataka spojeva

stupac 7: broj jedinstveno mapiranih read-ova koja prelaze spoj

stupac 8: broj read-ova višestrukog mapiranja koja prelaze spoj

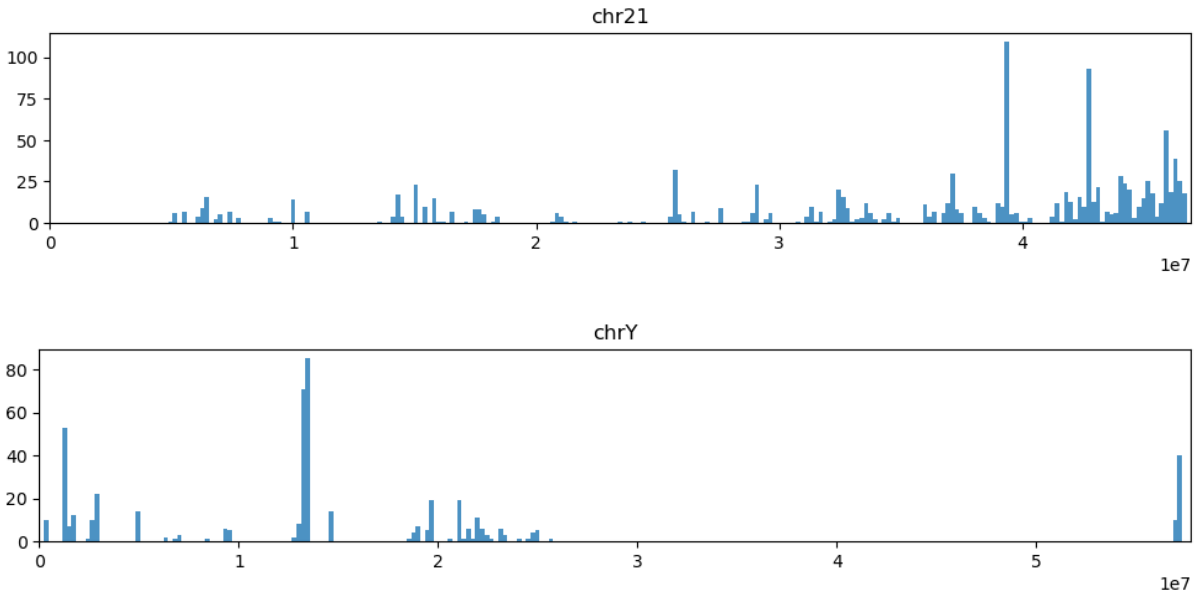
stupac 9: maksimalno podudaranje poravnanja spojeva

File SJ.out.tab koristimo i manipuliramo njime u našem algoritmu.

ASTALAVISTA

AStalavista je računalni program za izdvajanje događaja alternativnog izrezivanja (AS) iz zadane genomske oznake koordinata gena. Uspoređujući sve dane transkripte, AStalavista otkriva varijacije u njihovoj strukturi izrezivanja i identificira sve AS događaje (kao što je preskakanje egzona, alternativni donor, itd.) dodjeljivanjem svakom od njih AS kod. Izlazni dokument je u posebnom GTF formatu.

Vizualna reprezentacija podataka danih Astalavistom:



NAŠ ALGORITAM

Skripte koje smo koristili u algoritmu implemetirane su u programskim jezicima Python i C++.

Prvo je potrebno navesti importe za daljne korištenje u kodu.

```
import numpy as nc
import matplotlib as mpl
import matplotlib.pyplot as plt
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize
import csv
```

Generiramo TSV (Tab Seperated Values) datoteku koju ćemo koristiti pri provjeri podudaranja read-ova (generiranih STAR-om) i evenata (generiranih Astalavistom). U ovaj TSV stavljamo samo evente za neki određeni kromosom.

```
with open('output.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] == "chrY":
                tsv_writer.writerow([arr[0], arr[3], arr[4]])
                dict[arr[0]].append((arr[3], arr[4]))
```

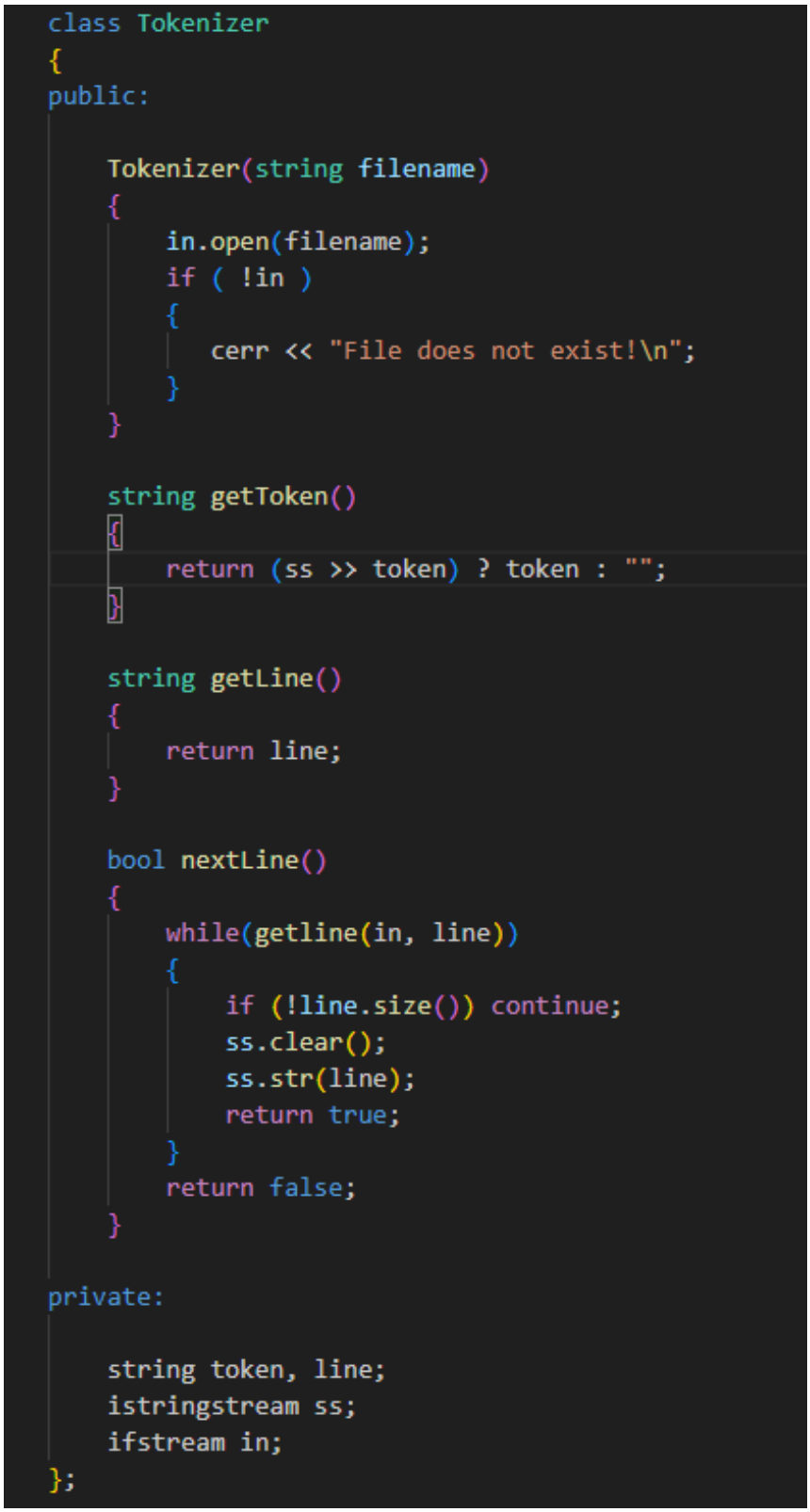
Par linija TSV datoteke:

```
chrY	276394	284167

chrY	290776	293035

chrY	299097	303356

chrY	319145	321332
```

Sada je bilo potrebno napraviti skriptu koja će uspoređivati SAM datoteku i našu TSV datoteku. Za taj dio smo koristili C++. Prvo treba navesti library-je koje ćemo koristiti u skripti.

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include "Tokenizer.cpp"
```

Tokenizer.cpp je skripta koja omogućava kretanje linijama neke datoteke riječ po riječ.



```
class Tokenizer
{
public:

    Tokenizer(string filename)
    {
        in.open(filename);
        if ( !in )
        {
            cerr << "File does not exist!\n";
        }
    }

    string getToken()
    {
        return (ss >> token) ? token : "";
    }

    string getLine()
    {
        return line;
    }

    bool nextLine()
    {
        while(getline(in, line))
        {
            if (!line.size()) continue;
            ss.clear();
            ss.str(line);
            return true;
        }
        return false;
    }

private:

    string token, line;
    istringstream ss;
    ifstream in;
};
```

Skripta koja uspoređuje svaku liniju SAM i TSV datoteka te pravi zaključak ovisno o imenu, početnoj i završnoj koordinati kromosoma. Podatke koji se podudaraju sprema u .txt datoteku.

```
int main(){
    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();
}
```

TraverseString funkcija je definirana na sljedeći način.

```
void TraverseString(string &str, int &Start_Coord, int &End_Coord)
{
      int n = str.length();
      string word = "";


      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' ) {

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

          }

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

Sljedeći dio algoritma je ponovno napravljen u Pythonu. Definiramo konstante koje su duljine svih kromosoma.

```
plt.rcParams['figure.figsize'] = [12, 2]
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]
```

Prolazimo kroz sve linije GTF datoteke i uzimamo samo podatke koji nas zanimaju. Svaku liniju dijelimo na riječi pomoću **word_tokenize()** funkcije. Vodimo računa i o tipu eventa koji može biti (Alternative Acceptor, Alternative Donor, 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 na .txt datoteci koju smo maloprije generirali svojim C++ kodom.

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

Pravimo statistiku frekvencija određenih evenata koje ćemo kasnije grafički prikazati.

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

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

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

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

Napravimo 3D matricu svih evenata za lakše snalaženje u podacima.

```
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)
```

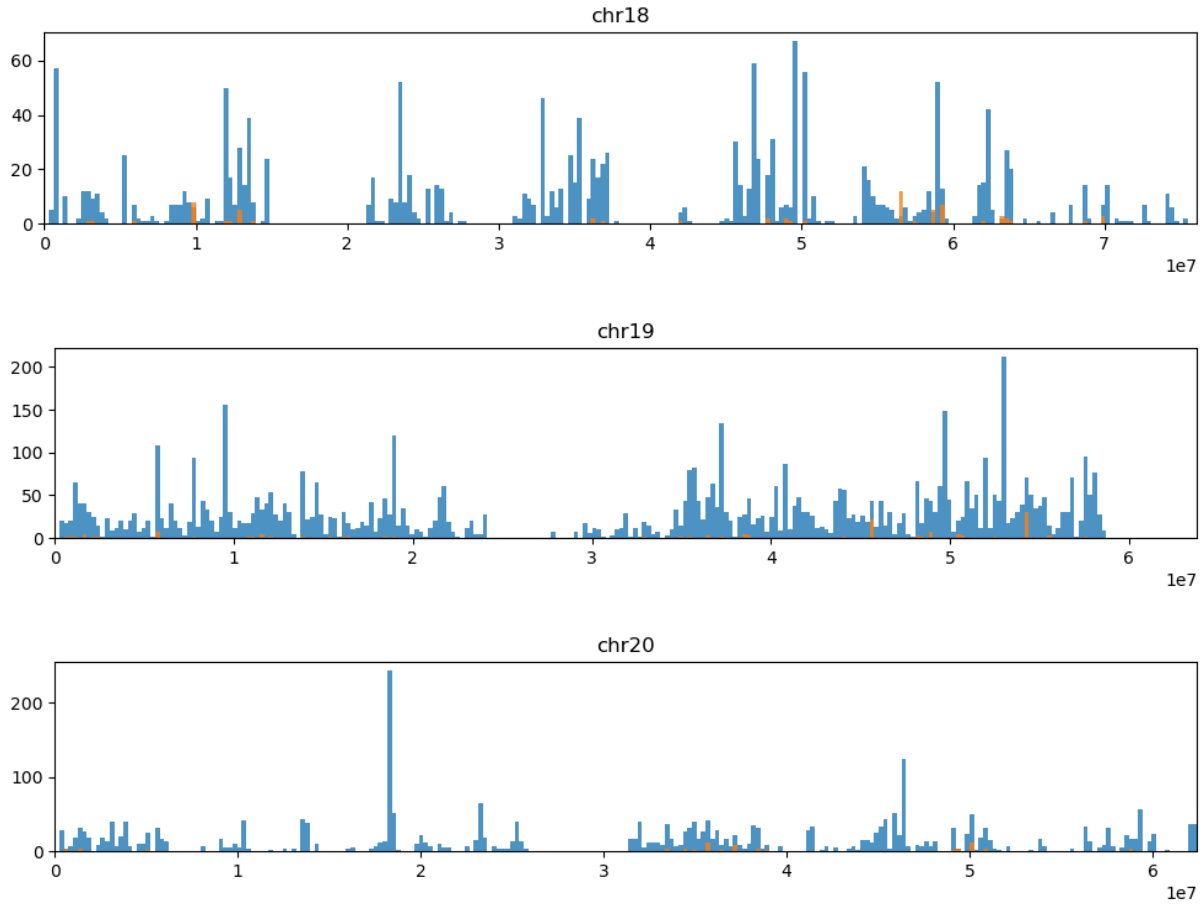
Biramo kromosom čiji histogram želimo prikazati, te preciznost istog histograma. U ovom slučaju, radi se o kromosomu Y.

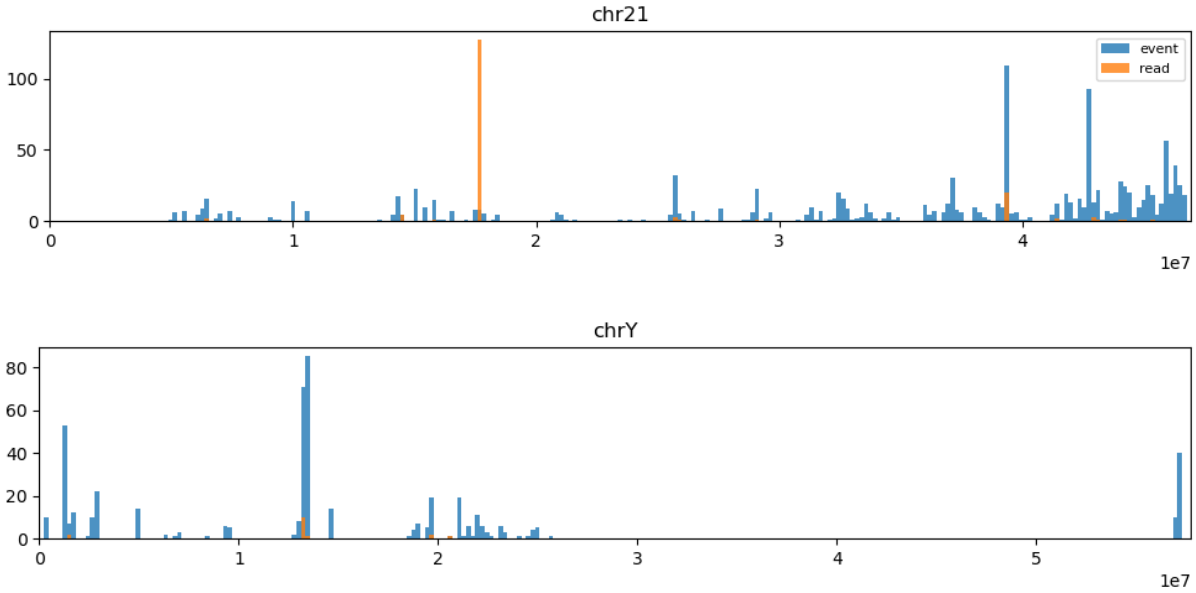
```
chr_id = 24
histogram_precision = 250
```

Sada treba iscrtati histograme tih kromosoma. Njih 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])
#plt.legend(loc='upper right', fontsize = 8)
```

Rezultati našeg algoritma izgledaju ovako:





Na grafovima su prikazani event-ovi i read-ovi zasebno za svaki kromosom. X os nam govori duljinu kromosoma po 10 milijuna, a y os količinu događaja ili read-ova na određenom području kromosoma. Poanta navedenih histograma je istaknuti količinu read-ova koji se podudaraju s određenim događajima i prikazati na kojoj točno lokaciji se nalaze. Te informacije su od velike koristi stručnjacima jer znaju na kojim područjima genoma provesti detaljnija istraživanja.

ZAKLJUČAK

Iz ovih histograma možemo primijetiti da su neki od evenata pridružili sebi jako puno read-ova, kao npr. na 21. kromosomu. To može ukazivati na anomalije pri alternativnom izrezivanju. Takvi podaci se dalje prosljeđuju biolozima koji onda mogu fokusirati i istraživati tu regiju genoma, te na taj način pokušati pronaći korelaciju između anomalija u njoj s autizmom.