Matematički fakultet

Klasterovanje gena i ćelija

Vladana Đorđević 89/2015

Mentor: prof. dr Nenad Mitić

Beograd

2019

Sadržaj

1 Uvod

Jedan od fundamentalnih pristupa razumevanju i učenju je organizovanje podataka u smislene grupe. Klaster analiza je formalna studija o metodima i algoritmima za prirodno grupisanje, odnosno klasterovanje, objekata na osnovu izmerenih ili uočenih unutrašnjih svojstava ili sličnosti. Uzorci za klasterovanje su prikazani kao vektori merenja ili, formalnije, kao tačka u multidimenzionom prostoru. Uzorci koji pripadaju jednom klasteru su sličniji međusobno nego uzorcima koji pripadaju drugim klasterima. Klaster metodologija je posebno pogodna za istraživanje međuodnosa između uzoraka da bi se dobila preliminarna procena o strukturi uzorka. Ljudske performanse se mogu porediti sa automatskim klasterovanjem kada to uključuje procedure klasterovanje u jednoj, dve ili tri dimenzije. Međutim, mnogi stvarni problemi uključuju klasterovanje u više dimenzija. Veoma je teško ljudima da intuitivno interpretiraju podatke koji su ugnježdeni u visoko dimenzionom prostoru [1].

1.1 Podaci

Podaci nad kojim će se vršiti klasterovanje u ovom radu se nalaze u datoteci 042_A549_cells_from_ATCC_and_Influenza_AWSN1933_virus_csv.csv Ova datoteka sadrži podatke dobijene iz mononuklearnih ćelija periferne krvi (eng. *peripheral blood mononuclear cells, PBMCs*). PBMC je svaka ćelija koja ima okrugli nukleus, i ona se sastoji iz limfocita (T ćelija, B ćelija, NK ćelija (eng. natural killer cells)), monocita i malog procenta dendritičnih ćelija. PBMC ćelije se koriste u istraživanju u različitim oblastima biomedicine, uključujući infektivne bolesti, imunologiju (uključujući i autoimune poremećaje), malignitet, transplantacionu imunologiju, razvoj vakcina, i skrining. Mada mogu da imaju različite funkcije, glavna funkcija PBMC ćelija je imuna odbrana organizma. Svaki tip ćelije ima karakteristične obrasce ('mustre') proteina i gena koje ih međusobno razlikuju i mogu da se koriste za podelu prema njihovom tipu. Ulazni materijal sadrži rezultate istraživanja skupova različitih humanih ćelija. Datoteka sadrži podatke vezane za ekspresiju 31221 gena na skupu izmedu 400 i 8500 PBMC ćelija. Nazivi svih gena imaju prefiks hg38 što znači da su podaci vezani za verziju 38 humanog genoma. Datoteka predstavlja različite tipove ćelija i uslova za Influenza AWSN1933 virus za koje je rađeno istraživanje. Datoteka sadrži podatke u CSV formatu u obliku ukrštene tabele (podatak u preseku prvog reda i prve kolone je prazan). Sastoji se iz 31222 reda i 3368 kolona, pri čemu prvi red sadrži redni broj ćelije za koje je vršeno istraživanje, a prva kolona sadrži identifikaciju gena. Vrednosti u tabeli, koje ne pripadaju prvom redu i prvoj koloni (31221x3368 vrednosti), sadrže broj transkripta gena (navedenog u prvoj koloni) u ćeliji [2].

1.2 Cilj istraživanja

Cilj istraživanja je uočiti sličnosti i vezu između gena i između ćelija, na osnovu podataka koji su nam dati. Ovo upravo potpada pod klasterovanje pa će se u ovom radu naći različite tehnike klasterovanja. Primenićemo alat CLUTO, algoritme

sakupljajućeg hijerarhijskog klasterovanja, algoritme DBSCAN i KMeans. Za hijerarhijsko klasterovanje, DBSCAN i KMeans korišćen je programski jezik Python i većinom biblioteka sklearn.cluster koja ima pomenute metode. Prilikom pretprocesiranja podataka i primene CLUTO alata korišćen je jezik C++ i biblioteka boost.

Biblioteka boost se instalira sledećom naredbom:

```
sudo apt-get install libboost-all-dev
```

Biblioteke koje su potrebne za izvršavanje programa u Python-u se mogu instalirati na sledeći način:

```
pip3 install ime_biblioteke
```

2 Pretprocesiranje

S obzirom na to da prva linija sadrži redne brojeve ćelija koje nisu od velikog značaja, prva linija se izbacuje. Nakon toga je potrebno ukloniti nulte gene. To je postignuto sumiranjem svih vrednosti u jednom redu i, ako je suma jednaka nuli, onda se taj red uklanja. Izdvajanje vrednosti iz jednog reda je postignuto korišćenjem biblioteke *boost*. Kod je prikazan listingom 1.

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <boost/algorithm/string.hpp>

bool jeste_zarez(char c) {
  return c == ',';
}

int main() {
  std::ifstream ulaz("042
    _A549_cells_from_ATCC_and_Influenza_AWSN1933_virus_csv.csv", std
    ::ifstream::in);
  std::ofstream izlaz("pretprocesirano.txt", std::ofstream::out);

  std::string linija;
  std::string prva;
  std::getline(ulaz, prva);
  int n = 0;
  while(ulaz >> linija) {
    std::vector<std::string> rezultat;
    boost::split(rezultat, linija, jeste_zarez);
    int suma = 0;
    for (int i = 1; i < rezultat.size(); i++) {
      suma += stoi(rezultat[i]);
    }
    if (suma != 0) {
      izlaz << linija << std::endl;
```

```
28        n++;
29      }
30    }
31    std::cout << "Broj redova nakon uklanjanja nultih gena je: " << n
        << std::endl;
32    ulaz.close();
33    izlaz.close();
34    return 0;
35  }
```

Listing 1: Pretprocesiranje podataka - pretprocesiranje.cpp

Da bi se program preveo i izvršio potrebno je pozicionirati se u direktorijum *IP2_ Vladana_ Djordjevic_ 89_ 2015/Pretprocesiranje* i izvršiti sledeće naredbe:

```
1     g++ pretprocesiranje.cpp -o pretprocesiranje
```

```
1     ./pretprocesiranje
```

U terminalu će se ispisati sledeća linija:

```
1     Broj redova nakon uklanjanja nultih gena je: 14428
```

Izlaz iz programa je datoteka *pretprocesirano.txt* koju ćemo nadalje koristiti kao ulaz u algoritme za klasterovanje gena.

Kada budemo klasterovali ćelije biće neophodno da nam svaki red bude jedna ćelija, jer tako algoritmi za klasterovanje čitaju podatke, jedan red - jedan objekat. Međutim, naši podaci su napravljeni tako da je jedan red jedan gen, a svaka kolona je jedna ćelija. Da bismo mogli da vršimo klasterovanje ćelija neophodno je transponovati podatke tako da svaki red bude jedna ćelija. To je postignuto kodom koji je dat listingom 2.

```
1   #include <iostream>
2   #include <cstdlib>
3   #include <string>
4   #include <iterator>
5   #include <vector>
6   #include <string>
7   #include <fstream>
8   #include <boost/algorithm/string.hpp>
9
10  bool jeste_zarez(char c) {
11      return c == ',';
12  }
13
14  int main(int argc, const char* argv[]) {
15      if (argc != 5) {
16    std::cout << "Program se pokrece na sledeci nacin: " << std::endl
      ;
17    std::cout << "./ime_programa ulazna_dat izlazna_dat n m" << std::
      endl;
```

```
18    std::exit(EXIT_FAILURE);
19      }
20
21      std::string ulazna_datoteka = argv[1];
22      std::string izlazna_datoteka = argv[2];
23      int n = std::stoi(argv[3]);
24      int m = std::stoi(argv[4]);
25
26      std::ifstream ulaz(ulazna_datoteka, std::ifstream::in);
27      std::ofstream izlaz(izlazna_datoteka, std::ofstream::out);
28
29      std::vector<std::vector<std::string>> matrica(n);
30      std::string linija;
31      int i = 0;
32
33      izlaz << m << " " << n << std::endl;
34      while(ulaz >> linija) {
35          std::vector<std::string> rezultat;
36          boost::split(rezultat, linija, jeste_zarez);
37
38          std::copy(rezultat.begin()+1, rezultat.end(), std::
     back_inserter(matrica.at(i)));
39          i++;
40      }
41
42      for (int i = 0; i < m; i++) {
43          for (int j = 0; j < n; j++) {
44              izlaz << matrica.at(j).at(i) << " ";
45          }
46          izlaz << std::endl;
47      }
48
49      ulaz.close();
50      izlaz.close();
51      return 0;
52  }
```

Listing 2: Transponovanje matrice podataka - transponovanje.cpp

Kod se nalazi u direktorijumu *IP2_ Vladana_ Djordjevic_ 89_ 2015/Pretprocesiranje*, a pokreće se i izvršava na sledeći način:

```
1     g++ transponovanje.cpp -o transponovanje
```

```
1     ./transponovanje pretprocesirano.txt transponovana_matrica.txt
     14428 3368
```

Izlazna datoteka pod nazivom *transponovana_ matrica.txt* sadrži transponovanu matricu koja će se koristiti kao ulaz za klasterovanje ćelija.

Kada budemo koristili alat CLUTO za klasterovanje gena biće potrebno pripremiti datoteku u kojoj se nalazi matrica sa podacima. Naime, vcluster program, koga koristi CLUTO alat, zahteva da prva linija u datoteci sadrži dimenziju matrice,

tj. dva cela broja jedan za drugim, gde prvi odgovara broju redova a drugi broju kolona. Takođe, matrica mora da ima samo numeričke vrednosti pa je potrebno izbaciti oznake gena koje se nalaze na početku svakog reda kao i zareze kojima su podaci odvojeni. To je postignuto kodom koji je dat kroz listing 3.

```
#include <iostream>
#include <cstdlib>
#include <iterator>
#include <vector>
#include <string>
#include <fstream>
#include <boost/algorithm/string.hpp>

bool jeste_zarez(char c) {
    return c == ',';
}

int main(int argc, const char* argv[]) {
    if (argc != 6) {
  std::cout << "Program se pokrece na sledeci nacin: " << std::endl
   ;
  std::cout << "./ime_programa ulazna_dat izlazna_dat1 izlazna_dat2
    n m" << std::endl;
  std::exit(EXIT_FAILURE);
    }

    std::string ulazna_datoteka = argv[1];
    std::string izlazna_datoteka1 = argv[2];
    std::string izlazna_datoteka2 = argv[3];
    int n = std::stoi(argv[4]);
    int m = std::stoi(argv[5]);

    std::ifstream ulaz(ulazna_datoteka, std::ifstream::in);
    std::ofstream izlaz1(izlazna_datoteka1, std::ofstream::out);
    std::ofstream izlaz2(izlazna_datoteka2, std::ofstream::out);

    std::string linija;

    izlaz1 << n << " " << m << std::endl;

    while(ulaz >> linija) {
        std::vector<std::string> rezultat;
        boost::split(rezultat, linija, jeste_zarez);

        izlaz2 << rezultat[0] << std::endl;
        std::copy(rezultat.begin()+1, rezultat.end(), std::
   ostream_iterator<std::string>(izlaz1, " "));
        izlaz1 << std::endl;
    }

    ulaz.close();
    izlaz1.close();
    izlaz2.close();
    return 0;
}
```

Listing 3: Priprema podataka za primenu CLUTO alata - sredjivanje_podataka.cpp

Za prevođenje i izvršavanje programa potrebno je pozicionirati se u direktorijum *IP2_ Vladana_ Djordjevic_ 89_ 2015/Pretprocesiranje* i izvršiti sledeće naredbe:

```
g++ sredjivanje_podataka.cpp -o sredjivanje_podataka
```

```
./sredjivanje_podataka pretprocesirano.txt matrica.txt
oznake_gena.txt 14428 3368
```

Kao ulazni datoteka korišćen je *pretprocesirano.txt* koji je dobijen u procesu pretprocesiranja, a izlazne datoteke su *matrica.txt* koja sadrži matricu koju koristi vcluster i *oznake_ gena.txt* gde su smeštene oznake gena koje izbacujemo iz podataka.

3 Klasterovanje

3.1 CLUTO

CLUTO je softverski paket za klasterovanje nisko i visoko dimenzionih skupova podataka i za analiziranje karakteristika različitih klastera. CLUTO pruža tri različite klase algoritama za klasterovanje koji rade ili u prostoru atributa objekta ili u prostoru sličnosti objekta. Ovi algoritmi su zasnovani na particionisanju, sakupljanju i grafovima. Glavna karakteristika većine CLUTO algoritama za klasterovanje je da tretiraju problem klasterovanja kao proces optimizacije koji pokušava da maksimizuje ili minimizuje određenu funkciju kriterijuma koja je definisana ili globalno ili lokalno u celom prostoru rešenja. CLUTO obezbeđuje sedam različitih funkcija kriterijuma koje se mogu koristiti i u particionišućim i u sakupljajućim algoritmima za klasterovanje. Većina tih funkcija kriterujuma se pokazala dobro u smislu da proizvode rešenja klasterovanja visokog kvaliteta u visoko dimenzionim skupovima podataka [3].

CLUTO obezbeđuje klasterovanje i analizu preko programa **vcluster** i **scluster**. Ključna razlika je da vcluster kao ulaz uzima multidimenzionu reprezentaciju objekata koju treba klasterovati (v je od vektor, *eng. vector*), dok scluster kao ulaz uzima matricu sličnosti (ili graf) ovih objekata (s je od sličnost, *eng. similarity*). Oba programa pružaju sličnu funkcionalnost. Programi vcluster i scluster se koriste da klasteruju objekte u unapred određeni broj klastera k. Vcluster program tretira svaki objekat kao vektor u visokodimenzionom prostoru i izračunava rezultat klasterovanja koristeći jedan od pet različitih pristupa. Četiri su razdvajajuća, a peti je sakupljajući. U slučaju naših podataka, koji su prikazani multidimenziono u obliku matrice, pogodno je koristiti vcluster program. Stoga ćemo nadalje pažnju posvetiti isključivo njemu [3].

Pri pozivu vcluster-a dva parametra su neophodna, a ima i nekoliko opcionih. Program se poziva na sledeći način:

vcluster [opcioni parametri] DatotekaSaMatricom BrojKlastera

DatotekaSaMatricom je ime datoteke u kojoj se nalazi N objekata koje je potrebno klasterovati. Vcluster posmatra svaki od tih objekata kao vektor u M-dimenzionom prostoru. Kolekcija ovih objekata se posmatra kao matrica dimenzije N x M, čiji redovi odgovaraju objektima, a kolone odgovaraju dimenzijama u prostoru karakteristika. Drugi argument, BrojKlastera, predstavlja broj klastera koji želimo da dobijemo [3].

Nakon uspešnog izvršavanja, vcluster ispisuje statistike vezane za kvalitet rezultata klasterovanja i potrebnu količinu vremena da se izvrši klasterovanje. Rezultat samog klasterovanja je smešten u datoteci pod nazivom DatotekaSaMatricom.clustering.BrojKlastera.

Ponašanje vcluster-a može biti kontrolisano navođenjem različitih opcionih parametara. Ovi parametri se mogu kategorisati u tri grupe. Prva grupa kontroliše različite aspekte algoritma klasterovanja, druga grupa kontroliše tip analize i izveštaja nad izračunatim klasterima, a treća kontroliše vizualizaciju klastera. Opcioni parametri se navode formatom **-paramname=vrednost** [3].

3.2 Opcioni parametri CLUTO alata

Ima ukupno 18 različitih opcionih parametara koji kontrolišu kako vcluster izračunava rešenje klasterovanja. Ovde će biti opisana samo 3 parametra koja su korišćena za klasterovanje naših podataka [3].

1. **-clmethod=string**: Ovaj parametar navodi metodu koja će biti korišćena za klasterovanje objekata. Moguće vrednosti parametra su:

   - **rb**: Da bismo dobili željenih k klastera ponavljamo k-1 bisekcija. Matrica se prvo klasteruje u dve grupe, a onda je jedna od te dve grupe izabrana i potom se bisekcija vrši nad njom. Ovaj proces se nastavlja sve dok se ne dobije željeni broj klastera. U svakom koraku, klaster se deli na dva dela tako da rezultat optimizuje određenu funkciju kriterijuma klastera (koja se navodi korišćenjem -crfun parametra). Napomena je da ovaj pristup lokalno optimizuje funkciju kriterijuma prilikom svake bisekcije, ali generalno nije globalno optimizovana. Klaster koji se bira je određen parametrom -cstype. Ovo je podrazumevana vrednost.
   - **rbr**: Ova metoda vrši klasterovanje koje je veoma slično ponovljenim bisekcijama, ali na kraju finalno rešenje je globalno optimizovano. U suštini, vcluster koristi rešenje koje dobijeno primenom rb metoda kao inicijalno rešenje a potom pokušava da dodatno optimizuje funkciju klasterovanja.
   - **direct**: Željenih k klastera se dobija njihovim istovremenim pronalaženjem. Generalno, izračunavanje k klastera direktno je sporije nego klasterovanje ponovljenim bisekcijama. Po pitanju kvaliteta, za male vrednosti k (obično manje od 10-20), ovaj pristup dobija bolje klastere nego pristup sa ponovljenim bisekcijama. Međutim, kako se k povećava,

ponovljene bisekcije predstavlaju bolji pristup od direktnog klasterovanja.

- **agglo**: Željenih k klastera je dobijeno korišćenjem sakupljajuće paradigme, čiji je cilj da lokalno optimizuje (minimizuje ili maksimizuje) određenu funkciju kriterijuma. Rešenje se dobija zaustavljanjem procesa kada ostane k klastera.

Pored ovih metoda postoje još i **graph** i **bagglo**, ali ih nećemo detaljno opisivati jer nisu korišćeni u ovom radu.

2. **-sim=string**: Bira se funkcija sličnosti koja se koristi za klasterovanje. Moguće vrednosti su:

   - **cos**: Sličnost između objekata je izračunata korišćenjem kosinusne funkcije. Ovo je podrazumevana vrednost.
   - **corr**: Sličnost između objekata je izračunata korišćenjem koeficijenta korelacije.

   Pored ovih postoje i **dist** i **jacc**, međutim, oni su primenjivi samo kada je metoda graph, a s obzirom na to da taj metod nije korišćen u našem slučaju, nisu ni ove dve funkcije.

   Još jedna napomena je da se vreme izvršavanja vcluster programa može povećati u slučaju korišćenja **-sim=corr** jer je neophodno da uskladišti i radi nad gustom N x M matricom.

3. **-crfun=string**: Ovaj parametar bira određenu funkciju kriterijuma klasterovanja koja će biti korišćena za pronalaženje klastera. Ukupno ima 7 različitih funkcija kriterijuma za klastere. Moguće vrednosti su:

   - **i1**: Bira se I1 funkcija kriterijuma.
   - **i2**: Bira se I2 funkcija kriterijuma. Ovo je podrazumevana vrednost za rb, rbr i direct metode klasterovanja.
   - **e1**: Bira se E1 funkcija kriterijuma.
   - **g1**: Bira se G1 funkcija kriterijuma.
   - **g1p**: Bira se G1' funkcija kriterijuma.
   - **h1**: Bira se H1 funkcija kriterijuma.
   - **h2**: Bira se H2 funkcija kriterijuma.

   Pored ovih postoje još i **slink**, **wslink**, **clink**, **wclink** i **upgma**, ali njih nećemo detaljno opisivati jer nisu korišćene u ovom istraživanju.

   Na slici 1 možemo videti preciznu matematičku definiciji funkcija koje smo nabrojali.

   Različite funkcije kriterijuma mogu dovesti do značajno različitih rezultata klasterovanja. Generalno I2 i H2 funkcije kriterijuma daju veoma dobre rezultate klasterovanja, dok E1 i G1' funkcije daju rešenja koja imaju klastere

Criterion Function	Optimazition Function	
$\mathcal{I}_1$	maximize $\sum_{i=1}^{k} \frac{1}{n_i} \left(\sum_{v,u \in S_i} \text{sim}(v,u) \right)$	(1)
$\mathcal{I}_2$	maximize $\sum_{i=1}^{k} \sqrt{\sum_{v,u \in S_i} \text{sim}(v,u)}$	(2)
$\mathcal{E}_1$	minimize $\sum_{i=1}^{k} n_i \frac{\sum_{v \in S_i, u \in S} \text{sim}(v,u)}{\sqrt{\sum_{v,u \in S_i} \text{sim}(v,u)}}$	(3)
$\mathcal{G}_1$	minimize $\sum_{i=1}^{k} \frac{\sum_{v \in S_i, u \in S} \text{sim}(v,u)}{\sum_{v,u \in S_i} \text{sim}(v,u)}$	(4)
$\mathcal{G}'_1$	minimize $\sum_{i=1}^{k} n_i^2 \frac{\sum_{v \in S_i, u \in S} \text{sim}(v,u)}{\sum_{v,u \in S_i} \text{sim}(v,u)}$	(5)
$\mathcal{H}_1$	maximize $\frac{\mathcal{I}_1}{\mathcal{E}_1}$	(6)
$\mathcal{H}_2$	maximize $\frac{\mathcal{I}_2}{\mathcal{E}_1}$	(7)

Slika 1: Funkcije kriterijuma CLUTO alata

uporedivih veličina. Međutim, izbor odgovarajuće funkcije kriterijuma zavisi od oblasti na koju se primenjuje i potrebno je eksperimentisati sa primenama ovih funkcija dok se ne pronađe odgovarajuća [3].

3.3 Analiza izlaza CLUTO programa

Izlaz iz programa vcluster sadrži deo pod nazivom "Solution" gde se, između ostalog, može naći vrednost funkcije kriterijuma. Na primer, [I2=3.31e+03], gde je 3.31e+03 vrednosti I2 funkcije kriterijuma u našem rešenju. U istoj liniji se nalazi koliko je originalnih objekata klasterovano. Na primer, [3368 of 3368]. Generalno, vcluster pokušava da klasteruje sve objekte, međutim, ako neki objekti ne dele nijednu dimenziju sa ostalim, onda je moguće da će vcluster klasterovati manje od ukupnog broja objekata [3].

Nakon toga, vcluster ispisuje tabelu u kojoj svaki red sadrži statistike za svaki od klastera. Kolone tabele su [3]:

- **cid**: Broj klastera, odnosno id klastera.
- **Size**: Broj objekata koji pripadaju klasteru.
- **ISim**: Prosečna sličnost između objekata svakog klastera (unutrašnja sličnost).
- **ISdev**: Standardna devijacija prosečnih unutrašnjih sličnosti (unutrašnja standardna devijacija).

- **ESim**: Prosečna sličnost objekata svakog klastera sa ostalim objektima (spoljašnja sličnost).
- **ESdev**: Standardna devijacija spoljašnjih sličnosti (spoljašnja standardna devijacija).

Napomena je da su dobijeni klasteri poređani u rastućem poretku po vrednosti (ISIM-ESIM). Drugim rečima, klasteri koji su blizu iznutra a daleko od ostatka objekata imaju manje vrednosti cid-a [3].

3.4 Klasterovanje gena korišćenjem CLUTO alata

Pre pokretanja vcluster programa potrebno je pokrenuti program sredjivanje_podataka.cpp koji je dat listingom 3. Kao izlaz program će vratiti datoteku *matrica.txt* koju je potrebno pozicionirati u direktorijum gde se nalazi program vcluster. Nakon toga izvršavamo vcluster korišćenjem različitih kombinacija parametara u potrazi za dobrim rezultatom klasterovanja.

U tabeli 1 se nalaze sve isprobane kombinacije parametara.

U narednom delu navešćemo datoteke koje su dobijene kao i komande koje su pokrenute da bi se te datoteke dobile. Napomena je da se komande moraju izvršavati u direktorijumu gde se nalazi izvršni program vcluster i da se u tom direktorijumu mora nalaziti i datoteka matrica.txt. Iznad svake komande biće navedene dve datoteke. Prva datoteka se nalazi u direktorijumu *IP2_ Vladana_ Djordjevic_ 89_ 2015/ CLUTO/Geni/Izlaz kluto* i predstavlja izlaz koji je dobijen u terminalu nakon pokretanja programa. Druga datoteka se nalazi u direktorijumu *IP2_ Vladana_ Djordjevic _ 89_ 2015/CLUTO/Geni/Podela po klasterima* i sastoji se od linija na kojima se nalazi broj klastera kome je dodeljen gen koji se nalazi na toj liniji u ulaznoj datoteci. Redosled navođenja komandi i datoteka odgovara redosledu u tabeli. Datoteke i pokretanja:

agglo_ cos_ 5_ g1p.txt i *agglo_ cos_ g1p.5*

```
./vcluster -clmethod=agglo -sim=cos -crfun=g1p matrica.txt 5
```

agglo_ cos_ 10_ g1p.txt i *agglo_ cos_ g1p.10*

```
./vcluster -clmethod=agglo -sim=cos -crfun=g1p matrica.txt 10
```

agglo_ cos_ 30_ g1p.txt i *agglo_ cos_ g1p.30*

```
./vcluster -clmethod=agglo -sim=cos -crfun=g1p matrica.txt 30
```

agglo_ cos_ 10_ i1.txt i *agglo_ cos_ i1.10*

```
./vcluster -clmethod=agglo -sim=cos -crfun=i1 matrica.txt 10
```

direct_ corr_ 10_ e1.txt i *direct_ corr_ e1.10*

Tabela 1: Sve isprobane kombinacije parametara prilikom klasterovanja gena programom vcluster

Metoda	**Funkcija sličnosti**	**Funkcija kriterijuma**	**Broj klastera**
agglo	cos	g1p	5
agglo	cos	g1p	10
agglo	cos	g1p	30
agglo	cos	i1	10
direct	corr	e1	10
direct	corr	g1	10
direct	corr	g1p	10
direct	corr	h1	10
direct	corr	h2	10
direct	corr	i1	10
direct	corr	i2	10
direct	cos	g1p	10
rb	cos	i2	10
rbr	corr	e1	10
rbr	corr	g1	5
rbr	corr	g1	10
rbr	corr	g1	20
rbr	corr	g1	50
rbr	corr	g1p	20
rbr	corr	h1	10
rbr	corr	h2	10
rbr	corr	i1	10
rbr	corr	i1	20
rbr	corr	i1	50
rbr	corr	i1	100
rbr	corr	i2	10
rbr	cos	g1p	5
rbr	cos	g1p	100

```
./vcluster -clmethod=direct -sim=corr -crfun=e1 matrica.txt 10
```

direct_corr_10_g1.txt i *direct_corr_g1.10*

```
./vcluster -clmethod=direct -sim=corr -crfun=g1 matrica.txt 10
```

direct_corr_10_g1p.txt i *direct_corr_g1p.10*

```
./vcluster -clmethod=direct -sim=corr -crfun=g1p matrica.txt 10
```

direct_corr_10_h1.txt i *direct_corr_h1.10*

```
./vcluster -clmethod=direct -sim=corr -crfun=h1 matrica.txt 10
```

direct_corr_10_h2.txt i *direct_corr_h2.10*

```
./vcluster -clmethod=direct -sim=corr -crfun=h2 matrica.txt 10
```

direct_corr_10_i1.txt i *direct_corr_i1.10*

```
./vcluster -clmethod=direct -sim=corr -crfun=i1 matrica.txt 10
```

direct_corr_10_i2.txt i *direct_corr_i2.10*

```
./vcluster -clmethod=direct -sim=corr -crfun=i2 matrica.txt 10
```

direct_cos_10_g1p.txt i *direct_cos_g1p.10*

```
./vcluster -clmethod=direct -sim=cos -crfun=g1p matrica.txt 10
```

rb_cos_10_i2.txt i *rb_cos_i2.10*

```
./vcluster -clmethod=rb -sim=cos -crfun=i2 matrica.txt 10
```

rbr_corr_10_e1.txt i *rbr_corr_e1.10*

```
./vcluster -clmethod=rbr -sim=corr -crfun=e1 matrica.txt 10
```

rbr_corr_5_g1.txt i *rbr_corr_g1.5*

```
./vcluster -clmethod=rbr -sim=corr -crfun=e1 gatrica.txt 5
```

rbr_corr_10_g1.txt i *rbr_corr_g1.10*

```
./vcluster -clmethod=rbr -sim=corr -crfun=g1 matrica.txt 10
```

rbr_corr_20_g1.txt i *rbr_corr_g1.20*

```
./vcluster -clmethod=rbr -sim=corr -crfun=g1 matrica.txt 20
```

rbr_corr_50_g1.txt i *rbr_corr_g1.50*

```
./vcluster -clmethod=rbr -sim=corr -crfun=g1 matrica.txt 50
```

rbr_corr_20_g1p.txt i *rbr_corr_g1p.20*

```
./vcluster -clmethod=rbr -sim=corr -crfun=g1p matrica.txt 20
```

rbr_corr_10_h1.txt i *rbr_corr_h1.10*

```
./vcluster -clmethod=rbr -sim=corr -crfun=h1 matrica.txt 10
```

rbr_corr_10_h2.txt i *rbr_corr_h2.10*

```
./vcluster -clmethod=rbr -sim=corr -crfun=h2 matrica.txt 10
```

rbr_corr_10_i1.txt i *rbr_corr_i1.10*

```
./vcluster -clmethod=rbr -sim=corr -crfun=i1 matrica.txt 10
```

rbr_corr_20_i1.txt i *rbr_corr_i1.20*

```
./vcluster -clmethod=rbr -sim=corr -crfun=i1 matrica.txt 20
```

rbr_corr_50_i1.txt i *rbr_corr_i1.50*

```
./vcluster -clmethod=rbr -sim=corr -crfun=i1 matrica.txt 50
```

rbr_corr_100_i1.txt i *rbr_corr_i1.100*

```
./vcluster -clmethod=rbr -sim=corr -crfun=i1 matrica.txt 100
```

rbr_corr_10_i2.txt i *rbr_corr_i2.10*

```
./vcluster -clmethod=rbr -sim=corr -crfun=i2 matrica.txt 10
```

rbr_cos_5_g1p.txt i *rbr_cos_g1p.5*

```
./vcluster -clmethod=rbr -sim=cos -crfun=g1p matrica.txt 5
```

rbr_cos_100_g1p.txt i *rbr_cos_g1p.100*

```
./vcluster -clmethod=rbr -sim=cos -crfun=g1p matrica.txt 100
```

Da bismo utvrdili koja kombinacija daje najbolje rezultate pregledane su sve datoteke koje su navedene prve, odnosno one koje sadrže izlaz iz vcluster programa. Cilj je bio pronaći kombinaciju koja daje velike vrednosti za sličnosti unutar samog klastera, a male vrednosti za sličnosti između različitih klastera. Upoređivanjem rezultata došli smo do zaključka da među navedenim kombinacijama najbolje rezultate daje rbr + corr + i1 kombinacija. Eksperimentirali smo sa brojem klastera: 10, 20, 50 i 100. Prikazaćemo rezultate za broj klastera 20 i 50.

rbr_corr_20_i1.txt

```
vcluster CLUTO 2.1.1 Copyright 2001-03, Regents of the University of Minnesota

Matrix Information ------------------------------------------------------------
  Name: matrica.mat, #Rows: 14428, #Columns: 3368, #NonZeros: 48593504

Options -----------------------------------------------------------------------
  CLMethod=RBR, CRfun=I1, SimFun=CorrCoef, #Clusters: 20
  RowModel=None, ColModel=None, GrModel=SY-DIR, NNbrs=40
  Colprune=1.00, EdgePrune=-1.00, VtxPrune=-1.00, MinComponent=5
  CSType=Best, AggloFrom=0, AggloCRFun=I1, NTrials=10, NIter=10

Solution ----------------------------------------------------------------------

------------------------------------------------------------------------
20-way clustering: [I1=7.28e+02] [14428 of 14428]
------------------------------------------------------------------------
cid  Size  ISim  ISdev   ESim  ESdev
------------------------------------------------------------------------
  0    10 +0.564 +0.151 +0.003 +0.001
  1   169 +0.634 +0.066 +0.100 +0.010
  2    25 +0.285 +0.107 +0.002 +0.006
  3   176 +0.324 +0.054 +0.066 +0.014
  4   340 +0.289 +0.049 +0.067 +0.012
  5   350 +0.262 +0.042 +0.066 +0.011
  6    99 +0.228 +0.066 +0.042 +0.014
  7    68 +0.096 +0.052 +0.004 +0.005
  8   272 +0.122 +0.029 +0.041 +0.011
  9   557 +0.110 +0.018 +0.038 +0.008
 10   929 +0.090 +0.016 +0.038 +0.007
 11   380 +0.050 +0.015 +0.022 +0.008
 12  1304 +0.033 +0.006 +0.023 +0.005
 13  1571 +0.026 +0.006 +0.021 +0.005
 14   823 +0.009 +0.003 +0.007 +0.004
 15  1085 +0.009 +0.003 +0.008 +0.005
 16  1126 +0.003 +0.002 +0.003 +0.004
 17  1268 +0.003 +0.001 +0.003 +0.003
 18  1785 +0.003 +0.001 +0.004 +0.004
 19  2091 +0.010 +0.002 +0.014 +0.004
------------------------------------------------------------------------

Timing Information ------------------------------------------------------------
   I/O:                                7.199 sec
   Clustering:                       349.890 sec
   Reporting:                          2.316 sec
```

Izvršavanje ovog programa nam je kao izlaz kreiralo datoteku *rbr_corr_i1.20* u kome se u svakoj liniji nalazi broj klastera kome pripada gen na odgovarajućoj liniji. Sada je potrebno povezati gen sa klasterom kome je dodeljen. To je postignuto kodom koji naveden listingom 4.

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <string>
#include <map>
#include <vector>
#include <set>
#include <iterator>

int main(int argc, const char* argv[]) {

  if (argc != 4) {
    std::cout << "Program se pokrece na sledeci nacin: " << std::
   endl;
    std::cout << "./uk ulazna_dat izlazna_dat broj_linija" << std::
   endl;
    std::exit(EXIT_FAILURE);
  }

  std::string datoteka1 = argv[1];
  std::string datoteka2 = argv[2];
  int broj_linija = std::stoi(argv[3]);

  std::ifstream ulaz1(datoteka1, std::ifstream::in);
        std::ifstream ulaz2("oznake_gena.txt", std::ifstream::in);
        std::ofstream izlaz(datoteka2, std::ofstream::out);

        std::map<int, std::vector<std::string>> mapa;

        std::string linija1, linija2;
        int broj;
        std::vector<std::string> lista;
        std::set<int> klasteri;

  for (int i = 0; i < broj_linija; i++) {

                ulaz1 >> linija1;
    broj = std::stoi(linija1);

                ulaz2 >> linija2;

                if (klasteri.count(broj)) {
                    lista = mapa.at(broj);
                    lista.push_back(linija2);
                } else {
                    lista.push_back(linija2);
                    klasteri.insert(broj);
                }

```

```
48                  mapa[broj] = lista;
49                  lista = {};
50          }
51
52          std::map<int, std::vector<std::string>>::iterator itr;
53          for (itr = mapa.begin(); itr != mapa.end(); ++itr) {
54              std::vector<std::string> geni = itr->second;
55              izlaz << itr->first << ": "<< std::endl;
56
57              std::copy(geni.cbegin(), geni.cend(), std::
    ostream_iterator<std::string>(izlaz, " "));
58              izlaz << std::endl;
59          }
60
61    ulaz1.close();
62    ulaz2.close();
63          izlaz.close();
64    return 0;
65  }
```

Listing 4: Ubacivanje gena u klastere - ubacivanje_u_klastere.cpp

Da bi se kod preveo i pokrenuo, potrebno je pozicionirati se u direktorijum *IP2_ Vladana_ Djordjevic_ 89_ 2015/CLUTO/Geni* i kopirati datoteku rbr_corr_i1 .20 (a ona se nalazi u *IP2_ Vladana_ Djordjevic_ 89_ 2015/CLUTO/Geni/Podela po klasterima*) u taj isti direktorijum.

Kod se prevodi i pokreće na sledeći način:

```
1    g++ ubacivanje_u_klastere.cpp -o uk
```

```
1    ./uk rbr_corr_i1.20 izlaz_klasteri_rbr_corr_i1_20.txt 14428
```

Izlaz iz ovog programa je datoteka pod nazivom *izlaz_ klasteri_ rbr_ corr_ i1_ 20.txt* u kome je naveden broj klastera, a potom lista gena koja mu pripada.

U tabeli 2 možemo videti koji geni se nalaze u kom klasteru. Ispisana su samo četiri gena iz svakog klastera, jer je nepraktično ispisati ih sve.

Sada ćemo prikazati rezultate klasterovanja za broj klastera 50.

Tabela 2: Klasterovani geni programom vcluster - prva kombinacija - izlaz_klasteri_rbr_corr_i1_20.txt

Klaster	**Geni**
0	hg38_ABCA4, hg38_AKR1D1, hg38_CETP, hg38_DLGAP2 ...
1	hg38_ANAPC11, hg38_AP2S1, hg38_APRT, hg38_ARPC2 ...
2	hg38_BTBD8, hg38_CASP1, hg38_CDRT1, hg38_CFB ...
3	hg38_ACTB, hg38_ACTG1, hg38_ADI1, hg38_ADRM1 ...
4	hg38_ACP1, hg38_ACTN4, hg38_ADSL, hg38_AHCY ...
5	hg38_ACOT13, hg38_AGPAT2, hg38_AMZ2, hg38_ANAPC16 ...
6	hg38_ANLN, hg38_ANP32E, hg38_ARHGAP11A, hg38_ARHGEF39 ...
7	hg38_ADAMTSL5, hg38_AHSG, hg38_ALDH4A1, hg38_ARHGAP25 ...
8	hg38_ABCC3, hg38_ABHD12, hg38_ABRACL, hg38_ACADVL ...
9	hg38_AAMP, hg38_ABCG2, hg38_ACAA2, hg38_ACAT1 ...
10	hg38_AAED1, hg38_ABI2, hg38_ACBD3, hg38_ACBD6 ...
11	hg38_ABCC1, hg38_ABCC2, hg38_ABHD3, hg38_ACAA1 ...
12	hg38_AAAS, hg38_AAGAB, hg38_AARS, hg38_AASDHPPT ...
13	hg38_A1BG, hg38_AACS, hg38_AAK1, hg38_AAMDC ...
14	hg38_AADAC, hg38_AANAT, hg38_ABCA12, hg38_ABCC12 ...
15	hg38_ABAT, hg38_ABCA1, hg38_ABCA3, hg38_ABCA5 ...
16	hg38_A1CF ,hg38_A4GALT, hg38_AARS2, hg38_AATK ...
17	hg38_ABCA2, hg38_ABCA7, hg38_ABCA8, hg38_ABCC11 ...
18	hg38_AARD, hg38_ABCA10, hg38_ABCB10, hg38_ABCB5 ...
19	hg38_AADAT, hg38_AAR2, hg38_AARSD1, hg38_ABCB8 ...

rbr_corr_50_i1.txt

```
vcluster CLUTO 2.1.1 Copyright 2001-03, Regents of the University of Minnesota

Matrix Information ------------------------------------------------------------
  Name: matrica.mat, #Rows: 14428, #Columns: 3368, #NonZeros: 48593504

Options -----------------------------------------------------------------------
  CLMethod=RBR, CRfun=I1, SimFun=CorrCoef, #Clusters: 50
  RowModel=None, ColModel=None, GrModel=SY-DIR, NNbrs=40
  Colprune=1.00, EdgePrune=-1.00, VtxPrune=-1.00, MinComponent=5
  CSType=Best, AggloFrom=0, AggloCRFun=I1, NTrials=10, NIter=10

Solution ----------------------------------------------------------------------

------------------------------------------------------------------------
50-way clustering: [I1=8.50e+02] [14428 of 14428]
------------------------------------------------------------------------
cid  Size  ISim  ISdev   ESim  ESdev
------------------------------------------------------------------------
  0     5 +0.704 +0.091 +0.003 +0.002
  1    74 +0.776 +0.042 +0.112 +0.006
  2     7 +0.641 +0.165 -0.000 +0.001
```

```
  3    10 +0.564 +0.151 +0.003 +0.001
  4     8 +0.511 +0.148 +0.004 +0.002
  5    33 +0.538 +0.110 +0.079 +0.019
  6   146 +0.465 +0.050 +0.089 +0.010
  7    14 +0.339 +0.128 +0.003 +0.002
  8    23 +0.310 +0.101 +0.003 +0.007
  9   127 +0.381 +0.054 +0.078 +0.011
 10   153 +0.360 +0.047 +0.077 +0.010
 11    81 +0.312 +0.047 +0.068 +0.012
 12    28 +0.242 +0.088 +0.001 +0.002
 13    18 +0.235 +0.102 +0.005 +0.003
 14    90 +0.246 +0.066 +0.044 +0.014
 15   135 +0.243 +0.043 +0.052 +0.010
 16    20 +0.178 +0.067 +0.002 +0.003
 17    27 +0.174 +0.074 +0.005 +0.003
 18   311 +0.206 +0.025 +0.059 +0.007
 19   122 +0.158 +0.040 +0.046 +0.012
 20   332 +0.160 +0.020 +0.050 +0.007
 21    44 +0.109 +0.049 +0.007 +0.008
 22    59 +0.081 +0.037 +0.004 +0.004
 23   305 +0.114 +0.017 +0.044 +0.007
 24   132 +0.086 +0.022 +0.026 +0.009
 25    61 +0.065 +0.032 +0.006 +0.004
 26   132 +0.071 +0.025 +0.015 +0.010
 27   478 +0.082 +0.012 +0.033 +0.006
 28    84 +0.054 +0.023 +0.007 +0.005
 29   696 +0.070 +0.011 +0.034 +0.005
 30   118 +0.041 +0.019 +0.008 +0.004
 31   137 +0.041 +0.021 +0.008 +0.005
 32   168 +0.032 +0.017 +0.009 +0.005
 33   239 +0.029 +0.014 +0.010 +0.005
 34   290 +0.024 +0.008 +0.010 +0.005
 35   740 +0.036 +0.006 +0.024 +0.004
 36   306 +0.021 +0.007 +0.010 +0.004
 37   623 +0.032 +0.007 +0.022 +0.005
 38   639 +0.031 +0.006 +0.021 +0.005
 39   506 +0.020 +0.006 +0.013 +0.005
 40   665 +0.025 +0.005 +0.018 +0.005
 41   458 +0.010 +0.003 +0.005 +0.004
 42   504 +0.011 +0.003 +0.006 +0.004
 43   477 +0.009 +0.003 +0.006 +0.004
 44   522 +0.008 +0.003 +0.005 +0.004
 45   577 +0.007 +0.003 +0.004 +0.004
 46   802 +0.003 +0.001 +0.002 +0.003
 47   818 +0.003 +0.001 +0.002 +0.003
 48   761 +0.005 +0.002 +0.005 +0.004
 49  1323 +0.012 +0.003 +0.014 +0.004
-------------------------------------------------------------------------

Timing Information ----------------------------------------------------------
   I/O:                                  7.236 sec
   Clustering:                         556.946 sec
   Reporting:                            2.404 sec
```

Da bi se kod preveo i pokrenuo, potrebno je pozicionirati se u direktorijum *IP2_ Vladana_ Djordjevic_ 89_ 2015/CLUTO/Geni* i kopirati datoteku rbr_corr_i1.50 (a ona se nalazi u *IP2_ Vladana_ Djordjevic_ 89_ 2015/CLUTO/Geni/Podela po klasterima*) u taj isti direktorijum.

Pozivamo prethodni program, ali sa drugim parametrima:

```
./uk rbr_corr_i1.50 izlaz_klasteri_rbr_corr_i1_50.txt 14428
```

Izlazna datoteka sadrži oznaku klastera, a potom listu gena koji mu pripadaju. U tabeli 3 smo naveli samo prva 4 gena iz svakog klastera jer bi ispis svih bio nepraktičan.

Tabela 3: Klasterovani geni programom vcluster - druga kombinacija - izlaz_klasteri_rbr_corr_i1_50.txt

Klaster	**Geni**
0	hg38_CD101, hg38_DMRTA2, hg38_MPPED2, hg38_MYO1G ...
1	hg38_EEF1A1, hg38_EEF1B2, hg38_EEF1D, hg38_FAU ...
2	hg38_ADCY10, hg38_CMYA5, hg38_ISM1, hg38_PKD2L1 ...
3	hg38_ABCA4, hg38_AKR1D1, hg38_CETP, hg38_DLGAP2 ...
4	hg38_AC027796.3, hg38_IFIT2, hg38_IL9R, hg38_METRNL ...
5	hg38_ATP5MD, hg38_ATP5MPL, hg38_BTF3, hg38_C8orf59 ...
6	hg38_ANAPC11, hg38_ANAPC16, hg38_AP2S1, hg38_APRT ...
7	hg38_ARHGEF5, hg38_CD72, hg38_CST2, hg38_CYP2C9 ...
8	hg38_BTBD8, hg38_CASP1, hg38_CDRT1, hg38_CFB ...
9	hg38_ACTB, hg38_ACTG1, hg38_ADI1, hg38_ADRM1 ...
...	...

3.5 Klasterovanje ćelija korišćenjem CLUTO alata

Da bismo klasterovali ćelije neophodno je da transponujemo matricu, međutim, to je već odrađeno u pretprocesiranju. Potrebno je datoteku sa transponovanom matricom, odnosno, transponovana_matrica.txt kopiratu u direktorijum gde se nalazi vcluster program a potom isprobavati različite kombinacije parametara.

U tabeli 4 možemo videti sve isprobane kombinacije parametara.

Tabela 4: Sve isprobane kombinacije parametara prilikom klasterovanja ćelija programom vcluster

Metoda	**Funkcija sličnosti**	**Funkcija kriterijuma**	**Broj klastera**
agglo	corr	g1	50
agglo	cos	g1	20
agglo	cos	g1	50
rb	cos	i2	20
rbr	corr	g1	10
rbr	corr	g1	20
rbr	corr	g1	50

Proces je isti kao i za gene, kao izlaz dobijamo dve datoteke. Navešćemo datoteke i naredbe u terminalu kojima su dobijene, i datoteke i naredbe su u redosledu datom u tabeli. Iznad svake komande nalaze se dve datoteke, prva se može naći u direktorjijumu *IP2_ Vladana_ Djordjevic_ 89_ 2015/CLUTO/Celije/Izlaz kluto* a druga u direktorijumu *IP2_ Vladana_ Djordjevic_ 89_ 2015/CLUTO/Celije/Podela po klasterima.*

agglo_ corr_ 50_ g1.txt i *agglo_ corr_ g1.50*

```
./vcluster -clmethod=agglo -sim=corr -crfun=g1
transponovana_matrica.txt 50
```

agglo_ cos_ 20_ g1.txt i *agglo_ cos_ g1.20*

```
./vcluster -clmethod=agglo -sim=cos -crfun=g1
transponovana_matrica.txt 20
```

agglo_ cos_ 50_ g1.txt i *agglo_ cos_ g1.50*

```
./vcluster -clmethod=agglo -sim=cos -crfun=g1
transponovana_matrica.txt 50
```

rb_ cos_ 20_ i2.txt i *rb_ cos_ i2.20*

```
./vcluster -clmethod=rb -sim=cos -crfun=i2
transponovana_matrica.txt 20
```

rbr_corr_10_g1.txt i *rbr_corr_g1.10*

```
./vcluster -clmethod=rbr -sim=corr -crfun=g1
transponovana_matrica.txt 10
```

rbr_corr_20_g1.txt i *rbr_corr_g1.20*

```
./vcluster -clmethod=rbr -sim=corr -crfun=g1
transponovana_matrica.txt 20
```

rbr_corr_50_g1.txt i *rbr_corr_g1.50*

```
./vcluster -clmethod=rbr -sim=corr -crfun=g1
transponovana_matrica.txt 50
```

Kada pogledamo statistike svih kombinacija koje smo dobili videćemo da je svuda ogromna sličnost unutar samog klastera, što je odlično, ali je isto tako i ogromna sličnost između samih klastera, što nije toliko dobro. Ono što se izdvaja je kombinacija rbr + corr + g1 gde su elementi po klasterima brojčano odlično raspoređeni, odnosno, svaki klaster ima približan broj elemenata. Zato ćemo tu kombinaciju izabrati kao najbolju. Prikazaćemo kombinaciju sa samo 10 klastera jer je najlakša za prikaz.

rbr_corr_10_g1.txt

```
vcluster CLUTO 2.1.1 Copyright 2001-03, Regents of the University of Minnesota

Matrix Information ------------------------------------------------------------
  Name: transponovana_matrica.txt, #Rows: 3368, #Columns: 14428, #NonZeros: 48593504

Options -----------------------------------------------------------------------
  CLMethod=RBR, CRfun=G1, SimFun=CorrCoef, #Clusters: 10
  RowModel=None, ColModel=None, GrModel=SY-DIR, NNbrs=40
  Colprune=1.00, EdgePrune=-1.00, VtxPrune=-1.00, MinComponent=5
  CSType=Best, AggloFrom=0, AggloCRFun=G1, NTrials=10, NIter=10

Solution ----------------------------------------------------------------------

------------------------------------------------------------------------
10-way clustering: [G1=9.80e+01] [3368 of 3368]
------------------------------------------------------------------------
cid  Size  ISim  ISdev   ESim  ESdev
------------------------------------------------------------------------
  0   332 +0.975 +0.005 +0.938 +0.010
  1   335 +0.974 +0.006 +0.946 +0.008
  2   336 +0.971 +0.006 +0.946 +0.009
  3   336 +0.970 +0.006 +0.947 +0.008
  4   336 +0.972 +0.008 +0.950 +0.009
  5   337 +0.966 +0.006 +0.945 +0.009
  6   337 +0.959 +0.008 +0.939 +0.013
```

```
  7   339 +0.957 +0.011 +0.942 +0.015
  8   339 +0.955 +0.010 +0.940 +0.015
  9   341 +0.891 +0.087 +0.886 +0.112
-----------------------------------------------------------------------

Timing Information ----------------------------------------------------------
   I/O:                                  7.235 sec
   Clustering:                         307.466 sec
   Reporting:                            2.354 sec
```

Sada je potrebno ubaciti ćelije u klastere. Pozicionirati se u direktorijum *IP2_Vladana_Djordjevic_89_2015/CLUTO/Celije*, u njega prebaciti datoteku rbr_corr_g1.10 i pokrenuti kod dat listingom 5 na sledeći način:

```
g++ ubacivanje_u_klastere.cpp -o uk
```

```
./uk rbr_corr_g1.10 izlaz_klasteri_rbr_corr_g1_10.txt 3368
```

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <string>
#include <map>
#include <vector>
#include <set>
#include <iterator>
#include <numeric>

int main(int argc, const char* argv[]) {

  if (argc != 4) {
    std::cout << "Program se pokrece na sledeci nacin: " << std::endl;
    std::cout << "./uk ulazna_dat izlazna_dat broj_linija" << std::endl;
    std::exit(EXIT_FAILURE);
  }

  std::string datoteka1 = argv[1];
  std::string datoteka2 = argv[2];
  int broj_linija = std::stoi(argv[3]);

  std::ifstream ulaz1(datoteka1, std::ifstream::in);
        std::ofstream izlaz(datoteka2, std::ofstream::out);

  std::vector<int> celije(broj_linija) ;
  std::iota (std::begin(celije), std::end(celije), 1);

        std::map<int, std::vector<int>> mapa;
```

```
30
31          std::string linija1;
32          int broj_klastera, broj_celije;
33          std::vector<int> lista;
34          std::set<int> klasteri;
35
36    for (int i = 0; i < broj_linija; i++) {
37
38                  ulaz1 >> linija1;
39      broj_klastera = std::stoi(linija1);
40
41                  broj_celije = celije.at(i);
42
43                  if (klasteri.count(broj_klastera)) {
44                      lista = mapa.at(broj_klastera);
45                      lista.push_back(broj_celije);
46                  } else {
47                      lista.push_back(broj_celije);
48                      klasteri.insert(broj_klastera);
49                  }
50
51                  mapa[broj_klastera] = lista;
52                  lista = {};
53          }
54
55          std::map<int, std::vector<int>>::iterator itr;
56          for (itr = mapa.begin(); itr != mapa.end(); ++itr) {
57              std::vector<int> celije = itr->second;
58              izlaz << itr->first << ": "<< std::endl;
59
60              std::copy(celije.cbegin(), celije.cend(), std::
     ostream_iterator<int>(izlaz, " "));
61              izlaz << std::endl;
62          }
63
64    ulaz1.close();
65          izlaz.close();
66    return 0;
67  }
```

Listing 5: Rasporedjivanje ćelija po klasterima - ubacivanje_u_klastere.cpp

Kao izlaz dobijamo datoteku izlaz_klasteri_rbr_corr_g1_10.txt. U tabeli 5 za svaki klaster prikazano je prvih 10 ćelija.

Tabela 5: Klasterovane ćelije programom vcluster - izlaz_klasteri_rbr_corr_g1_10.txt

Klaster	**Ćelije**
0	5, 6, 16, 34, 36, 37, 38, 48, 58, 65 ...
1	1, 4, 7, 22, 33, 42, 49, 61, 72, 83 ...
2	21, 43, 56, 77, 78, 89, 106, 116, 130, 150 ...
3	10, 17, 29, 35, 40, 52, 57, 62, 64, 86 ...
4	11, 14, 23, 32, 39, 41, 46, 66, 67, 73 ...
5	13, 18, 27, 47, 50, 75, 90, 100, 123, 125 ...
6	12, 19, 20, 25, 28, 44, 45, 51, 55, 63 ...
7	15, 24, 26, 30, 31, 53, 80, 92, 101, 104 ...
8	2, 54, 59, 60, 68, 69, 74, 79, 81, 82 ...
9	3, 8, 9, 76, 93, 94, 99, 102, 113, 119 ...

4 Hijerarhijsko klasterovanje

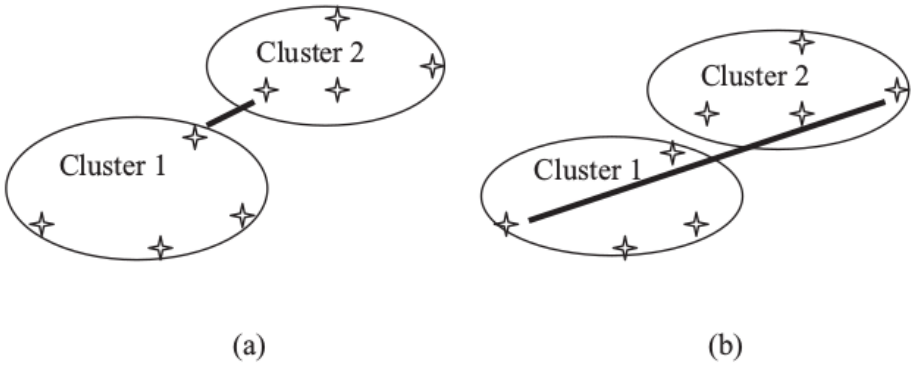
Hijerarhijsko klasterovanje je skup ugnježdenih klastera organizovan u obliku hijerarhijskog stabla (koje nazivamo dendogram) i daje dublje informacije o samoj strukturi podskupova. Broj klastera zavisi od dubine na kojoj posmatramo stablo. Odnosno, sekući dendogram na nekom nivou dobijamo klastere.

Prilikom hijerarhijskog klasterovanja ne navodimo broj klastera kao ulaz. Ulaz predstavlja (X, s), gde je X skup uzoraka, a s mera sličnosti. Izlaz je hijerarhija klastera. Većina procedura za hijerarhijsko klasterovanje nije zasnovana na konceptu optimizacije i cilj je pronaći približna, suboptimalna rešenja, koristeći iteracije za poboljšanje deljenja dok ne konvergira. Algoritmi za hijerarhijsko klasterovanje su podeljeni u dve kategorije, deljivi i sakupljajući algoritmi. Deljivi algoritmi započinju od celog skupa uzorka X i dele skup u podskupove, onda svaki podskup deli u manje podskupove itd. Dakle, deljivi algoritmi generišu niz particija koje su uređene od grubljih ka finijim. Sakupljajući algoritmi prvo posmatraju svaki objekat kao inicijalni klaster. Klasteri se spajaju u grublju particiju i proces spajanja se nastavlja dok se ne dođe do trivijalne particije - svi objekti su u jednom velikom klasteru. Ovakav proces klasterovanja je od dna ka vrhu, gde particije idu od finijih ka grubljim [1].

Uglavnom se sakupljajući algoritmi češće koriste u aplikacijama nego deljivi algoritmi, stoga je u ovom radu predstavljeno sakupljajuće hijerarhijsko klasterovanje i odsad ćemo više pažnje posvetiti njemu.

Većina sakupljajućih algoritama je varijanta pojedinačne veze ili kompletne veze algoritama. Ove dve vrste algoritama se razlikuju samo u načinu određivanja sličnosti između parova klastera. Metoda pojedinačne veze, rastojanje između dva klastera je minimum rastojanja između svih parova uzoraka uzetih iz oba klastera (jedan element iz prvog klastera i jedan element iz drugog). Metoda kompletna veza, rastojanje između dva klastera je maksimum rastojanja između svih parova iz oba klastera. Grafička ilustracija ova dva metoda je prikazana na slici 2 [1].

U oba slučaja, dva klastera se spajaju i time formiraju veći klaster na osnovu



Slika 2: Hijerarhijsko klasterovanje: (a) Pojedinačna veza (b) Kompletna veza

kriterijuma minimalnog rastojanja. Iako je pojedinačna računarski jednostavnija, sa praktičnog stanovišta je ustanovljeno da kompletna veza algoritmi proizvode korisnije hijerarhije u većini aplikacija. Za oba slučaja, osnovni koraci sakupljajućeg algoritma klasterovanja su isti. Ovi koraci su [1]:

1. Postaviti svaki uzorak u sopstveni klaster. Napraviti listu rastojanja između klastera za svaki neuređeni par uzoraka i sortirati listu u rastućem poretku.

2. Proći kroz sortiranu listu rastojanja i napraviti za svaku različitu vrednost praga d_k graf uzoraka gde su parovi uzoraka koji su bliži od d_k spojeni u novi klaster granom grafa. Ako su svi uzorci čvorovi povezanog grafa, zaustaviti se. U suprotnom, ponoviti ovaj korak.

3. Izlaz iz algoritma je ugnježdena hijerarhija grafova, koja može biti posečena na željenom nivou gde ima razlikovanja i time formira klastere koji se identifikuju spojenim komponentama u odgovarajućim podgrafima.

Za primenu sakupljajućeg hijerarhijskog klasterovanja na naš skup podataka koristićemo programski jezik Python, kao i metod *AgglomerativeClustering* iz biblioteke *sklearn.cluster* [4].

Metod *AgglomerativeClustering* ima više parametara, mnogi od njih su opcioni. Ovde će biti navedeni samo oni koji će biti korišćeni u našem istraživanju. Parametri su sledeći [4]:

1. **affinity:** Definiše metriku koja se koristi da se izračuna linkage. Moguće vrednosti su:

   - **euclidean:** Euklidsko rastojanje. Ovo je podrazumevana vrednost.
   - **manhattan:** Menheten rastojanje.
   - **cosine:** Kosinusno rastojanje.
   - **precomputed:** Ukoliko je ovaj argument postavljen onda je potrebno proslediti matricu rastojanja (umesto matrice sličnosti) kao ulazni parametar za metod fit.

- **I1:** $maximize \sum_{i=1}^{k} \frac{1}{n_i} (\sum_{v,u \in S_i} sim(v,u))$
- **I2:** $maximize \sum_{i=1}^{k} \frac{1}{n_i} \sqrt{\sum_{v,u \in S_i} sim(v,u)}$

2. **linkage:** Definiše koji kriterijum povezivanja se koristi, a kriterijum povezivanja određuje koje rastojanje će se koristiti između posmatranih klastera. Algoritam će spojiti parove klastera koji minimizuju ovaj kriterijum. Moguće vrednosti su:
   - **ward:** Minimizuje varijansu klastera koji se spajaju. Ovo je podrazumevana vrednost.
   - **average:** Koristi prosečnu vrednost rastojanja između svaka dva uzorka iz klastera.
   - **complete/maximum:** Maksimalno rastojanje između svaka dva uzorka iz klastera.
   - **single:** Minimalno rastojanje između svaka dva uzorka iz klastera.
3. **distance_threshold:** Prag za rastojanje iznad kog klasteri neće biti spajani. Ako koristimo ovaj parametar onda n_clusters mora biti None.

4.1 Hijerarhijsko klasterovanje gena

Da bismo pronašli odgovarajuće klasterovanje potrebno je da ispitamo nekoliko kombinacija parametara - affinity, linkage i distance_threshold. Program dat listingom 6 nam upravo to omogućava. Potrebno je pozicionirati se u direktorijum *IP2_Vladana_Djordjevic_89_2015/Hijerarhijsko/Geni* i pokrenuti sledeću naredbu:

```
python3 geniHijerarhijsko.py pretprocesirano.txt 3369 affinity
 linkage distance_threshold
```

Uz izmenu da se navedu željene vrednosti parametara za poslednja tri argumenta.

```
import sys
import numpy as np
from sklearn.cluster import AgglomerativeClustering
import collections as cs

def main():

  if (len(sys.argv) != 6):
    print("Program se pokrece na sledeci nacin:")
    print("python3 naziv_programa.py ulazna_dat broj_kolona metrika
    rastojanje prag")
    exit(1)

  ulazna_datoteka = sys.argv[1]
  broj_kolona = int(sys.argv[2])
  metrika = sys.argv[3]
```

```
16    rastojanje = sys.argv[4]
17    prag = int(sys.argv[5])
18
19    kolone = [x for x in range(1, broj_kolona)]
20    podaci = np.loadtxt(ulazna_datoteka, delimiter = ",", usecols =
       kolone)
21
22    agglo = AgglomerativeClustering(n_clusters = None, affinity =
       metrika, linkage = rastojanje, distance_threshold = prag).fit(
       podaci)
23
24    klasteri = agglo.labels_
25
26    with open("klasteriHijerarhijsko_" + str(metrika) + "_" + str(
       rastojanje) + "_" + str(prag) + ".txt", "w") as datoteka:
27      for x in klasteri:
28        datoteka.write(str(x) + "\n")
29
30    velicina = cs.Counter(klasteri).most_common()
31    with open("klasteriHijerarhijsko_id_velicina_" + str(metrika) + "
       _" + str(rastojanje) + "_" + str(prag) + ".txt", "w") as
       datoteka_id_velicina:
32      for x in velicina:
33        datoteka_id_velicina.write("ID klastera: {} -> Velicina
       klastera: {}\n".format(x[0], x[1]))
34
35
36  if __name__ == '__main__':
37    main()
```

Listing 6: Klasterovanje gena - geniHijerarhijsko.py

U ovom istraživanju je ispitano nekoliko kombinacija parametara, a ispod će biti navedene izlazne datoteke koje su dobijene, kao i komande u terminalu koje su pokrenute da bi one bile dobijene. Sve dobijene datoteke se mogu naći u direktorijumu *IP2_Vladana_Djordjevic_89_2015/Hijerarhijsko/Geni*.

klasteriHijerarhijsko_euclidean_average_2.txt i *klasteriHijerarhijsko_id_velicina_euclidean_average_2.txt*

```
1   python3 geniHijerarhijsko.py pretprocesirano.txt 3369 euclidean
     average 2
```

klasteriHijerarhijsko_euclidean_average_5.txt i *klasteriHijerarhijsko_id_velicina_euclidean_average_5.txt*

```
1   python3 geniHijerarhijsko.py pretprocesirano.txt 3369 euclidean
     average 5
```

klasteriHijerarhijsko_euclidean_average_10.txt i *klasteriHijerarhijsko_id_velicina_euclidean_average_10.txt*

```
1   python3 geniHijerarhijsko.py pretprocesirano.txt 3369 euclidean
     average 10
```

klasteriHijerarhijsko_euclidean_average_50.txt i *klasteriHijerarhijsko_id_velicina_euclidean_average_50.txt*

```
python3 geniHijerarhijsko.py pretprocesirano.txt 3369 euclidean average 50
```

klasteriHijerarhijsko_euclidean_complete_50.txt i *klasteriHijerarhijsko_id_velicina_euclidean_complete_50.txt*

```
python3 geniHijerarhijsko.py pretprocesirano.txt 3369 euclidean complete 50
```

klasteriHijerarhijsko_euclidean_complete_100.txt i *klasteriHijerarhijsko_id_velicina_euclidean_complete_100.txt*

```
python3 geniHijerarhijsko.py pretprocesirano.txt 3369 euclidean complete 100
```

klasteriHijerarhijsko_euclidean_complete_500.txt i *klasteriHijerarhijsko_id_velicina_euclidean_complete_500.txt*

```
python3 geniHijerarhijsko.py pretprocesirano.txt 3369 euclidean complete 500
```

klasteriHijerarhijsko_euclidean_complete_100.txt i *klasteriHijerarhijsko_id_velicina_euclidean_complete_100.txt*

```
python3 geniHijerarhijsko.py pretprocesirano.txt 3369 euclidean ward 100
```

klasteriHijerarhijsko_euclidean_ward_500.txt i *klasteriHijerarhijsko_id_velicina_euclidean_ward_500.txt*

```
python3 geniHijerarhijsko.py pretprocesirano.txt 3369 euclidean ward 500
```

klasteriHijerarhijsko_euclidean_ward_750.txt i *klasteriHijerarhijsko_id_velicina_euclidean_ward_750.txt*

```
python3 geniHijerarhijsko.py pretprocesirano.txt 3369 euclidean ward 750
```

klasteriHijerarhijsko_manhattan_complete_750.txt i *klasteriHijerarhijsko_id_velicina_manhattan_complete_750.txt*

```
python3 geniHijerarhijsko.py pretprocesirano.txt 3369 manhattan complete 750
```

klasteriHijerarhijsko_cosine_complete_750.txt i *klasteriHijerarhijsko_id_velicina_cosine_complete_750.txt*

```
python3 geniHijerarhijsko.py pretprocesirano.txt 3369 cosine complete 750
```

klasteriHijerarhijsko_cosine_complete_50.txt i *klasteriHijerarhijsko_id_velicina_cosine_complete_50.txt*

```
python3 geniHijerarhijsko.py pretprocesirano.txt 3369 cosine complete 50
```

Ako pogledamo sve datoteke koje u sebi sadrže broj i veličinu klastera videćemo da su svi rezultati loši. Ima mnogo klastera sa samo jednim elementom i jedan koji ima mnogo elemenata. U nekim situacijima je algoritam dao samo jedan element. Kao najmanje loš od ovih modela izdvajamo kombinaciju parametara euclidean + ward + 500. Klastera ima ukupno 129, a njihove veličine možete videti u navedenoj datoteci.

klasteriHijerarhijsko_id_velicina_euclidean_ward_500.txt

```
ID klastera: 45 -> Velicina klastera: 8522
ID klastera: 59 -> Velicina klastera: 2549
ID klastera: 25 -> Velicina klastera: 1599
ID klastera: 49 -> Velicina klastera: 629
ID klastera: 15 -> Velicina klastera: 332
ID klastera: 3 -> Velicina klastera: 237
ID klastera: 52 -> Velicina klastera: 128
ID klastera: 1 -> Velicina klastera: 82
ID klastera: 60 -> Velicina klastera: 82
ID klastera: 30 -> Velicina klastera: 36
ID klastera: 22 -> Velicina klastera: 30
ID klastera: 9 -> Velicina klastera: 20
ID klastera: 14 -> Velicina klastera: 15
ID klastera: 54 -> Velicina klastera: 12
ID klastera: 24 -> Velicina klastera: 9
ID klastera: 0 -> Velicina klastera: 7
ID klastera: 50 -> Velicina klastera: 6
ID klastera: 4 -> Velicina klastera: 5
ID klastera: 5 -> Velicina klastera: 4
ID klastera: 57 -> Velicina klastera: 4
ID klastera: 6 -> Velicina klastera: 3
ID klastera: 2 -> Velicina klastera: 2
ID klastera: 7 -> Velicina klastera: 2
```

```
ID klastera: 10 -> Velicina klastera: 2
ID klastera: 11 -> Velicina klastera: 2
ID klastera: 12 -> Velicina klastera: 2
ID klastera: 13 -> Velicina klastera: 2
ID klastera: 26 -> Velicina klastera: 2
ID klastera: 28 -> Velicina klastera: 2
ID klastera: 29 -> Velicina klastera: 2
ID klastera: 8 -> Velicina klastera: 1
ID klastera: 16 -> Velicina klastera: 1
ID klastera: 17 -> Velicina klastera: 1
ID klastera: 18 -> Velicina klastera: 1
ID klastera: 19 -> Velicina klastera: 1
ID klastera: 20 -> Velicina klastera: 1
ID klastera: 21 -> Velicina klastera: 1
ID klastera: 23 -> Velicina klastera: 1
ID klastera: 27 -> Velicina klastera: 1
ID klastera: 31 -> Velicina klastera: 1
ID klastera: 32 -> Velicina klastera: 1
ID klastera: 33 -> Velicina klastera: 1
ID klastera: 34 -> Velicina klastera: 1
ID klastera: 35 -> Velicina klastera: 1
ID klastera: 36 -> Velicina klastera: 1
ID klastera: 37 -> Velicina klastera: 1
ID klastera: 38 -> Velicina klastera: 1
ID klastera: 39 -> Velicina klastera: 1
ID klastera: 40 -> Velicina klastera: 1
ID klastera: 41 -> Velicina klastera: 1
ID klastera: 42 -> Velicina klastera: 1
ID klastera: 43 -> Velicina klastera: 1
ID klastera: 44 -> Velicina klastera: 1
ID klastera: 46 -> Velicina klastera: 1
ID klastera: 47 -> Velicina klastera: 1
ID klastera: 48 -> Velicina klastera: 1
ID klastera: 51 -> Velicina klastera: 1
ID klastera: 53 -> Velicina klastera: 1
ID klastera: 55 -> Velicina klastera: 1
ID klastera: 56 -> Velicina klastera: 1
ID klastera: 58 -> Velicina klastera: 1
ID klastera: 61 -> Velicina klastera: 1
ID klastera: 62 -> Velicina klastera: 1
ID klastera: 63 -> Velicina klastera: 1
ID klastera: 64 -> Velicina klastera: 1
ID klastera: 65 -> Velicina klastera: 1
ID klastera: 66 -> Velicina klastera: 1
ID klastera: 67 -> Velicina klastera: 1
ID klastera: 68 -> Velicina klastera: 1
ID klastera: 69 -> Velicina klastera: 1
ID klastera: 70 -> Velicina klastera: 1
ID klastera: 71 -> Velicina klastera: 1
ID klastera: 72 -> Velicina klastera: 1
ID klastera: 73 -> Velicina klastera: 1
ID klastera: 74 -> Velicina klastera: 1
ID klastera: 75 -> Velicina klastera: 1
ID klastera: 76 -> Velicina klastera: 1
ID klastera: 77 -> Velicina klastera: 1
```

```
ID klastera: 78 -> Velicina klastera: 1
ID klastera: 79 -> Velicina klastera: 1
ID klastera: 80 -> Velicina klastera: 1
ID klastera: 81 -> Velicina klastera: 1
ID klastera: 82 -> Velicina klastera: 1
ID klastera: 83 -> Velicina klastera: 1
ID klastera: 84 -> Velicina klastera: 1
ID klastera: 85 -> Velicina klastera: 1
ID klastera: 86 -> Velicina klastera: 1
ID klastera: 87 -> Velicina klastera: 1
ID klastera: 88 -> Velicina klastera: 1
ID klastera: 89 -> Velicina klastera: 1
ID klastera: 90 -> Velicina klastera: 1
ID klastera: 91 -> Velicina klastera: 1
ID klastera: 92 -> Velicina klastera: 1
ID klastera: 93 -> Velicina klastera: 1
ID klastera: 94 -> Velicina klastera: 1
ID klastera: 95 -> Velicina klastera: 1
ID klastera: 96 -> Velicina klastera: 1
ID klastera: 97 -> Velicina klastera: 1
ID klastera: 98 -> Velicina klastera: 1
ID klastera: 99 -> Velicina klastera: 1
ID klastera: 100 -> Velicina klastera: 1
ID klastera: 101 -> Velicina klastera: 1
ID klastera: 102 -> Velicina klastera: 1
ID klastera: 103 -> Velicina klastera: 1
ID klastera: 104 -> Velicina klastera: 1
ID klastera: 105 -> Velicina klastera: 1
ID klastera: 106 -> Velicina klastera: 1
ID klastera: 107 -> Velicina klastera: 1
ID klastera: 108 -> Velicina klastera: 1
ID klastera: 109 -> Velicina klastera: 1
ID klastera: 110 -> Velicina klastera: 1
ID klastera: 111 -> Velicina klastera: 1
ID klastera: 112 -> Velicina klastera: 1
ID klastera: 113 -> Velicina klastera: 1
ID klastera: 114 -> Velicina klastera: 1
ID klastera: 115 -> Velicina klastera: 1
ID klastera: 116 -> Velicina klastera: 1
ID klastera: 117 -> Velicina klastera: 1
ID klastera: 118 -> Velicina klastera: 1
ID klastera: 119 -> Velicina klastera: 1
ID klastera: 120 -> Velicina klastera: 1
ID klastera: 121 -> Velicina klastera: 1
ID klastera: 122 -> Velicina klastera: 1
ID klastera: 123 -> Velicina klastera: 1
ID klastera: 124 -> Velicina klastera: 1
ID klastera: 125 -> Velicina klastera: 1
ID klastera: 126 -> Velicina klastera: 1
ID klastera: 127 -> Velicina klastera: 1
ID klastera: 128 -> Velicina klastera: 1
```

S obzirom na to da je ovaj model loš i da ima veliki broj klastera nećemo prikazivati ćelije koje pripadaju svakom klasteru, ali se grupisane ćelije po klasterima

mogu naći u datoteci *spojeni_klasteri_i_geni_euclidean_ward_500.txt*.

Pokušajmo drugim pristupom. Pokušajmo da nacrtamo dendogram podataka i da sa njega zaključimo nešto o našim klasterima. Za to nam je potreban sledeći kod (Napomena: deo koda je preuzet od [5]).

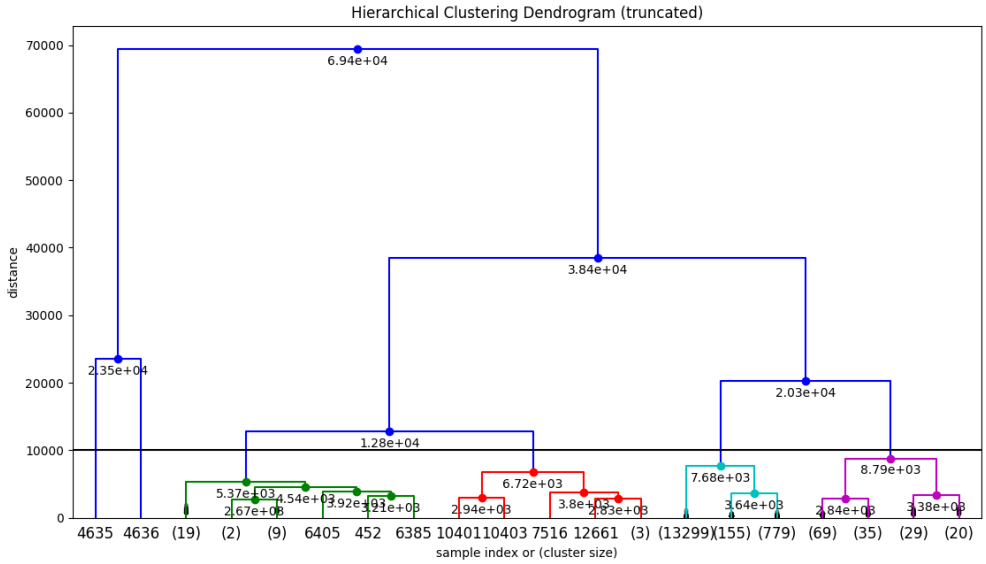
```
import scipy.cluster.hierarchy as shc
import numpy as np
import matplotlib.pyplot as plt

def fancy_dendrogram(*args, **kwargs):
    max_d = kwargs.pop('max_d', None)
    if max_d and 'color_threshold' not in kwargs:
        kwargs['color_threshold'] = max_d
    annotate_above = kwargs.pop('annotate_above', 0)

    ddata = shc.dendrogram(*args, **kwargs)

    if not kwargs.get('no_plot', False):
        plt.title('Hierarchical Clustering Dendrogram (truncated)')
        plt.xlabel('sample index or (cluster size)')
        plt.ylabel('distance')
        for i, d, c in zip(ddata['icoord'], ddata['dcoord'], ddata['color_list']):
            x = 0.5 * sum(i[1:3])
            y = d[1]
            if y > annotate_above:
                plt.plot(x, y, 'o', c=c)
                plt.annotate("%.3g" % y, (x, y), xytext=(0, -5),
                             textcoords='offset points',
                             va='top', ha='center')
        if max_d:
            plt.axhline(y=max_d, c='k')
    return ddata

def main():
  kolone = [x for x in range(1, 3369)]
  podaci = np.loadtxt("pretprocesirano.txt", delimiter = ",", usecols = kolone)
  plt.figure(figsize = (25, 10))

  dend = fancy_dendrogram(shc.linkage(podaci, method = 'ward'), truncate_mode = 'lastp', p = 20, show_contracted = True, annotate_above = 20, max_d = 10000)
  plt.show()

if __name__ == '__main__':
  main()
```

Listing 7: Dendogram geni - geniHijerarhijskoSaDendogramom.py

Za pokretanje navedenog programa potrebno je pozicionirati se u direktorijum *IP2_Vladana_Djordjevic_89_2015/Hijerarhijsko/Geni* i pokrenuti:

```
python3 geniHijerarhijskoSaDendogramom.py
```

Kao izlaz je dobijena slika 3 na kojoj smo presekli dendogram na rastojanju 10000 i kao rezultat dobijamo 6 klastera.



Slika 3: Dendogram hijerarhijskog klasterovanja gena - geniHijerarhijskoSaDendogramom.py

4.2 Hijerarhijsko klasterovanje ćelija

Klasterovanju ćelija pristupamo na isti način kao i klasterovanju gena, odnosno isprobavanjem kombinacija parametara. U ovom slučaju smo isprobali manje kombinacija jer je ovaj pristup dao dobre rezultate pa nije bilo potrebe nastavljati dalje sa isprobavanjima. U narednom delu teksta biće navedene datoteke koju su dobijene kao i komande iz terminala kojima su generisane. Da bi se komande izvršile potrebno je pozicionirati se u direktorijum *IP2_ Vladana_ Djordjevic_ 89_ 2015/Hijerarhijsko/ Celije*, i u istom tom direktorijumu biće smeštene gorepomenute datoteke.

klasteriHijerarhijsko_ euclidean_ complete_ 10.txt i *klasteriHijerarhijsko_ id _ velicina_ euclidean_ complete_ 10.txt*

```
python3 celijeHijerarhijsko.py transponovana_matrica.txt 14428
 euclidean complete 10
```

klasteriHijerarhijsko_ euclidean_ complete_ 500.txt i *klasteriHijerarhijsko_ id _ velicina_ euclidean_ complete_ 500.txt*

```
python3 celijeHijerarhijsko.py transponovana_matrica.txt 14428
 euclidean complete 500
```

klasteriHijerarhijsko_ euclidean_ complete_ 750.txt i *klasteriHijerarhijsko_ id _ velicina_ euclidean_ complete_ 750.txt*

```
python3 celijeHijerarhijsko.py transponovana_matrica.txt 14428
 euclidean complete 750
```

klasteriHijerarhijsko_ euclidean_ ward_ 750.txt i *klasteriHijerarhijsko_ id _ velicina_ euclidean_ ward_ 750.txt*

```
python3 celijeHijerarhijsko.py transponovana_matrica.txt 14428
 euclidean ward 750
```

klasteriHijerarhijsko_ euclidean_ ward_ 1000.txt i *klasteriHijerarhijsko_ id _ velicina_ euclidean_ ward_ 1000.txt*

```
python3 celijeHijerarhijsko.py transponovana_matrica.txt 14428
 euclidean ward 1000
```

klasteriHijerarhijsko_ euclidean_ ward_ 1250.txt i *klasteriHijerarhijsko_ id _ velicina_ euclidean_ ward_ 1250.txt*

```
python3 celijeHijerarhijsko.py transponovana_matrica.txt 14428
 euclidean ward 1250
```

klasteriHijerarhijsko_ euclidean_ ward_ 1500.txt i *klasteriHijerarhijsko_ id _ velicina_ euclidean_ ward_ 1500.txt*

```
python3 celijeHijerarhijsko.py transponovana_matrica.txt 14428
 euclidean ward 1500
```

klasteriHijerarhijsko_ euclidean_ ward_ 2000.txt i *klasteriHijerarhijsko_ id _ velicina_ euclidean_ ward_ 2000.txt*

```
python3 celijeHijerarhijsko.py transponovana_matrica.txt 14428
 euclidean ward 2000
```

Kada pogledamo sadržaj datoteka koje u sebi sadrže klastere i njihove veličine videćemo da smo generalno dobili dobre rezultate. Najbolje rezultate daje kombinacija euclidean + ward + 2000 koja kao izlaz vraća 21 klaster sa dobro raspoređenim brojem elemenata u klasterima. Klasteri i njihove veličine se mogu videti u datoteci:

`klasteriHijerarhijsko_id_velicina_euclidean_ward_2000.txt`

```
ID klastera: 15 -> Velicina klastera: 371
ID klastera: 10 -> Velicina klastera: 316
ID klastera: 14 -> Velicina klastera: 293
```

```
ID klastera: 20 -> Velicina klastera: 283
ID klastera: 3 -> Velicina klastera: 281
ID klastera: 11 -> Velicina klastera: 239
ID klastera: 12 -> Velicina klastera: 231
ID klastera: 13 -> Velicina klastera: 194
ID klastera: 18 -> Velicina klastera: 172
ID klastera: 8 -> Velicina klastera: 166
ID klastera: 4 -> Velicina klastera: 160
ID klastera: 0 -> Velicina klastera: 157
ID klastera: 6 -> Velicina klastera: 109
ID klastera: 17 -> Velicina klastera: 81
ID klastera: 2 -> Velicina klastera: 71
ID klastera: 5 -> Velicina klastera: 70
ID klastera: 16 -> Velicina klastera: 62
ID klastera: 9 -> Velicina klastera: 58
ID klastera: 1 -> Velicina klastera: 36
ID klastera: 7 -> Velicina klastera: 17
ID klastera: 19 -> Velicina klastera: 1
```

Sada je potrebno rasporediti ćelije u klastere. Pozivamo program *ubacivanje_u_klastere.cpp* na sledeći način:

```
g++ ubacivanje_u_klastere.cpp -o uk
```

```
./uk klasteriHijerarhijsko_euclidean_ward_2000.txt
 spojeni_klasteri_i_celije_euclidean_ward_2000.txt 3368
```

Kao izlaz dobijamo datoteku *spojeni_klasteri_i_celije_euclidean_ward_2000.txt*. U tabeli 6 možemo da vidim prvih 10 (ako ih ima toliko) ćelija iz svakog klastera.

Pokušajmo i pristup sa dendogramom. U direktorijumu *IP2_Vladana_Djordjevic_89_2015/Hijerarhijsko/Celije* pokrenuti naredbu:

```
python3 celijeHijerarhijskoSaDendogramom.py
```

Kao izlaz dobijamo sliku *celijeDendogram.png* koji se nalazi u istom direktorijumu. Presecanje dendograma je izvršeno na rastojanju 4500 i na taj način dobijamo 7 klastera.

Tabela 6: Klasterovane ćelije primenom hijerarhijskog klasterovanja - spojeni_klasteri_i_celije_euclidean_ward_2000.txt

Klaster	**Ćelije**
0	61, 91, 92, 143, 226, 249, 254, 265, 291, 294 ...
1	11, 88, 108, 287, 366, 441, 562, 588, 613, 629 ...
2	27, 199, 217, 221, 255, 274, 345, 348, 354, 359 ...
3	5, 19, 43, 44, 55, 59, 65, 70, 83, 100 ...
4	57, 58, 64, 73, 87, 115, 118, 122, 144, 165 ...
5	49, 84, 181, 201, 234, 240, 263, 264, 292, 431 ...
6	39, 81, 212, 230, 256, 278, 293, 300, 397, 422 ...
7	332, 401, 730, 954, 1485, 1667, 1732, 1739, 1784, 2038 ...
8	8, 45, 67, 76, 125, 154, 160, 189, 192, 193 ...
9	77, 140, 155, 190, 228, 295, 374, 413, 457, 568 ...
10	10, 29, 31, 32, 40, 66, 72, 78, 86, 121 ...
11	9, 12, 14, 25, 51, 63, 93, 99, 128, 151 ...
12	3, 20, 24, 30, 82, 98, 104, 129, 132, 134 ...
13	1, 4, 16, 22, 33, 37, 38, 103, 116, 146 ...
14	7, 28, 35, 41, 46, 47, 50, 52, 53, 54 ...
15	15, 17, 21, 23, 56, 68, 69, 89, 90, 95 ...
16	48, 71, 162, 277, 316, 350, 364, 392, 407, 415 ...
17	2, 13, 18, 74, 94, 127, 141, 245, 352, 378 ...
18	60, 75, 80, 126, 147, 204, 209, 223, 224, 231 ...
19	1650
20	6, 26, 34, 36, 42, 85, 106, 110, 117, 131 ...

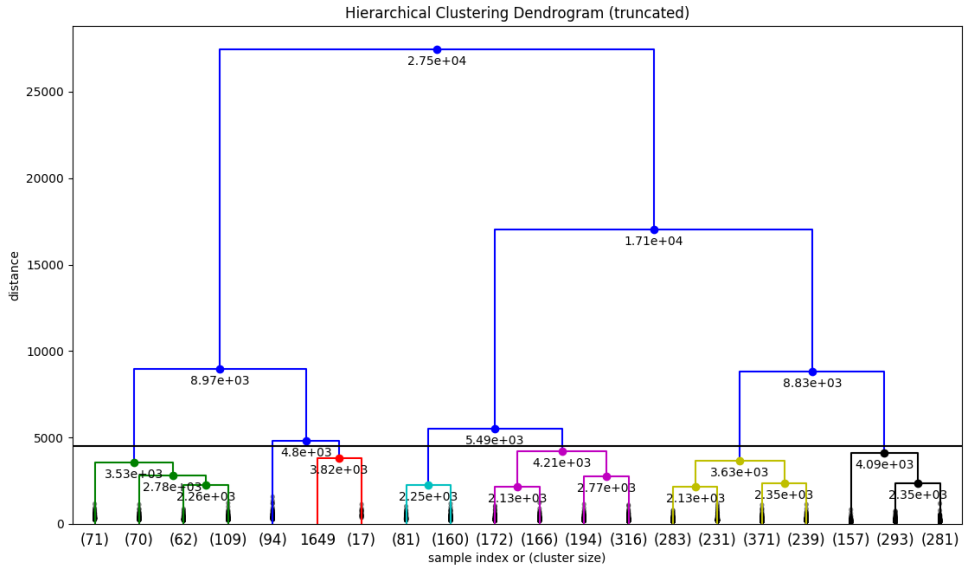
5 Klasterovanje korišćenjem algoritma DBSCAN

Klasterovanje zasnovano na gustini posmatra klastere kao guste regione objekata u prostoru podataka koji su razdvojeni regionima male gustine (šuma). Ovi regioni mogu imati proizvoljan oblik. Ključni koncepti ovog pristupa su gustina i povezanost koji su izračunati lokalnom raspodelom najbližih suseda. Algoritam DBSCAN cilja podatke niske dimenzije je glavni predstavnik ove vrste klasterovanja. Glavni razlog što DBSCAN prepoznaje klastere je taj da unutar svakog klastera imamo gustinu tačaka koja je značajno veća nego izvan klastera. Gustina tačaka u regionima gde se nalazi šum je je manja od gustine unutar bilo kog klastera [1].

DBSCAN algoritam ima dva ulazna parametra: veličina susedstva (ϵ) i minimalan broj tačaka u klasteru (m). Ideja algoritma je da, za svaku tačku u klasteru, susedstvo datog poluprečnika ϵ mora da sadrži barem m tačaka, tj. gustina susedstva mora da pređe definisani prag.

Proces klasterovanja je zasnovan na klasifikaciji tačaka u skupu kao tačke u jezgru klastera, tačke na granici klastera i šum [1].

- Tačka je tačka jezgra klaster ako ima više od zadatog broja tačaka (m) u susedstvu poluprečnika ϵ. Ove tačke čine unutrašnjost klastera.



Slika 4: Dendogram hijerarhijskog klasterovanja ćelija - celijeHijerarhijskoSaDendogramom.py

- Tačka na granici klastera ima manje od m tačaka u susedstvu poluprečnika ϵ, ali je sused tačke jezgra.
- Šum je bilo koja tačka koja nije ni tačka jezgra ni tačka na granici.

Idealno bi bilo ako bismo znali odgovarajuće vrednosti parametara ϵ i m za svaki klaster. Ali ne postoji lak način da se to zna unapred za svaki klaster. Stoga, DBSCAN koristi globalne vrednosti za ϵ i m, tj. koristi iste vrednosti za sve klastere. Takođe, brojni eksperimenti su pokazali da se DBSCAN klasteri za $m > 4$ ne razlikuju značajno od slučaja kad je $m = 4$, a algoritam u prvom slučaju ima mnogo više izračunavanja. Zato se praksi često koristi $m = 4$.

Glavni koraci DBSCAN algoritma su [1]:

- Proizvoljno bira tačku p.
- Dohvata sve tačke koje su dostižne iz p u pogledu na ϵ i m.
- Ako je p tačka jezgra pravi se novi klaster ili se proširuje već postojeći.
- Ako je p tačka na granici klastera nijedna tačka nije dostižna iz p, pa DBSCAN bira narednu tačku iz skupa.
- Nastavlja proces sa svim tačkama u skupu sve dok sve tačke nisu obrađene.
- S obzirom na to da se koriste globalne vrednosti za ϵ i m, DBSCAN može da spoji dva klastera u jedan, ako su oni ”blizu” jedan drugom. Blizu su ako je rastojanje između dva klastera manje od ϵ.

Glavne prednosti DBSCAN algoritma su sledeće [1]:

1. DBSCAN ne zahteva da se unese željeni broj klastera, za razliku od KMeans i ostalih popularnih algoritama.
2. DBSCAN može da pronađe klastere proizvoljnog oblika.
3. DBSCAN ima pojam o šumu i eliminiše elemente van granica iz klastera.
4. DBSCAN zahteva samo dva parametra i većinom je neosetljiv na uređenje tačaka u skupu.

5.1 DBSCAN klasterovanje gena

Kod za klasterovanje gena DBSCAN algoritmom dat je listingom 8.

```
import sys
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import DBSCAN
import collections as cs

def main():

  if (len(sys.argv) != 5):
    print("Program se pokrece na sledeci nacin:")
    print("python3 naziv_programa.py ulazna_dat broj_kolona epsilon
    min_broj_tacaka")
    exit(1)

  ulazna_datoteka = sys.argv[1]
  broj_kolona = int(sys.argv[2])
  eps = int(sys.argv[3])
  m = int(sys.argv[4])

  kolone = [x for x in range(1, broj_kolona)]
  podaci = np.loadtxt(ulazna_datoteka, delimiter = ",", usecols =
   kolone)

  ss = StandardScaler().fit(podaci)
  podaci = ss.transform(podaci)

  model = DBSCAN(eps = eps, min_samples = m).fit(podaci)

  klasteri = model.labels_

  with open("klasteriDBSCAN_" + str(eps) + "_" + str(m) + ".txt", "
   w") as datoteka:
    for x in klasteri:
      datoteka.write(str(x) + "\n")

  velicina = cs.Counter(klasteri).most_common()
  with open("klasteriDBSCAN_id_velicina_" + str(eps) + "_" + str(m)
    + ".txt", "w") as datoteka_id_velicina:
    for x in velicina:
```

```
36          datoteka_id_velicina.write("ID klastera: {} -> Velicina
        klastera: {}\n".format(x[0], x[1]))
37
38  if __name__ == '__main__':
39    main()
```

Listing 8: Kod za klasterovanje gena korišćenjem DBSCAN - geniDBSCAN.py

Potrebno je pozicionirati se u direktorijum *IP2_ Vladana_ Djordjevic_ 89_ 2015 /DBSCAN/Geni* i izvršiti program na sledeći način za odgovarajuće epsilon i m:

```
1   python3 geniDBSCAN.py ulazna_datoteka broj_kolona epsilon m
```

U sledećem delu rada biće prikazane kombinacije vrednosti epsilon i m koje su isprobane, odnosno, navešćemo komandu u terminalu, a iznad nje datoteke koje se dobijaju. Sve datoteke se nalaze u direktorijumu *IP2_ Vladana_ Djordjevic_ 89_ 2015 /DBSCAN/Geni*.

klasteriDBSCAN_ 5_ 10.txt i *klasteriDBSCAN_ id_ velicina_ 5_ 10.txt*

```
1   python3 geniDBSCAN.py pretprocesirano.txt 3369 5 10
```

klasteriDBSCAN_ 5_ 100.txt i *klasteriDBSCAN_ id_ velicina_ 5_ 100.txt*

```
1   python3 geniDBSCAN.py pretprocesirano.txt 3369 5 100
```

klasteriDBSCAN_ 50_ 100.txt i *klasteriDBSCAN_ id_ velicina_ 50_ 100.txt*

```
1   python3 geniDBSCAN.py pretprocesirano.txt 3369 50 100
```

klasteriDBSCAN_ 100_ 4.txt i *klasteriDBSCAN_ id_ velicina_ 100_ 4.txt*

```
1   python3 geniDBSCAN.py pretprocesirano.txt 3369 100 4
```

klasteriDBSCAN_ 100_ 100.txt i *klasteriDBSCAN_ id_ velicina_ 100_ 100.txt*

```
1   python3 geniDBSCAN.py pretprocesirano.txt 3369 100 100
```

klasteriDBSCAN_ 500_ 100.txt i *klasteriDBSCAN_ id_ velicina_ 500_ 100.txt*

```
1   python3 geniDBSCAN.py pretprocesirano.txt 3369 500 100
```

Svi rezultati su veoma loši. Pronalazi uglavnom dva klastera, ali neravnomerno raspoređenih po broju elemenata, negde pronalazi i tri. Ovakav rezultat ne bi trebalo da iznenadi, s obzirom na to da DBSCAN radi sa podacima malih dimenzija, a u našim podacima svaki gen je predstavljem sa po 3368 kolona, što je izuzetno veliko.

5.2 DBSCAN klasterovanje ćelija

Kod za klasterovanje ćelija primenom DBSCAN algoritma dat je listingom 9.

```
import sys
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import DBSCAN
import collections as cs

def main():

  if (len(sys.argv) != 5):
    print("Program se pokrece na sledeci nacin:")
    print("python3 naziv_programa.py ulazna_dat broj_kolona epsilon
    min_broj_tacaka")
    exit(1)

  ulazna_datoteka = sys.argv[1]
  broj_kolona = int(sys.argv[2])
  eps = int(sys.argv[3])
  m = int(sys.argv[4])

  kolone = [x for x in range(0, broj_kolona)]
  podaci = np.loadtxt(ulazna_datoteka, delimiter = " ", usecols =
   kolone, skiprows = 1)

  ss = StandardScaler().fit(podaci)
  podaci = ss.transform(podaci)

  model = DBSCAN(eps = eps, min_samples = m).fit(podaci)

  klasteri = model.labels_

  with open("klasteriDBSCAN_" + str(eps) + "_" + str(m) + ".txt", "
   w") as datoteka:
    for x in klasteri:
      datoteka.write(str(x) + "\n")

  velicina = cs.Counter(klasteri).most_common()
  with open("klasteriDBSCAN_id_velicina_" + str(eps) + "_" + str(m)
    + ".txt", "w") as datoteka_id_velicina:
    for x in velicina:
      datoteka_id_velicina.write("ID klastera: {} -> Velicina
   klastera: {}\n".format(x[0], x[1]))

if __name__ == '__main__':
  main()
```

Listing 9: Kod za klasterovanje ćelija korišćenjem DBSCAN - celijeDBSCAN.py

Da bi se kod izvršio potrebno je pozicionirati se u direktorijum *IP2_ Vladana _ Djordjevic_ 89_ 2015/DBSCAN/Celije* i izvršiti sledeću naredbu za odgovarajuće epsilon i m.

```
python3 celijeDBSCAN.py ulazna_dat broj_kolona epsilon m
```

U sledećem delu rada biće prikazane kombinacije vrednosti epsilon i m koje su isprobane, odnosno, navešćemo komandu u terminalu, a iznad nje datoteke koje se dobijaju. Sve datoteke se nalaze u direktorijumu *IP2_ Vladana_ Djordjevic_ 89_ 2015 /DBSCAN/Celije*.

klasteriDBSCAN_ 1_ 5.txt i *klasteriDBSCAN_ id_ velicina_ 1_ 5.txt*

```
python3 celijeDBSCAN.py transponovana_matrica.txt 14428 1 5
```

klasteriDBSCAN_ 1_ 10.txt i *klasteriDBSCAN_ id_ velicina_ 1_ 10.txt*

```
python3 celijeDBSCAN.py transponovana_matrica.txt 14428 1 10
```

klasteriDBSCAN_ 2_ 5.txt i *klasteriDBSCAN_ id_ velicina_ 2_ 5.txt*

```
python3 celijeDBSCAN.py transponovana_matrica.txt 14428 2 5
```

klasteriDBSCAN_ 2_ 10.txt i *klasteriDBSCAN_ id_ velicina_ 2_ 10.txt*

```
python3 celijeDBSCAN.py transponovana_matrica.txt 14428 2 10
```

klasteriDBSCAN_ 5_ 10.txt i *klasteriDBSCAN_ id_ velicina_ 5_ 10.txt*

```
python3 celijeDBSCAN.py transponovana_matrica.txt 14428 5 10
```

klasteriDBSCAN_ 5_ 100.txt i *klasteriDBSCAN_ id_ velicina_ 5_ 100.txt*

```
python3 celijeDBSCAN.py transponovana_matrica.txt 14428 5 100
```

klasteriDBSCAN_ 100_ 4.txt i *klasteriDBSCAN_ id_ velicina_ 100_ 4.txt*

```
python3 celijeDBSCAN.py transponovana_matrica.txt 14428 100 4
```

klasteriDBSCAN_ 100_ 10.txt i *klasteriDBSCAN_ id_ velicina_ 100_ 10.txt*

```
python3 celijeDBSCAN.py transponovana_matrica.txt 14428 100 10
```

I u ovom slučaju su rezultati veoma loši. Većina ima samo jedan klaster koji se sastoji od svih ćelija. Tek poneki imaju po dva klastera. Opet, to nije iznenađujuće, s obzirom na to DBSCAN radi najbolje sa podacima niske dimenzionalnosti, a svaka ćelija ima po 14428 kolona.

I u slučaju gena i u slučaju ćelija nismo prikazali podelu po klasterima, jer su modeli veoma loši i nema svrhe.

6 Klasterovanje korišćenjem algoritma KMEANS

Tehnika klasterovanja K-sredina je jednostavna. Prvo izaberemo K početnih centroida, gde je K parametar koji navodi korisnik, odnosno broj klastera koji želimo da dobijemo. Svaka tačka je potom dodeljena najbližem centroidu, a svaka kolekcija tačaka koja je dodeljena jednom centroidu čini klaster. Nakon toga se centroid svakog klastera ažurira na osnovu tačaka koje su mu dodeljene. Ponavljamo postupak i ažuriramo korake sve dok nijedna tačka ne promeni klaster, odnosno, sve dok centroidi ne ostanu isti [6].

Za primenu algoritma K-sredina na naš skup podataka koristićemo programski jezik Python, kao i metod *KMeans* iz biblioteke *sklearn.cluster* [7].

Metod *KMeans* ima više parametara koji su opcioni. Biće navedeni samo oni koji su korišćeni u ovom istraživanju. Korišćeni parametri su sledeći:

1. **n_clusters:** Broj klastera koji će biti formirani, a to je ujedno i broj centroida koji će se generisati. Podrazumevana vrednost je 8.

2. **n_init:** Broj puta koji će se algoritam K-sredina izvršiti sa različitim "seed" za centroide. Krajnji rezultat će biti najbolji izlaz iz n_init uzastopnih izvršavanja u pogledu na inertia. Podrazumevana vrednost je 10.

6.1 KMEANS klasterovanje gena

S obzirom na to da u klasterovanju ne znamo koji je broj klastera pre same primene algoritma, a u ovom slučaju kao argument metodu prosleđujemo isti, moramo prvo da vidimo za koji broj klastera algoritam KMeans daje najbolje rešenje. Za procenu istog koristićemo vrednost *inertia* koja predstavlja sumu kvadratnih rastojanja uzoraka do njihovih najbližih centroida. Želimo da ta vrednost bude što manja, jer želimo da svaki uzorak bude što bliže centroidu klastera koji mu je dodeljen.

Kod programa *brojKlastera.py* koji je prikazan listingom 10 nam to omogućava. Kod se pokreće na sledeći način:

```
1  python3 brojKlastera.py brojKolona brojKlastera
```

Rezultati programa će biti ispisani u datoteku pod nazivom *inertia_za_broj_klastera_od_1_do_N.txt*, pri čemu će umesto *N* u nazivu datoteke biti broj klastera koji unosimo pri izvršavanju programa.

```
1  import sys
2  import numpy as np
3  from sklearn.cluster import KMeans
4
5  def main():
6
7    if (len(sys.argv) != 4):
8      print("Program se pokrece na sledeci nacin: ")
9      print("python3 naziv_programa.py ulazna_dat broj_kolona N")
10     exit(1)
```

```
11
12    ulazna_dat = sys.argv[1]
13    broj_kolona = int(sys.argv[2])
14    N = int(sys.argv[3])
15
16    # ucitavamo podatke, ali preskacemo prvu kolonu koja sadrzi
       oznake gena i rastavljamo kolone po zarezu
17    kolone = [x for x in range(1, broj_kolona)]
18    podaci = np.loadtxt(ulazna_dat, delimiter = ",", usecols = kolone
       )
19
20    # isprobavamo kvalitet rezultata KMeans za broj klastera u rangu
       [1,N]
21    for n in range(1, N+1):
22      kmeans = KMeans(n_clusters = n, random_state = 3368).fit(podaci
       )
23      inertia = kmeans.inertia_
24      with open("inertia_za_broj_klastera_od_1_do_" + str(N) + ".txt"
       , "a+") as datoteka:
25        datoteka.write("N: {} -> Inertia: {}\n".format(n, inertia))
26
27  if __name__ == '__main__':
28    main()
```

Listing 10: Trazenje odgovarajuceg broja klastera - brojKlastera.py

Da bi se program uspešno izvršio potrebno je pozicionirati se u direktorijum *IP2_Vladana_Djordjevic_89_2015/KMeans/Geni*, a potom izvršiti program na sledeći način:

```
1   python3 brojKlastera.py 3369 30
```

Kao broj kolona smo naveli 3369, a kao broj klastera 30. Izlazna datoteka pod nazivom *inertia_za_broj_klastera_od_1_do_30.txt* sadrži vrednosti inertia za rezultat izvršavanja algoritma KMeans za svaku vrednost K od 1 do 30.

Sadržaj pomenute datoteke je :

inertia_za_broj_klastera_od_1_do_30.txt

```
N: 1 -> Inertia: 3959224398.8249946
N: 2 -> Inertia: 1550001933.0538597
N: 3 -> Inertia: 756638518.719278
N: 4 -> Inertia: 480014145.71927446
N: 5 -> Inertia: 316085709.5706096
N: 6 -> Inertia: 239315124.7128746
N: 7 -> Inertia: 196222086.5321392
N: 8 -> Inertia: 167863969.13320205
N: 9 -> Inertia: 151270219.64675683
N: 10 -> Inertia: 137475086.99062264
N: 11 -> Inertia: 131561270.45043735
N: 12 -> Inertia: 121718007.92601605
N: 13 -> Inertia: 117333760.61271882
N: 14 -> Inertia: 110229942.67216705
```

```
N: 15 -> Inertia: 102320725.6282352
N: 16 -> Inertia: 98854974.62301697
N: 17 -> Inertia: 93488765.66621329
N: 18 -> Inertia: 91256621.49478382
N: 19 -> Inertia: 85320139.12826756
N: 20 -> Inertia: 81627160.0945359
N: 21 -> Inertia: 78599694.19604957
N: 22 -> Inertia: 77687908.6833566
N: 23 -> Inertia: 74810753.20936841
N: 24 -> Inertia: 72429686.90833949
N: 25 -> Inertia: 71508125.90653655
N: 26 -> Inertia: 69466682.45517103
N: 27 -> Inertia: 68585521.37580878
N: 28 -> Inertia: 66728650.09080883
N: 29 -> Inertia: 65480981.38591023
N: 30 -> Inertia: 64747669.22217878
```

Najmanja vrednost *inertia* se dobija upravo za broj klastera = 30. Primenimo algoritam KMeans za broj klastera i vidimo kakav je kvalitet klastera koji dobijamo.

To nam omogućava kod dat listingom 11. Program se pokreće na sledeći način:

```
python3 geniKMeans.py 3369 30 20
```

Gde je 3369 broj kolona, 30 broj klastera, a 20 željena vrednost argumenta n_init.

```
import sys
import numpy as np
from sklearn.cluster import KMeans
import collections as cs

def main():
  if (len(sys.argv) != 4):
    print("Potrebno je navesti broj zeljenih klastera na sledeci nacin: ")
    print("python3 naziv_programa.py broj_kolona N n_init")
    exit(1)

  broj_kolona = int(sys.argv[1])
  n = int(sys.argv[2])
  ninit = int(sys.argv[3])

  # ucitavamo podatke, ali preskacemo prvu kolonu koja sadrzi oznake gena i rastavljamo kolone po zarezu
  kolone = [x for x in range(1, broj_kolona)]
  podaci = np.loadtxt("pretprocesirano.txt", delimiter = ",", usecols = kolone)

  # primenjujemo algoritam KMeans sa zadatim parametrima
  kmeans = KMeans(n_clusters = n, n_init = ninit, random_state = 2322).fit(podaci)

```

```
23    klasteri = kmeans.labels_
24
25    # u datoteku ispisujemo za svaki red kom klasteru pripada
26    with open("klasteriKMeans_n=" + str(n) + "_ninit=" + str(ninit) +
       ".txt", "w") as datoteka:
27      for x in klasteri:
28        datoteka.write(str(x) + "\n")
29
30    # u drugu datoteku ispisujemo id klastera i njegovu velicinu
31    velicina = cs.Counter(klasteri).most_common(n)
32    with open("klasteriKMeans_id_velicina_n=" + str(n) + "_ninit=" +
       str(ninit) + ".txt", "w") as datoteka_id_velicina:
33      for x in velicina:
34        datoteka_id_velicina.write("ID klastera: {} -> Velicina
       klastera: {}\n".format(x[0], x[1]))
35
36  if __name__ == '__main__':
37    main()
```

Listing 11: Primena algoritma KMeans za odgovarajući broj klastera - geniKMeans.py

Izlaz iz programa predstavljaju dve datoteke: *klasteriKMeans_n=N_ninit=ninit.txt* i *klasteriKMeans_id_velicina_n=N_ninit=ninit.txt*, gde umesto N stoji broj klastera, a umesto ninit stoji vrednost argumenta n_init. Prva datoteka se sastoji od oznake klastera u svakom redu. Odnosno, gen koji se nalazio na toj liniji u ulaznim podacima pripada klasteru koji se nalazi na toj liniji u izlaznim podacima. Dok druga datoteka sadrži redne brojeve klastera i njihove veličine. U našem slučaju, pošto smo pozvali program sa argumentima N = 30 i ninit = 20 dobijamo *klasteriKMeans_n=30_ninit=20.txt* i *klasteriKMeans_id_velicina_n=30_ninit=20.txt*. Obe datoteke se nalaze u direktorijumu *IP2_Vladana_Djordjevic_89_2015/KMeans/Geni*. Pogledajmo prvo drugu datoteku, da bismo zaključili o raspodeli broja elemenata u svakom klasteru.

klasteriKMeans_id_velicina_n=30_ninit=20.txt

```
ID klastera: 20 -> Velicina klastera: 11833
ID klastera: 0 -> Velicina klastera: 1574
ID klastera: 19 -> Velicina klastera: 530
ID klastera: 5 -> Velicina klastera: 238
ID klastera: 15 -> Velicina klastera: 87
ID klastera: 24 -> Velicina klastera: 46
ID klastera: 11 -> Velicina klastera: 32
ID klastera: 7 -> Velicina klastera: 22
ID klastera: 8 -> Velicina klastera: 11
ID klastera: 28 -> Velicina klastera: 11
ID klastera: 9 -> Velicina klastera: 8
ID klastera: 26 -> Velicina klastera: 7
ID klastera: 27 -> Velicina klastera: 6
ID klastera: 3 -> Velicina klastera: 3
ID klastera: 2 -> Velicina klastera: 2
ID klastera: 17 -> Velicina klastera: 2
ID klastera: 22 -> Velicina klastera: 2
```

```
ID klastera: 25 -> Velicina klastera: 2
ID klastera: 1 -> Velicina klastera: 1
ID klastera: 4 -> Velicina klastera: 1
ID klastera: 6 -> Velicina klastera: 1
ID klastera: 10 -> Velicina klastera: 1
ID klastera: 12 -> Velicina klastera: 1
ID klastera: 13 -> Velicina klastera: 1
ID klastera: 14 -> Velicina klastera: 1
ID klastera: 16 -> Velicina klastera: 1
ID klastera: 18 -> Velicina klastera: 1
ID klastera: 21 -> Velicina klastera: 1
ID klastera: 23 -> Velicina klastera: 1
ID klastera: 29 -> Velicina klastera: 1
```

Izlaz koji smo dobili nije toliko sjajan, od 30 klastera koje smo naveli da želimo, 12 (što je skoro polovina) su klasteri koji se sastoje samo od jednog elementa. Pored toga, raspored broja elemenata po klasterima nije ravnomeran. Samim tim nije iznenađujuće što je vrednost *inertia* za ovaj broj klastera najmanja, jer je rastojanje jednog elementa od centroida, tj. samog sebe 0. Pokušajmo ovaj postupak ponovo, ali ovog puta sa manjim brojem elemenata.

Posmatrajući ponovo datoteku *inertia_za_broj_klastera_od_1_do_30.txt*, izaberimo ovog puta broj klastera da bude 15. Za 15 vrednost inertia nije previše velika a ni previše mala.

Program izvršavamo na sledeći način:

```
python3 geniKMeans.py 3369 15 20
```

Kao i u prethodnom slučaju izlaz se sastoji od dve datoteke, ovog puta sa nazivima *klasteriKMeans_n=15_ninit=20.txt* i *klasteriKMeans_id_velicina_n=15_ninit=20.txt* koje se nalaze u direktorijumu *IP2_Vladana_Djordjevic_89_2015/KMeans/Geni*. Pogledajmo sadržaj druge.

klasteriKMeans_id_velicina_n=15_ninit=20.txt

```
ID klastera: 0 -> Velicina klastera: 13178
ID klastera: 9 -> Velicina klastera: 882
ID klastera: 5 -> Velicina klastera: 209
ID klastera: 7 -> Velicina klastera: 64
ID klastera: 10 -> Velicina klastera: 36
ID klastera: 8 -> Velicina klastera: 24
ID klastera: 2 -> Velicina klastera: 19
ID klastera: 3 -> Velicina klastera: 6
ID klastera: 13 -> Velicina klastera: 3
ID klastera: 6 -> Velicina klastera: 2
ID klastera: 1 -> Velicina klastera: 1
ID klastera: 4 -> Velicina klastera: 1
ID klastera: 11 -> Velicina klastera: 1
ID klastera: 12 -> Velicina klastera: 1
ID klastera: 14 -> Velicina klastera: 1
```

Situacija nije bolja. Imamo nekoliko klastera sa jednim elementom, a ostatak je i dalje neravnomerno raspoređen. Možemo zaključiti da ovaj postupak neće dati bolje rezultate ma koliko menjali broj klastera. Veći broj klastera će nam dati veći broj klastera sa po jednim elementom, ali će najveći klaster imati manje elemenata, dok ako navedemo manje klastera, imaćemo manji broj sa jednim elementom, ali će najveći klaster imati mnogo elemenata. Ipak, prikažimo kako je KMeans algoritam rasporedio gene u klastere.

U tu svrhu pozivamo program *ubacivanje_u_klastere.cpp* koji je ranije naveden u listingu 4. Potrebno je pozicionirati se u direktorijum *IP2_Vladana_Djordjevic_89_2015/Kmeans/Geni* i prevesti kod na sledeći način:

```
g++ ubacivanje_u_klastere.cpp -o uk
```

I ovako ga izvršiti:

```
uk klasteriKMeans_n=15_ninit=20.txt
 spojeni_klasteri_i_oznake_gena_15_20.txt 14428
```

Izlazna datoteka pod nazivom *spojeni_klasteri_i_oznake_gena_15_20.txt* može se naći u direktorijumu *IP2_Vladana_Djordjevic_89_2015/Kmeans/Geni*. Mi ćemo u tabeli 7 prikazati samo nekoliko elemenata iz svakog klastera.

6.2 KMEANS klasterovanje ćelija

Situacija sa ćelijama je slična genima. Potrebno je prvo pronaći odgovarajući broj klastera. Koristimo skoro isti program *brojKlastera.py* kao i u slučaju gena, međutim, sa nekim izmenama. Izmene su sledeće:

```
kolone = [x for x in range(0, broj_kolona)]
podaci = np.loadtxt(ulazna_dat, delimiter = " ", usecols = kolone,
    skiprows = 1)
...
kmeans = KMeans(n_clusters = n, random_state = 14428).fit(podaci)
```

Listing 12: Izmene u programu brojKlastera.py u slučaju ćelija

Da bi se kod izvršio, potrebno je pozicionirati se u direktorijum *IP2_Vladana_Djordjevic_89_2015/KMeans/Celije* i izvršiti sledeću naredbu:

```
 python3 brojKlastera.py transponovana_matrica.txt 14428 30
```

Izlaz je datoteka pod nazivom *inertia_za_broj_klastera_od_1_do_30.txt* koja se nalazi u direktorijumu *IP2_Vladana_Djordjevic_89_2015/KMeans/Celije* i njen sadržaj je sledeći:

Tabela 7: Klasterovani geni primenom algoritma KMeans N=15 n_init=20 - spojeni_klasteri_i_oznake_gena_15_20.txt

Klaster	**Geni**
0	hg38_A1BG, hg38_A1CF, hg38_A4GALT, hg38_AAAS ...
1	hg38_FTL
2	hg38_EEF1A1, hg38_GAPDH, hg38_KRT18, hg38_PFN1 ...
3	hg38_MT-CO1, hg38_MT-CO3, hg38_RPL10, hg38_RPL13 ...
4	hg38_FTH1
5	hg38_ACP1, hg38_AGR2, hg38_ALDOA, hg38_ANAPC11 ...
6	hg38_RPS19, hg38_RPS2
7	hg38_AKR1C2, hg38_ALDH1A1, hg38_ALDH3A1, hg38_ANXA1 ...
8	hg38_ACTG1, hg38_ANXA2, hg38_MT-ATP6, hg38_MT-ND4 ...
9	hg38_ACAT2, hg38_ACOT13, hg38_ACTN4, hg38_ADI1 ...
10	hg38_ACTB, hg38_AKR1B1, hg38_EIF1, hg38_FAU ...
11	hg38_KRT81
12	hg38_AKR1B10
13	hg38_RPL13A, hg38_RPLP1, hg38_RPS16
14	hg38_MT-CO2

inertia_za_broj_klastera_od_1_do_30.txt

```
N: 1 -> Inertia: 846571064.527017
N: 2 -> Inertia: 435387672.05541605
N: 3 -> Inertia: 312085992.11456347
N: 4 -> Inertia: 261559969.57191
N: 5 -> Inertia: 232032821.91403276
N: 6 -> Inertia: 215401128.83757833
N: 7 -> Inertia: 203638640.80225277
N: 8 -> Inertia: 194639154.06288886
N: 9 -> Inertia: 186608419.26705685
N: 10 -> Inertia: 179740694.4881051
N: 11 -> Inertia: 173323970.1735455
N: 12 -> Inertia: 166763662.6533052
N: 13 -> Inertia: 163635451.61513314
N: 14 -> Inertia: 160231573.11693323
N: 15 -> Inertia: 158000350.7823272
N: 16 -> Inertia: 155551768.6737525
N: 17 -> Inertia: 153168161.27076113
N: 18 -> Inertia: 151479167.53633052
N: 19 -> Inertia: 149839630.03688872
N: 20 -> Inertia: 148211632.1971123
N: 21 -> Inertia: 147093001.4995384
N: 22 -> Inertia: 145233620.77479666
N: 23 -> Inertia: 144210377.7372194
N: 24 -> Inertia: 142565493.2652048
N: 25 -> Inertia: 141261304.37927222
N: 26 -> Inertia: 140343534.66674486
N: 27 -> Inertia: 138919339.7577644
```

```
N: 28 -> Inertia: 137974806.3612218
N: 29 -> Inertia: 137357084.72530374
N: 30 -> Inertia: 135933401.4042807
```

Najmanja vrednost za inertia je kada je N=30, mada možemo primetiti da se u poslednjih nekoliko iteracija nije značajno smanjila. Nakon što smo izabrali broj klastera potrebno je pokrenuti algoritam KMeans sa zadatim parametrima. U tu svrhu koristimo program *celijeKMeans.py*. Pozicionirati se u direktorijum *IP2_ Vladana_ Djordjevic_ 89_ 2015/KMeans/Celije* a potom izvršiti program na sledeći način:

```
1    python3 celijeKMeans.py 14428 30 20
```

Izlaz iz programa je datoteka pod nazivom *klasteriKMeans_ id_ velicina_ n=30 _ ninit=20.txt* koja se nalazi u direktorijumu *IP2_ Vladana_ Djordjevic_ 89_ 2015/ KMeans/Celije*. U njoj se nalaze klasteri kao i njihova veličina.

klasteriKMeans_id_velicina_n=30_ninit=20.txt

```
ID klastera: 9 -> Velicina klastera: 314
ID klastera: 0 -> Velicina klastera: 248
ID klastera: 29 -> Velicina klastera: 242
ID klastera: 7 -> Velicina klastera: 235
ID klastera: 13 -> Velicina klastera: 225
ID klastera: 19 -> Velicina klastera: 223
ID klastera: 5 -> Velicina klastera: 185
ID klastera: 25 -> Velicina klastera: 181
ID klastera: 20 -> Velicina klastera: 156
ID klastera: 22 -> Velicina klastera: 124
ID klastera: 17 -> Velicina klastera: 123
ID klastera: 23 -> Velicina klastera: 119
ID klastera: 8 -> Velicina klastera: 110
ID klastera: 28 -> Velicina klastera: 109
ID klastera: 4 -> Velicina klastera: 87
ID klastera: 27 -> Velicina klastera: 82
ID klastera: 16 -> Velicina klastera: 72
ID klastera: 2 -> Velicina klastera: 70
ID klastera: 3 -> Velicina klastera: 69
ID klastera: 21 -> Velicina klastera: 64
ID klastera: 26 -> Velicina klastera: 59
ID klastera: 15 -> Velicina klastera: 52
ID klastera: 14 -> Velicina klastera: 50
ID klastera: 1 -> Velicina klastera: 46
ID klastera: 18 -> Velicina klastera: 35
ID klastera: 12 -> Velicina klastera: 34
ID klastera: 24 -> Velicina klastera: 21
ID klastera: 6 -> Velicina klastera: 18
ID klastera: 10 -> Velicina klastera: 14
ID klastera: 11 -> Velicina klastera: 1
```

Dobijen je odličan rezultat za date parametre. Ćelije su dobro klasterovane. Sad je preostalo rasporediti ćelije u klastere kojima su dodeljene i ispisati rezultat. Postupak je isti kao i za gene. Pozicionirati se u direktorijum *IP2_ Vladana_ Djordjevic _89_ 2015/KMeans/Celije* i prevesti kod na sledeći način:

```
g++ ubacivanje_u_klastere.cpp -o uk
```

A potom ovako izvršiti:

```
./uk klasteriKMeans_n=30_ninit=20.txt
spojeni_klasteri_i_oznake_celija_30_20.txt 3368
```

Kao izlaz se dobija datoteka pod nazivom *spojeni_ klasteri_ i_ oznake_ celija_ 30 _ 20.txt* koja se nalazi u direktorijumu *IP2_ Vladana_ Djordjevic_ 89_ 2015/KMeans/ Celije*. Oznake klastera i prvih 10 ćelija iz gorepomenute datoteke su prikazani u tabeli 8.

Tabela 8: Klasterovane ćelije primenom algoritma KMeans N=30 n_init=20 - spojeni_klasteri_i_oznake_celija_30_20.txt

Klaster	**Ćelije**
0	20, 24, 30, 63, 68, 93, 95, 104, 106, 112 ...
1	49, 77, 140, 155, 295, 348, 413, 568, 605, 690 ...
2	78, 132, 243, 416, 417, 420, 424, 463, 471, 475 ...
3	2, 13, 27, 74, 94, 127, 199, 279, 345, 354 ...
4	57, 73, 75, 87, 116, 141, 165, 185, 246, 273 ...
5	19, 44, 55, 59, 70, 79, 102, 113, 133, 159 ...
6	226, 393, 546, 561, 696, 753, 1138, 1227, 1520, 1644 ...
7	6, 36, 42, 85, 98, 117, 131, 167, 170, 194 ...
8	8, 126, 147, 154, 193, 223, 242, 276, 325, 357 ...
9	7, 15, 17, 21, 23, 26, 53, 56, 62, 69 ...
10	332, 401, 730, 954, 1485, 1667, 1732, 1739, 2038, 2211 ...
11	1650
12	11, 88, 190, 228, 287, 366, 441, 562, 588, 807 ...
13	61, 83, 91, 92, 100, 101, 119, 135, 143, 195 ...
14	39, 217, 221, 230, 256, 274, 482, 497, 522, 569 ...
15	84, 162, 181, 201, 212, 234, 240, 263, 277, 292 ...
16	144, 222, 293, 316, 339, 397, 422, 444, 518, 564 ...
17	29, 60, 72, 121, 183, 204, 214, 224, 231, 310 ...
18	108, 374, 431, 457, 613, 629, 718, 768, 880, 884 ...
19	3, 9, 12, 14, 99, 109, 111, 124, 128, 158 ...
20	22, 32, 34, 38, 40, 152, 191, 197, 200, 233 ...
21	48, 71, 122, 180, 248, 312, 363, 364, 380, 392 ...
22	25, 45, 51, 76, 125, 145, 157, 160, 202, 207 ...
23	1, 4, 16, 37, 58, 64, 103, 115, 229, 247 ...
24	255, 359, 439, 565, 756, 820, 946, 1021, 1408, 1539 ...
25	66, 67, 80, 86, 138, 153, 187, 189, 210, 244 ...
26	81, 264, 278, 300, 583, 593, 714, 729, 735, 860 ...
27	18, 33, 118, 146, 175, 179, 209, 245, 368, 386 ...
28	10, 31, 149, 164, 188, 205, 206, 218, 232, 271 ...
29	5, 28, 35, 41, 43, 46, 47, 50, 52, 54 ...

7 Zaključak

Klasterovanje gena je predstavljalo izazov, dok su ćelije bile mnogo uspešnije. Najgori rezultati su zabeleženi kod DBSCAN algoritma koji je i kod gena i kod ćelija imao veoma loše rezultate. Međutim, kao što smo već zaključili, to nije iznenađujuće, s obzirom na to da je dimenzija podataka bila izuzetno velika a DBSCAN najbolje funkcioniše sa podacima niske dimenzije. CLUTO je dao solidne rezultate za gene, najveći problem je bio pronaći klastere koji su unutar sebe slični. CLUTO je kod ćelija bio uspešniji, međutim, problem je ovde bio suprotan - klasteri su unu-

tar sebe veoma slični, ali takođe i među sobom. Broj elemenata u svakom klasteru je bio ravnomerno raspoređen što je dobra stvar. Hijerarhijsko je kod gena bilo neuspešno, stoga je pokušan drugi pristup, a to je preko dendograma i to je dalo bolji rezultat, dok je kod ćelija bilo mnogo uspešnije. Kod KMeans je situacija slična - loše kod gena, dobro kod ćelija. U svakom slučaju, klasterovanje je veoma moćan alat za uviđanje veza među objektima, ali ne postoji univerzalni recept za njegovo izvršavanje. Svaki skup podataka je drugačiji i da bi se došlo do nekih kvalitetnih zaključaka potrebno je eksperimentisati i isprobavati različite metode i načine.

Literatura

[1] M. Kantardzic. *Data Mining Concepts, Models, Methods, and Algorithms*. A John Wiley and Sons Publication, IEEE PRESS, 2011.

[2] dr Nenad Mitić. Profesorova uputstva, 2019. on-line at: http://poincare.matf.bg.ac.rs/~nenad/ip2/zadaci_klasterovanje.txt.

[3] G. Karypis. *CLUTO A Clustering Toolkit*. University of Minnesota, Department of Computer Science, 2003.

[4] INRIA et al. sklearn.cluster.AgglomerativeClustering, 2019. on-line at: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.Agglomerative\Clustering.html.

[5] Jörn Hees. Deo koda preuzet sa, 2019. on-line at: https://joernhees.de/blog/2015/08/26/scipy-hierarchical-clustering-and-dendrogram-tutorial/.

[6] V. Kumar P. Tan, M. Steinbach. *Introduction to Data Mining*. Pearson, 2006.

[7] INRIA et al. sklearn.cluster.KMeans, 2019. on-line at: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.