# KLASTEROVANJE GENA

Seminarski rad u okviru kursa
Istraživanje podataka 2
Matematički fakultet, avgust 2019.

Darko Veizović, 310/2015

## Sadržaj

# 1 Uvod

**Istraživanje podataka** predstavlja proces otkrivanja korisnih informacija u skupovima podataka. Tehnike koje se koriste pretražuju podatke u cilju pronalaska neobičnih i korisnih obrazaca. **Klasterovanje** je jedna od tehnika istraživanja podataka koja za cilj ima pronalaženje objekata sličnih osobina i njihovu podelu u grupe, odnosno klastere, čineći ih tako preglednijim i upotrebljivim.
Rad predstavlja prikaz različitih tehnika i algoritama klasterovanja koji su korišćeni za istraživanje podataka na primeru baze podataka o ćelijama. Za obradu podataka korišćeni su programski jezici **C++**, **Python** i **R**, kao i softver **Orange**. Upotrebljene su različite metode klasterovanja, a u nastavku su iste detaljno obrazložene.

# 2 Opis skupa podataka

Izvorna datoteka naziva **005_Lymphoblastoid_cell_line_GM12891_csv.csv** sadrži podatke o **limfoblastičnim ćelijama(LCL)**. Naime, izolovanje DNK materijala korak je koji ograničava brzinu istraživanja, postojanje LCL ćelija koje predstavljaju surogat izolovanim perifernim krvnim ćelijama značajno ubrzava proces bioloških ispitivanja. LCL ćelije mogu se izdvojiti *in vitro* infekcijom perifernih B-ćelija, što rezultira kontinuiranim izvorom, podnoseći zanemarljive fenotipske i genotipske promene. Budući da postoji spontani izvor razmnožavanja, LCL ispunjava zahtev za stalnim snabdevanjem polaznim materijalom za različite testove, štedeći potrebu za ponovnim uzorkovanjem. Postoji razlog za verovanje da su LCL u bliskoj srodnosti sa matičnim limfocitima na osnovu obimnih potpornih opažanja iz različitih studija koja pokazuju značajan nivo korelacije na molekularnom i funkcionalnom nivou. LCL ćelije, koje nose kompletan set genetskog materijala iz klice, bile su generalno instrumentalni izvor biomolekula i sistem za sprovođenje različitih imunoloških i epidemioloških ispitivanja. Štaviše, u novije vreme njihova upotreba za analizu celokupnog ljudskog genoma opširno je dokumentovana. Ovo dokazuje korisnost LCL-a u raznim genetičkim i funkcionalnim studijama. Postoji nekoliko kontradiktornih izveštaja koji dovode u pitanje zapošljavanje LCL-a kao roditelja surogata. Bez obzira na neka svojstvena ograničenja, LCL se sve više smatraju važnim resursom za genetička i funkcionalna istraživanja.

Ulazni materijal sadrži rezultate istraživanja skupova različitih humanih ćelija. Ulazna datoteka sadrži podatke vezane za ekspresiju **31221** gena. Nazivi svih gena imaju prefiks *hg38* koji označava da su podaci vezani za verziju 38 humanog genoma.

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| hg38_A1CF | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| hg38_A2M | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| hg38_A2M-AS1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| hg38_A2ML1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| hg38_A2ML1-AS1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| hg38_A2ML1-AS2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| hg38_A3GALT2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| hg38_A4GALT | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 14 | 0 | 0 | 0 | 0 | 0 |
| hg38_A4GNT | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| hg38_AAAS | 1 | 0 | 0 | 2 | 0 | 0 | 3 | 0 | 0 | 0 | 6 | 1 | 0 | 0 | 0 | 0 | 0 | 9 | 0 | 0 |
| hg38_AACS | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 1 | 2 | 0 | 2 | 4 | 0 | 0 | 0 | 0 |
| hg38_AADAC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| hg38_AADACL2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| hg38_AADACL2-AS1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| hg38_AADACL3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| hg38_AADACL4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| hg38_AADAT | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| hg38_AAED1 | 0 | 0 | 5 | 2 | 0 | 0 | 1 | 0 | 0 | 2 | 0 | 3 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 |
| hg38_AAGAB | 0 | 0 | 0 | 4 | 0 | 5 | 0 | 3 | 0 | 3 | 0 | 0 | 0 | 0 | 3 | 5 | 0 | 1 | 0 | 0 |
| hg38_AAK1 | 1 | 0 | 0 | 4 | 5 | 0 | 7 | 0 | 0 | 3 | 2 | 0 | 2 | 1 | 2 | 1 | 0 | 7 | 1 | 0 |
| hg38_AAMDC | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| hg38_AAMP | 1 | 0 | 1 | 11 | 0 | 3 | 1 | 0 | 0 | 0 | 0 | 2 | 20 | 0 | 4 | 11 | 0 | 3 | 1 | 0 |
| hg38_AANAT | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| hg38_AAR2 | 0 | 2 | 0 | 4 | 0 | 2 | 0 | 0 | 0 | 0 | 2 | 2 | 2 | 0 | 1 | 2 | 1 | 0 | 0 | 0 |
| hg38_AARD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Slika 1: primer podataka

Ulazna datoteka sadrži podatke u CSV formatu u obliku ukrštene tabele (podatak u preseku prvog reda i prve kolone je prazan). Datoteka ima **31221** red i **6484** kolone, pri čemu prvi red sadrži redni broj ćelije za koju je vršeno istraživanje, a prva kolona sadrži identifikaciju gena.

# 3 Priprema podataka

U okviru istraživanja podataka, ključna uloga pripada podacima, stoga određivanje istih sa pogodnim osobinama za dalju obradu, jeste prvi korak u procesu klasterovanja. Na prvi pogled, može se učiniti da podaci imaju pogodne osobine, međutim u stvarnosti oni sa sobom nose mnoštvo problema. Proces **pretprocesiranja** podataka predstavlja skup tehnika kojima se dobijaju pogodni podaci koji će kasnije predstavljati ulazni skup za algoritme klasterovanja. Izvorni skup podataka uglavnom mora biti podvrgnut pretprocesiranju kako bi postao pogodniji za dalju analizu. Programi za pretprocesiranje pisani su u programskom jeziku **C++**, a kao rezultat imali su prilagođavanje podataka konkretnim tehnikama i alatima za klasterovanje.

## 3.1 Uklanjanje nultih redova

```cpp
#include <iostream>
#include <algorithm>
#include <cstdlib>
#include <fstream>
#include <vector>

void errorMessage() {
    std::cout << "Usage: ./findZeros inputFile outputFile" << std::endl;
    exit(EXIT_FAILURE);
}

bool notZero(char c) {
    return c != ',' && c != '0' && !std::isspace(c);
}

int main(int argc, char **argv) {

    if(argc < 3)
        errorMessage();

    std::ifstream input;
    std::ofstream output;

    input.open(argv[1]);
    output.open(argv[2]);

    std::string start;
    getline(input, start);

    output << start << std::endl;

    while(input.good()) {
        std::string name;
        std::string line;

        getline(input, name, ',');
        getline(input, line);

        auto it = std::find_if(line.begin(), line.end(), notZero);
        if(it != line.end())
            output << name << ',' << line << std::endl;

    }

    return 0;
}
```

Slika 2: findZeros.cpp

U našem slučaju, ulazna datoteka sadržala je veliki broj redova, tj. gena koji su za svaku od 6383 ćelije, kao numeričku reprezentaciju iste imali vrednost - **0**. Kako ovako definisani geni ne bi imali uticaj na proces klasterovanja, a njihovim izbacivanjem iz dalje obrade isti bi bio značajno efikasniji, pomenuti geni uklonjeni su iz skupa podataka.

Prevođenje programa potrebno je izvršiti komandom:

```
g++ findZeros.cpp -o findZeros
```

nakon čega se isti pokreće izvršavanjem naredbe:

```
./findZeros inputFile outputFile
```

Pre uklanjanja nultih gena, broj redova bio je **31221**, dok se nakog istog taj broj smanjio na **10561**. Ovakva redukcija značajno utiče na efikasnost samih algoritama. Algoritmi klasterovanja su u najvećem broju slučajeva kvadratne složenosti, pa se smanjivanjem skupa podataka dobija na brzini i vremenu izvršavanja. Ovako pripremljen skup podataka biće ulaz za sve navedene algoritme klasterovanja, uz izvesna prilagođavanja.

## 3.2 Transponovanje podataka

Ovako pripremljeni podaci, pogodni su za klasterovanja na osnovu gena. Međutim, klasterovanje na osnovu ćelija zahteva nešto drugačiji ulazni skup. Naime, da bi se podaci klasterovali na osnovu ćelija, neophodno je najpre transponovati iste, tako da ćelije predstavljaju redove u tabeli, dok geni popunjavaju kolone. Program koji transponuje podatke, napisan je u programskom jeziku **C++**, pri čemu je korišćena biblioteka **boost**.

```cpp
#include <iostream>
#include <fstream>
#include <assert.h>
#include <cstdlib>
#include <algorithm>
#include <vector>
#include <boost/algorithm/string.hpp>

#define ROWS 10561
#define COLUMNS 6483

void errorMessage() {
    std::cout << "Usage: ./transposeData inputFile outputFile" << std::endl;
    exit(EXIT_FAILURE);
}

int main(int argc, char const *argv[])
{
    std::ifstream input;
    std::ofstream output;
    std::vector<std::vector<std::string>> dataMatrix(ROWS);
    if(argc < 3)
        errorMessage();

    input.open(argv[1]);
    output.open(argv[2]);
    std::string start;
    getline(input,start);

    for(int i = 0; i < ROWS; i++){
        std::vector<std::string> numbers;
        std::string name;
        std::string line;
        getline(input,name,',');
        getline(input,line);
        boost::split(numbers,line,boost::is_any_of(","));
        std::transform(numbers.begin(),numbers.end(),numbers.begin(),
                []{std::string& str){ boost::trim(str); return str;});
        std::copy(numbers.begin(),numbers.end(),std::back_inserter(dataMatrix[i]));
    }

    for(int i = 0; i < COLUMNS; i++){
        for(int j = 0; j < ROWS; j++)
            output << dataMatrix[j][i]<<" ";
        output<<std::endl;
    }


    return 0;
}
```

Slika 3: transposeData.cpp

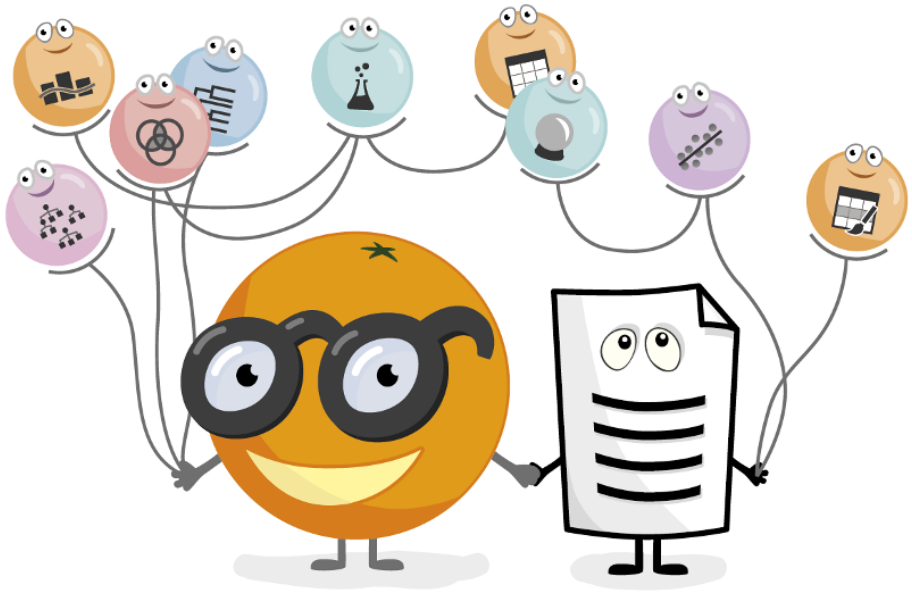
Prilikom prevođenja programa neophodno je proslediti putanju do boost biblioteke na sledeći način:

```
g++ transposeData.cpp -o transposeData - I /path/to/boost/
```

# 4 Orange

**Orange** je open-source softver za mašinsko učenje, vizualizaciju i istraživanje podataka. Razvijen je 10. oktobra 1996. godine od strane Univerziteta u Ljubljani. Softver je pisan na programskim jezicima C++, Python i C, a slobodan

je u upotrebi ili uz grafički-korisnički interfejs ili kao spoljašnja biblioteka jezika Python. Dostupan je za operativne sisteme *Linux*, *Windows* i *Mac*.



Slika 4: orange

Instalacija **Orange** softvera na operativnom sistemu *Linux* može se izvršiti pozivom sledećih komandi iz komandne linije.

```
sudo apt install virtualenv build-essential python3-dev
```

```
virtualenv --python=python3 --system-site-packages orange3venv
```

```
pip install PyQt5 PyQtWebEngine
```

```
pip install orange3
```

Pokretanje programa moguće je komandom:

```
orange-canvas
```

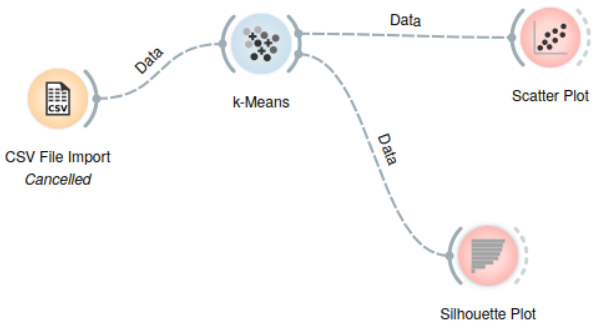
ili

```
python3 -m Orange.canvas
```

## 4.1 K-means

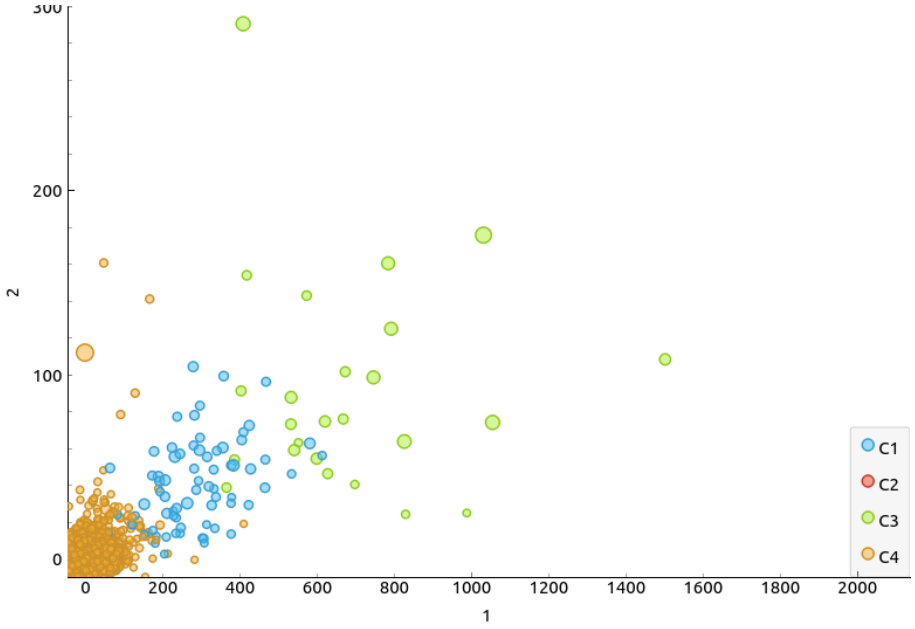
### 4.1.1 Klasterovanje gena

Za učitavanje podataka iskorišćen je čvor **CSV File Import**. Učitani su unapred pripremljeni podaci, odnosno skup podataka kome su uklonjeni nulti redovi, stoga je ukupan broj redova 10561 od kojih svaki sadrži osobine odgovarajućeg gena. Čvor **k-Means** koji se koristi za proces klasterovanja podešen je tako da softver sam odredi optimalan broj klastera. Pre određivanja broja klastera, pokušalo se sa klasterovanjem podataka za različit broj klastera, pri čemu se zaključilo da za broj klastera veći od 8 podaci nisu dobro raspoređeni, stoga je programu prosleđen fiksiran interval u kom traži optimalan broj klastera, odnosno interval od 2 do 8. Maksimalan broj iteracija postavljen je na 300.



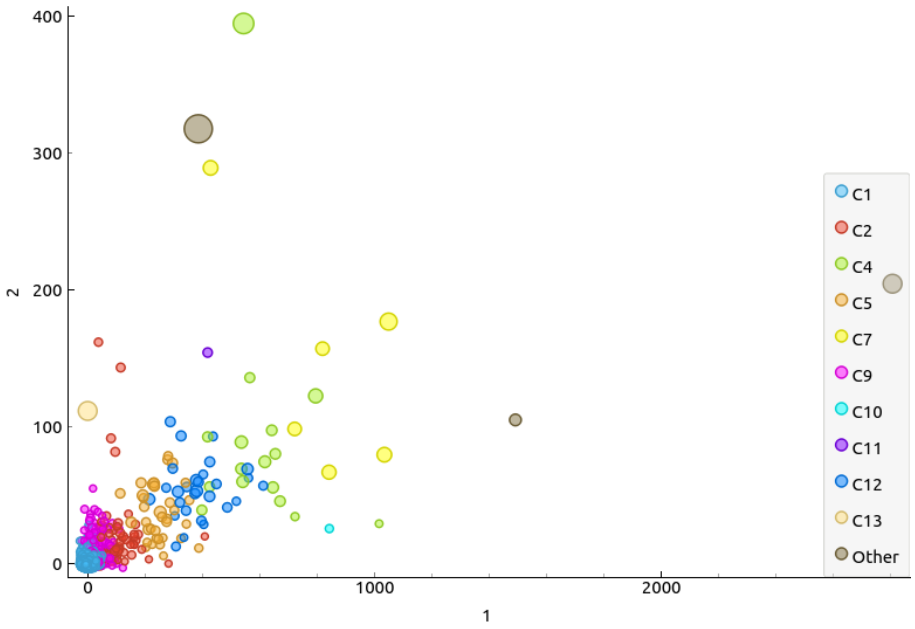

Slika 5: algoritam Kmeans

Pomoću čvora **Scatter plot** vizualizovani su dobijeni klasteri.




Slika 6: vizualizacija dobijenih klastera

Sa slike se može zaključiti, da se uz ovako podešene parametre za klasterovanje metodom K-means, kao optimalan broj klastera izdvaja 4. Podaci su neravnomerno raspoređeni u klastere, takođe gustina klastera se izuzetno razlikuje. U okviru jednog klastera grupisana je većina podataka, dok preostala tri klastera sadrže neznatan broj istih. Te stoga možemo zaključiti da ovakvom metodom nismo uspeli da valjano klasterujemo ulazni skup podataka. Silueta rastojanja između klastera data je u direktorijumu **clusters** u vidu slike odgovarajućeg naziva.




Slika 7: vizualizacija dobijenih klastera

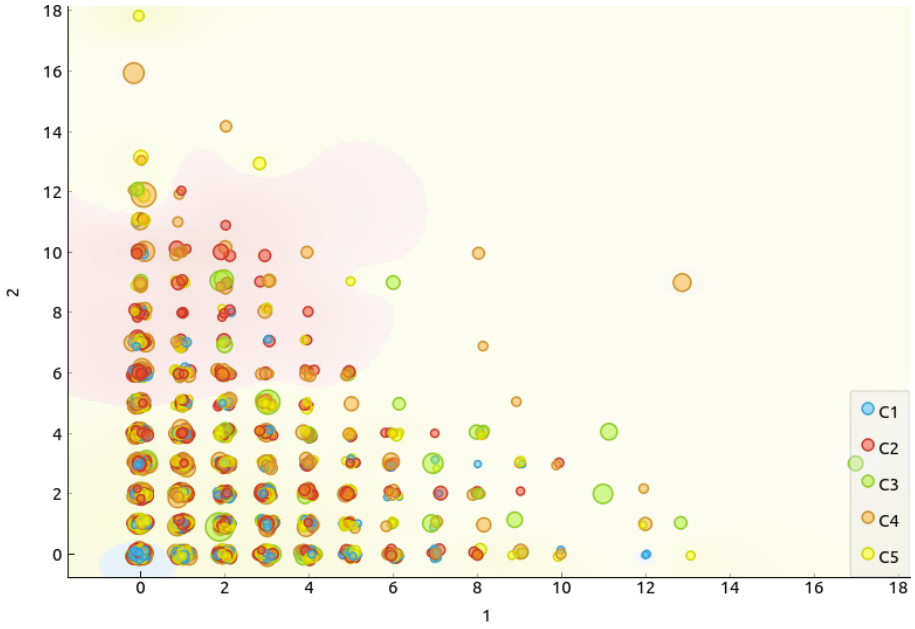
Promenom ulaznih parametara algoritma, odnosno fiksiranjem broja klastera na 13 nisu dobijeni značajno bolji rezultati. Klasteri su nešto ujednačenijih veličina, međutim podaci ponovo nisu kvalitetno raspoređeni. Uočeno je veće preklapanje među grupisanim podacima, takođe pronađen je i veći broj elemenata van granica.

```
CLUSTER0 - 71
CLUSTER1 - 1
CLUSTER2 - 27
CLUSTER3 - 10462
```

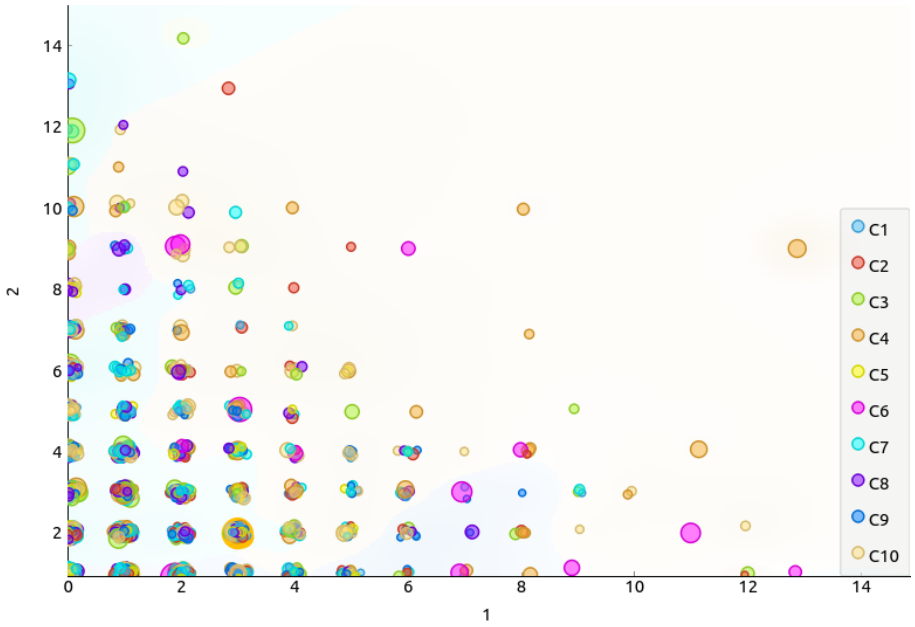
Raspored veličina klastera dobijen za 4 klastera. Izdvaja se klaster najveće gustine u kom se nalazi veliki deo podataka.

### 4.1.2 Klasterovanje ćelija

Prilikom klasterovanja ćelija iskorišćen je sličan postupak kao prilikom klasterovanja gena, pri čemu se za ulaznu datoteku uzima skup transponovanih podataka.




Slika 8: 5 klastera




Slika 9: 10 klastera

U ovom slučaju, pokušalo se sa različitim brojem klastera što je rezultiralo slabijim rezultatima. Podaci su grupisani u klastere tako da zbog velikog broja preklapanja sa slike nije moguće utvrditi tačan raspored klastera.

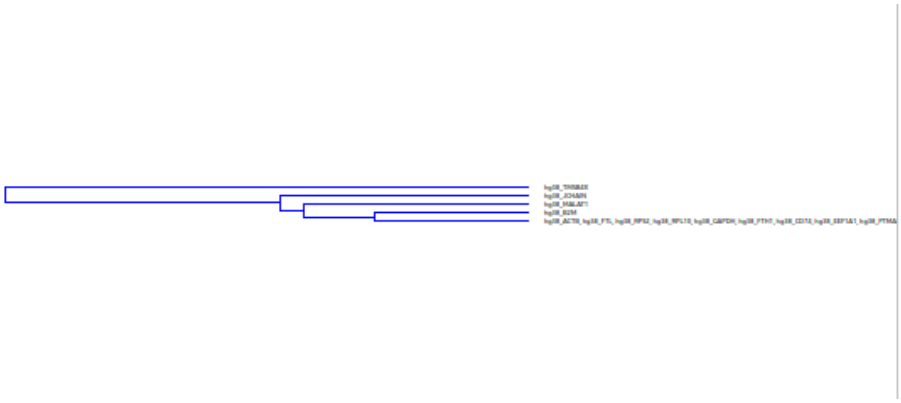
```
C1 - 2568
C2 - 1358
C3 - 94
C4 - 429
C5 - 2078
```

Rezultati koji su dobijeni prilikom grupisanja u pet klastera jasno izdvajaju 3 klastera koja gustinom dominiraju nad ostatkom. Sa dobijene siluete, zaključuje se da postoji veliki broj podataka koji su smešteni u jedan klaster, pri čemu imaju veliku sličnost sa podacima iz drugog klastera, što zapravo potvrđuje i vizualizaciju dobijenu čvorom Scatter plot.

## 4.2 Hijerarhijsko klasterovanje

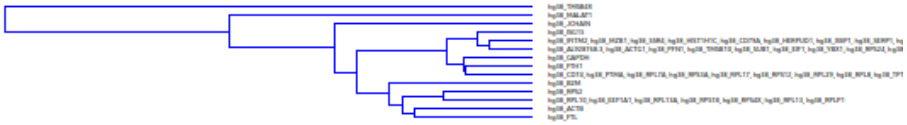
Podaci su ponovo učitani pomoću čvora **CSV File Import**. Pre primene metode hijerarhijskog klasterovanja nad ulaznim podacima, čvorom **Distances** izračunata su rastojanja između datih podataka. Prilikom klasterovanja ovim algoritmom, korišćeni su **Euklidsko**, **Menhetn** i **Kosinusno** rastojanje.

Dendrogrami koji su dobijeni različitim postavljanjem veza(*single, average, complete*) i različitom dubinom sečenja stabla dati su u nastavku.



Slika 10: dendrogram1

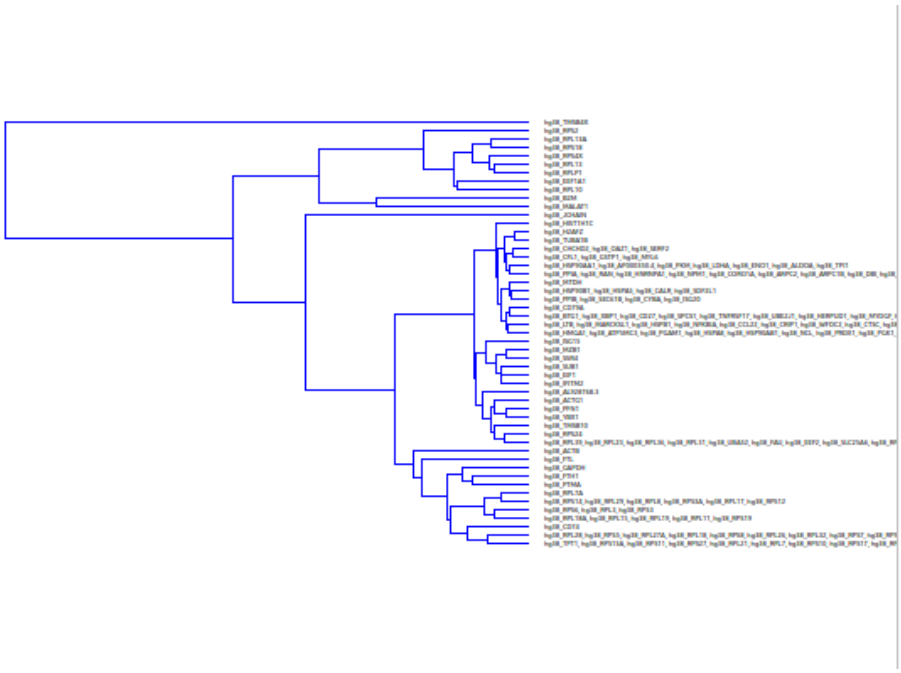
Rezultat dobijen pojedinačnom vezom, pri čemu nema sečenja stabla. Jasno se vidi da podaci nisu dobro raspoređeni u klastere. Postoji jedan klaster velike gustine, dok se u ostalim nalazi mali broj podataka.



Slika 11: dendrogram2

Prosečna veza, uz dubinu sečenja stabla - 7, dala je nešto bolje rezultate. Ovoga puta, izdvaja se grupa od nekoliko klastera koji sadrže veliku većinu podataka, sa druge strane postoje klasteri i podklasteri u kojima se nalazi neznatan broj gena.

Slika 12: dendrogram3



Slika 13: dendrogram4

Povećanjem dubine sečenja stabla, te različitim načinima grupisanja podklastera, moguće je povećati kvalitet klasterovanih podataka. Prilikom korišćenja različitih veza, najlošije se pokazala pojedinačna veza, dok se u slučaju prosečnog i ward povezivanja za određeni stepen ujednačuje raspored podataka u klasterima. Ipak, iz navedenih rezultata ne može se zaključiti da je ovakav metod pogodan za klasterovanje datog skupa podataka.

## 4.3 Louvain

**Louvain** metod predstavlja metod za pronalaženje veza između podataka u velikim skupovima ili mrežama. Metod je objavljen od strane belgijskog profesora *Vinsenta Blondela* na Univerzitetu Louvain. Ideja same metode je optimizacija modularnosti veza u mreži. Modularnost predstavlja vrednost između -1 i 1 na osnovu koje se određuje povezanost između unutrašnjih i spoljašnjih komponenti mreže. Najpogodnija struktura za predstavljanje podataka prilikom korišćenja ovakvog algoritma jeste graf, pri čemu se svaki čvor identifikuje sa odgovarajućim podatkom. Složenost u najgorem slučaju je O($nlogn$) .

$$Q = \frac{1}{2m} \sum \left[ A_{ij} - \frac{k_j k_i}{2m} \right] \delta(c_i c_j)$$

gde je:

- $A_{ij}$ - grana između čvorova $i$ i $j$
- $k_i, kj$ - suma težina svih grana koje prolaze kroz čvor $i$, odnosno $j$

- $2m$ - suma težina svih grana u grafu
- $c_i, c_j$ - stepen povezanosti
- $\delta$ - delta funkcija

**Louvain Clusttering** čvor predstavlja osnovu za izvršavanje algoritma u softveru Orange. Čvor najpre konvertuje ulazni skup podataka u graf k-najbližih suseda nakon čega se u toku izvršavanja algoritma vrši optimizacija modularnosti, koja ima za cilj da ustanovi povezanost između čvorova i podatke grupiše u klastere.

Pre procesa klasterovanja poželjno je definisati parametre:

- PCA - stepen ukljanjanja šuma među podacima
- Distance Metric - definisanje načina na koji će se tražiti veze među podacima
- K - broj suseda
- Resolution - mera kojom se utiče na broj pronađenih klastera

U nastavku su dati prikazani rezultati primene algoritma prilikom klasterovanja gena i klasterovanja ćelija.

### 4.3.1 Klasterovanje gena

| | Unnamed: 0 | Cluster | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | hg38_A1BG | C7 | 1 | 0 | 10 | 6 | 1 | 0 | 2 | 0 | 0 | 1 |
| 2 | hg38_A4G... | C10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | hg38_AAAS | C2 | 1 | 0 | 0 | 2 | 0 | 0 | 3 | 0 | 0 | 0 |
| 4 | hg38_AAC5 | C17 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 2 |
| 5 | hg38_AAED1 | C3 | 0 | 0 | 5 | 2 | 0 | 0 | 1 | 0 | 0 | 2 |
| 6 | hg38_AAGAB | C2 | 0 | 0 | 0 | 4 | 0 | 5 | 0 | 3 | 0 | 3 |
| 7 | hg38_AAK1 | C5 | 1 | 0 | 0 | 4 | 5 | 0 | 7 | 0 | 0 | 3 |
| 8 | hg38_AAM... | C3 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 |
| 9 | hg38_AAMP | C15 | 1 | 0 | 1 | 11 | 0 | 3 | 1 | 0 | 0 | 0 |
| 10 | hg38_AAR2 | C2 | 0 | 2 | 0 | 4 | 0 | 2 | 0 | 0 | 0 | 0 |
| 11 | hg38_AAR5 | C6 | 1 | 0 | 5 | 4 | 0 | 0 | 0 | 0 | 0 | 2 |
| 12 | hg38_AARS2 | C4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | hg38_AAR... | C2 | 2 | 1 | 0 | 0 | 0 | 2 | 6 | 0 | 0 | 1 |
| 14 | hg38_AASDH | C3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 |
| 15 | hg38_AAS... | C5 | 3 | 0 | 0 | 2 | 0 | 6 | 0 | 0 | 0 | 5 |
| 16 | hg38_ABAT | C4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | hg38_ABCA5 | C2 | 2 | 0 | 0 | 6 | 0 | 3 | 3 | 0 | 0 | 0 |
| 18 | hg38_ABCA6 | C9 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | hg38_ABCA7 | C1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 20 | hg38_ABC... | C2 | 0 | 0 | 0 | 3 | 2 | 0 | 1 | 0 | 0 | 3 |
| 21 | hg38_ABCB4 | C9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 22 | hg38_ABCB7 | C3 | 0 | 0 | 0 | 0 | 4 | 3 | 5 | 0 | 0 | 3 |
| 23 | hg38_ABCB8 | C3 | 0 | 3 | 0 | 0 | 1 | 8 | 0 | 0 | 0 | 0 |
| 24 | hg38_ABCB9 | C7 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 25 | hg38_ABCC1 | C1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 26 | hg38_ABC... | C1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 27 | hg38_ABCC4 | C3 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 2 |
| 28 | hg38_ABCC5 | C4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 29 | hg38_ABCD1 | C4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 30 | hg38_ABCD2 | C1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 31 | hg38_ABCD3 | C3 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 32 | hg38_ABCD4 | C7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 33 | hg38_ABCE1 | C15 | 10 | 0 | 0 | 17 | 4 | 3 | 12 | 0 | 1 | 13 |
| 34 | hg38_ABCF1 | C13 | 2 | 0 | 3 | 13 | 2 | 1 | 4 | 2 | 0 | 5 |
| 35 | hg38_ABCF2 | C10 | 4 | 0 | 1 | 4 | 0 | 3 | 0 | 1 | 0 | 14 |
| 36 | hg38_ABCF3 | C7 | 2 | 1 | 0 | 4 | 0 | 0 | 0 | 0 | 1 | 2 |
| 37 | hg38_ABCG1 | C1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Slika 14: 19 klastera

Veličine klastera:

```
C0: 1098
C1: 1093
C2: 956
C3: 954
C4: 843
C5: 806
C6: 699
C7: 631
C8: 595
C9: 537
C10: 491
C11: 480
C12: 399
C13: 359
C14: 178
C15: 156
C16: 112
C17: 91
C18: 83
```

| | Unnamed: 0 | Cluster | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | hg38_A1BG | C19 | 1 | 0 | 10 | 6 | 1 | 0 | 2 | 0 | 0 | 1 |
| 2 | hg38_A4G... | C16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | hg38_AAAS | C9 | 1 | 0 | 0 | 2 | 0 | 0 | 3 | 0 | 0 | 0 |
| 4 | hg38_AACS | C6 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 2 |
| 5 | hg38_AAED1 | C1 | 0 | 0 | 5 | 2 | 0 | 0 | 1 | 0 | 0 | 2 |
| 6 | hg38_AAGAB | C10 | 0 | 0 | 0 | 4 | 0 | 5 | 0 | 3 | 0 | 3 |
| 7 | hg38_AAK1 | C4 | 1 | 0 | 0 | 4 | 5 | 0 | 7 | 0 | 0 | 3 |
| 8 | hg38_AAM... | C3 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 |
| 9 | hg38_AAMP | C8 | 1 | 0 | 1 | 11 | 0 | 3 | 1 | 0 | 0 | 0 |
| 10 | hg38_AAR2 | C10 | 0 | 2 | 0 | 4 | 0 | 2 | 0 | 0 | 0 | 0 |
| 11 | hg38_AARS | C14 | 1 | 0 | 5 | 4 | 0 | 0 | 0 | 0 | 0 | 2 |
| 12 | hg38_AARS2 | C12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | hg38_AAR... | C9 | 2 | 1 | 0 | 0 | 0 | 2 | 6 | 0 | 0 | 1 |
| 14 | hg38_AASDH | C3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 |
| 15 | hg38_AAS... | C9 | 3 | 0 | 0 | 2 | 0 | 6 | 0 | 0 | 0 | 5 |
| 16 | hg38_ABAT | C2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | hg38_ABCA5 | C16 | 2 | 0 | 0 | 6 | 0 | 3 | 3 | 0 | 0 | 0 |
| 18 | hg38_ABCA6 | C7 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | hg38_ABCA7 | C1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 20 | hg38_ABC... | C21 | 0 | 0 | 0 | 3 | 2 | 0 | 1 | 0 | 0 | 3 |
| 21 | hg38_ABCB4 | C7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 22 | hg38_ABCB7 | C3 | 0 | 0 | 0 | 0 | 4 | 3 | 5 | 0 | 0 | 3 |
| 23 | hg38_ABCB8 | C10 | 0 | 3 | 0 | 0 | 1 | 8 | 0 | 0 | 0 | 0 |
| 24 | hg38_ABCB9 | C19 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 25 | hg38_ABCC1 | C12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 26 | hg38_ABC... | C1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 27 | hg38_ABCC4 | C3 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 2 |
| 28 | hg38_ABCC5 | C2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 29 | hg38_ABCD1 | C2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 30 | hg38_ABCD2 | C1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 31 | hg38_ABCD3 | C6 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 32 | hg38_ABCD4 | C3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 33 | hg38_ABCE1 | C8 | 10 | 0 | 0 | 17 | 4 | 3 | 12 | 0 | 1 | 13 |
| 34 | hg38_ABCF1 | C13 | 2 | 0 | 3 | 13 | 2 | 1 | 4 | 2 | 0 | 5 |
| 35 | hg38_ABCF2 | C18 | 4 | 0 | 1 | 4 | 0 | 3 | 0 | 1 | 0 | 14 |

Slika 15: 10 klastera

```
C0: 9%
C1: 9%
C2: 10%
C3: 11%
C4: 6%
C5: 9%
C6: 9%
C7: 12%
C8: 16%
C9: 9%
```

Iz dobijenih rezultata može se zakjučiti da Louvian metod klasterovanja postiže zadovoljavajuće rezultate. U slučaju klasterovanja podataka u 19 klastera, podaci nisu sasvim ravnomerno raspoređeni, dok su u slučaju povećanja parametra **resolution** izdvaja 10 klastera približno slične veličine. Klasterovanje je vršeno koristeći Menhetn i Euklidsko rastojanje, pri čemu nije bilo značajnije razlike između ove dve metode. Takođe, valja napomenuti da je u softveru Orange ispis nakon klasterovanja moguć samo u okviru tabele, odnosno nije moguće grupisati nazive dobijenih gena, stoga su pomoću siluete date veličine i rastojanja između klastera, kao i skica same tabele.

### 4.3.2 Klasterovanje ćelija

| | Cluster | 1 | 0 | 0.1 | 1.1 | 0.2 | 0.3 | 0.4 | 1.2 | 0.5 | 1.3 | 0.6 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | C7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| 2 | C5 | 10 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 1 | 0 |
| 3 | C6 | 6 | 0 | 0 | 2 | 1 | 2 | 4 | 4 | 0 | 11 | 4 |
| 4 | C14 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 |
| 5 | C6 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 3 | 2 |
| 6 | C3 | 2 | 0 | 0 | 3 | 0 | 1 | 0 | 7 | 3 | 1 | 0 |
| 7 | C5 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 |
| 8 | C16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | C2 | 1 | 0 | 0 | 0 | 2 | 2 | 3 | 3 | 0 | 0 | 0 |
| 10 | C1 | 2 | 0 | 0 | 6 | 0 | 0 | 0 | 2 | 0 | 0 | 2 |
| 11 | C1 | 0 | 0 | 2 | 1 | 1 | 3 | 0 | 0 | 0 | 2 | 2 |
| 12 | C2 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 2 | 0 | 20 | 2 |
| 13 | C16 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 14 | C6 | 0 | 0 | 14 | 0 | 2 | 5 | 3 | 2 | 0 | 4 | 1 |
| 15 | C2 | 0 | 0 | 0 | 0 | 4 | 0 | 5 | 1 | 1 | 11 | 2 |
| 16 | C11 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 17 | C1 | 0 | 0 | 0 | 9 | 0 | 0 | 1 | 7 | 0 | 3 | 0 |
| 18 | C3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 19 | C1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 26 | 4 |
| 20 | C1 | 3 | 2 | 0 | 0 | 0 | 2 | 0 | 2 | 0 | 0 | 2 |
| 21 | C11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 22 | C4 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 3 | 2 |
| 23 | C3 | 4 | 0 | 0 | 0 | 0 | 4 | 6 | 1 | 1 | 6 | 0 |
| 24 | C1 | 3 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 1 | 7 |
| 25 | C11 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 |
| 26 | C6 | 0 | 3 | 5 | 0 | 2 | 0 | 0 | 4 | 4 | 2 | 0 |
| 27 | C8 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 3 |
| 28 | C15 | 1 | 0 | 4 | 0 | 0 | 5 | 7 | 4 | 2 | 9 | 0 |
| 29 | C13 | 1 | 4 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 2 | 1 |
| 30 | C6 | 0 | 0 | 13 | 0 | 0 | 0 | 0 | 7 | 0 | 11 | 0 |
| 31 | C7 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 32 | C7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 2 |
| 33 | C4 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 3 | 0 |

Slika 16: 10 klastera

```
C0: 7%
C1: 6%
C2: 10%
C3: 12%
C4: 10%
C5: 10%
C6: 11%
C7: 11%
C8: 10%
C9: 13%
```

# 5 CLARA

**CLARA**(Clustering Large Applications) je algoritam koji predstavlja nadogradnju alogritma **k-medioida**, u cilju brže obrade velikih skupova podataka. Za razliku od klasičnog algoritma k-medioda koji računa medioide za kompletan skup podataka, **CLARA** u obzir uzima mali uzorak skupa fiksirane veličine na koji primenjuje **PAM** algoritam izračunavajući tako optimalan skup medioida. Kako se u procesu računanja ne koristi kompletan skup, dobijeni rezultati se mogu značajno razlikovati od stvarnih, ipak **CLARA** ponavalja proces slučajnog pronalaženja uzorka i primene **PAM** algoritma nad istim čime se minimizuje greška. Implementacija algoritma data je u programskom jeziku R.

Neophodno je instalirati pakete *factoextra* i *cluster*. Navedeno se može uraditi naredbama:

```
install.packages("factoextra")
install.packages("cluster")
```

a potom iste i učitati:

```
library(factoextra)
library(cluster)
```

```
 dataset = read.csv(header = FALSE, sep = ",",
            file = "data.txt")

head(dataset, nrows = 3)
x = as.matrix(sapply(dataset, as.numeric))


fviz_nbclust(x = dataset, FUNcluster = clara,
            method = "silhouette",
            diss = NULL, k.max = 10, nboot = 100,
            verbose = interactive(),
            barfill = "steelblue", barcolor = "steelblue",
            linecolor = "steelblue", print.summary = TRUE)

clara.results = clara(x, 4, samples = 50, pamLike = TRUE)
print(clara.results)

fviz_cluster(clara.results, data = NULL, choose.vars = NULL,
                stand = TRUE,   axes = c(1, 2),
                geom = c("point", "text"),
                repel = FALSE,
                show.clust.cent = TRUE, ellipse = TRUE,
                ellipse.type = "convex",
                ellipse.level = 0.95, \
                ellipse.alpha = 0.2, shape = NULL,
                pointsize = 1.5, labelsize = 12,
                main = "Cluster plot", xlab = NULL,
                ylab = NULL, outlier.color = "black",
                outlier.shape = 19,
                ggtheme = theme_grey())
```

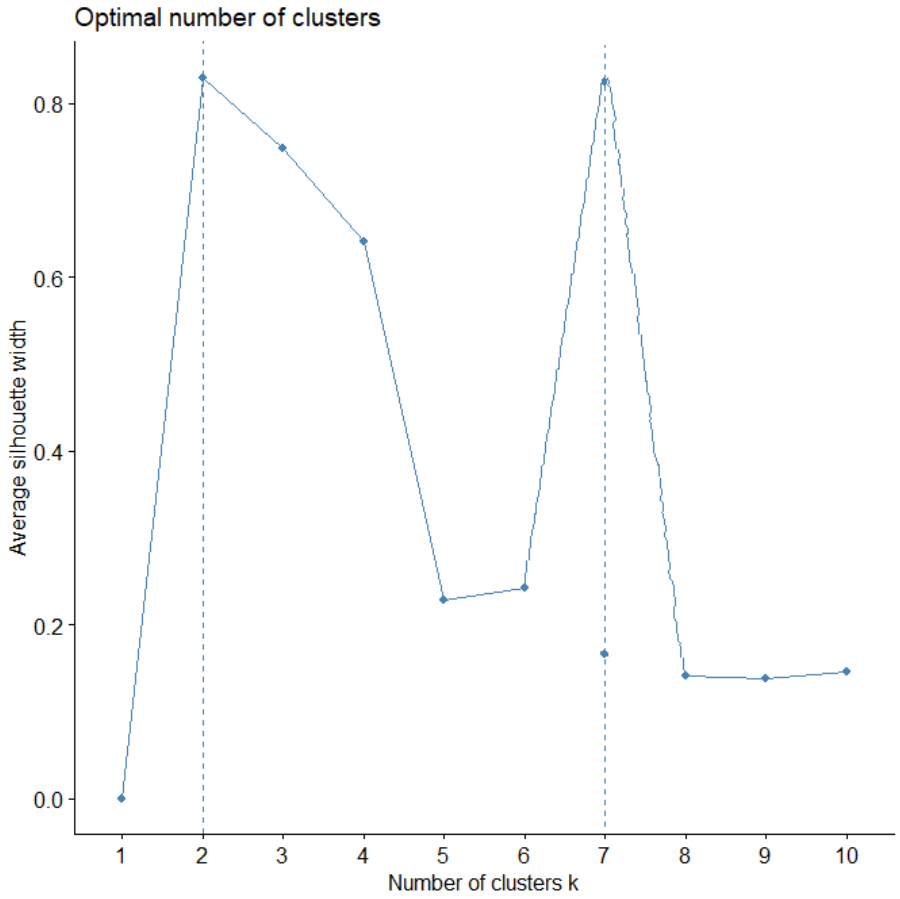
Nakon učitavanja skupa podataka, iste je neophodno pretvoriti u numeričku matricu koja predstavlja ulaz za **CLARA** algoritam. Pre samog procesa klasterovanja, poželjno je odrediti optimalan broj klastera.
Navedeno se može uraditi funkcijom *fviz_nbclust* iz paketa *factoextra*.

- x - ulazni skup podataka
- FUNCluster - algoritam klasterovanja
- method - mera koja se koristi za određivanje broj klastera
- k.max - maksimalan broj klastera
- nboot - broj iteracija

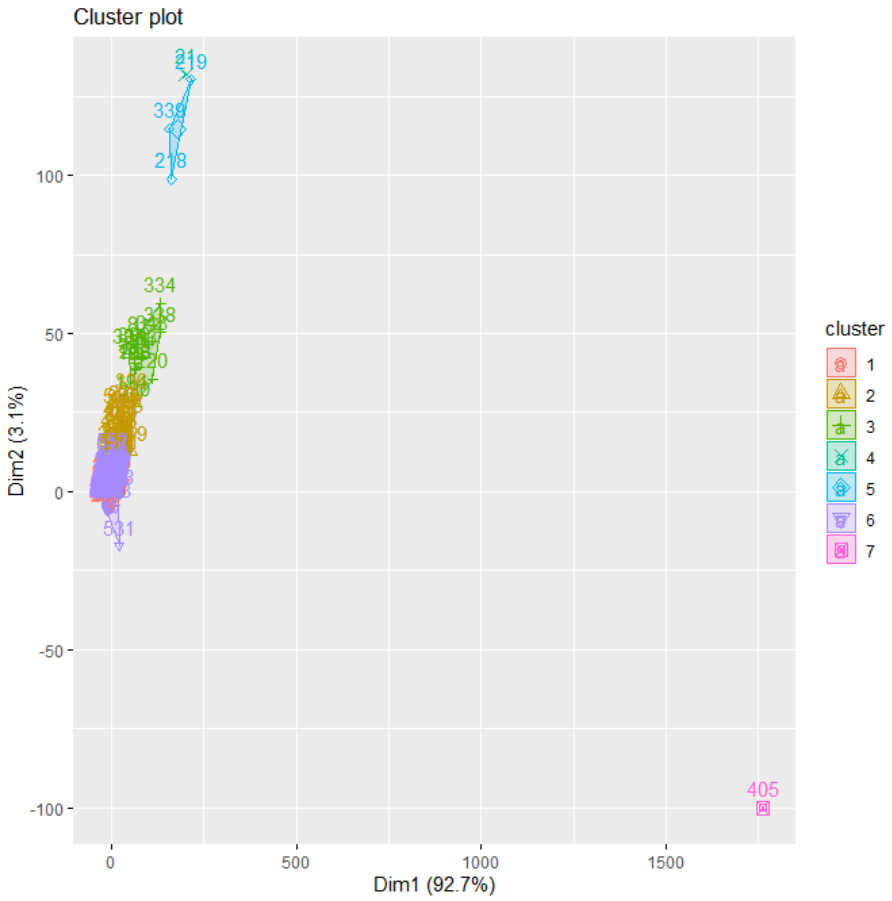
### 5.0.1 Klasterovanje gena

Prilikom klaterovanja gena korišćen je pripremljeni skup podataka. Primenom funkcije za određivanje broja klastera, grafik dostiže maksimalnu vrednost

za ulazne parametre 2 i 7, što implicira da podatke treba podeliti u 2, odnosno 7 klastera.




Slika 17: rezultat primene funkcije fviz_nbclust



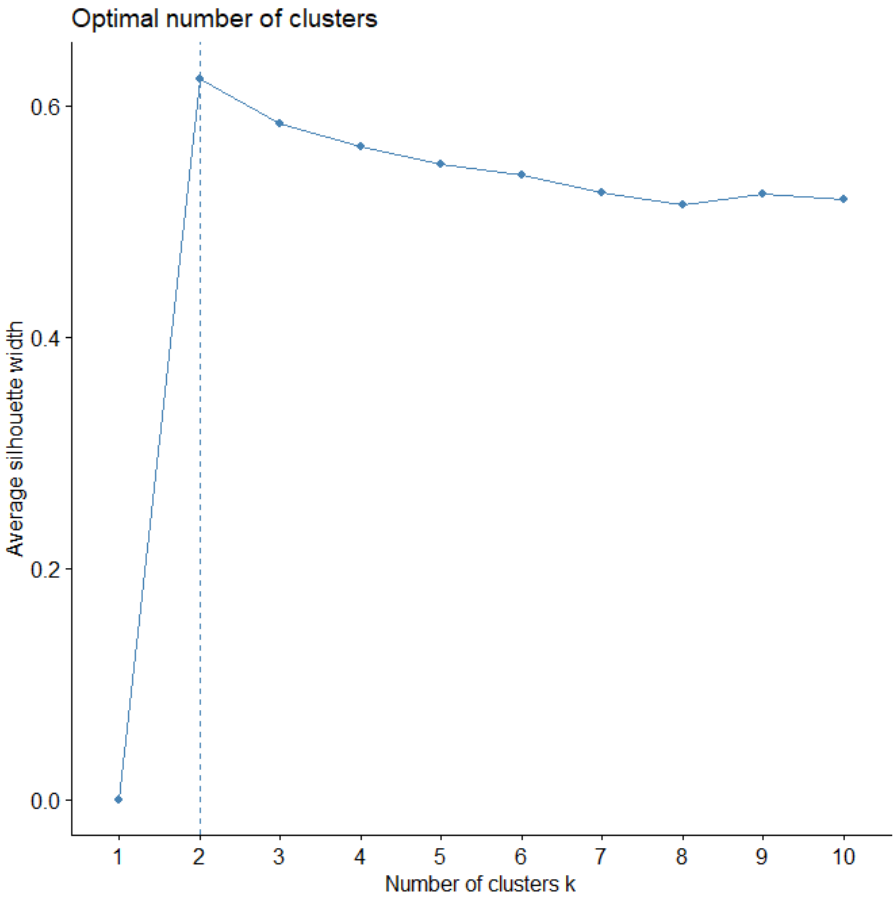

Slika 18: vizualizacija dobijenih klastera

Klasterovanje sa ovakvim parametrima rezultiralo je prilično ujednačenim klasterima, međutim prilikom grupisanja podataka u 7 klastera jasno su bila vidljiva određena preklapanja među klasterima, tako da na slici nije moguće jasno utvrditi granice između klastera, čak su i pojedini klasteri usled preklapanja nevidljivi.

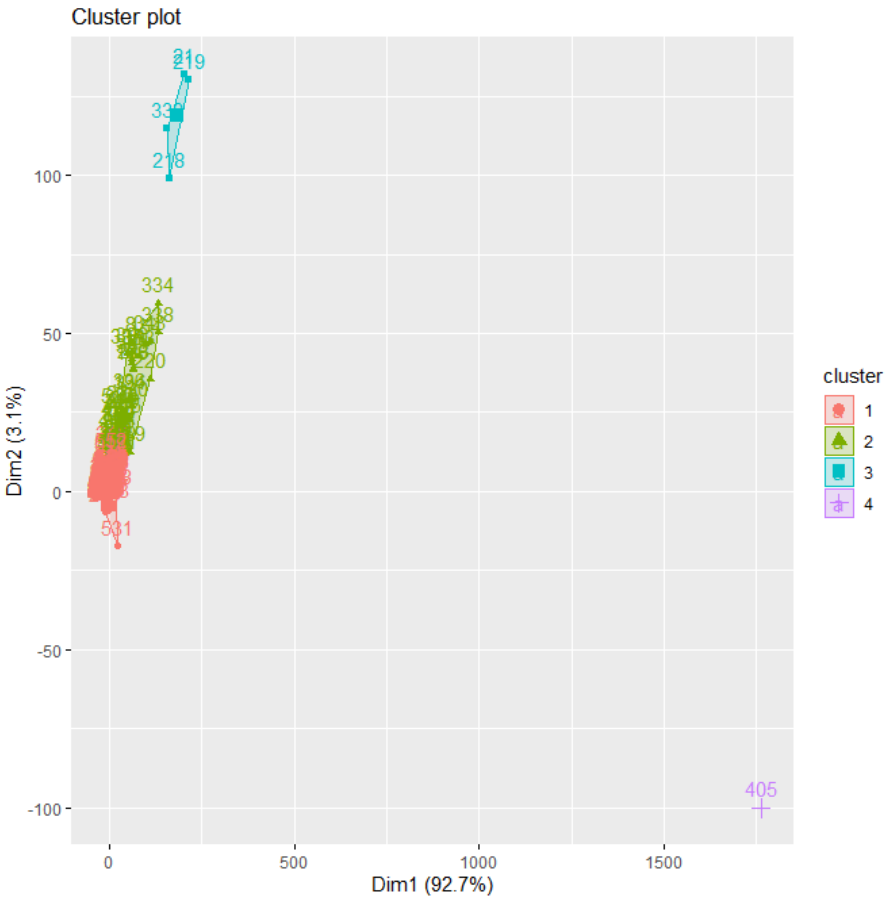
```
Objective function: 308.2523
Clustering vector: int [1:10561] 1 1 1 2 1 2 3 1 4 5 1 5 1...
Cluster sizes: 1321 1041 312 2235 2002 850 3700
```

### 5.0.2 Klasterovanje ćelija

Za klasterovanje ćelija, kao i u prethodnim slučajevima, biće iskorišćeni transponovani podaci. Izračunat je optimalan broj klastera i on iznosi dva, međutim, kako su i ostale vrednosti(u rasponu od 1 do 10) imale sličnu vrednost algoritam je primenjen za različite vrednosti broja klastera.




Slika 19: rezultat primene funkcije fviz_nbclust




Slika 20: vizualizacija dobijenih klastera

Klasterovanje ćelija u slučaju 4 klastera dalo je povoljne rezultate. Klasteri su dobro grupisani, i sličnih veličina, s tim što se jedan klaster tretirao kao elementi van granica.

```
Objective function: 14.531
Clustering vector: int [1:6484] 12 2 1 2 3 2 3 4 1 1 1 1 1 2...
Cluster sizes: 1609 x 1918 2957
```

# 6 Algoritam DBSCAN

**DBSCAN** algoritam predstavlja klasterovanje zasnovano na gustini, odnosno, prepoznaje regione visoke gustine koji su međusobno razdvojeni regionima niske gustine. Gustina se definiše kao broj tačaka u okviru određenog poluprečnika(*eps*). DBSCAN može da prepoznaje klastere različitih oblika i gustina. Odabirom različitih vrednosti za parametre *eps* i *minPts* mogu se pogodno prepoznati klasteri određene gustine, međutim u tom slučaju drugi(retki) klasteri

mogu izgledati kao šum.

```python
import numpy as np
import sys
import collections
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler
from sklearn.externals import joblib

def main():
    if len(sys.argv) != 2:
        print("Usage: python3 genesDbscan inputFile")
        exit(1)

    col = [x for x in range(1, 6484)]

    data = np.loadtxt(sys.argv[1], delimiter = ",", usecols = col, skiprows = 1)
    scaler = StandardScaler()
    scaler.fit(data)

    data = scaler.transform(data)

    z = DBSCAN(eps = 0.02, min_samples = 2)
    z.fit(data)

    clusters = z.labels_

    with open("dbscanCl.txt", "w") as file:
        for i in clusters:
            file.write(str(int(i)) + "\n")
    numberOfClusters = collections.Counter(clusters)
    for x in numberOfClusters:
        if x == -1:
            print("Outliers: ", numberOfClusters[x])
        else:
            print("CLUSTER", x, " -- SIZE: ", numberOfClusters[x])
    joblib.dump(z, "genesDbscanModel.sav")

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

Slika 21: implementacija DBSCAN algoritma

## 6.1 Klasterovanje gena

Klasterovanje algoritmom DBSCAN nije dalo željene rezultate. Uprkos definisanju različitih vrendosti za parametre *eps* i *minpts*, sve vrednosti su bile predstavljene kao elementi van granica iz čega zaključujemo da ovaj algoritam zasnovan na gustini nije pogodan za klasterovanje ovakvog tipa podataka. U nastavku su prikazani rezultati za različite vrednosti parametara *eps* i *minPts*.

```
Eps = 0.7
MinPts = 14

Outliers:   10561
```

```
Eps = 0.4
MinPts = 11

CLUSTER0 -- SIZE: 14
CLUSTER1 -- SIZE: 25
CLUSTER2 -- SIZE: 11
CLUSTER3 -- SIZE: 7

Outliers:   10504
```

```
Eps = 0.3
MinPts = 9

Outliers:   10561
```

```
Eps = 0.1
MinPts = 5

Outliers:   10561
```

```
Eps = 0.02
MinPts = 2

Outliers:   10561
```

# 7 Klasterovanje metodom K-means

Klasterovanje metodom **K-means** predstavlja metod za analizu grupisanja koja za cilj ima podelu objekata u **k** klastera, pri čemu svaki objekat pripada klasteru sa najsličnijim osobinama. Klasterovanje će biti vršeno u programskom jeziku Python, uz pomoć bibiloteka *sklearn, pandas, numpy, matplotlib*.

## 7.1 Klasterovanje gena

Pre početka procesa klasterovanja, kao i u slučaju **CLARA** algoritma, najpre je neophodno odrediti optimalan broj klastera. Broj klastera moguće je odrediti na osnovu vrednosti *inertia*. Kako bi se pristupilo datoj promenljivoj, izvršićemo najpre klasterovanje za vrednosti **k** u opsegu 1-20 nakon čega ćemo odrediti optimalan broj klastera.

```python
import sys
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans

def main():
    if len(sys.argv) != 2:
        print("Usage: python3 findKmeans.py inputFile")
        exit(1)

    col = [x for x in range(1, 6483)]
    data = np.loadtxt(sys.argv[1], delimiter = ",", usecols = col, skiprows = 1)

    for k in range (1, 21):
        kmeans_clustering = KMeans(n_clusters = k, random_state = 1).fit(data)
        iterate = kmeans_clustering.inertia_

        print("k: ", k, " ------ ", iterate)


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

Slika 22: findKmeans.py

Pozivom programa iznad dobijamo sledeći rezultat.

```
k:  1  ------  112468838126.22235
k:  2  ------  52403795184.77562
k:  3  ------  25666530479.058987
k:  4  ------  17245417713.175568
k:  5  ------  14139942030.161459
k:  6  ------  11997712819.261986
k:  7  ------  10914223612.17275
k:  8  ------  9964115336.273674
k:  9  ------  9277906309.844564
k:  10  ------  8581792905.926445
k:  11  ------  8010461744.837661
k:  12  ------  7176757398.016832
k:  13  ------  6836614168.836437
k:  14  ------  6308053233.808801
k:  15  ------  5989325457.669791
k:  16  ------  5376111680.354345
k:  17  ------  5169671282.9423895
k:  18  ------  4814736658.926268
k:  19  ------  4664123191.664272
k:  20  ------  4426681029.103273
```

Vrednost cost dobijamo pomoću promenljive inertia koja predstavlja sumu kvadratnih rastojanja uzoraka do njihovih najbližih centroida. Trebalo bi da se k izabere u trenutku kada vrednost cost prestane da se razlikuje značajno. Vrednost cost se ne menja značajno kada se za k uzimaju vrednosti od 12 do 17, pa će broj klastera biti 15, jer predstavlja sredinu intervala kada je vrednost cost

prestala naglo da opada. Sada kada znamo na koliko klastera ćemo klasterovati podatke pokrenućemo klasterovanje metodom K-means.

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


def main():
    if len(sys.argv) != 2:
        print("Usage: python genesKmeans.py inputFile")
        exit(1)

    col = [x for x in range(1, 6483)]
    data = np.loadtxt(sys.argv[1], delimiter = ",", usecols = col, skiprows = 1)
    kmeans_clustering = KMeans(n_clusters = 15, random_state = 1).fit(data)

    clusters = kmeans_clustering.labels_
    with open("clustersKmeans.txt", "w") as file:
        for i in clusters:
            file.write(str(int(i)) + "\n")

    numberOfClusters = collections.Counter(clusters)
    for x in numberOfClusters:
        print("C", x, "  size: ", numberOfClusters[x])

    joblib.dump(kmeans_clustering, "genesKmeans.sav")

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

Slika 23: genesKmeans.py

Rezultati koje smo dobili dati su u sledećoj tabeli.

```
C 0   size:  9872
C 5   size:  533
C 11   size:  1
C 1   size:  35
C 4   size:  65
C 9   size:  12
C 3   size:  1
C 6   size:  2
C 12   size:  1
C 8   size:  1
C 10   size:  3
C 13   size:  27
C 7   size:  5
C 14   size:  2
C 2   size:  1
```

Iz priloženih rezultata vidimo da klasteri nisu ravnomerno raspoređeni, te stoga zaključujemo da metoda **K-means** ne uspeva dovoljno dobro da klasteruje gene.

## 7.2 Klasterovanje ćelija

U slučaju klasterovanja ćelija koristićemo transponovane podatke.

```
import sys
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans

def main():
    if len(sys.argv) != 2:
        print("Usage: python3 findKmeans.py inputFile")
        exit(1)

    col = [x for x in range(1, 6483)]
    data = np.loadtxt(sys.argv[1], delimiter = ",", usecols = col, skiprows = 1)

    for k in range (1, 21):
        kmeans_clustering = KMeans(n_clusters = k, random_state = 1).fit(data)
        iterate = kmeans_clustering.inertia_

        print("k: ", k, " ------ ", iterate)


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

Slika 24: cellsKmeans.py

Rezultati koje smo dobili dati su u sledećoj tabeli.

```
Cid:     8  Size:   530
Cid:     0  Size:  1037
Cid:    14  Size:   493
Cid:     3  Size:   281
Cid:     7  Size:   426
Cid:     1  Size:   289
Cid:    12  Size:   880
Cid:    13  Size:   131
Cid:     4  Size:   722
Cid:     6  Size:   725
Cid:    11  Size:   251
Cid:    10  Size:   571
Cid:     5  Size:    41
Cid:     9  Size:    90
Cid:     2  Size:    16
```

Kod klasterovanja ćelija podaci su ravnomernije raspoređeni u klastere u odnosu na klasterovanje gena.

# 8 Hijerarhijsko klasterovanje

Hijerarhijsko klasterovanje jedna je od osnovnih tehnika klasterovanja, koja iako staromodna, i dalje široko rasprostranjena. U slučaju ovakvog pristupa,

nema potrebe za određivanjem broja klastera. Hijerahijsko klasterovanje se često grafički prikazuje u obliku drveta koje se naziva dendrogram. **Dendrogram** pokazuje klastere i podklastere. Željeni broj klastera ostvaruje se skraćivanjem dendrograma, odnosno sečenjem dobijenog drveta. U osnovi, postoje dva tipa ovakvog klasterovanja:

- Sakupljajuće - polazi se od činjenice da svaka tačka predstavlja jedan klaster, nakon čega se spajaju po dve najbliže tačke sve dok se ne dođe do 1 ili k klastera
- Razdvajajuće - polazi se od jednog velikog klastera, nakon čega se isti razbija sve dok se ne dođe do 1 ili k klastera

## 8.1 Klasterovanje gena

Pre klasterovanja podataka, odredićemo broj klastera. Optimalan broj klastera biće određen grafičkim putem, pomoću dendrograma.

```
import pandas as pd
import numpy as np
from scipy.cluster.hierarchy import dendrogram, linkage
from matplotlib import pyplot as plt

def main():
    datafile = 'dend.txt'
    df = pd.read_csv(datafile,sep=',')
    df = df.set_index('gene')
    cells = list(df.columns.values)
    dendrogrammer(df, cells)

def dendrogrammer(df, leaf_labels):
    D = df.values
    if len(leaf_labels) != len(D):
        D = np.transpose(D)
    Z = linkage(D, method='ward', metric='euclidean')
    plt.figure(figsize=(10, 6))
    ax = plt.subplot()
    plt.subplots_adjust(left=0.07, bottom=0.3, right=0.98, top=0.95,
    wspace=0, hspace=0)
    plt.xlabel('Cell Line')
    plt.ylabel('Distance')
    dendrogram(Z, leaf_rotation=90., leaf_font_size=10.,
    labels=leaf_labels)
    plt.savefig('dendrogram.png')

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

Slika 25: dendrogram

Biće prikazana različita rešenja, gde će biti promenjen broj klastera, mera sličnosti i metode povezivanja. Najčešće se koristi **Lance-Vilijamsova** grupa metoda:

- Single linkage - mera rastojanja između dva klastera predstavlja minimalno rastojanje između parova obekata koji pripadaju ovim klasterima.

- Complete linkage - rastojanje između dva klastera predstavlja maksimalno rastojanje između parova objekata koji pripadaju tim klasterima.
- Average linkage - rastojanje se određuje prema prosečnom rastojanju svih objekata koji pripadaju dvema grupama.
- Ward - poznata i kao metoda minimalne varijanse. Kao i ostale metode povezivanja, kreće se od m klastera (svaki klaster sadrži jedan objekat), ali se ne računa udaljenost između klastera, već se maksimizira homogenost unutar klastera. Ukupna suma kvadrata unutar klastera (SSE) se računa u cilju utvrđivanja koje se dve grupe spajaju u svakom koraku algoritma.

```python
import numpy as np
from sklearn.cluster import AgglomerativeClustering
import collections
import sys
from sklearn.externals import joblib

def main():
    if(len(sys.argv) != 2):
        print("Usage: python genesHierarchical.py in")
        exit(1)
    columns = [i for i in range(1,6483)]
    data = np.loadtxt(sys.argv[1],delimiter=",",skiprows=1,usecols=columns)
    cluster = AgglomerativeClustering(n_clusters=3, affinity='euclidean', linkage='average')
    cluster.fit_predict(data)
    clusters = cluster.labels_
    with open("clustersHierarchical.txt","w") as f:
        for i in clusters:
            f.write(str(int(i))+"\n")
    clustersNum = collections.Counter(clusters)
    for x in clustersNum:
        print("Cid: {:>5}".format(x)," Size: {:>5}".format(clustersNum[x]))
    joblib.dump(cluster, "genesHierarchical.sav")

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

Slika 26: genesHierarchical.py

U nastavku će biti izloženi rezultati klasterovanja za različite parametre.

```
Cid:     4  Size: 10486
Cid:    11  Size:     1
Cid:     7  Size:     1
Cid:     0  Size:    57
Cid:    13  Size:     1
Cid:    12  Size:     1
Cid:     9  Size:     1
Cid:     5  Size:     1
Cid:     8  Size:     1
Cid:     3  Size:     1
Cid:     1  Size:     3
Cid:     2  Size:     5
Cid:     6  Size:     1
Cid:    10  Size:     1
```

```
Cid:     2  Size: 10549
Cid:     0  Size:    11
Cid:     1  Size:     1
```

Klasterovanje prosečnom vezom, pri čemu je broj klastera postavljen na 14 nije dalo željene rezultate. Jedan klaster sadrži ogromnu većinu podataka, dok se u preostalih 13 klastera nalazi minimalan broj istih. Stoga možemo zaključiti da ovakvi parametri nisu pogodni za klasterovanje datih gena. Slična situacija je bila i u slučaju 3 klastera.

```
Cid:     1  Size: 5256
Cid:     2  Size: 970
Cid:     0  Size: 4355
```

Ward metodotom dobijeni su značajno bolji rezultati. Izdvojena su tri klastera, međutim podaci nisu i dalje u potpunosti ravnomerno raspoređeni.

```
Cid:     0  Size: 10551
Cid:     2  Size:     9
Cid:     1  Size:     1
```

Rezultati dobijeni za kompletnu vezu.

## 8.2 Klasterovanje ćelija

Kao i kod metode K-means koristićemo iste transponovane podatke.

```python
import numpy as np
from sklearn.cluster import AgglomerativeClustering
import collections
import sys
from sklearn.externals import joblib

def main():
    if(len(sys.argv) != 2):
        print("Usage: python hierarchicalCells.py in")
        exit(1)
    data = np.loadtxt(sys.argv[1])
    cluster = AgglomerativeClustering(n_clusters=11, affinity='euclidean', linkage='ward')
    cluster.fit_predict(data)
    clustersNum = collections.Counter(cluster.labels_)
    for x in clustersNum:
        print("Cid: {:>5}".format(x)," Size: {:>5}".format(clustersNum[x]))
    joblib.dump(cluster, 'hierarchicalCellsModel.sav')

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

Slika 27: cellsHierarchical.py

```
Cid:     8  Size:   642
Cid:     3  Size:  1796
Cid:     2  Size:   329
Cid:     4  Size:   409
Cid:     5  Size:   534
Cid:     7  Size:   868
Cid:    10  Size:   410
Cid:     6  Size:   744
Cid:     9  Size:   559
Cid:     0  Size:   159
Cid:     1  Size:    33
```

Ward metod dao je zadovoljavajuće rezultate u slučaju klasterovanja ćeija. Podaci su ravnomerno raspoređeni, klasteri su slične veličine izuzev dva klastera koja odstupaju od ostatka.

```
Cid:     1  Size:  6421
Cid:     7  Size:    45
Cid:     0  Size:     8
Cid:     3  Size:     4
Cid:     2  Size:     2
Cid:     5  Size:     1
Cid:     6  Size:     1
Cid:     4  Size:     1
```

Rezultati za prosečnu vezu i 8 klastera. Podaci su velikom većinom grupisani u jedan klaster, dok ostatak grupa sadrži izuzetno mali broj podataka.

# 9 Samoorganizujuće mape

Samoorganizujuća mapa predstavlja tip veštačke neuronske mreže čija obuka se vrši nenadgledanim učenjem kako bi se dobila niskodimenzionalna reprezentacija ulaznih uzoraka. Samoorganizujuće mape razlikuju se od drugih tipova neuronskih mreža po tome što čuvajuinformaciju o topološkim svojstvima ulaza pomoću funkcije susednih neurona.

Rad samoorganizujućih mapa sastoji se iz dve faze:

- faza učenja - izgrađuje se mapa pomoću ulaznih podataka
- faya preslikavanja - vrši se klasifikacija ulaznog vektora

Samoorganizujuća mapa sastoji se iz čvorova, odnosno nerona. Svaki neuron definisan je svojim vektorom težina, pri čemu isti mora biti iste dimenzije kao i ulazni vektor podataka. Proces smeštanja vektora iz prostora podataka u mapu sastoji se iz pronalaženja neurona čiji vektor težina ima vrednosti najbliže vektoru iz prostora podataka, nakon čega sledi dodeljivanja koordinata mape neurona odgovarajućem vektoru.

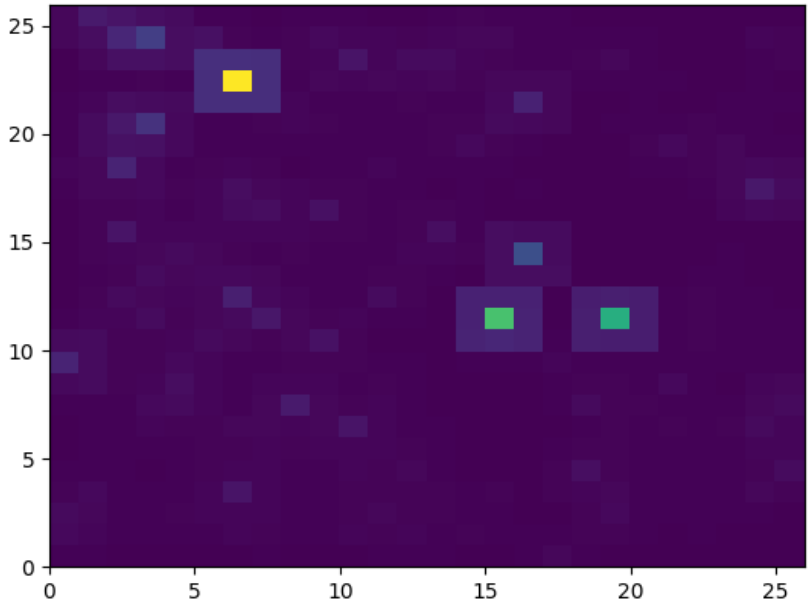
```
from minisom import MiniSom
import matplotlib.pyplot as plt
import numpy as np
import sys
from sklearn.externals import joblib

def main():
    if(len(sys.argv) != 2):
        print("Usage: python genesSom.py input")
        exit(1)
    columns = [i for i in range(1,6483)]
    X = np.loadtxt(sys.argv[1],delimiter=",",skiprows=1,usecols=columns)
    mdl = MiniSom(26, 26, 6482)
    #mdl.random_weights_init(X)
    mdl.pca_weights_init(X)
    mdl.train_batch(X, 100)
    joblib.dump(mdl, '"genesSomModel.sav')

    plt.pcolor(mdl.distance_map().T)
    plt.show()

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

Slika 28: genesSom.py

Dimenzija matrice (mape) kod samoorganizujućih mapa dobija se po formuli:

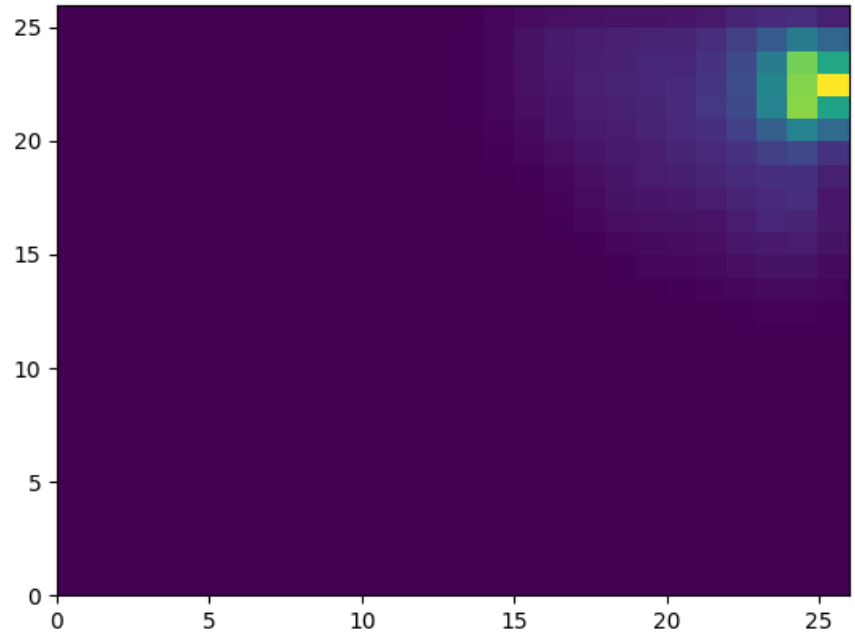
$$M^2 \approx 5\sqrt{N}, \tag{1}$$

de je M broj neurona, koji predstavlja ceo broj blizak jednačini sa desne strane, gde je N broj objekata. Samoorganizujuća mapa će biti korišćena radi vizuelizacije podataka nakon čega ćemo moći preko toplotnih mapa da zaključimo broj klastera. U implementaciji samoorganizujućih mapa u na dva načina se mogu podaci uzimati iz skupa, prvi je nasumičnim izborom gde se koristi metoda **traininrandom** a drugi je sekvencijalno, odnosno podaci se uzimaju redom gde se koristi metoda **trainbatch**.



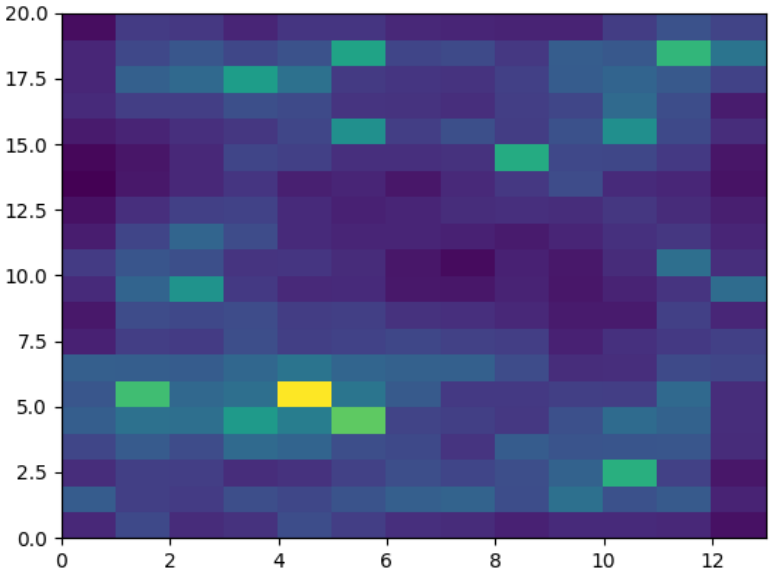

Slika 29: Toplotna mapa dobijena nasumičnim izborom gena

Izdvojena su četiri skupa podataka, pri čemu ostatak podataka nije prepoznat ovakvim nasumičnim izborom gena. Sa ovako dobijene mape moglo bi se pretpostaviti da podatke treba podeliti u četiri klastera, međutim upitne su veličine istih, jer veliki broj podataka nije u potpunosti uočen.



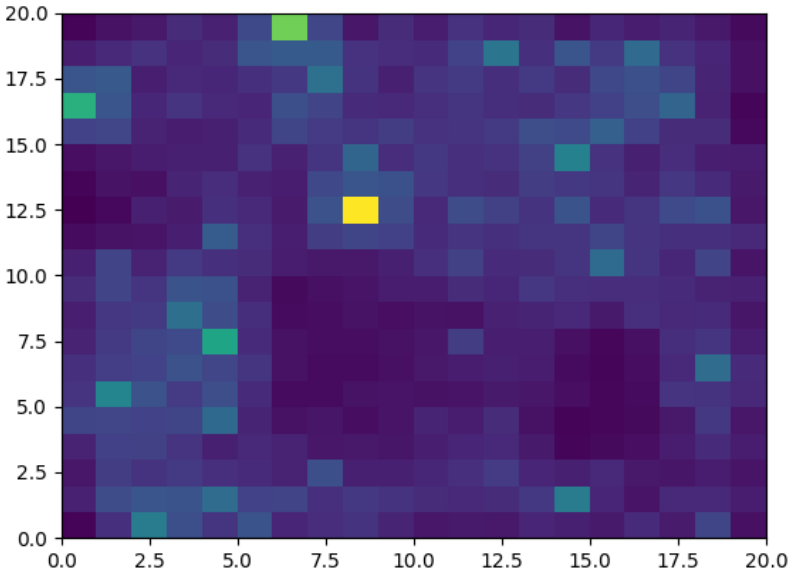

Slika 30: Toplotna mapa dobijena izborom gena po redosledu

Određeni broj podataka grupisan je zajedno u jednu grupu dok ostatak podataka nije prepoznat u ovom slučaju.




Slika 31: Toplotna mapa dobijena nasumičnim izborom ćelija

Kako se može videti sa dobijene mape, podatke bi bilo pravilno rasporediti u četiri klastera, pri čemu su isti sličnih veličina. Takođe postoji i određeni broj podataka koji se nalazi između dva ili više klastera te stoga mogu biti u sastavu više od jednog klastera.




Slika 32: Toplotna mapa dobijena nasumičnim izborom ćelija

Sa date slike može se videti da su podaci rasuti po mapi i da nisu kvalitetnu grupisani u klastere.

# 10 Zaključak

Modeli algoritama koji su implementirani u programskom jeziku Python sačuvani su u direktorijumu **models**, sa odgovarajućim nazivom. Isti se mogu učitati komandom *joblib.load(ime_fajla)*. Rešenja koja su dobijena, odnosno raspored ćelija i gena po klasterima nalaze se u direktorijumu **clusters**, sačuvana kao txt datoteke odgovarajućeg naziva. U istom direktorijumu nalaze se i dobijene toplotne mape.

Nakon primene različitih metoda i tehnika, možemo zaključiti da ulazni skup podataka nije u potpunosti pogodan za klasterovanje. U okviru klasterovanja gena, poželjne rezultate postigli smo klasterovanjem CLARA algoritmom, nešto lošiji bili su rezultati primene Louviane metode, dok se ostale tehnike nisu pokazale dobro. Slučaj klasterovanja ćelija bio je nešto efikasniji, pa se kvalitetan raspored podataka postigao u većem broju slučajeva. U sledećoj tabeli date su procene kvaliteta klasterovanja različitim metodama zasnovane na **Davis-Bouldinovom** indeksu.
**Davis-Bouldinov** indeks definiše se kao prosečna sličnost između svakog klastera sa klasterom koji mu je najsličniji.

$$DB = \frac{1}{k} \sum max_{i \neq j} R_{ij}$$

gde je:

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

- $s_i$ - prosečno rastojanje tačaka klastera od centra istog
- $d_{ij}$ - rastojanje između centara klastera $i$ i $j$

| K-means | DBSCAN | H - Average | H - Single | H - Ward |
|---|---|---|---|---|
| 0.78 | 1.89 | 0.3 | 0.47 | 0.4 |

KLASTEROVANJE GENA

Kako su kvalitet razdvojenosti klastera i Davis-Bouldinov indeks obrnuto proporcionalni, odnosno, veća vrednost indeksa označava i lošiji kvalitet klasterovanja, zaključujemo da su se u slučaju klasterovanja gena najbolje pokazao algoritam Hijerarhijskog klasterovanja, pri čemu su rezultati najbolji za prosečnu vezu. DBSCAN algoritam očekivano ima najlošije rezultate.

| K-means | H - Average | H - Single | H - Ward |
|---|---|---|---|
| 1.68 | 0.2 | 0.3 | 0.44 |

KLASTEROVANJE ĆELIJA

Kao i u slučaju klasterovanja gena, prilikom klasterovanja ćelija najbolji rezultati ponovo su dobijeni hijerarhijskim klasterovanjem. Sa druge strane, algoritam k-means, iako je na prvi pogled dao kvalitetne rezultate prilikom grupisanja ćelija ipak ima lošiji DB indeks.

Važno je napomenuti da se rezultati navedeni u prethodne dve tabele odnose na algoritme klasterovanja koji su implementirani u programskom jeziku Python, takođe kako su prilikom klasterovanja korišćeni različiti parametri u tabeli su navedeni samo najbolji rezultati za svaku od tehnika. Kako softver Orange ne raspolaže testovima koji bi odredili kvalitet klasterovanja, rezultati primenjenih tehnika u istom dati su u vidu slika koje se odnose na veličinu dobijenih klastera.

# References


[1] Introduction to data mining
Pang-ning Tan and Michael Steinbach and Vipin Kumar, 2005

[2] Clustering Algorithms in data mining
Jure Leskovec and Anand Rajaraman, Stanford Univeristy

[3] Datanovia: Data Analysis and Visualization

[4] Skit-learn library
`https://scikit-learn.org`

[5] Orange Data Mining Software
`https://https://orange.biolab.si/`

[6] Neo4J: Graph Algorithms
`https://neo4j.com/docs/graph-algorithms/current/algorithms/louvain/`