Upotreba R-a u SWISH-u

Leja, Andreas

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Andreas Leja

UPOTREBA R-A U SWISH-U

ZAVRŠNI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Andreas Leja

JMBAG: 0016141488

Studij: Informacijski sustavi

UPOTREBA R-A U SWISH-U

ZAVRŠNI RAD

Mentorica :

Vlatka Sekovanić, mag. educ. inf.

Varaždin, rujan 2022.

Andreas Leja

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Ovaj se rad bavi tematikom obrade podataka u statističkom programskom jeziku R-u, korištenjem logičkog programskog jezika Prologa te njihovog zajedničkog međudjelovanja. Obradit će se prednosti i mane korištenja SWISH-a i regularnog SWI-Prolog razvojnog okruženja, opisati što ove tehnologije mogu zasebno postići, istaknuti prednosti i mane svake od njih te kako se međusobno nadopunjuju. Prikazat će se popularni načini korištenja R-a i SWISH-a te će biti međusobno uspoređeni na temelju obavljanja istog seta zadataka te će na taj način biti prikazane prednosti i nedostatci. Uspostavit će se međudjelovanje R-a i SWISH-a na jednom praktičnom primjeru koristeći metodu EM klasteriranja iz područja podatkovne znanosti kao rješenje za zadani problem. Na samom kraju bit će izvučen zaključak iz provedenih istraživanja te opisan ukupan dojam izrade rada.

Ključne riječi: R; SWISH; Prolog; SWI-Prolog; programiranje; statistika; obrada podataka

Sadržaj

1. Uvod 1

2. Statistički programski jezici 3
2.1. R 3
2.2. RStudio 4

3. Prolog 15
3.1. Logičko programiranje 15
3.2. SWI-Prolog 16
3.3. Primjer korištenja Prologa 16
3.4. SWISH 22

4. Podatkovna znanost 24
4.1. Podatkovna znanost danas 24
4.2. Strojno učenje 25

5. Usporedba korištenja 27
5.1. Korištenje CSV datoteke 27
5.2. Korištenje paketa Rserve 29
5.3. Korištenje paketa rolog 35
5.4. Osvrt i usporedba 39

6. EM klasteriranje (praktičan primjer) 41
6.1. Teorijska podloga 41
6.2. Programsko rješenje 44

7. Zaključak 49

Popis literature 53

Popis slika 56

1. Uvod

U suvremeno doba nastoji se postići simbioza i međusobna suradnja dva ili više programska jezika. Postoji više razloga, a samim time i nekoliko ciljeva koji se žele postići kombiniranjem više programskih jezika u jedan projekt. Neke domene zadataka mogu se izvesti jedino korištenjem više tehnologija u projektu. Neki programski jezici u svojim bibliotekama sadrže sučelja za kod drugih programskih jezika. To je jedan od načina na koji se može povezati više programskih jezika u jedan projekt. Također se može poslužiti nekom tehnologijom koja bi bila posrednik. Primjerice, moguće je izvesti CSV (*Comma-Seperated Values*) datoteku iz jednog programskog jezika, tj. jednog programa, te tu istu datoteku koristiti kao ulaz u drugi program u drugom programskom jeziku.

Jednostavnije je, prema [1], kada se različiti jezici kompajliraju u istu vrstu koda. Na primjeru C i C++ koji se uobičajeno kompajliraju u strojni jezik. Također isti izvor navodi još jedan primjer, a to je kod jezika C# i VB.Net koji se kompajliraju u IL (*Intermediate Language*). Situacija postane malo kompliciranija kada programski jezici, odnosno njihovi kompajleri koriste sustave različite vrste.

Stručnjak iz [2] tvrdi da također postoji mogućnost da jedan kompajler može kompajlirati više jezika u isti jezik, ali ta se metoda ne koristi često u praksi. U slučaju da jezici koriste isto okruženje (primjer bi bile .NET tehnologije, JVM i drugi), jednostavno možemo pristupiti bibliotekama drugih jezika i služiti se njima. Naziv *Foreign Function Interface* vežemo upravo uz ovu domenu, a autor [3] govori da se radi o slučaju u kojem jezik može pozivati rutine, funkcije, metode napisane u drugom programskom jeziku.

Sada će biti rečeno o prednostima i nedostatcima korištenja više programskih jezika zajedno, tj. u kojim je situacijama pogodnije koristiti više programskih jezika zajedno. Popis razloga nam donosi korisnik sa [4]. Smanjenje troškova očiti je razlog korištenja više jezika u jednom projektu zato što je u zadnje vrijeme veća dostupnost *open-source* programa te je također lakše pronaći na Internetu komponente koda koje bi odgovarale željenom slučaju. Zahvaljujući ovom razmišljanju, stara metodologija u kojoj se sve počinje kodirati iznova, bez "recikliranja" prethodno korištenog koda, polako pada u zaborav. Ekonomski se korporacijama više isplati kada su projekti ranije završeni i kada čine ono što se od njih i traži pogotovo ako ti isti projekti koriste kod koji je već ranije kvalitetno napisan. Navodi se i dostupnost programera u nekom jeziku kao jedan od razloga. Racionalno je razmišljati o projektu tako da se pridružuje projektu jezik za koji je najpogodniji, a ne da se forsira projekt na jezik koji zna većina radnika ili koji znaju donositelji tih odluka. Gleda li se na projekt bazirano na njegovim funkcionalnim cjelinama, ponekad se može doći do zaključka da manje poznati jezik treba samo za implementaciju nekog algoritma ili neku sporednu funkcionalnost. Tako će se na projekt postaviti i manje ljudi koji programiraju u tom jeziku, a što projektu može pomoći na način da se ostvari bolje sučelje za izmjenjivanje podataka među jezicima. Uz sve te situacije, moglo bi se posegnuti za ovakvim pristupom i zbog lošije organizacije i vođenja projekta prilikom neplanskog korištenja više jezika na projektu. To je situacija u kojoj nije preporučljivo koristiti više jezika i "komplicirati" sam projekt, ali je zbog slabijeg planiranja situacija tražila takav pristup iako možda nije

pogodan za projekt.

2. Statistički programski jezici

Andra u [5] daje definiciju statističkog računalstva i programiranja. Riječ je o "tehnikama računanja koje pomažu pri analizi podataka i pridodaju smisao samim podatcima korištenjem statističkih koncepata i metoda unutar programskog koda" [5]. Izvođenje takvih metoda naziva se statističko programiranje i koriste ga podatkovni znanstvenici na dnevnoj razini kroz razne projekte vezane uz podatkovnu znanost. Konkretno, "takve se metode primjenjuju najčešće u farmaceutskim, telekomunikacijskim, financijskim i drugim industrijama" [5].

Neki jezici nude razne biblioteke i dodatke za statističku i grafičku obradu podataka te su iz tog razloga povoljni za provođenje statističkog računalstva. Također, ti jezici daju mogućnost klasičnog grafičkog prikaza obrade tih podataka. Podatkovni znanstvenici često koriste takve jezike kako bi proveli razne statističke analize, ponovnu konfiguraciju nestrukturiranih podataka te za predviđanja.

Andra također u [5] predstavlja popis najpopularnijih programskih jezika (2022. godina) za statističko računalstvo među kojima su:

- R
- Python
- SQL
- Java
- C/C++

2.1. R

Prema autorima [6], R je vrlo moćan alat za statistiku i statističko programiranje. Riječ je o statističkom programskom jeziku kojeg koriste desetci tisuća ljudi na dnevnoj razini. Prije svega, besplatan je i *open-source*, a također sadrži podršku za preko 2000 dodataka te samim time konkurira ostalim statističkim programskim jezicima. Na njemu će biti fokus u ovom radu.

Kako bi se osposobio i instalirao R za rad, potrebno je slijediti nekoliko jednostavnih koraka. Prvo je potrebno preuzeti sam R paket. Instalacija je vrlo jednostavna i ne zahtijeva posebnu pažnju. Budući da računalo sada prepoznaje da postoji R, moguće je pokretati programe pisane u R-u.

Patwal u svom [7] tvrdi da R može provoditi osnovne komande, aritmetičke operacije i logičke operacije. S obzirom na to da za R nije potreban *compiler*, nego se njegov kod može interpretirati kroz interpreter, moguće je jednostavno napisati željeni izračun te će R ponuditi rješenje tijekom samog izvođenja. Primjer nekoliko takvih operacija vidljiv je na slici 1.

Evidentno je kako prije svakog rezultata stoji jedinica u uglatim zagradama. To se dogodilo zato što se svaki podatak u R-u predstavlja kao vektor, pa tako i rezultat korisničkih

```
[Previously saved workspace restored]

> 2+2
[1] 4
> 2-2
[1] 0
> 3**2
[1] 9
> TRUE & FALSE
[1] FALSE
> TRUE & TRUE
[1] TRUE
> TRUE | FALSE
[1] TRUE
> 5==5
[1] TRUE
> 5==10
[1] FALSE
> sqrt(2)
[1] 1.414214
>
```

Slika 1: Osnovne operacije u R-u [Autorski rad]

računica. Na taj se način dobije vektor od jednog elementa te se samo taj jedan element ispisuje.

2.2. RStudio

Za lakše služenje R-om, moguće je preuzeti i RStudio koji se prema službenoj dokumentaciji ([8]) navodi kao razvojno okruženje koje omogućuje jednostavniju interakciju s R-om i samim programom koji se pokreću. Zahvaljujući njemu, u svakom trenutku izvođenja R programa moguće je pratiti rad, izlaze, varijable koje se koriste, što se nalazi u njima itd. Također, to okruženje samostalno može prepoznati i detektirati komponente funkcije ili seta podataka koji se koriste i koji će se najvjerojatnije dalje koristiti.

Za ovo okruženje Tomić u [9] tvrdi da omogućuje korištenje tzv. "okna". Riječ je o oknu izvora, konzole, okruženja i datoteka (eng. *source, console, environment, files*). Konzola je već prikazana na slici 1 gdje su se R naredbe pokretale preko terminala. Ova konzola nije ništa puno drugačija, ali omogućava pristup konzoli unutar samog razvojnog okruženja. Okno izvora omogućuje rad s aktivnim datotekama, odnosno otvara one datoteke u kojima se program trenutno izvodi. Okruženje predstavlja pregled svih objekata koji se nalaze unutar programa koji je trenutno aktivan u ovoj sesiji. Također, postoji i *history* koji sadrži popis od maksimalno 512 prethodnih naredbi koje je korisnik zadao RStudiju da izvrši. Na kraju okno datoteka predstavlja prilagodbu radnog direktorija svim datotekama koje se nalaze u tom direktoriju.

Tomić u [9] također navodi da RStudio razlikuje nekoliko vrsta datoteka koje pokreću R programe. Počevši od *R Script* koja je jednostavna tekstualna datoteka u koju upisujemo naredbe za R. Riječ je o klasičnoj datoteci s .r ekstenzijom. Zadane naredbe izvode se na pritisak kombinacije tipki Ctrl + Enter za naredbu u pojedinom redu. Druga vrsta datoteke je *R Markdown*. To je vrsta R datoteke u kojoj možemo ispreplitati rezultate izvođenja programa, tekst i

sam programski kod. Ovo je vrlo korisno zato što možemo izraditi HTML, PDF ili Word dokument iz ove vrste datoteke. U cilju je najčešće da se prikažu rezultati obrade željenih podataka u R-u pa će se za to najčešće koristiti ta vrsta datoteke. Naredbe u ovakvim datotekama se pišu u blokove kao i u mnogim drugim programskim jezicima, a blok je moguće generirati pritiskom na Ctrl + Alt + I. Blok izgleda malo drugačije nego kod ostalih programskih jezika u kojima se takvi dijelovi koda navode unutar vitičasih zagrada koje označavaju blok naredbi. Kao i kod R skripte, ovdje je također moguće izvesti pojedinu naredbu pritiskom na kraticu Ctrl + Enter, ali također je moguće izvesti i sve naredbe koje se nalaze u jednom bloku pomoću kratice Ctrl + Shift + Enter.

Kao što i kod drugih programskih jezika, tako i R raspoznaje nekoliko osnovnih ili atomskih tipova podataka. Tomić u izvoru [9] navodi sljedeće tipove:

- realni (*double*)
- cjelobrojni (*integer*)
- znakovni (*character*)
- logički (*logical*)
- kompleksni (*complex*)
- sirovi (*raw*)

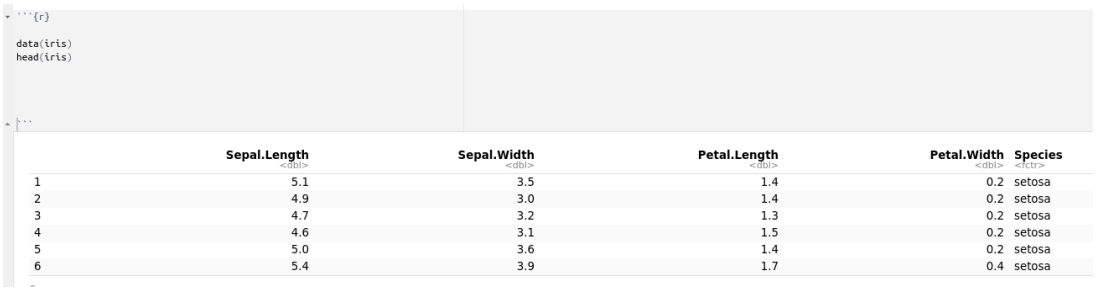
Većinu ovih tipova podataka moguće je prepoznati iz ostalih programskih jezika pa će biti objašnjeni oni za koje se slabije čuje. Kompleksni tip podataka prikazuje kompleksne brojeve u njihovom obliku a+bi. *Raw* prema izvoru [10] predstavlja podatak u svom sirovom obliku. Primjerice, neki string bi se spremio kao niz parova heksadecimalnih znakova.

Tomić u [9] navodi da R također razlikuje funkcije *is.* i *as.* koje se mogu koristiti u kombinaciji s tipom podataka. Uz pomoć njih je moguće provjeriti tip podatka objekta koji se proslijedi kao argument (upotrebom funkcije is.) ili pretvoriti tip podataka proslijeđenog argumenta (ako je moguće) u tip podataka po želji. Te funkcije se mogu kombinirati uz bilo koji objekt u R-u. Također, R nudi i posebnu vrijednost za beskonačnost koja se označava s Inf. Nju je moguće dobiti na način da se neki broj koji nije nula pokuša podijeliti s nulom. S vrijednosti Inf je moguće provoditi i ostale operacije kao što je to moguće i u matematici. Zbrajanjem, primjerice, dva Inf, dobije se Inf, ali u slučaju oduzmanja, dobije se NaN jer nije moguće zaključiti koja je beskonačnost veća. Inače se Inf klasificira kao numerički, realni tip podataka. Također ne postoji funkcija is.Inf, ali postoji is.finite koja vraća logičku istinu ako objekt nije beskonačan, a u suprotnom vraća logičku laž. Nedostajuće vrijednosti u R-u označavaju se oznakom NA kao i u ostalim programskim jezicima, a nedefinirane se označavaju s NaN. Ove tipove je moguće testirati upotrebom funkcija is.na() ili is.nan().

Glavna prednost ovog okruženja je sam sustav monitoringa svega što se događa tijekom izvođenja programa. Za korištenje RStudija, potrebno je prvo učitati željeni set podataka za obradu. U slučaju da se isti ne učita, moguće je koristiti set podataka koji već postoji u RStudiju, a među kojima su prema listi na [11] najpopularniji:

- mtcars
- iris
- ToothGrowth
- PlantGrowth
- USArrests

Za potrebe ovog primjera koristit će se set podataka pod nazivom iris - set podataka o cvijeću. Primjer jednostavne obrade podataka prikazan je na slici 2.



```r
data(iris)
head(iris)
```

	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fctr>
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

Slika 2: Korištenje seta podataka iris [Autorski rad]

Sada je s tim setom podataka moguće raditi što god se namjerava. Primjer jednog takvog korištenja moguće je vidjeti na slici 2 gdje se koristi ugrađena head funkcija head() koja iz opisa na izvoru [12] vraća prvi dio nekog vektora, matrice, tablice, podatkovnog okvira ili funkcije. R ima mogućnost prikaza bilo kojeg seta podataka u grafičkom obliku za vizualizaciju podataka. Autori [6] navode da RStudio koristi već učitani paket pod nazivom "graphics" za kreiranje dijagrama. Neki od dijagrama koje možemo kreirati su: linijski dijagram (*line*), pita dijagram (*pie*), stupčasti dijagram (*barplot*), brkata kutija (*boxplot*), histogram (*hist*) itd.

Kako bi bilo moguće kreirati dijagrame u odnosu na neku varijablu koju set podataka posjeduje, tj. za ispis varijabli koje željeni set podataka posjeduje, koristi se funkcija names() i kao argument joj se prosljeđuje set podataka po želji. U primjeru se proslijeđuje set podataka iris te sve izmjerene varijable tog seta podataka moguće je vidjeti na slici 3.

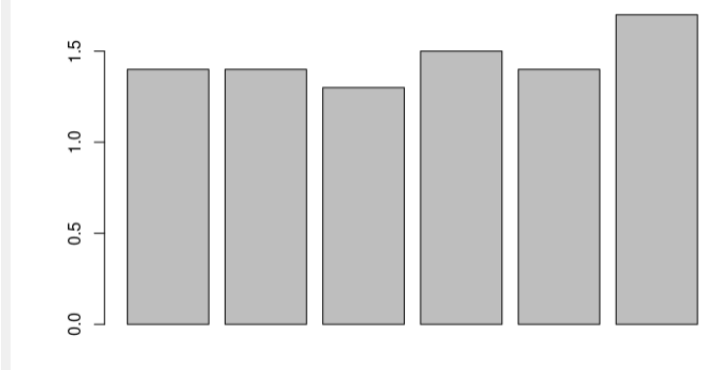
```r
names(iris)
```

```
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

Slika 3: Set podataka iris [Autorski rad]

Za potrebe kreiranja dijagrama od pojedinih varijabli, moguće im je pristupiti preko iris$Sepal.Width. Primjer stupčastog dijagrama s obzirom na varijablu petal.length nalazi se na slici 4, brkata kutija varijable petal.length u odnosu na vrstu cvijeta na slici 5 te histogram varijable petal.length na slici 6. Napomena: korištenje funkcije head kako bi se prikazalo manje podataka radi preglednosti. Stvarnih podataka u setu podataka iris ima puno više nego što je prikazano na slici 2.

```
barplot(head(iris$Petal.Length))
```

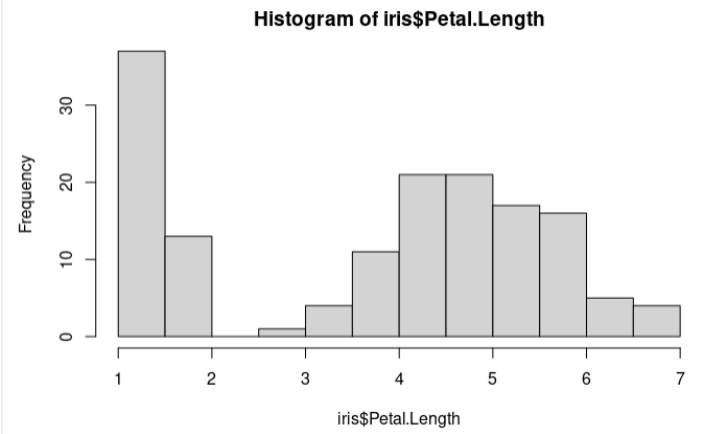


Slika 4: Stupčasti dijagram [Autorski rad]

```
boxplot(Petal.Length~Species, data=iris)
```

Slika 5: Brkata kutija [Autorski rad]

```
hist(iris$Petal.Length)
```



Slika 6: Histogram [Autorski rad]

Vrijednosti varijablama u R-u prema [13] moguće je dodijeliti na više načina. Jedan od njih je onaj učestali u drugim programskim jezicima korištenjem znaka =. Vrijednost je moguće dodijeliti i strelicama lijevo i desno (<- i ->). Primjer dodjeljivanja vrijednosti varijablama u R-u vidljiv je na slici 7.

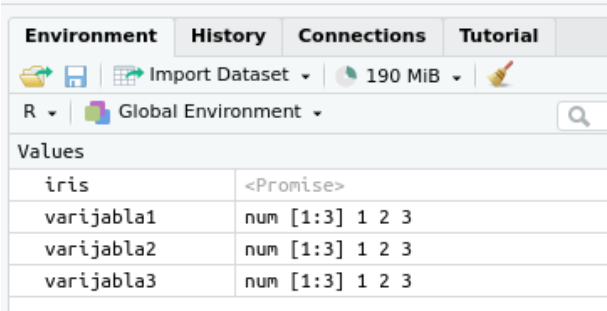
```
varijabla1  = c(1, 2, 3)

varijabla2 <- c(1, 2, 3)

c(1,2,3) -> varijabla3
```

Slika 7: Dodjela vrijednosti varijablama u R-u [Autorski rad]

Na slici 8 vidi se okno okruženja koje prikazuje sve trenutno aktivne objekte koji se koriste u programu i koje vrijednosti su pohranjene u njima. To je jedan od načina monitoringa o kojima je ranije bila riječ.



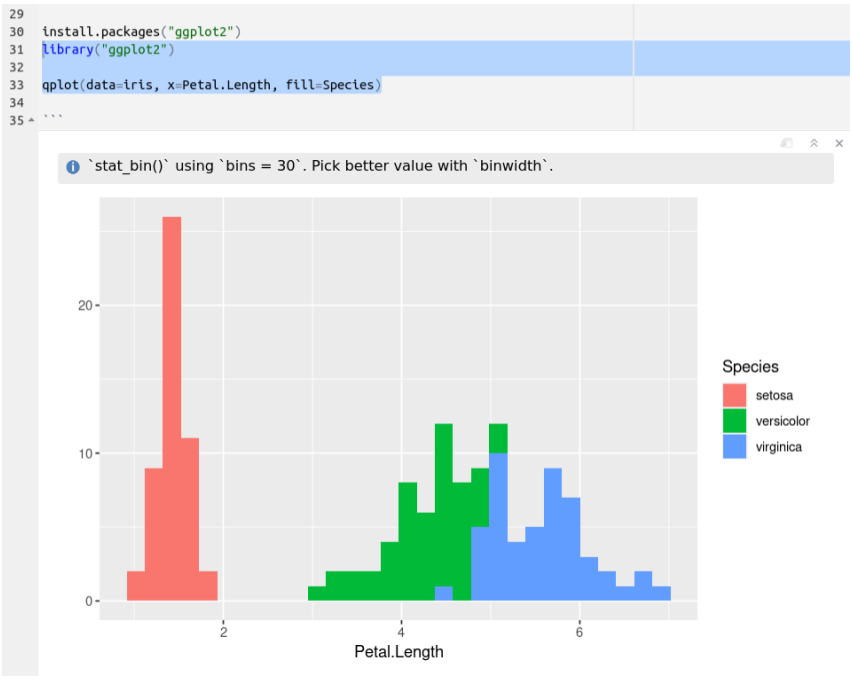
Slika 8: Monitor trenutnih varijabli u RStudiju [Autorski rad]

Uz sve to, R omogućuje i korištenje sučelja prema Prologu o kojem će biti riječi naknadno. To se sučelje naziva rolog i predstavlja paket u RStudiju preko kojeg je moguće zadavati naredbe, oblikovati i generirati bazu znanja, definirati pravila i sve ostalo kao u Prologu.

Paketi se općenito u RStudiju instaliravaju na način da se pozove funkcija install.package() te kao argument se proslijedi string koji predstavlja naziv paketa. Jednom kada se instalira paket, više nije potrebno provoditi tu liniju s obzirom da će RStudio upozoriti da će samo reinstalirati cijeli paket, a taj postupak traje određeno vrijeme te stvarno nije potrebno da se svaki put prilikom pokretanja programa instaliravaju paketi. Paketi se preuzimaju online stoga je potrebna stabilna internetska veza za instaliravanje paketa. Također, sadržaj samih paketa koji se isntaliraju moguće je uključiti i početi koristiti uz pomoć funkcije library() kojoj se u zagradi prosljeđuje naziv željenog paketa.

Prikazat će se jedan primjer korištenja paketa za izradu tzv. "brzih" plot dijagrama. Riječ je o paketu ggplot2 te ga je potrebno instalirati u RStudio razvojno okruženje kako bi se koristile njegove sastavnice. Za primjer će se prikazati jednostavan ggplot2 dijagram od podataka iz podatkovnog seta iris. Slika 9 prikazuje sinstaksu instaliravanja paketa ggplot2, korišteje samog paketa te graf izrađen od podatkovnog seta iris. Evidentno je da je samo potrebno upisati funkciju qplot() te u zagradama definirati koji se podatkovni set koristi, što se

nalazi na x osi te što je tzv. "ispuna". RStudio prikazuje dijagram ovisnosti duljine latice cvijeta s vrstom cvijeta iris. Na ovom se primjeru označavaju redovi 31 do 33 zato što se u redu 30 nalazila komanda za intaliravanje paketa, a koja se pokenula ranije. Označavanjem željenih linija koga i pritiskom na prečac Ctrl + Enter izvšit će se samo komande iz odabranih redova koda, a ostale će se zanemariti.



Slika 9: Instalacija i korištenje paketa ggplot2 unutar RStudija [Autorski rad]

Moguće je primijetiti kako su duljina latice i vrsta cvijeta u podatkovnom setu iris unaprijed određeni te su, pogledom na sliku 2, podatci prikazani unutar tablice te sadrže različite varijable. U R-u je također moguće kreirati jedan takav set podataka koristeći podatkovni okvir. Podatkovni set iris primjer je jednog takvog okvira. Podatkovne okvire Tomić u [9] opisuje kao dvodimenzionalne skupove podataka koji sadrže elemente različitih tipova. "Elementi svakog stupca sadrže isti tip podataka za taj stupac, a sami stupci su jednakih duljina što znači da sadrže jednak broj podataka" [9]. Stupce je moguće gledati kao vektore te je na taj način moguće kreirati vlastiti podatkovni okvir, dok se redak u podatkovnom okviru smatra listom. "Podatkovni okviri se također smatraju i listama vektora istih duljina te za njih onda vrijede pravila i matrica i listi" [9]. Nad podatkovnim okvirima provode se mnoge funkcije od kojih su neke već isporbane ranije u radu. Riječ je o funkcijama: names(), colnames(), rownames(), dim(), length(), ncol(), nrow(), rbind(), cbind() i ostalima.

Primjerice, potrebno je kreirati jedan takav podatkovni okvir. Moguće ga je, kako navodi Tomić u [9], kreirati "ručno" ili pak učitati podatkovni okvir i iz vanjske datoteke (CSV i slične datoteke). Prije svega je potrebno nekoliko vektora istih duljina koji se zajedno spojaju u jedan podatkovni okvir. Zahtjeva se da se obrade podatci o studentima na FOI-ju i nekim njihovim podatcima koji su od relevantne važnosti poput imena, prezimena, godine studija te smjera. Na slici 10 vidljiv je proces kreiranja te krajnji proizvod izrade jednog takvog podatkovnog okvira. Za početak se definira po jedan vektor za svaki podatak koji je važan i koji je potrebno prikazati

u podatkovnom okviru. Zatim se koristi funkcija data.frame() te se u zagrade upisujeu nazivi stupaca te ih se izjednačava sa željenim vektorima za prikaz u tom stupcu. Podatkovni okviri mogu imati proizvoljan broj stupaca što znači da postoje podatkovni okviri i samo s jednim stupcem. Prilikom izvoza podataka, svaki red podatkovnog okvira dobiva svoj id. Definiranjem podatkovnog okvira, on počinje zauzimati mjesto u memoriji, međutim, korisno bi ga bilo prikazati na neki način. Korištenjem funkcije print() te upisivanjem naziva varijable željenog podatkovnog okvira dobijemo tablicu sa slike 10.

```
ime <- c("Andreas", "Filip", "Ana", "Goran", "Tomislav", "Ivan")
prezime <- c("Leja", "Balder", "Horvat", "Bogatić", "Milec", "Zavadil")
godina <- c(3, 3, 3, 2, 2, 1)
smjer <- c("IS", "IS", "IS", "IPS", "IPS", "IPS")

studenti <- data.frame(ime=ime, prezime=prezime, godina_studija=godina, smjer=smjer)
print(studenti)
```

Description: df [6 × 4]

ime <chr>	prezime <chr>	godina_studija <dbl>	smjer <chr>
Andreas	Leja	3	IS
Filip	Balder	3	IS
Ana	Horvat	3	IS
Goran	Bogatić	2	IPS
Tomislav	Milec	2	IPS
Ivan	Zavadil	1	IPS

6 rows

Slika 10: Izrada i prikaz podatkovnog okvira studenti [Autorski rad]

Kada se kreira podatkovni okvir, može se koristiti za daljnju statističku obradu. Tomić u [9] navodi da gotovo svaka statistička obrada u R-u sadrži do neke razine rad s podatkovnim okvirima. Nad kreiranim podatkovnim okvirom moguće je provesti spomenute operacije te usporediti što se dobije kao izlaz. Slika 11 prikazuje što se dobije kao rezultat kada se nad podatkovnim okvirom studenti provedu funkcije dim(), length(), nrow() i ncol(). Kao što je već ranije napomenuto, odgovor na svaku funkciju u R-u je također vektor. Funkcija dim() vraća dimenziju podatkovnog okvira, length() u slučaju podatkovnih okvira vraća broj stupaca podatkovnog okvira, a ne duljinu pojedinog vektora kako bi to bilo očekivano te na taj način funkcionira na isti način kao i ncol(), a nrow() vraća broj redaka u podatkovnom okviru.

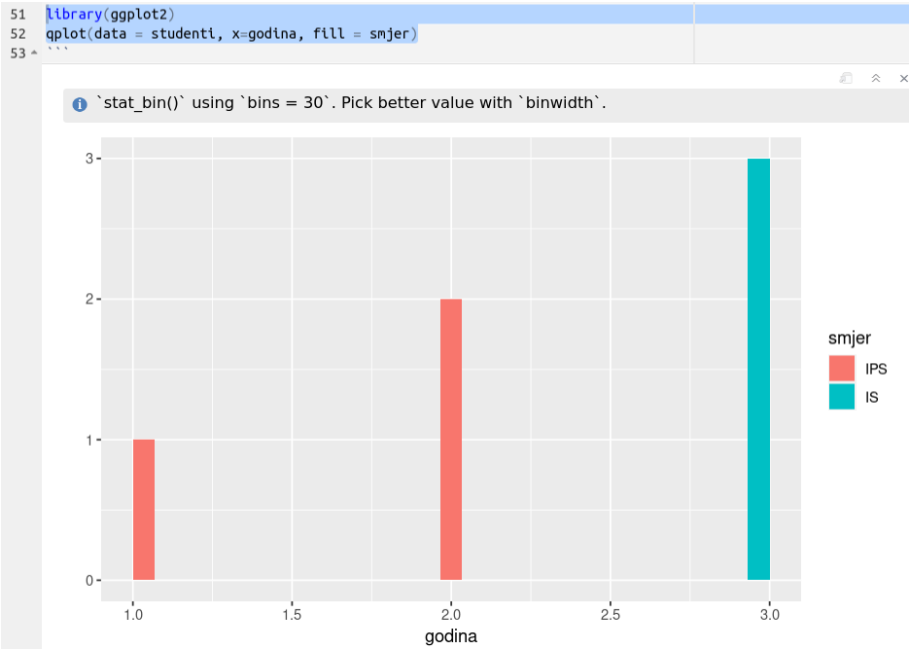
```
dim(studenti)
length(studenti)
nrow(studenti)
ncol(studenti)
```

```
[1] 6 4
[1] 4
[1] 6
[1] 4
```

Slika 11: Operacije nad podatkovnim okvirom studenti [Autorski rad]

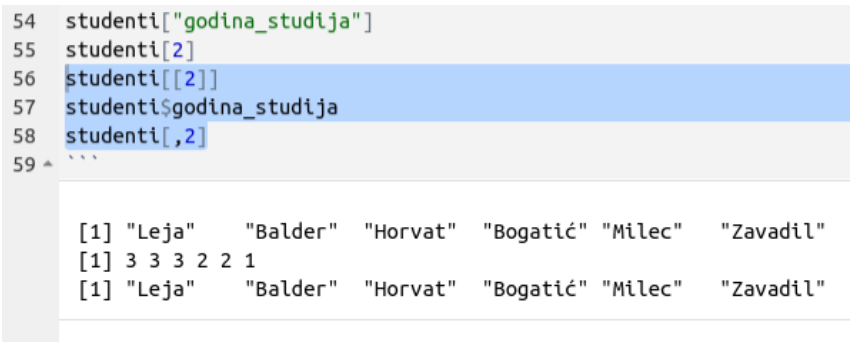
Nad podatkovnim okvirima također je moguće provoditi i neke značajnije i naprednije obrade. Tako se dobiju, primjerice, grafička obrada podataka iz podatkovnog okvira i mnogi drugi "korisniji" rezultati. Koristeći već instalirani paket ggplot2, moguće je prikazati brzi plot

dijagram našeg podatkovnog seta. Potrebno je prikazati koliko je studenata iz podatkovnog seta na kojoj godini studija. Koristit će se znanje o brzom plot dijagramu koje se već koristilo na primjeru podatkovnog seta iris te ga je moguće primijeniti da odgovara trenutnoj situaciji. Slika 12 prikazuje brzi plot dijagram distribucije godine studiranja po broju studenata na godini.



Slika 12: Brzi plot dijagram nad podatkovnim okvirom studenti [Autorski rad]

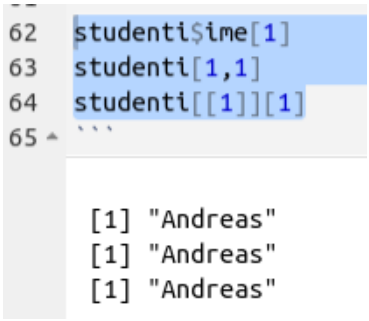
U slučaju da je potrebno pristupiti pojedinim podatcima iz podatkovnog seta, to se može lako izvesti korištenjem ranije stečenog znanja. Pojedini se elementi selektiraju operatorima [,] te operatorom $ s obzirom na to da podatkovni okviri imaju svojstva i matrica i listi. Tomić također u [9] navodi da je "moguće koristiti dvostruke uglate zagrade ([[ ]]), međutim, taj pristup se u praksi ne primijenjuje". Slika 13 prikazuje neke od načina na koje je moguće izdvojiti pojedine podatke iz podatkovnog okvira. Primijetimo da označeni redovi vraćaju vektor. To je zato što jednostruke uglate zagrade vraćaju podatkovne okvire, a ostali operatori vraćaju podatke u obliku vektora.



Slika 13: Izdvajanje podataka iz podatkovnog okvira studenti [Autorski rad]

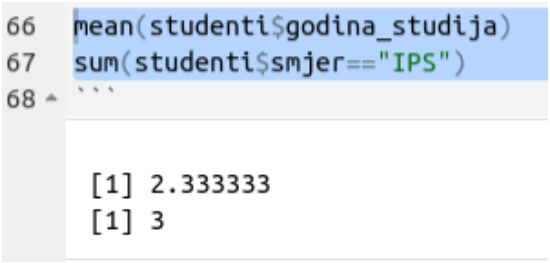
Na takav način se odabiru i pojedinačne vrijednosti iz podatkovnog seta. Primjerice, potrebno je odabrati prvi red i prvi stupac, to je moguće jednostavno napraviti na nekoliko načina. Uzimanjem prvog stupca kao vektora te odabirom prvog podatka iz tog vektora ili pak

odabirom prvi podatak iz prvog vektora u podatkovnom okviru. Slika 14 prikazuje navedene načine pristupanja elementu prvog stupca i prvog reda podatkovnog okvira. Evidentno je da sva tri načina vraćaju isti podatak iz podatkovnog okvira studenti.



Slika 14: Ispis elementa prvog reda i prvog stupca [Autorski rad]

Također, iz podatkovnih okvira moguće je provoditi operacije nad podatcima poput sume, aritmetičke sredine i sličnih operacija. Na slici 15 vidljivo je kako se iz podatkovnog okvira studenti izračunava prosječna godina studija na kojoj se nalaze studenti te dohvaćanje broja studenata koji se nalaze na smjeru IPS.



Slika 15: Prosjek i suma nad podatcima u podatkovnom okviru [Autorski rad]

Obratit će se pažnja malo na to kako R obrađuje i bira koje podatke koristiti, a koje ne kada mu se postavi neko ograničenje te će biti prikazano na primjeru podskupa podataka iz podatkovnog okvira. Za tu namjenu će se koristiti funkcija subset() u koju je potrebno proslijediti koji podatkovni okvir se koristi te ograničenje, odnosno po kojem kriteriju se podatci izdvajaju iz podatkovnog okvira. Način na koji R odabire redove jest taj da se "ispod površine" svakog vektora nalazi još jedan vektor koji sadrži samo *boolean* vrijednosti te tumači koji podatci će se koristiti, a koji ne. Ako R *interpreter* naiđe na element koji sadrži *TRUE*, odnosno koji zadovoljava postavljeni kriterij, R će ga ispisati, a ako pak sadrži *FALSE*, podatak se neće ispisati ili koristiti. Tako je moguće razmišljati i kada je potrebno ispisati sve podatke iz nekog vektora jer se tu radi o vektoru punom *TRUE* vrijednosti. Kada se postavi neko ograničenje, mjesta na kojima se nalaze elementi koji odgovaraju ograničenju ostaju *TRUE*, a ostali poprimaju vrijednost *FALSE* te se samim time ne koriste u obradi. S tim na umu nastaje slika 16 koja prikazuje podskup studenata koji su trenutno na trećoj godini studija.

Podatke iz podatkovnog okvira moguće je sortirati. Za tu potrebu će se koristiti funkcija order() koja funkcionira na način da joj se proslijedi željeni podatkovni okvir te ga ona sortira silazno. Točnije, izlaz iz ove funkcije je zapravo vektor koji sadrži id-eve poredane po kriteriju koji mu je priložen u zagradi. Funkcija po *defaultu* sortira podatke silazno, ali moguće je pridodati i

```
subset(studenti, godina_studija==3)
```

Description: df [3 × 4]

	ime <chr>	prezime <chr>	godina_studija <dbl>	smjer <chr>
1	Andreas	Leja	3	IS
2	Filip	Balder	3	IS
3	Ana	Horvat	3	IS

3 rows

Slika 16: Podskup podataka iz podatkovnog okvira studenti [Autorski rad]

opciju "decreasing = T" kao jedan od argumenata kako bi se ostvario odabir podataka sortiranih silazno. Podatci iz primjera su već sortirani na takav način tako da tu dodatnu opciju nije potrebno koristiti. Slika 17 prikazuje kako bi se prikazali podatci sortirani ovom funkcijom. S obzirom da je poznatno da funkcija vraća vektor koji sadrži id-eve redova koji zadovoljavaju kriterij, moguće ih je ubaciti u uglatu zagradu. Na taj način će biti ispisani podatci 6, 4, 5, a zatim 1, 2 i 3.

```
order(studenti$godina_studija)
studenti[order(studenti$godina_studija),]
```

Description: df [6 × 4]

	ime <chr>	prezime <chr>	godina_studija <dbl>	smjer <chr>
6	Ivan	Zavadil	1	IPS
4	Goran	Bogatić	2	IPS
5	Tomislav	Milec	2	IPS
1	Andreas	Leja	3	IS
2	Filip	Balder	3	IS
3	Ana	Horvat	3	IS

Slika 17: Korištenje funkcije order i ispis sortiranih podataka [Autorski rad]

Prikazat će se i kako se briše sadržaj iz podatkovnog okvira. Brisanje se također, ovisno o namjeni, može izvesti na više različitih načina. Moguće je obrisati cijele stupce, cijele zapise ili je pak moguće obrisati pojedine ćelije. Za brisanje cijelog jednog stupca, potrebno je odabrati željeni stupac koji i dodijeliti mu vrijednsot *NULL* ili pak u uglate zagrade podatkovnog okvira napisati broj željenog stupca s predznakom minus (-). Valja napomenuti da dodjeljivanje cijelom stupcu vrijednost NULL briše sadržaj tog stupca iz memorije i drugim riječima mijenja sadržaj podatkovnog okvira, dok je upisivanjem broja stupca s negativnim predznakom brisanje samo u svrhu prezentacije. Podatkovni okvir nakon druge metode ostaje nepromijenjen u memoriji. Na slici 18 vidljiv je primjer brisanja redova. Evidentno je da se uistinu nakon pridruživanja vrijednosti izgubi cijeli stupac dok to kod druge metode nije tako. Iz tog razloga postoje dva *chunka* R koda: prvi koji pokazuje da su se obrisali željeni stupci te drugi koji pokazuje kako je brisanje imena studenta bilo samo za potrebe prikaza.

Naravno, drugi način brisanja moguće je također pohraniti u memoriju na način da se dodijeli izlaz iz te operacije na već postojeću (ili novu) varijablu. Brisanje pojedinog retka odvija se na sličan način kao selekcija pojedinog retka, ali ga je za ovu namjenu potrebno napisati s negativnim predznakom. Na sličan je način moguće ukloniti i više redaka koristeći funkciju c() u čije se zagrade upisuju brojevi redaka koje je potrebno obrisati. Na sličan način koriste se i sekvence korištenjem funkcije seq() za koju Tomić u [9] tvrdi da je slična operatoru ":",

```
studenti$godina_studija <- NULL
studenti[-1]
```

Description: df [6 × 2]

prezime <chr>	smjer <chr>
Leja	IS
Balder	IS
Horvat	IS
Bogatić	IPS
Milec	IPS
Zavadil	IPS

6 rows

```{r}
studenti
```

Description: df [6 × 3]

ime <chr>	prezime <chr>	smjer <chr>
Andreas	Leja	IS
Filip	Balder	IS
Ana	Horvat	IS
Goran	Bogatić	IPS
Tomislav	Milec	IPS
Ivan	Zavadil	IPS

Slika 18: Brisanje stupaca u R-u [Autorski rad]

ali uz pomoć te funkcije se mogu uređivati uvjeti po kojima se sekvenca generira. Na takav je način moguće dobiti samo parne brojeve, samo neparne brojeve, svaki treći broj u nizu, svaki deseti itd. To omogućuje i prilagođeni prikaz podataka. Za potrebe primjera korištena je jednostavna funkcija c(). Brisanje pojedine vrijednosti funkcionira na način da se "targetira" specifična vrijednost koju je potrebno obrisati primjenjujući uvjete nad podatkovnim okvirom i postavljanjem vrijednosti "NA". Tomić nam u [9] također savjetuje da "je pametno dodjeljivati vrijednost 'NA' jer nam ona neće poremetiti tipove podataka unutar podatkovnog okvira" [9]. Primjerice, za dodjelu praznog stringa kao elementa brisanja, cijeli stupac će se pretvoriti u tip podataka *character* što najčešće nije poželjno. Slika 19 prikazuje brisanje reda korištenjem dvije metode te brisanje vrijednosti po određenom kriteriju.

```{r}
studenti <- studenti[-c(2, 4, 6),]
studenti <- studenti[-1,]
studenti$smjer[studenti$ime=="Tomislav"] <- NA
studenti
```

Description: df [2 × 3]

	ime <chr>	prezime <chr>	smjer <chr>
3	Ana	Horvat	IS
5	Tomislav	Milec	*NA*

2 rows

Slika 19: Brisanje reda i vrijednosti u R-u [Autorski rad]

3. Prolog

Autori [14] tumače da je Prolog kratica za PROgramming in LOGic. "Programski jezik razvili su 1972. godine Alain Colmerauer i Philippe Roussel kao prvi logički programski jezik. Temelji se na predikatnoj logici ili logici prvog reda" [14] koja se iz [15] koristi kvantificirane varijable nad nelogičnim objektima. Najčešće se povezuje uz područje umjetne inteligencije i računalne lingvistike. Postoje i računala čiji su dijelovi programirani upravo u ovom programskom jeziku. Primjer jednog takvog računala je superračunalo pod nazivom IBM Watson SuperComputer. Prolog se koristi za ekspertne sustave, procesiranje prirodnog jezika, inteligentne baze podataka te mnoge druge primjene korištenja umjetne inteligencije. Također, sama NASA u svom [16] navodi da koristi Prolog, odnosno svoju modificiranu verziju Prologa za svoj ekspertni sustav. Ta verzija se sastoji od Prologa koji je pisan u Forthu. Na taj način su stučnjaci uspjeli spojiti Forth i Prolog, nešto će biti ostvareno i u ovom radu radu s R-om i Prologom. Taj njihov "ekspertni sustav se sastoji od baze podataka koja sadrži detaljne operacijske upute za svaki eksperiment koji provode, a klauzule bazirane na pravilima u Prologu određuju koji je sljedeći korak u izvođenju samog eksperimenta" [16].

3.1. Logičko programiranje

Prolog je logički programski jezik te se vodi pravilima logičkog programiranja. Imam na [17] navodi da "logičko programiranje svoje početke gradi na vremenima kada su se proceduralno i deklarativno programiranje koristili kako bi se predstavilo znanje u području umjetne inteligencije. Prvi takav programski jezik zvao se Planner" [17].

Logički programski jezici prate programiranje bazirano na logici, što znači da takvi jezici sadrže pojmove koji prate logiku kako bi izrazili činjenice i razna pravila. Takvi jezici također koriste set već postojeće logike u obliku baze kroz predikate. Oni se koriste kako bi se izrazile neke činjenice kroz logičko programiranje. Postoje i različite varijante logičkog programiranja koje rad neće obrađivati.

Imam na [17] tvrdi da iako logičko programiranje nije toliko popularno i efektivno primjenjivo za sve zadatke, takvi programski jezici obavljaju odličan posao pretvaranja logičkih iskaza u računalne programe. Takvi se jezici koriste često u poslovnim i istraživačkim okruženjima zbog svoje sigurnosti i preciznosti - dvije osobine koje su poželjne prilikom izbora programskog jezika. Najčešće se koriste na područjima:

- Umjetne inteligencije
- Obrade prirodnog jezika
- Big data
- Baza podataka
- Analize i predviđanja
- Pronalaska uzorka i dr.

3.2. SWI-Prolog

[18] navodi da je SWI-Prolog svestrana implementacija Prologa kao jezika i predstavlja besplatno okruženje za rad s Prologom. Najčešće se koristi pri poučavanju i za izradu semantičkih web aplikacija. To okruženje pruža širok spektar raznih sučelja prema drugim tehnologijama, paketa, potpora za razne tipove dokumenata, protokola itd. Na [19] se navodi da se ne smije miješati SWI-Prolog i sam programski jezik Prolog. SWI-Prolog je IDE koji omogućuje da se korisnici uopće služe svime onime što Prolog kao programski jezik pruža. Postoji i *online* verzija ovog okruženja pod imenom SWISH na koju će se ovaj rad fokusirati, a koji će biti opisan malo kasnije.

3.3. Primjer korištenja Prologa

[14] navodi da se sam Prolog vodi principom zatvorenog svijeta. To znači da Prolog ima bazu znanja koju provjerava kada god mu je neki upit zadan. Primjer jedne takve baze znanja vidljiv je na slici 20. Na spomenutoj slici vidi se nekoliko različitih činjenica te jedno pravilo, a koje glasi: "Kolegij postoji, ako postoji nastavnik koji je nositelj kolegija". Nositelj činjenicu čita se kao: "Nositelj kolegija X je nastavnik Y". Princip zatvorenog svijeta govori da je istinito sve što se nalazi u bazi znanja, a sve ostalo nije istinito. U bazu se upisuju klauzule, a koje mogu biti ili činjenice ili pravila. Svaka klauzula u Prologu ima svoju kratnost, odnosno broj argumenata koje zahtjeva određena klauzula.

Prema navedenoj je bazi znanja moguće slati upite. Autori [14] navode da upiti u prologu funkcioniraju na način prepoznavanja i ispunjavanja uzorka u uvjetu. Postoje li u trenutnoj bazi znanja činjenice koje ispunjavaju tzv. cilj našeg upita, prolog će vratiti istinu, a ako ne, onda će "failati", odnosno, vratit će laž. Upiti se dijele na jednostavne i složene. Primjer nekoliko takvih upita se nalazi na slici 21.

Primjer jednostavnog upita bio bi primjer u kojem se pita postoji li kolegij webdip ili prog2. Iako je poznato da kolegij prog2 postoji u stvarnosti, nije deiniran u bazi znanja pa u primjeru ni ne postoji. Zato Prolog javlja laž kao odgovor na taj upit. Isto vrijedi i za smjer ipi koji nije definiran u bazi znanja. Usklađivanje na ovakvim upitima funkcionira po principu da se prolazi kroz cijelu bazu znanja i Prolog pokušava uskladiti ono što se od njega traži. Uvjeti kada usklađivanje prolazi kod jednostavnih upita:

- ako je isto imenovan predikat i u cilju i u bazi znanja
- ako je kratnost i upita i činjenice ista
- ako i upit i činjenica imaju jednake argumente

Također, na istoj slici (slici 21) vidljivi su i upit gdje su napisani obvezan(X, is), odnosno, gdje se provodi upit oko toga koji su sve obvezni kolegiji na smjeru is. Evidentno je da tu zapravo postoji jedna varijabla X u upitu. Zbog toga usklađivanje funkcionira malo drugačije. Nikako se ne bi trebale zamijeniti varijable u Prologu za uobičajene varijable iz ostalih programskih jezika

jer funkcioniraju malo drugačije, pa se tako i nazivaju logičke varijable. Proces usklađivanja je sličan kao u prvom slučaju, ali ovoga puta varijabla X može poprimiti sve vrijednosti kod kojih u bazi znanja dođe do podudaranja. Autori [14] navode kako se taj postupak naziva vezivanje varijable te se zato u primjeru dobije više odgovora, odnosno X je poprimila više vrijednosti.

```
smjer(is).
smjer(ps).

nositelj(webdip, kermek).
nositelj(nfm, schatten).
nositelj(oi1, zugecb).
nositelj(po, begicevicredep).
nositelj(ppp, detelj).
nositelj(pi, stapic).
nositelj(vp, zajdelahrustek).

obvezan(webdip, is).
obvezan(oi1, is).
obvezan(pi, is).
obvezan(oi1, ps).
obvezan(po, ps).
obvezan(ppp, ps).

izborni(webdip, ps).
izborni(pi, ps).
izborni(po, is).
izborni(ppp, is).
izborni(vp, is).
izborni(vp, ps).
izborni(nfm, is).
izborni(nfm, ps).

kolegij(X):- nositelj(X, _).
```

Slika 20: Činjenice i pravilo u bazi znanja Prologa [Autorski rad]

Također, postoje i složeni upiti koji su sastavljeni od nekoliko jednostavnih upita koji moraju biti usklađeni. Takvi upiti prema [14], koriste tehniku zvanu *backtracking*. Prilikom tog postupka se od složenijeg upita sastavi popis jednostavnijih ciljeva te zatim krećemo u usklađivanje prvog cilja. Ako usklađivanje prođe, onda idemo na sljedeći cilj s istom varijablom. Ako zadnji cilj prolazi, vrijednost se ispisuje i ako se ne dođe do kraja popisa jednostavnih ciljeva, ide se natrag i ciklus se ponavlja s drugačijom varijablom. Na kraju svaki takav upit vraća false jer na popisu ne preostaje više vrijednosti koje varijabla može poprimiti.

U primjeru će se izvesti složeni upit na način da ima smisla u kontekstu kolegija na fakultetu. Dodat će se nova činjenica u bazu znanja student/1 i stvorit će se novo pravilo koje će dopustiti odabir izbornih kolegija. Kolegij se može izabrati ako kolegij postoji (a što je poznato iz pravila kolegij/1), ako postoji student na smjeru te ako je kolegij na popisu izbornih kolegija za smjer koji studira odabrani student. Primjer promijenjene baze znanja nalazi se na slici 22, a odgovor na upite na slici 23. Prvi upit ne prolazi jer student ide na smjer is, a webdip

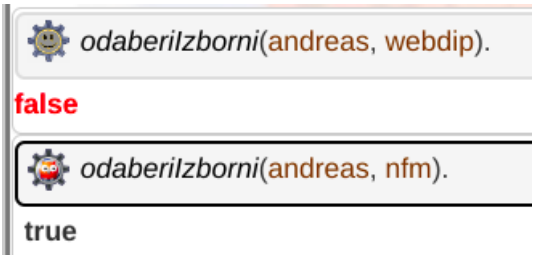
Slika 21: Upiti na bazu znanja sa slike 9. [Autorski rad]

je na popisu obveznih kolegija na tom smjeru (što znači da nije na popisu izbornih). Drugi upit prolazi jer kolegij nfm zadovoljava sve uvjete za odabranog studenta.

```
student(andreas, is).

odaberiIzborni(X, Y):- kolegij(Y), student(X, Z), izborni(Y, Z).
```

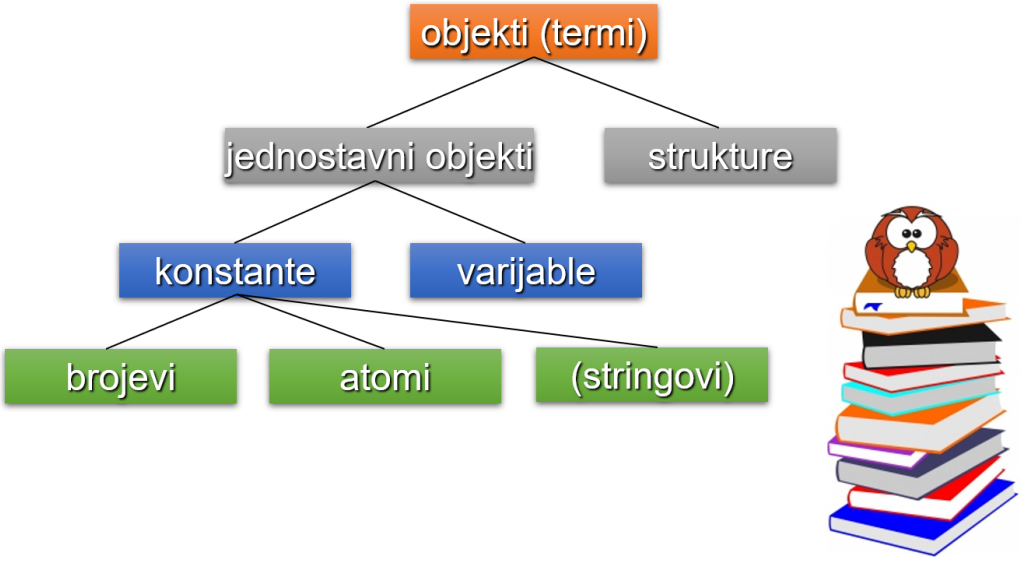
Slika 22: Ažurirana baza znanja [Autorski rad]



Slika 23: Primjeri novog upita [Autorski rad]

Još će se obratiti pažnja na tipove podataka koji se nalaze u Prologu. Kao što je vidljivo na slici 24, objekti se u Prologu dijele na jednostavne objekte i na strukture. Strukture autori [14] opisuju kao objekte koji se sastoje od više komponenti. Mogu biti građeni od objekata ili također od drugih struktura pa ih je moguće prikazati kao stablo. Jednostavni objekti se dijele na konstante (ne mijenjaju vrijednost) i varijable. Pravila imenovanja varijabli je slična kao u regularnim programskim jezicima uz dodatak tzv. anonimnih varijabli koje su se koristile i koje su prikazane na slici 7 kada se opisuje pravilo za kolegij. Takve varijable se koriste kada se neka varijabla javlja samo jednom u nekoj klauzuli te se označavaju znakom "_". Prolog bi također upozorio da se koristi tzv. *singleton* varijabla u slučaju da se umjesto "_" postavi

regularna varijabla s obzirom na to da se nigdje drugdje ne koristi. Konstante mogu biti brojevi, atomi i stringovi. Brojevi i stringovi su poprilično intuitivni pa će atomi biti objašnjeni. Autori [14] navode da su atomi izrazi koji počinju malim slovom, a nazivi atoma se sastoje od slova, brojki i "_", posebnih znakova osim znakova koji u Prologu imaju ugrađeno posebno značenje poput ; : - ** ==> ili niz znakova unutar jednostrukih navodnika.



Slika 24: Tipovi podataka u Prologu [20]

Prolog se također sastoji i od nekih ugrađenih predikata. Oni su definirani u samom Prologu i koriste se prilikom složenih uvjeta za razne namjene. Autori [14] nude nekoliko primjera ugrađenih predikata i njihove namjene. Write/1 služi za ispisivanje proslijeđenog argumenta (koji može biti i varijabla) na zaslon. Predikat nl/0 je onaj koji se koristi kada je potrebno prijeći u novi red prilikom ispisa na zaslon. Također, postoji i tab/1, predikat koji služi kada želimo ispisati razmak. Broj razmaka koji će se ispisati ovisi o broju koji mu je proslijeđen kao argument. Usklađivanje ovih predikata će uvijek uspjeti kod samog poziva, ali neće uspjeti prilikom ponavljanja. Za razliku od ovih koji su navedeni, postoji i predikat fail/0 čije usklađivanje nikada neće uspjeti. Koristi se kada postoji uvjet koji ne smije proći kao napredak pa je potrebno vratiti laž na cijelu klauzulu.

Upit sa slike 23 zapravo ne radi ništa korisno za program osim što govori da je moguće da student andreas odabere izborni kolegij nfm. Kako bi se nekako označilo da je student andreas stvarno upisao kolegij nfm, koristit će se ugrađeni predikati za ažuriranje baze znanja. Autori [14] objašnjavaju takve predikate i nude opis svakog. Asserta/1 je predikat koji se koristi kako bi Prolog postavio klauzulu koja se nalazi u njegovom argumentu kao prvu klauzulu tog predikata u dinamičku bazu znanja. Na vrlo sličan način funkcionira i ugrađeni predikat assertz/1, s tim da se pri tome postavlja klauzula kao posljednja klauzula predikata. Kao što postoje predikati za unos u dinamičku bazu znanja, tako postoje i predikati za brisanje iz dinamičke baze znanja. Za to se koristi ugrađeni predikat retract/1 kojim se brišu sve klauzule koje se mogu uskladiti s onom koja je proslijeđena u argumentu predikata.

Kao i kod ostalih programskih jezika, tako i u Prologu postoji negacija. Koristit će se

preko još jednog ugrađenog predikata not/1. Autori [14] za predikat not/1 tvrde da on kao svoj argument prima neki cilj koji pokušavamo uskladiti. U slučaju da taj cilj ne uspije, not će uspjeti, a u slučaju da cilj uspije, onda not neće uspjeti. Također, valja napomenuti da će Prolog po principu zatvorenog svijeta, a kojeg smo već opisali ranije, vratiti istinu na sve što se nalazi u bazi znanja, a sve što se ne nalazi u njoj ili se ne zna će vratiti laž. To treba imati na umu prilikom korištenja not/1 predikata.

Postoji još jedan koncept u vezi Prologa koji će se obraditi, a to je rez. On se u Prologu predstavlja kao znak "!", a uz pomoć njega se isključuje backtracking. To je vrlo korisno u situacijama kada, primjerice, nije potrebno da se nakon usklađivanja nekog cilja nastavi dalje usklađivanje. Autori [14] za rez kažu da će "odrezati sve ciljeve ispred" [14], tj. ti ciljevi će moći dati rezultate samo tijekom prvog usklađivanja.

Uz sve to će se izmijeniti primjer za upis kolegija. Prije svega potrebno je pretvoriti predikat upisanIzborni/2 u *dynamic* kako bi ga se moglo modificirati. Prilagodit će se pravilo odaberiIzborni/2 tako da se pokriju i neki mogući scenariji. Ovdje se vrlo dobro vidi velika prednost Prologa kao logičkog programskog jezika jer će se na ovaj način prilagoditi svaki scenarij i opisati što se događa u svakom slučaju. Pravilo za upis kolegija je prilagođeno kao na slici 25.

```
odaberiIzborni(X, Y):- kolegij(Y), student(X, Z), izborni(Y, Z), upisanIzborni(X, Y),
    write("Student "), write(X), write(" već je upisao kolegij "), write(Y), nl, !.
odaberiIzborni(X, Y):- kolegij(Y), student(X, Z), not(izborni(Y, Z)), not(upisanIzborni(X, Y)),
    write("Kolegij "), write(Y), write(" se ne nalazi na listi izborbnih kolegija na smjeru studenta ("), write(Z), write(")"), nl, !.
odaberiIzborni(X, Y):- kolegij(Y), student(X, Z), izborni(Y, Z), not(upisanIzborni(X, Y)),
    asserta(upisanIzborni(X, Y)), write("Student "), write(X), write(" upisao je kolegij "), write(Y), nl, !.
```

Slika 25: Izmijenjena pravila za upis kolegija [Autorski rad]

Moguće je upisati studenta andreas na kolegij nfm dva puta da se isproba što će Prolog ponuditi kao odgovor. Nakon toga će ga se pokušati upisati na kolegij koji postoji, a koji nije na listi izbornih kolegija za njegov smjer (is) iz istog razloga. Na slici 26 vidljiv je rezultat svega navedenog.

```
odaberiIzborni(andreas,nfm), odaberiIzborni(andreas, nfm),
odaberiIzborni(andreas, pi), odaberiIzborni(andreas, vp).
Student andreas upisao je kolegij nfm
Student andreas već je upisao kolegij nfm
Kolegij pi se ne nalazi na listi izborbnih kolegija na smjeru studenta (is)
Student andreas upisao je kolegij vp
true
```

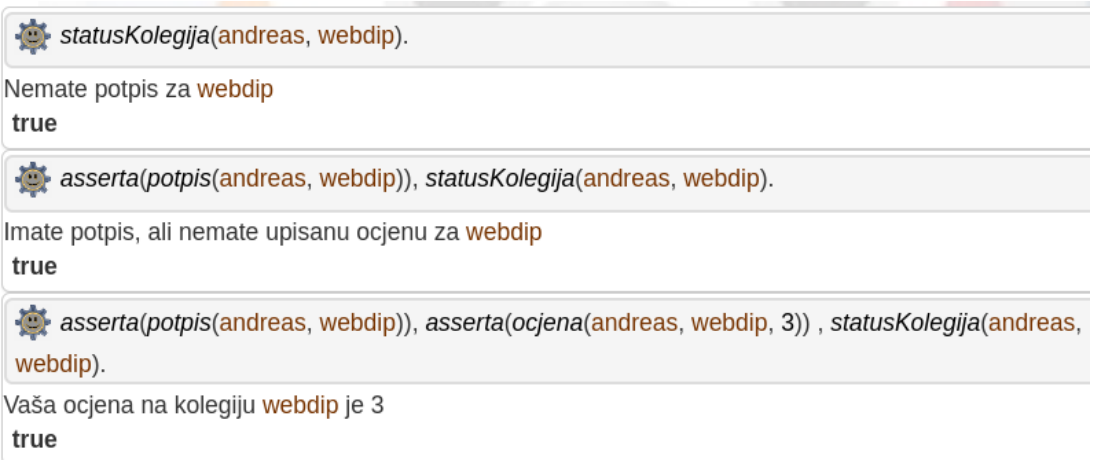
Slika 26: Pokušaj upisa studenta na kolegije [Autorski rad]

Prolog također može provoditi selekcije kao primjerice u ostalim programskim jezicima koje se u Prologu nazivaju još i kontrolnim strukturama. U takve strukture ulaze strukture *if-else* i *repeat*. Predikat repeat/0 ponavlja cilj koji se nalazi iza njega te na taj način onemogućuje *backtracking*. Sintaksa se, očekivano, razlikuje od sintakse kod ostalih programskih jezika za *if-else* selekciju. Cilj1 -> cilj2; cilj3. znači da će Prolog krenuti na cilj1 i ako on uspije, onda

će pokušati cilj2, a ako cilj1 ne uspije, onda će Prolog pokušati cilj3. Primjer provjere statusa na kolegiju poslužit će kao prikaz ovog pravila u praksi. Potrebno je osmisliti klauzulu koja će za navedenog studenta i kolegij ispisati studentov status na tom kolegiju (potpis i ocjenu ako postoje). Slika 27 prikazuje pravilo koje pokriva svaku mogućnost u tom slučaju. Polazi se od pretpostavke da korisnik ne pokušava doznati status kolegija kojeg student nije upisao te tako provjerava postoji li postavljena činjenica o potpisu za studenta i kolegij te ako taj cilj prođe, nastavlja u sljedeće grananje koje ispituje ima li student upisanu ocjenu za taj kolegij te ako taj cilj prolazi, ispisuje se ocjena, a ako ne, onda se ispisuje da ne postoji evidentiran podatak o ocjeni za taj kolegij. Na slici 28 je prikazan primjer izvođenja statusKolegija/2 na tri različita primjera: kada je kolegij upisan, neodslušan, kada je kolegij odslušan, neocjenjen te kada je kolegij odslušan i ocjenjen. Vidljiva je razlika u ispisu za svaki predočeni scenarij.

```
44 statusKolegija(Student, Kolegij):-
45     not(potpis(Student, Kolegij))->
46     write("Nemate potpis za "), write(Kolegij);
47     not(ocjena(Student, Kolegij, _)) ->
48     write("Imate potpis, ali nemate upisanu ocjenu za "), write(Kolegij);
49     ocjena(Student, Kolegij, Ocjena), write("Vaša ocjena na kolegiju "),
50     write(Kolegij),write(" je "), write(Ocjena).
```

Slika 27: Status kolegija korištenjem selekcije [Autorski rad]



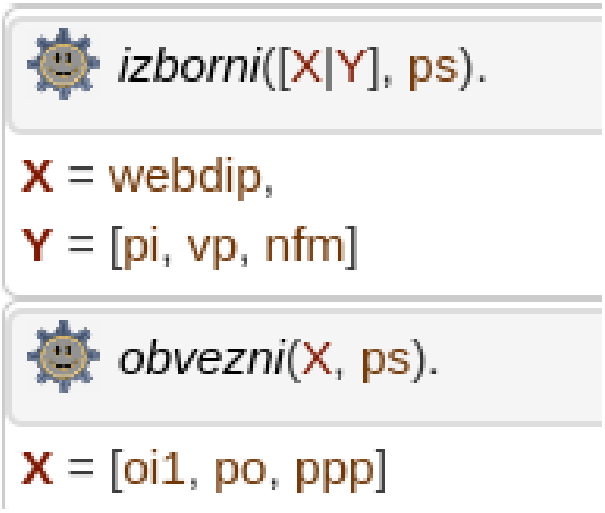
Slika 28: Primjer upita za status na kolegiju [Autorski rad]

Kao i mnogi popularni programski jezici, tako i Prolog sadrži liste. One su odličan način za povezivanje više jednostavnih termina u složeniji. Tijekom izvođenja programa se broj elemenata u listama mijenja. To je značajna razlika u odnosu na strukture, primjerice. Valja napomenuti kako se u Prologu (a tako i u ostalim programskim jezicima) razlikuje zaseban termin od jednočlane liste. Autori [14] tvrde da bi se mogle provoditi operacije nad listama, potrebno je omogućiti operacije pretraživanja liste određenim elementom, brisanje pojedinog elementa iz liste, dodavanje novog elementa u listu te dohvaćanje željenog elementa iz liste. Ta pravila nisu ugrađena i potrebno ih je samostalno definirati. Međutim, postoji poseban način prikaza liste u obliku [X|Y] pri čemu X predstavlja prvi element liste kao termin, a Y predstavlja tzv. "rep" liste, a koji je zapravo samo polazna lista bez prvog elementa. Konkretno je, primjerice, moguće u listu upisati sve podatke o izbornim i obveznim kolegijima kako bi se lakše dohvaćali. Kolegije

će biti zapisani unutar uglatih zagrada, a termin koji govori o kojem se smjeru radi bit će napisan izvan tih uglatih zagrada. Slika 29 prikazuje skraćeni način popisa izbornih i obveznih kolegija za svaki smjer, a na slici 30 vidljiv je odgovor na upite za kolegije zapisane u listama na smjeru ps. Također, na istoj slici (slici 30) evidenta je i mogućnost izvlačenja prvog elementa iz liste kao termina te prikaz ostatka (repa) liste. Koristeći se takvim prikazom moguće je čak i manipulirati elementima u listi te se tako mogu ostvariti gore navedene operacije nad elementima u listama.

```
39 obvezni([webdip, oi1, pi], is).
40 obvezni([oi1, po, ppp], ps).
41 izborni([po, ppp, vp, nfm], is).
42 izborni([webdip, pi, vp, nfm], ps).
```

Slika 29: Obvezni i izborni kolegiji u listi [Autorski rad]

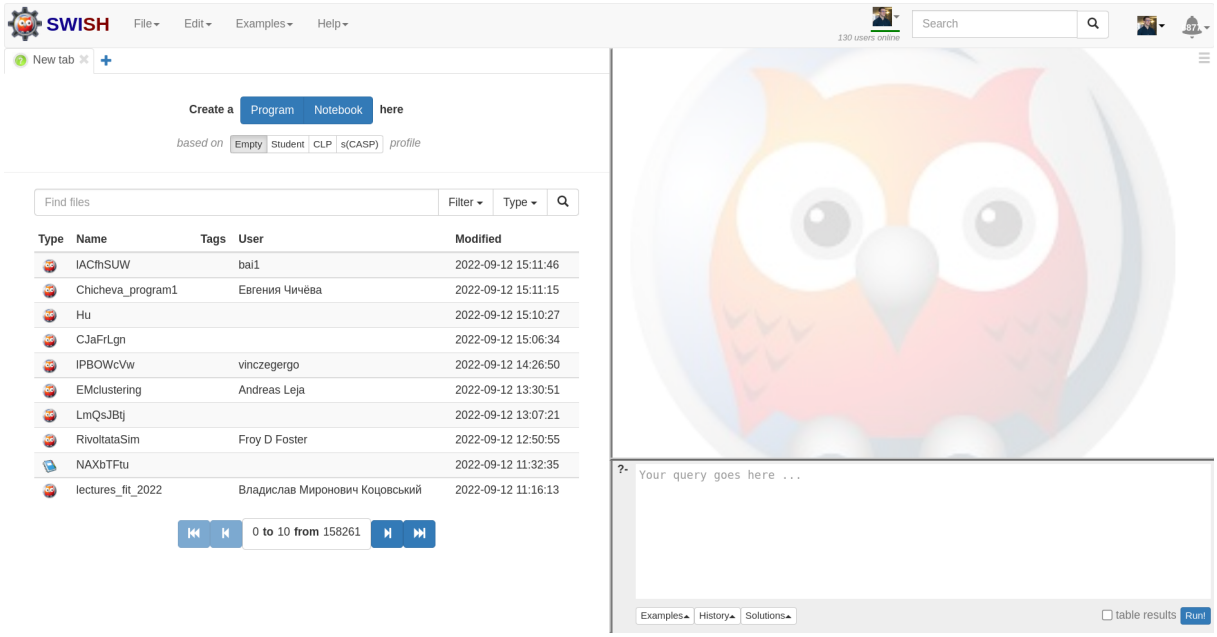


Slika 30: Upiti nad listama kolegija [Autorski rad]

3.4. SWISH

Kao što je već spomenuto, a kako navodi [21], SWISH je kratica od *SWI-Prolog for SHaring*. "Predstavlja *online* okruženje za korištenje SWI-Prolog razvojnim okruženjem, ali i više od toga. Također nam omogućava kolaboraciju s drugim korisnicima što je također velika prednost SWISH-a u poučavanju Prologa" [21]. Uz sve to, SWISH se može kombinirati s raznim drugim tehnologijama jer "pruža razne biblioteke za korištenje drugih tehnologija unutar samog okruženja" [21]. S obzirom da je riječ o pokretanju okruženja na zajedničkom poslužitelju, potrebno je osigurati cijelo okruženje od napada. Iz tog se razloga nameće tzv. *sandbox* okruženje, odnosno okruženje koje ne dopušta pokretanje opasnog dijela koda ili koda za koji se ne može dokazati da je siguran" [21]. U odnosu na klasično SWI-Prolog sučelje, SWISH nam omogućuje veću razinu vizualizacije ulaza i izlaza zahvaljujući C3.js vizualizacijskoj biblioteci. Organizacija i prikaz sučelja se nalazi na slici 31. S lijeve strane prikazan je izbornik u kojem se pretražuju programi na SWISH-u. Odabrani program se prikazuje u tom lijevom okviru. Desno

se nalazi okvir s upitima gdje se šalju upiti. Moguće je vidjeti i nekoliko nedavno korištenih upita te ih ponovno pokrenuti u slučaju da je potrebno. Velika prednost i značajka ovog sučelja je mogućnost kolaboracije s drugim korisnicima i razmjena programa. Na taj način nije potreban servis poput Git Huba. Naravno, potrebno je podesiti prava pristupa drugim korisnicima te odabrati koji dio sadržaja se može dijeliti.



Slika 31: Izgled SWISH sučelja [Autorski rad]

SWISH je, prema navodima [21], stvorio Torbjörn Lager kao "svojedobnu početnu stranicu SWI-Prologa" [21]. Osmišljen je kao univerzalni alat kojeg njegovi korisnici mogu prilagoditi i proširiti da zadovoljava njihov željeni scenarij. Također, "korištenje drugih tehnologija je omogućeno uz korištenje raznih biblioteka" [21] koje su uz to vrlo dobro dokumentirane.

Kao što u R-u postoje ugrađeni setovi podataka, tako u SWISH-u postoje i ugrađeni primjeri koje je moguće koristiti prilikom učenja Prologa. Svaki od tih primjera je podijeljen u kategorije ovisno o tome koji je dio potreban. Tako je moguće koristiti njihov primjer s bazom znanja o zaljubljenim parovima. Kao što je ranije već spomenuto, SWISH postavlja veliku važnost na poučavanje Prologa te se tako i primjeri redaju od jednostavne baze znanja pa kroz neke naprednije koncepte i primjere. Od svih primjera, rad će se najviše bazirati na konceptima strojnog učenja u SWISH-u, a dodat će i sučelje prema R-u. Ostali napredniji primjeri uključuju sudoku, problem N kraljica, čitanje i pisanje, grafički prikaz podataka, ekspertni sustavi i mnoge druge.

4. Podatkovna znanost

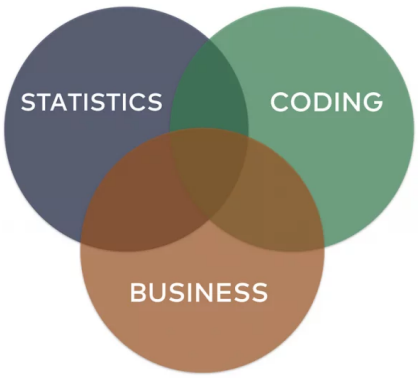
Kao što je ranije rečeno, SWI-Prolog i R se najčešće koriste u području podatkovne znanosti. To područje obuhvaća strojno učenje, analizu podataka, umjetnu inteligenciju, duboko učenje itd. Službeno i znanstvenici [22] tvrde da se "podatkovna znanost koristi kroz znanstvene metode, matematiku, statistiku, specijalizirano programiranje, naprednu analizu i mnoga druga područja" [22] kako bi se lakše objasnile i otkrili važne poslovne informacije i saznanja iz podataka.

Ovaj će rad, uz primjer zajedničkog korištenja R-a i SWISH-a na jednostavnim primjerima obrade podataka i grafičkog prikaza, obraditi i malo šire područje podatkovne znanosti, točnije, strojnog učenja. Ove dvije tehnologije mogu zajedno poslužiti za izradu modela za potrebe raznih slučajeva iz područja podatkovne znanosti, a o kojima će biti više riječi u nastavku.

4.1. Podatkovna znanost danas

Stručnjaci [22] tvrde da organizacije danas prikupljaju i pohranjuju puno podataka koje kasnije koriste kao pokazatelje za svoje poslovanje. Takvi podatci su korisni *stakeholderima* kako bi mogli lakše donositi odluke vezane za predviđanje budućeg poslovanja organizacije. Ipak su podatci bitniji i primarno koriste organizacijama za kvalitetnije poslovanje, potporu odlučivanju, donošenje odluka i slično.

Mester [23] govori da se podatkovna znanost bavi izvlačenjem nečeg korisnog iz velike količine podataka. "Uz pomoć toga moguće je izvući određene trendove i korelacije među podatcima" [23]. Uz takve podatke, organizacije mogu provoditi razna istraživanja i donositi zaključke koji bi im mogli koristiti za nastavak odlučivanju vezanog uz samo poslovanje kompanije. "Podatkovna znanost je spoj statistike, kodiranja i poslovanja" [23] (prikazano na slici 32). Kodiranje se najčešće koristi uz pomoć SQL-a, R-a, Python-a i bash-a, ali i uz druge možda manje popularne jezike. "Statistika sadrži samo značenje podataka, a poslovno znanje pak predstavlja sam koncept u kojem se podatci nalaze." [23] Potrebno je svrstati podatke koji se dobiju u poslovnu domenu kojoj pripadaju kako bi iz njih moglo biti izvučeno nešto korisno.



Slika 32: Područja koja čine podatkovnu znanost [23]

Mester [23] također govori o nekim popularnim konceptima koji se koriste u podatkovnoj znanosti počevši s analizom podataka. "Ona je poprilično konvencionalan način korištenja podataka jer se koriste podatci, provode se analize kako bi se prikazalo gdje se organizacija nalazi sada (poslovno) u odnosu na prošlost". Prediktivna analiza se bavi pitanjem što će se u budućnosti dogoditi bazirano na podatcima koje imamo trenutno, omogućuje predviđanje. Jednostavan primjer ove metodologije je funkcija cilja koja predstavlja funkciju kojoj bi podatci u budućnosti mogli težiti.

4.2. Strojno učenje

Kao što je već spomenuto, a kako navodi Mester [23], strojno učenje je pojam koji se veže uz podatkovnu znanost. Boljim uvidom na podatke koji se prikupljaju, vidljivo je da svaki dio ima neku svoju varijancu. Zato, primjerice, poslovni podatci znaju toliko varirati i odstupati. Za potrebe lakšeg razumijevanja podataka, a što će služiti za predviđanje raznih budućih situacija, koristimo strojno učenje.

Stručnjaci [24] tvrde da je "strojno učenje zapravo grana umjetne inteligencije i podatkovne znanosti koja u svom fokusu ima podatke i algoritme za oponašanje ljudskog načina učenja" [24]. S obzirom na to da se radi o računalnom programu, nerijetko se ovim postupkom i postepeno poboljšava preciznost samog procesa učenja. "Strojno učenje je kroz nekoliko desetljeća omogućilo mnoge inovacije zbog raznih nadogradnji u skladištenju podataka i procesorskoj snazi. Jedan od takvih inovativnih koncepata jest Netflixov sustav za preporuku filmova ili pak samovozeći automobili" [24]. Algoritmi strojnog učenja se uobičajeno kreiraju korištenjem raznih *frameworka* među kojima su i TensorFlow i PyTorch.

Isti stručnjaci [24] također objašnjavaju razliku između strojnog učenja, dubokog učenja i neuronskih mreža. Kao što je moguće i pretpostaviti, sve tri navedene tehnologije su zapravo s područja umjetne inteligencije. Iako su tehnički neuronske mreže područje strojnog učenja, a duboko učenje područje neuronskih mreža, sve to spada pod područje umjetne inteligencije. "Razlika između dubokog učenja i strojnog učenja je u samom algoritmu po kojem računalo (program) uči" [24]. Duboko učenje se tu ističe po tome što "za svoje učenje (za razliku od strojnog učenja) može koristiti i nestrukturirane podatke u svom sirovom obliku. Strojno učenje, za razliku od dubokog učenja, više ovisi o ljudskom faktoru za potpomognuto učenje" [24]. Ljudi su ti koji pripreme "materijale", odnosno set podataka za učenje i set podataka za testiranje naučenog. Doseći razinu dubokog učenja bi u principu značilo da postoji računalo koje tehnički vidi podatke i razumije što vidi. "Kod neuronskih mreža se radi o mreži čvorova koji pokušavaju oponašati ljudski mozak, odnosno neurone u ljudskom tijelu" [24]. Svaki čvor ili "neuron" je spojen na drugi i uz sebe ima povezanu određenu težinu i prag. Izlaz aktivnog čvora šalje podatke kao ulaz u sljedeću razinu čvorova u mreži. Neuronske mreže i duboko učenje akceleriraju u području računalnog vida, područja obrade prirodnog jezika i prepoznavanja govora.

Eksperti na ovom području [24] navode i učestalo korištene algoritme za strojno učenje koji uključuju: neuronske mreže, linearne regresije, logističke regresije, klastering, stabla odlučivanja i dr. Proces učenja se odvija kroz tri glavna dijela. Prvi dio je sam proces predvi-

đanja i klasifikacije podataka. Bazirano na ulazu koji se ponudi izgrađenom modelu, on bi nam trebao prepoznati neku sličnosti i obrazac u danim podatcima. Drugi je dio baziran na funkciji pogreške koja uspoređuje rezultat predviđanja i uspoređuje ga s primjerima koje model već poznaje od ranije. Na taj način se dobije postotak preciznosti, odnosno koliko je model siguran da se radi o primjeru nečega što je naučio. Na kraju postoji i proces optimizacije modela gdje si model samostalno postavlja težinu na podatke koje obrađuje kako bi postigao traženu razinu preciznosti.

SWISH i R svojim zajedničkim djelovanjem mogu postići strojno učenje korištenjem nekoliko različitih algoritama i metoda. Obje tehnologije su kompatibilne za provođenje navedenih algoritama te mogu međusobno surađivati pokrivajući međusobne slabe točke. Prednost SWISH-a (Prologa) je njegova sigurnost te stoga upiti za koje se ne može tvrditi jesu li istiniti ili nisu, smatraju se lažnima. Na taj način algoritam neće raditi osim ako nema sve moguće *inpute* koje treba uz pretpostavku da je algoritam ispravno napisan i strukturiran u programskom jeziku. R pruža raznoliki izbor prikaza podataka koji su krucijalni za razumijevanje rada samog algoritma i za provjeru ispravnosti rada algoritma. Međudjelovanje ove dvije tehnologije moguće je izvesti na nekoliko različitih načina koji slijede u nastavku.

Model koji će obraditi ovaj rad koristit će EM algoritam za maksimizaciju izlaza te korištenjem *Gaussian Mixture Modela* koji će biti naknadno dodatno objašnjeni. Ovo je šire područje od klasičnog načina zajedničkog korištenja R-a i SWISH-a, ali je bliži stvarnim primjerima i realnijim situacijama.

5. Usporedba korištenja

Kao što je ranije spomenuto, međusobni rad R-a i SWISH-a se može ostvariti na nekoliko različitih načina. Ovaj rad će obraditi tri ponuđena načina među kojima su: korištenje CSV datoteke, korištenjem Rserve paketa (R servera unutar SWISH okruženja) i korištenjem rologa (paketa u R-u). Svaki od ovih načina sadrži svoje prednosti i mane te su neki od njih jednostavniji i intuitivniji za korištenje od drugih.

5.1. Korištenje CSV datoteke

Za potrebe ovog primjera će se kreirati dvije CSV datoteke u okruženju RStudija će se *exportati*. Prikazat će se primjer upisa treće godine na Fakultetu organizacije i informatike na način da se popišu svi kolegiji koji se upisuju u petom semestru te sve koji se upisuju u šestom semestru. Kolegiji će biti zapisani u zaseban podatkovni okvir kako bi se mogli ispisati u jednu CSV datoteku. Druga će CSV datoteka biti oblikovana na način da sadrži popis studenata koje je potrebno upisivati na treću godinu studija. Slika 33 prikazuje *chunk* R koda koji izvršavanjem izvozi podatke u obliku CSV datoteka. Koristit će se R funkcija write.CSV() koja kao argumente prima vektor koji je potrebno izvesti te naziv izlazne CSV datoteke. Datoteka s kolegijima će se zvati "kolegiji.CSV", a ona koja sadrži popis studenata će se nazvati "studenti.CSV".

```
petiskolegiji <- c("Baze podataka 2", "Projektiranje informacjiskih sustava", "Računovodstvo", "Upravljanje
informatičkim uslugama", "Algoritmi", "Sustavi temeljeni na znanju", "Engleski jezik za menadžere",
"Logistika", "Mreže računala 2", "Multimedijski sustavi", "Njemački jezik za menadžere", "Odabrane teme iz
biometrije", "Otkrivanje znanja u podacima", "Poduzetništvo", "Računalno i informatičko pravo", "Upravljanje
odnosima s klijentima", "Uvod u teoriju informacija")

sestiskolegiji <- c("Operacijska istraživanja 1", "Programsko inženjestvo", "Web dizajn i programiranje",
"Završni rad", "Kontroling", "Napdredne formalne metode", "Obrambeni informacijski sustavi", "Poslovno
odlučivanje", "Poslovno planiranje i projekti", "Vođenje projekata")

studenti <- c("Andreas Leja", "Filip Balder", "Ana Horvat", "Mia Maran")

okvirpeti=data.frame(kolegij=petiskolegiji, semestar=5)
okvirsesti=data.frame(kolegij=sestiskolegiji, semestar=6)
kolegiji <- data.frame(rbind(okvirpeti, okvirsesti))

write.csv(kolegiji, "kolegiji.csv")
write.csv(studenti, "studenti.csv")
```

Slika 33: Izvoz vektora iz RStudija [Autorski rad]

CSV datoteke koje se izvezu iz RStudio okruženja potrebno je prvo prenijeti na neku platformu s koje ih se može jednostavno dohvatiti. Za potrebe ovog primjera koristiti će se Git Hub. Nakon prenošenja datoteka odabire se opcija *raw* kako bi se dobio sirovi prikaz prenesene CSV datoteke. Primjer takvog prikaza nalazi se na slici 34. Svaki kolegij ima svoj id koji je automatski generiran i automatski se inkrementira za jedan po svakom zapisu. S obzirom da kod studenata nije detaljnije definiran podatkovni okvir, atribut tih podatka se zove "X", dok u primjeru s kolegijima atributi imaju smislene nazive. Mogli su se u RStudiju urediti podatkovni okviri datoteke studenata da piše i godina studija na kojoj se student nalazi, međutim primjer služi samo za upisivanje na treću godinu te se polazi od pretpostavke da su studenti na listi uistinu na trećoj godini studija.

```
"","kolegij","semestar"
"1","Baze podataka 2",5
"2","Projektiranje informacjiskih sustava",5
"3","Računovodstvo",5
"4","Upravljanje informatičkim uslugama",5
"5","Algoritmi",5
"6","Sustavi temeljeni na znanju",5
"7","Engleski jezik za menadžere",5
"8","Logistika",5
"9","Mreže računala 2",5
"10","Multimedijski sustavi",5
"11","Njemački jezik za menadžere",5
"12","Odabrane teme iz biometrije",5
"13","Otkrivanje znanja u podacima",5
"14","Poduzetništvo",5
"15","Računalno i informatičko pravo",5
"16","Upravljanje odnosima s klijentima",5
"17","Uvod u teoriju informacija",5
"18","Operacijska istraživanja 1",6
"19","Programsko inženjestvo",6
"20","Web dizajn i programiranje",6
"21","Završni rad",6
"22","Kontroling",6
"23","Napdredne formalne metode",6
"24","Obrambeni informacijski sustavi",6
"25","Poslovno odlučivanje",6
"26","Poslovno planiranje i projekti",6
"27","Vođenje projekata",6
```

Slika 34: Sirovi prikaz CSV datoteke na Githubu [Autorski rad]

Datoteke je potrebno uvesti u SWISH korištenjem predikata data_source/2. U zagrade ovog predikata se upisuje prvo naziv atoma koji će sadržavati željene podatke koji se uvoze, a zatim i dodatne specifikacije izvora podataka. Koristit će se specifikator CSV da se naglasi činjenica da se radi o CSV datoteci. U zagradu se upisuje poveznica do sirovog pregleda željene datoteke te se postavljaju prazne uglate zagrade jer nisu potrebne dodatne modifikacije kojima se dodatno kontroliraju pretvorbe podataka. Tako će se koristiti imati dva izvora: jedan za kolegije i drugi za studente. Slika 35 prikazuje dio Prolog koda koji je zadužen za konverziju podataka iz CSV datoteke u SWISH razvojno okruženje. Također, ta ista slika prikazuje i dva pravila koja su izvedena korištenjem podataka iz tih datoteka. Definirani su svi studenti kao studenti treće godine te su za kolegije definirani naziv i semestar u kojem se izvode. Također, moguće je bilo napisati "ljetni" i "zimski" semestar s obzirom na to da studenti mogu upisivati sve izborne kolegije s ljetnog i zimskog semestra respektabilno.

UsingRinSWISHcsv

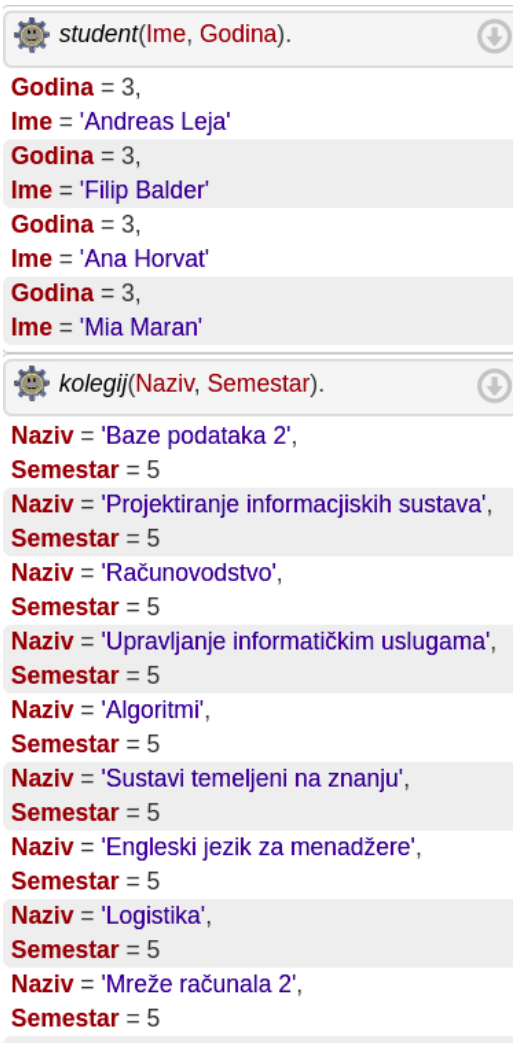
```
:- data_source(kolegiji,
               csv('https://raw.githubusercontent.com/racciel/finalwork/main/kolegiji.csv',[])).
:- data_source(popisstudenata,
               csv('https://raw.githubusercontent.com/racciel/finalwork/main/studenti.csv',[])).

student(Ime, 3) :- popisstudenata{x:Ime}.
kolegij(Naziv, Semestar) :- kolegiji{kolegij:Naziv, semestar:Semestar}.
```

Slika 35: Baza znanja primjera sa CSV datotekama [Autorski rad]

Sada je nad ovom bazom znanja moguće provoditi i nekoliko upita. Primjerice, moguće je tražiti da se ispišu svi studenti u bazi znanja te njihove pripadajuće godine. Doduše, *hardco-*

dirana je činjenica da su svi studenti na trećoj godini tako da će odgovor na godinu za svakog studenta biti taj da je student trenutno na trećoj godini. Također, moguće je ispisati i sve kolegije koji se nalaze u bazi znanja, kao i kolegije po određenim semestrima. Upis kolegija i dodavanje novih studenata kao i dodavanje obveznih i izbornih kolegija moguće je obaviti po primjeru koji je već ranije bio prikazan. Slika 36 sadrži opisane upite nad bazom znanja.

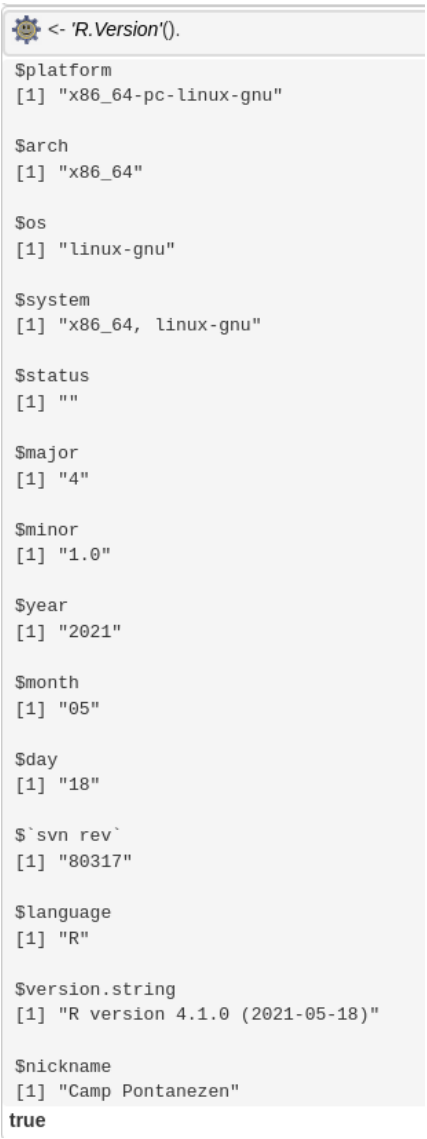


Slika 36: Upiti nad bazom znanja nakon uvođenja CSV datoteke [Autorski rad]

5.2. Korištenje paketa Rserve

Kreatori [21] navode da je u SWISH-u moguće koristiti paket pod imenom rsrve_client. Radi se o sučelju prema Rserve-u koji stučnjaci [25] navode kao TCP/IP server koji omogućuje raznim drugim programima korištenje sadržaja i mogućnosti R-a. Ova funkcija ne zahtjeva nikakvo pozivanje paketa, nego je već ugrađena u sam SWISH. Dostupnost, verziju i ostale podatke o statusu servera moguće je pogledati upitom kao na slici 37. U slučaju da odgovor na ovaj upit prođe, server radi i moguće je slati R upite bez problema i server će reagirati i slati odgovore na njih (ukoliko su valjani).

R server reagira na upite koji mu se šalju te je moguće početi s obradom podataka.



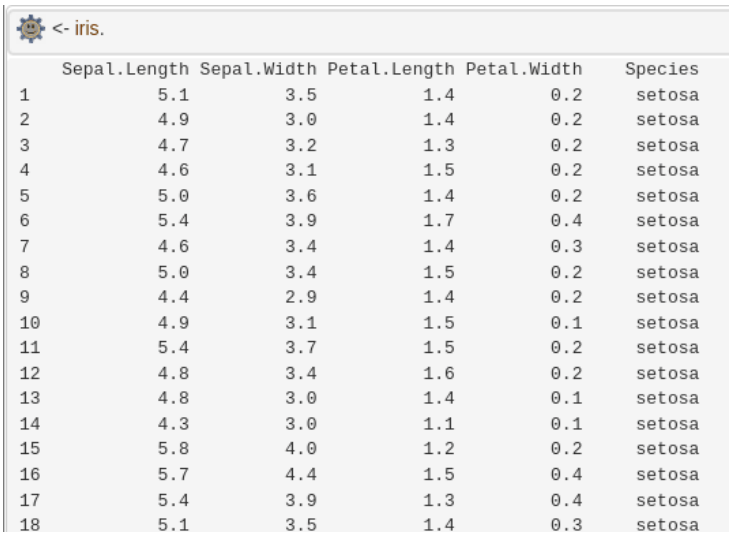
Slika 37: Provjera statusa R servera kroz SWISH [Autorski rad]

Rserve tako omogućuje da se u konzolu upisuje sve što bi se inaće pisalo u R konzolu. Tako je moguće dodijeliti vrijednost varijabli koristeći strelicu pridruživanja kao u R-u (<-) ili pak poslati upit bez varijabli te će željeni sadržaj biti ispisan kao odgovor. Varijabla kojoj se dodjeljuje vrijednost može biti ili varijabla u Prologu ili u R-u.

Također, navode stručnjaci [26] da je moguće izvesti i tzv. "Quasi" notaciju. Takva notacija omogućava ugradnju (dugačkih) stringova korištenjem sintakse iz nekog vanjskog jezika (koji nije Prolog) u Prolog tekst. Uz pomoć te notacije je također moguće koristiti dugačke stringove u Prologu. Obje notacije su u redu i obje se jednako koriste.

Za primjer korištenja rserve paketa, moguće je koristiti prethodno korišteni primjer sa studentima i upisima kolegija, međutim, taj podatkovni set ima vrlo mali broj podataka te će iz tog razloga biti prikazan korištenjem prethodno poznatog podatkovnog seta iris. Taj set sadrži 150 mjerenja cvijeta te je jednostavnije na temelju tog seta prikazati mogućnosti R servera u praksi.

Koristeći prethodno opisan set podataka iris bit će prikazan primjer obrade podataka korištenjem rserve paketa. Jednostavan upit poput "<- iris", kao izlaz daje sve podatke koji se nalaze u setu podataka iris. Za prikaz tih podataka moguće je samo upisati <- iris kao upit i ispisat će se svi podatci iz seta podataka iris kao što je prikazano na slici 38.



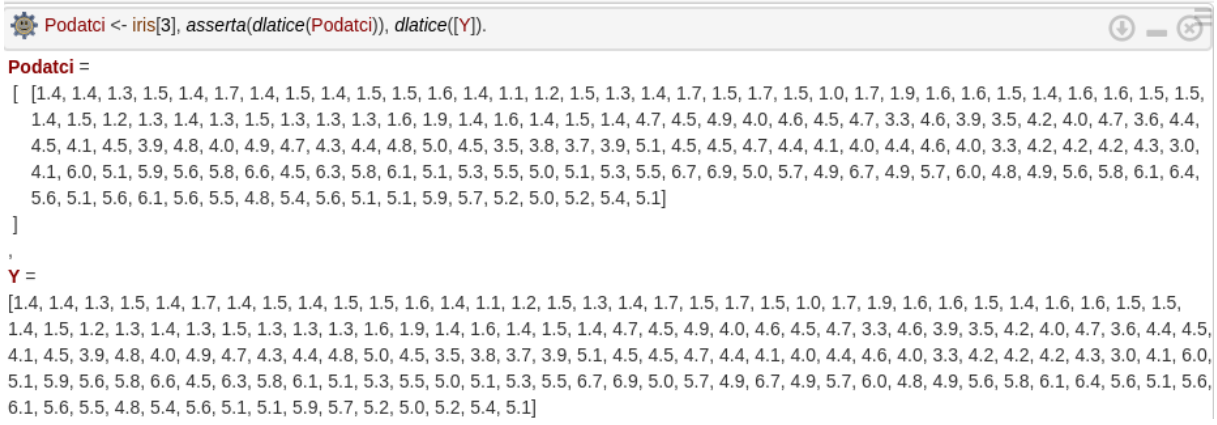
<- iris.

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa
11	5.4	3.7	1.5	0.2	setosa
12	4.8	3.4	1.6	0.2	setosa
13	4.8	3.0	1.4	0.1	setosa
14	4.3	3.0	1.1	0.1	setosa
15	5.8	4.0	1.2	0.2	setosa
16	5.7	4.4	1.5	0.4	setosa
17	5.4	3.9	1.3	0.4	setosa
18	5.1	3.5	1.4	0.3	setosa

Slika 38: Ispis podatkovnog seta iris unutar SWISH-a [Autorski rad]

Najčešće neće biti potreban cijeli podatkovni set iris za promatranje, primjerice, duljine latice kod svih jedinki cvijeta iris. Ovdje će se koristiti činjenica da je ranije poznato koji je stupac u našem podatkovnom okviru duljina latica. Na slici 36, moguće je vidjeti da se radi o trećem stupcu te će se iz tog razloga u uglate zagrade iza naziva podatkovnog seta upisati 3. Te podatke je sada moguće pohraniti u Prolog varijablu. Upisivanjem naziva varijable velikim početnim slovom, dobije se Prolog varijabla s podatcima iz trećeg stupca podatkovnog seta iris. Te iste podatke moguće je pohraniti u bazu znanja i koristiti kasnije. Takva situacija prikazana je na slici 39.

Evidentno je (na slici 39) da se potrebni podatci nalaze u vektoru koji je jedini element vektora prikazanih podataka. U tom slučaju se varijabla Y postavlja u uglate zagrade. Na taj način se dobije jedan element u kojem se nalaze željeni podatci odvojeni zarezom. Međutim, ovoga puta se radi o Prolog varijabli

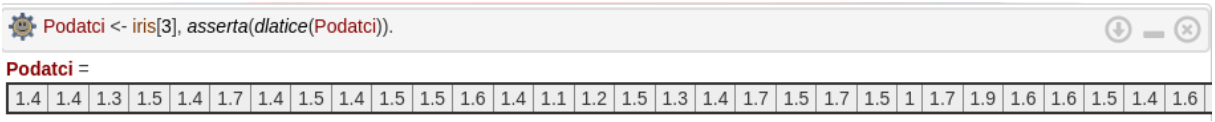


Podatci <- iris[3], *asserta*(*dlatice*(Podatci)), *dlatice*([Y]).

Podatci =
[[1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.5, 1.5, 1.6, 1.4, 1.1, 1.2, 1.5, 1.3, 1.4, 1.7, 1.5, 1.7, 1.5, 1.0, 1.7, 1.9, 1.6, 1.6, 1.5, 1.4, 1.6, 1.6, 1.5, 1.5, 1.4, 1.5, 1.2, 1.3, 1.4, 1.3, 1.5, 1.3, 1.3, 1.3, 1.6, 1.9, 1.4, 1.6, 1.4, 1.5, 1.4, 4.7, 4.5, 4.9, 4.0, 4.6, 4.5, 4.7, 3.3, 4.6, 3.9, 3.5, 4.2, 4.0, 4.7, 3.6, 4.4, 4.5, 4.1, 4.5, 3.9, 4.8, 4.0, 4.9, 4.7, 4.3, 4.4, 4.8, 5.0, 4.5, 3.5, 3.8, 3.7, 3.9, 5.1, 4.5, 4.5, 4.7, 4.4, 4.1, 4.0, 4.4, 4.6, 4.0, 3.3, 4.2, 4.2, 4.2, 4.3, 3.0, 4.1, 6.0, 5.1, 5.9, 5.6, 5.8, 6.6, 4.5, 6.3, 5.8, 6.1, 5.1, 5.3, 5.5, 5.0, 5.1, 5.3, 5.5, 6.7, 6.9, 5.0, 5.7, 4.9, 6.7, 4.9, 5.7, 6.0, 4.8, 4.9, 5.6, 5.8, 6.1, 6.4, 5.6, 5.1, 5.6, 6.1, 5.6, 5.5, 4.8, 5.4, 5.6, 5.1, 5.1, 5.9, 5.7, 5.2, 5.0, 5.2, 5.4, 5.1]
]
,
Y =
[1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.5, 1.5, 1.6, 1.4, 1.1, 1.2, 1.5, 1.3, 1.4, 1.7, 1.5, 1.7, 1.5, 1.0, 1.7, 1.9, 1.6, 1.6, 1.5, 1.4, 1.6, 1.6, 1.5, 1.5, 1.4, 1.5, 1.2, 1.3, 1.4, 1.3, 1.5, 1.3, 1.3, 1.3, 1.6, 1.9, 1.4, 1.6, 1.4, 1.5, 1.4, 4.7, 4.5, 4.9, 4.0, 4.6, 4.5, 4.7, 3.3, 4.6, 3.9, 3.5, 4.2, 4.0, 4.7, 3.6, 4.4, 4.5, 4.1, 4.5, 3.9, 4.8, 4.0, 4.9, 4.7, 4.3, 4.4, 4.8, 5.0, 4.5, 3.5, 3.8, 3.7, 3.9, 5.1, 4.5, 4.5, 4.7, 4.4, 4.1, 4.0, 4.4, 4.6, 4.0, 3.3, 4.2, 4.2, 4.2, 4.3, 3.0, 4.1, 6.0, 5.1, 5.9, 5.6, 5.8, 6.6, 4.5, 6.3, 5.8, 6.1, 5.1, 5.3, 5.5, 5.0, 5.1, 5.3, 5.5, 6.7, 6.9, 5.0, 5.7, 4.9, 6.7, 4.9, 5.7, 6.0, 4.8, 4.9, 5.6, 5.8, 6.1, 6.4, 5.6, 5.1, 5.6, 6.1, 5.6, 5.5, 4.8, 5.4, 5.6, 5.1, 5.1, 5.9, 5.7, 5.2, 5.0, 5.2, 5.4, 5.1]

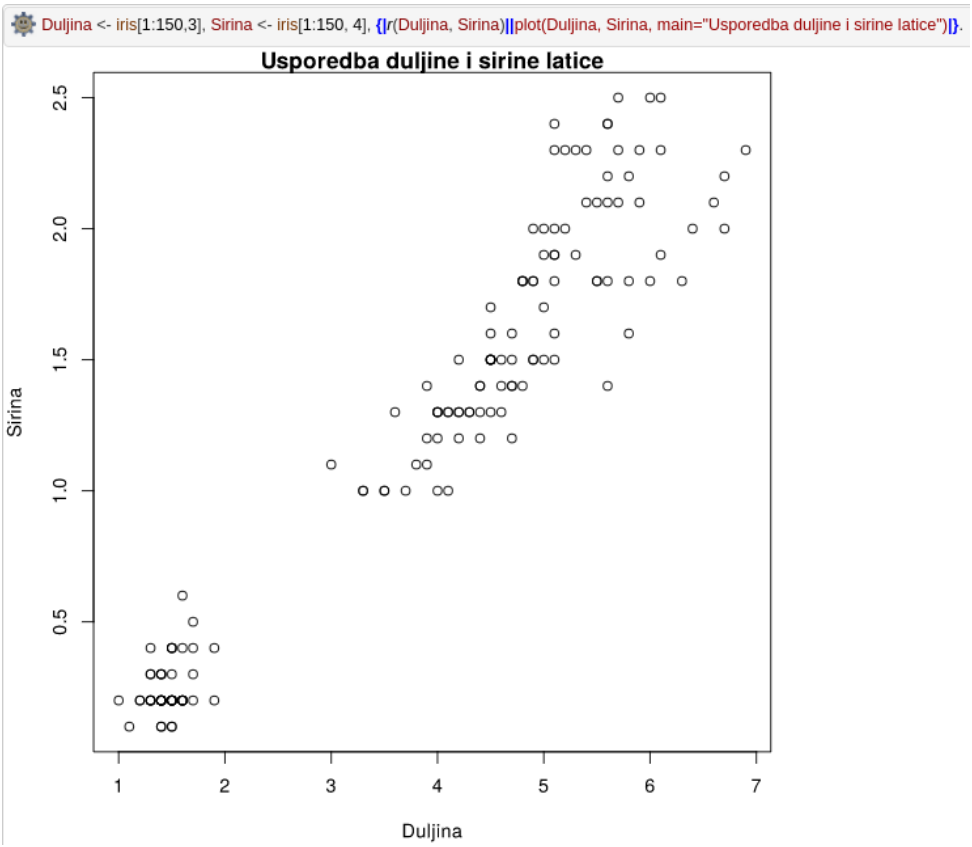
Slika 39: Duljina latica u SWISH-u [Autorski rad]

U slučaju da je podatke potrebno prikazati u obliku tablice, koristit će se biblioteka use_rendering(table). Na taj način se dobije lijepi prikaz tablice s podatcima kao na slici 40.



Slika 40: Tablica u SWISH-u [Autorski rad]

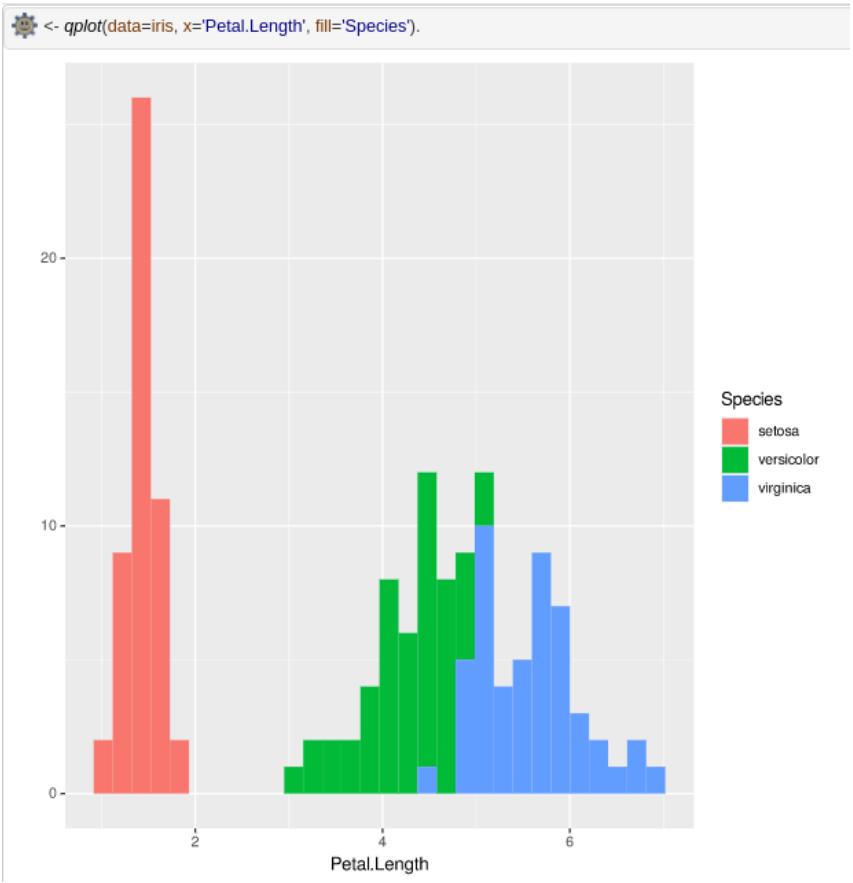
Na sličan način moguće je i prikazati podatke koji se obrađuju u grafičkom obliku. Kao i u R-u, postoji mogućnost prikaza podataka u tzv. plot dijagramu koji se jednostavno može kreirati korištenjem R-ove funkciju plot(). S obzirom da je program u SWISH razvojnom okruženju, moguće je koristiti i varijable iz Prologa za stvaranje plot dijagrama. Tako će se za primjer usporediti duljina i širina latica cvijeta iris. Koristit će se dvije Prolog varijable u ovom slučaju: jedna za duljinu latice, a druga za širinu latice. Koristit će se Quasi notacija za ovaj primjer koja se oblikuje na način da se u vitičaste zagrade kreiraju dvije "kutije" koristeći znak "|", u prvu "kutiju" se upisuje "r" te u zagrade se upisuju imena željenih Prolog varijabli koje je u tom izrazu potrebno koristiti kao R variajble. Primjer koristi varijable "Duljina" i "širina". Zatim se u drugu "kutiju" zapisuje naziv željee R funkcije te gdje se točno prosljeđuju varijable iz prve "kutije" . Slika 41 prikazuje rezultat ovog izraza.



Slika 41: Usporedba duljine i širine latice cvijeta iris[Autorski rad]

Moguće je također kao i u R-u kreirati ggplot2 ili drugim riječima "Quick plot" dijagram. Za njega je samo potrebno uključiti R paket u aktivni SWISH program. To se može postići na

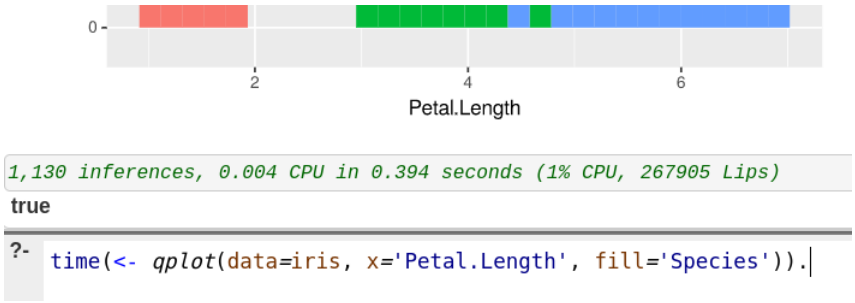
način da se na početku samog programa (cijele baze znanja) napiše ":- <- library("ggplot2").". Moguće je primijeniti kako je ta biblioteka korištena i u R-u na primjeru prikaza takvog dijagram te strelica sugerira da se vrši upit na R server i da se na njemu traži R biblioteka, a ne Prolog biblioteka. Prikazat će se, dakle, odnos duljine latica u odnosu na vrstu cvijeta na ggplot2 dijagramu. Slika 40 prikazuje finalni rezultat.



Slika 42: Duljina latice u odnosu na vrstu cvijeta u SWISH-u [Autorski rad]

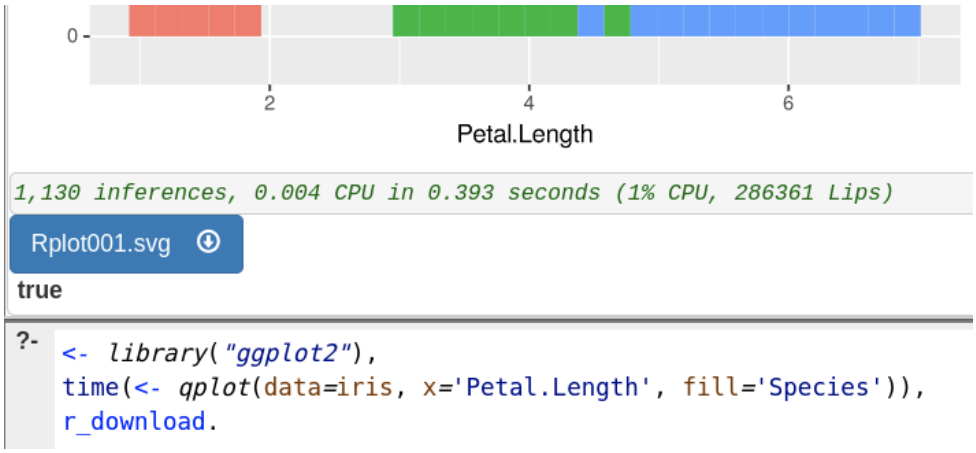
Za potrebe ovog rada poslužiti će jedan predikat koji se zove time/1. Predikat funkcionira na način da izvrši cilj u zagradi i vrati svom korisniku statistiku rada. Ta statistika se zapravo "izvlači pozivanjem jednog drugog predikata, thread_statistics/3, a koji povlači predikat statistics/2 za aktivnu dretvu" [21]. Statistics/2 sadrži puno argumenata, ali time/1 od svega toga vraća samo najbitnije: vrijeme potrebno za izvođenje cilja, iskorištenost procesora tijekom izvođenja te broj zaključivanja. Uz pomoć predikata time/1 moguće je izmjeriti koliko je vremena i resursa potrebno za izvođenje zadatka. Dodat će se prethodni primjer unutar zagrada predikata time/1 i izmjeriti gore navedene dimenzije. Rezultati su prikazani na slici 43. Na taj način se obavlja uloga monitoringa unutar samog programa.

Uz sve to, postoji mogućnost preuzimanja kreiranog grafa. Za to se koristi ugrađeni predikat r_download/0 i r_download/1. U prvom slučaju se dobije gumb za preuzimanje SVG datoteke (vektorske slike) kreiranog grafa. Za tu je situaciju jednostavno ispod grafa potrebno upisati r_download i pokrenuti program. Slika 44 prikazuje graf i gumb ispod njega. Također, graf je moguće postaviti i u PDF datoteku provođenjem iste komande, ali potrebno je ranije dodati upit u kojem se navodi da izlazna datoteka mora biti PDF. Taj primjer vidljiv je na slici

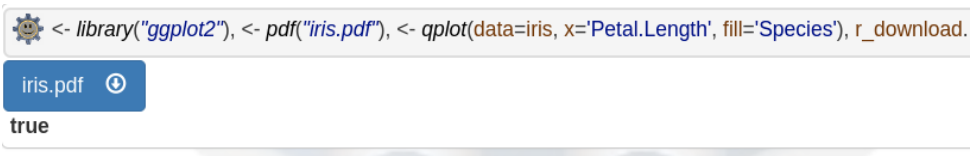


Slika 43: Mjerenje iskorištenosti resursa i vremena u SWISH-u [Autorski rad]

45. Zanimljivost kod korištenja r_download/0 jest ta da on pozove graphics.off() funkciju prije nego što pokuša preuzeti željenu datoteku. Zato na slici 46 ne vidimo grafički prikaz grafa. Na slici 44 ga vidimo jer se radi o vektorskoj slici i kao takva može ostati prikazana kao izlaz našeg upita.



Slika 44: Preuzimanje slike grafa u SVG formatu [Autorski rad]



Slika 45: Preuzimanje slike grafa u PDF formatu [Autorski rad]

Na sličan se način mogu preuzeti svi podatci iz nekog podatkovnog seta. Za to se koristi ista R komandu write.CSV i prilaže koje je točno podatke potrebno upisati i kako će se zvati izlazna datoteka. Na ovom je primjeru korišten predikat r_download/1 gdje se kao cilj zadaje naziv datoteke koja se izgradila iznad. Primjer preuzimanja CSV datoteke se nalazi na slici 46. Ovo je vrlo korisna mogućnost cijelog razvojnog okruženja jer je podatke koji se obrađuju i programi koji se nerijetko potrebno nekako korisno i praktično prikazati i izvan razvojnog okruženja. Za te potrebe se najčešće koristi funkcija preuzimanja da obrada podataka ne bude uzaludna ili da ju je moguće primjerice negdje proslijediti.

Slika 46: Preuzimanje podataka podatkovnog seta u CSV formatu [Autorski rad]

5.3. Korištenje paketa rolog

Interakciju između SWISH-a (Prologa) i R-a moguće je ostvariti i korištenjem R paketa zvanog rolog. Treba napomenuti kako ovaj paket nije podržan na verziji R-a koja je starija od 4.2.0 te samim time neće raditi na starijim verzijama. postupak započinje instalacijom paketa "rolog" te korištenjem tog paketa unutar samog R programa.

Ovaj paket je kreirao Matthias Gondan te dokumentirao do 09. svibnja 2022. godine, a prvu radnu inačicu je objavio na SWI-Prolog forumu za diskusije 12. prosinca 2021. godine. Autori [27] tvrde da je rolog paket koji ugrađuje SWI-Prolog u R kako bi mu se mogli slati upiti slično kao u Prolog konzoli. Jedan od najčešćih načina korištenja ovog paketa je korištenje vanjske baze znanja kroz funkciju consult(). Ta funkcija omogućuje korištenje podataka iz vanjske datoteke lokalno ili na nekom *cloud* servisu. Funkciji je potrebno proslijediti string koji sadrži putanju željene datoteke na lokalnom računalu. Uvijek je moguće koristiti ovu funkciju bez argumenata kako bi se koristila bazu znanja koja služi kao *defaultna* biblioteka pod nazivom "ancestors". Ta baza znanja je ekvivalentna tzv. "Hello world" programu u drugim programskim jezicima. Riječ je o skupu činjenica i pravila koji prikazuju obiteljsko stablo. Za potebe ovog primjera, poslužit će i ranije izrađena i malo prilagođena baza znanja, potrebno ju je postaviti u isti radni direktorij kao i aktivni RStudio program. Slika 47 prikazuje bazu znanja koja se koristi. Baza znanja služi za evidenciju studenata na FOI-ju.

Iako dokumentacija u samom okruženju RStudija prikazuje jedan način "uklapanja" Prolog datoteke u program, taj se primjer pokazao prikladnim samo za primjer s *defaultnom* bazom znanja, dok autori [27] nude sintaksu koja funkcionira, a koja je spomenuta u prethodnom paragrafu. Prolog program sa slike 45 naziva se "zavrsni.pl" te je postavljen u radni direktorij iz kojeg se pokreće R *markdown* datoteka u RStudiju. Instalacija paketa već je prethodno obavljena tako da jedino što je još preostalo jest upotrijebiti library() funkciju kojom se naglašava da se koristi rolog paket. Izvest će se postavljanje upita na učitanu bazu znanja te će se odgovor zapisati kao R varijabla. To je moguće učiniti na više načina od kojih će biti prikazano korištenje funkcije as.prolog() koja služi za prevođenje pojednostavljenog prikaza u kanonski prikaz. Na taj se način daje R-u do znanja da interpretira ono što mu je postavljeno u zagradama kao Prolog upit. Unutar programa će se koristiti varijabla Q kao željeni upit koji je potrebno interpretirati kao Prolog upit. Ovaj korak nije neophodan, ali će kod biti pregledniji i lakše je objasniti o čemu se radi. Željene Prolog varijable koje se šalju kroz R potrebno je označiti na način da se upiše točka (.) i napiše naziv željene varijable koju je potrebno koristiti u upitu. Primjerice, potrebno je u programu obraditi podatke o svim studentima koji se nalaze u bazi znanja. Ranije je rečeno

```
smjer(is).
smjer(ips).
smjer(ps).

nositelj(webdip, kermek).
nositelj(nfm, schatten).
nositelj(oil, zugecb).
nositelj(po, begicevicredep).
nositelj(ppp, detelj).
nositelj(pi, stapic).
nositelj(vp, zajdelahrustek).

obvezan(webdip, is).
obvezan(oil, is).
obvezan(pi, is).
obvezan(oil, ps).
obvezan(po, ps).
obvezan(ppp, ps).

izborni(webdip, ps).
izborni(pi, ps).
izborni(po, is).
izborni(ppp, is).
izborni(vp, is).
izborni(vp, ps).
izborni(nfm, is).
izborni(nfm, ps).

student(andreas, is, 3).
student(ana, is, 3).
student(mia, ps, 3).
student(ivan, ips, 1).
student(goran, ips, 2).
student(domagoj, ips, 2).
student(david, ps, 2).
student(filip, is, 3).

kolegij(X):- nositelj(X, _).
```

Slika 47: Baza znanja korištena u rolog primjeru [Autorski rad]

kako je Prolog poznat po svojoj sigurnosti zbog primjene pristupa zatvorenog svijeta. Iz tog razloga poznato je da su podatci koji se dobiju kao odgovor uistinu istiniti.

Postoji više načina na koje je moguće prevesti odabrani tekst u Prolog upit. Korištenjem funkcije once() moguće je poslati upit od kojeg se očekuje i prihvaća samo jedan odgovor. Unutar zagrada koristi se funkcija call() u koju se upisuje string, a koji sadrži željenu klauzulu koja se šalje kao upit te expression() unutar čijih zagrada se upisuje varijabla koja se koristi da se dobije odgovor. Uz funkciju once() postoji i funkcija findall() koja funkcionira na principu pretrage svih odgovora koji zadovoljavaju uneseni upit. Ta će se funkciju češće koristiti jer vraća sve moguće odgovore koji zadovoljavaju poslani upit.

Također, postoji i jedna više "manualna" metoda slanja upita prilikom koje je moguće kontrolirati koliko odgovora je potrebno vratiti. Za to će se koristiti funkcija query() u čije zagrade se upisuje isti sadržaj kao i u prethodna dva primjera. Taj query trenutno ne radi ništa, ali ga je moguće poslati za dobivanje jednog odgovor ili pak očistiti query. U slučaju da se odluči zadati novi upit, stari upit će se automatski obrisati kao je bila upotrijebljena funkcija clear(). Funkcija preko koje se šalje upit od query() funkcije se zove submit() te on kao i clear() ne traži argumente. Potrebno je kao primjer, koristeći funkciju findall, pronaći sve smjerove na fakultetu. Slika 48 prikazuje kako se to postiže i što će se u toj situaciji dogoditi.

Dobivene odgovore je sada moguće spremiti u neku R varijablu kako bi bile od koristi

```
once(call("smjer", expression(X)))

findall(call("smjer", expression(X)))

```

```
[[1]]
[[1]]$X
is


[[2]]
[[2]]$X
ips


[[3]]
[[3]]$X
ps
```

Slika 48: Korištenje funckija once() i findall() [Autorski rad]

kasnije u obradi. Također, bit će prikazan i primjer korištenjem query() funkcije. Kao što je već ranije spomenuto, ova funkcija se koristi u kombinaciji s funkcijama submit() i clear() koje ne traže argumente za pozivanje te je uz pomoć njih moguće kontrolirati koliko se odgovora dobiva iz upita, a može se i obrisati sam upit ako više nije potreban. Za primjer korištenja ovih funkcija zatražit će se primjer kolegija kako bi se zapisao te se zato postavlja Prolog upit u obliku kolegij(Y). Kolegij postoji ako postoji nositelj kolegija što je vidljivo i iz pravila na slici 47. Slika 49 prikazuje gore navedeni primjer. Može se primijetiti da je dva puta pozvana funkcija submit() te se zato i dobivaju samo dva odgovora. Važno je napomenuti da $Y znači istu stvar kao što u Prolog-u piše "Y=".

```
query(call("kolegij", expression(Y)))
submit()
submit()
clear()
```

```
[1] TRUE
attr(,"query")
[1] "kolegij(Y)"
$Y
webdip

$Y
nfm
```

Slika 49: Dohvaćanje dva kolegija preko funkcije query() [Autorski rad]

Znanje o paketu moguće je upotrijebiti da se obavi jednostavna obrada podataka u R-u o kojoj je bilo riječi ranije. Bit će potrebni podatci o studentima od ranije. Njih je moguće jednostavno pronaći u bazi znanja kao činjenice. Koristit će se student/3 te se tako posebno mogu dohvatiti ime studenta, smjer studiranja te godina studija. Radi preglednosti poslužit će ranije spomenuta funkcija quote kako bi kod bio pregledniji. Također, u ovom će se primjeru koristiti još jedna funkcija, ali ovoga se puta radi o R-ovoj funkciji koja se zove unlist(), a koja vraća elemente iz liste (izlistava listu). To je vrlo bitno jer odgovor koji se dobije od Prologa poprimi oblik liste u listi, a što je bitno za razumijevanje načina na koji su podatci strukturirani. Za lakšu obradu koristit će se izlistana lista te će se u zasebne varijable pospremiti imena studenata, zatim smjerovi studenata te godine na kojima se nalaze studenti. Na slici 50 moguće je pratiti opisani proces. Ovo za sobom također nosi i jedan problem. Naime, Prolog i R dosta drugačije interpretiraju string kao tip podataka. Samim time ime studenta i smjer nije moguće tako jednostavno ubaciti u podatkovni okvir. S godinom studiranja s druge pak strane nema problema jer se ipak radi o numeričkom zapisu kojeg R i Prolog interpretiraju slično (najčešće R numeričku vrijednost interpretira kao double osim ako se ne priloži L uz broj što označava da se radi o integeru) ili pak isto kao u ovom slučaju.

S obzirom da s godinama nije potrebno ništa dodatno raditi, potrebno je posvetiti se načinu baratanja preostalim podatcima. Jedno od rješenja je da se cijela lista pretvori u jedan veliki string te da ga se sreže po nekom određenom kriteriju. Za to će biti potrebne funkcije toString() te strplit() koje su također R-ove funkcije. Koristeći privremenu varijablu tmp u koju se upisuje lista pretvorena u string. U memoriji je zapisan jedan dugačak string od svih dohvaćenih podataka. Kreirat će se nove varijable koje će sadržavati uređenu listu podataka koji se obrađuju. Funkcija strsplit() koristi se tako da se prvo upiše željeni string koji je potrebno srezati, a zatim se upisuje kriterij po kojem se "reže" po stringu. U ovom slučaju to je podstring zareza i razmaka (,). Postupak se ponavlja za podatke o smjerovima na kojima se nalaze studenti. Zatim, može se kreirati podatkovni okvir od tih podataka. Preostalo je još samo promijeniti imena nad podatkovnim okvirom u nešto smisleno te se zato koristi funkcija setNames() kojoj se prosljeđuje podatkovni okvir tipa podatkovni okvir koji je potrebno urediti te kao drugi argument se šalju željeni nazivi stupaca. Nakon toga, moguće je ispisati podatkovni okvir te će podatci biti vidljivi u tabličnom prikazu. Slika 48 prikazuje cijeli opisani postupak, a slika 51 prikazuje tablični prikaz novokreiranog podatkovnog okvira.

Ove podatke moguće je prikazati i u obliku grafa korištenjem već dobro poznatog paketa ggplot2. Bit će prikazano koliko je studenata upisalo koji smjer, ali od podataka dohvaćenih iz baze znanja. Na ovakav način bilo tko tko ažurira datoteku ažurira i samu bazu znanja te sama analiza može preciznije i brže teći prikazujući relevantnije podatke. To posebice vrijedi za baze znanja u nekom *cloudu*, ali može se slično izvesti i s lokalnim datotekama. Na slici 52 vidi se koliko je studenata upisalo koji smjer i na kojoj su godini ti studenti. Ovakvi podatci mogu služiti za nekakvu vrstu statistike i predviđanja, ali i za povratne informacije samim studijima. Na sličan način na koji su obrađeni podatci općenito o studentima moguće je i prikazati koliko je studenata upisalo koji kolegij. Potrebne su samo neke modifikacije u bazi znanja vezane uz pravila i uz činjenice.

```
11 library("rolog")
12 consult("zavrsni.pl")
13
14 Q <- quote(student(.X, ._, ._))
15 imena <- unlist(findall(as.rolog(Q)))
16 Q <- quote(student(._, .Y, ._))
17 smjerovi <- unlist(findall(as.rolog(Q)))
18 Q <- quote(student(._, ._, .Z))
19 godine <- unlist(findall(as.rolog(Q)))
20
21 tmp <- toString(imena)
22 splittanaimena <- strsplit(tmp, ", ")
23
24 tmp <- toString(smjerovi)
25 splittanismjerovi<- strsplit(tmp, ", ")
26
27 studenti <- data.frame(student = splittanaimena,
28                        smjer=splittanismjerovi, godina_studija=godine)
29
30 studenti <- setNames(studenti, c("student", "smjer" , "godina"))
```

Slika 50: Postupak dohvaćanja i pospremanja podataka [Autorski rad]

```
32 studenti
33 ```
```

student <chr>	smjer <chr>	godina <int>
andreas	is	3
ana	is	3
mia	ps	3
ivan	ips	1
goran	ips	2
domagoj	ips	2
david	ps	2
filip	is	3

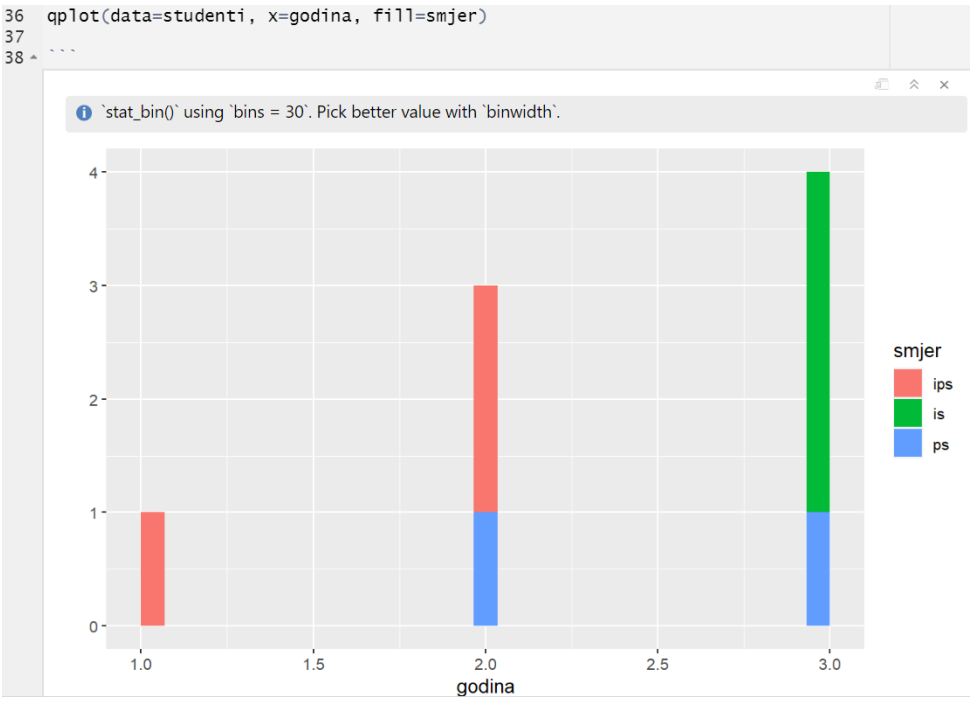
8 rows

Slika 51: Prikaz dohvaćenih podataka u tabličnom prikazu [Autorski rad]

5.4. Osvrt i usporedba

Navedeno je nekoliko načina na se može postići međusobni rad R-a i Prologa. Obrađena je situacija u kojoj se obavlja obrada podataka u R-u, izvoze se podatci u CSV datoteku koja se može uvesti u SWISH i koristiti podatke za izgradnju baze znanja. Taj je pristup koristan u slučaju da se podatci ne mijenjaju često te, najbitnije, da se ne mijenjaju nazivi stupaca u podatkovnim okvirima. Iako se podatci nalaze negdje u *cloudu*, sama Prolog obrada ovisi o njezinom sadržaju te u slučaju da se, primjerice, promijene nazivi nekih stupaca u podatkovnom okviru koji se izvozi, Prolog dio neće raditi. Metoda je vrlo jednostavna za korištenje te zahtjeva vrlo malo predznanja, a sintaksa nije teška za razumjeti samim time što podsjeća na uvoz CSV datoteke u ostalim programskim jezicima.

Glavna prednost korištenja paketa rservea jest ta što nije potrebno ni napuštati SWISH



Slika 52: Brzi plot dijagram broja studenata na smjerovima (rolog) [Autorski rad]

okruženje za obradu podataka u R-u pa ih prebaciti u SWISH. Sintaksa koja se šalje na rserve server je vrlo intuitivna jer svojom sintaksom podsjeća na pridruživanje vrijednosti varijablama u R-u. Mana ovog pristupa jest ta da jako ovisi o statusu rserve servera (koji je većim dijelom aktivan) te verzijom R-a koja se na njemu pokreće. Na slici 37 prikazani su detalji o rserve serveru te je evidentno da se na njemu pokreće (u vrijeme pisanja rada) R verzija 4.1.0, dok je za korištenje rologa bila potrebna novija verzija 4.2.0. Također, moguće je uljepšati prikaz podataka korištenjem biblioteke use_rendering(table) te se time nije potrebno brinuti o duplim zagradama unutar varijable. Ova metoda je također mjerljiva jer možemo koristiti predikat time/1 te na taj način pratiti performanse izvođenja upita.

Korištenje paketa rolog se definitivno najviše razlikuje od ostalih načina korištenja. Potreban je nešto drugačiji način razmišljanja koji ne mora nužno biti mana ovog pristupa. S druge strane, paket je poprilično nov i slabije dokumentiran, a sama službena dokumentacija sadrži nejasan sadržaj. Nije intuitivan kao prethodno navedeni primjeri, ali nakon nekoliko dana vježbanja ne čini veliku razliku u samom načinu izvođenja. Mogućnost iščitavanja podataka direktno iz baze znanja jamči istinitost podataka (ali ne i točnost) koji se koriste u daljnjoj obradi. Na taj je način jednostavno moguće kreirati razne izvještaje, uočavati uzorke, prikazivati i analizirati podatke itd. Ovaj pristup također daje veću razinu kontrole rezultata iz postavljenih upita zbog mogućnosti kontrole izgleda upita i samih odgovora kroz različite funkcije. Jednostavnija je mogućnost pretvorbe Prolog varijabli u R varijable i obratno čak i onda kada nisu u jednostavnom obliku za obradu. U takvim situacijama nastupaju R-ove metode i funkcije za obradu podataka kako bi se na izlazu našli željeni podatci u željenom obliku.

6. EM klasteriranje (praktičan primjer)

Svi primjeri do sada prikazivali su jednostavne programe u kojima je moguće zajedničko korištenje R-a i Prolog-a zajedno unutar SWISH ili RStudio razvojnih okruženja. Međutim, R i Prolog se koriste u mnogim područjima podatkovne znanosti poput analize podataka te raznih vrsta učenja. Jedno takvo učenje bit će prikazano u primjeru zajedničkog rada. Provest će se kroz upotrebu EM (*Expectation Maximization*) algroitma kojeg stručnjaci [28] navode kao jednog od primjena strojnog učenja u praksi. Vrlo se često koristi u slučajevima kada postoji puno relevantnih značajki za učenje, ali set podataka koji računalo koristi da ih nauči je relativno malen. Tako je moguće naučiti računalo da na primjeru značajki koji su mu predočene, predvidi značajke novih jedinki, odnosno u slučajevima kada ne postoje vidljive značajke.

6.1. Teorijska podloga

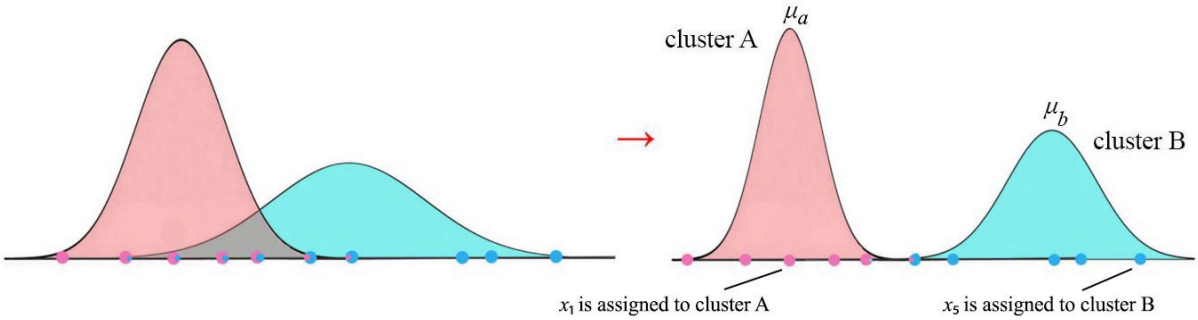
Glavna poanta cijelog algoritma leži u tome da postoje dva koraka koji se nazivaju E korak i M korak. Tijekom koraka E pokušava se uz pomoć promatranih podataka zaključiti, odnosno pretpostaviti vrijednost podataka koji nedostaju. Zatim se prelazi na M korak u kojem se pokušava uz pomoć izlaza iz E koraka prilagoditi parametre za novu iteraciju. Taj postupak se ponavlja sve dok vrijednost ne dosegnu konvergenciju. Na taj način se zapravo traži lokalni maksimum neke funkcije koja se promatra. Konvergencija se postiže u trenutku kada se dođe do tog lokalnom maksimuma. Valja napomenuti da je također vrlo bitan početni položaj odakle algoritam kreće s radom jer on pronalazi prvi lokalni maksimum te ako, primjerice, postoji dalje još neki maksimum, potencijalno veći od prvog, do njega algoritam neće ni doći.

Primjerice, provodi se EM algoritam na funkciji, E i M korake moguće je prikazati na način da se odabere početni položaj odakle se kreće s algoritmom, u toj točki se crta nova funkcija koja se u točki spaja s originalnom funkcijom, a ostale točke su dio funkcije koja prolazi ispod originalno zadane funkcije. Drugim riječima, nova funkcija i originalna funkcija imaju zajedničku vrijednost u točki koja je inicijalno odabrana, a u svim ostalim točkama nova funkcija ima vrijednost manju od originalne funkcije. Na takav način se dobije konvergencija kada se dođe do prvog lokalnog maksimuma. Također, zbog toga se ne dostiže ni do sljedećeg lokalnog maksimuma (ako postoji) jer korak maksimizacije uvijek vuče na onaj prvi od kojeg se kreće.

Algoritam služi za određivanje najveće lokalne vrijednosti, ali rad će ga prikazati u suradnji s tzv. *Gaussian Mixture Modelom*. Takav model stručnjaci [29] opisuju kao model vjerojatnosti koji služi za prikaz podataka koji spadaju u različite podskupove u odnosu na cijeli skup podataka. Koristit će se već spomenuti algoritam u suradnji s ovim modelom kako bi se računali parametri modela s obzirom da je ipak cilj što preciznije pronaći skup podataka koji pripada jednom od određenih klastera.

Kao jednostavan primjer predstavit će se slučaj u kojem se koriste podatci koji su na istom pravcu, ali mogu poprimiti jednu od dvije vrijednosti. Znači da se radi o promatranju samo jedne dimenzije i postoje dvije skupine kojima podatci mogu pripadati. U tom se slučaju koriste dvije različite distribucije, odnosno načina na koje su raspoređeni podatci. Izvede li se

distribucija za svaki set podataka, moguće je predvidjeti za nasumični podatak kojoj skupini pripada uz neku određenu preciznost. Tako računalo ruspijeva naučiti da izvlačenjem nekog podatka može prosuditi radi li se o skupini (klasteru) jedan ili dva (ili koliko ih ima na nekom setu podataka). Polazi se od pretpostavke da se koristi tzv. tehnika *hard clusteringa* jer je poznato da neki podatak pripada jednom od dva klastera, te da ne postoji mogućnost pripadanja određenim postotkom u dva ili više različitih klastera. S druge pak strane postoji tehnika *soft clusteringa* gdje, primjerice, podatak može pripadati 30% u jednom klasteru i 70% u drugom. Slika 53 prikazuje podatke koji spadaju u dvije različite kategorije (klastere). Potrebno je napomenuti kako se već na temelju postotka pripadnosti pojedinačnom klasteru s lijeve slike (uz pomoć algoritma) prilagodi distribucija za *hard clustering* s obzirom da jedan podatak može pripadati potpuno samo jednom od dva klastera. Moguće je primijetiti kako nije poznato kako su podatci zapravo raspoređeni, ali promatrajući njihovu distribuciju, moguće je zaključiti koji bi mogli pripadati kojoj skupini. Najčešće će se teže odrediti pripadnost podatcima koji se nalaze na preklapanju dvije distribucije kao primjerice peti i šesti podatak s lijeve strane (sa slike 53). Postoji mogućnost da se zamijeni pripadnost tim podatcima.



Slika 53: EM u jednoj dimenziji i dva klastera [30]

Situacija s jednom dimenzijom (podatcima na pravcu) je poprilično jednostavna. Međutim, ovaj algoritam se primjenjuje i za višedimenzionalne sustave. Ovdje će se algoritam iskoristiti za podatke kod kojih su dvije dimenzije od visoke važnosti, dok se u stvarnosti koristi i trodimenzionalna primjena ovog algoritma. Cilj kod tog slučaja je odrediti približno centar svakog klastera oko kojeg se nalazi najviše pripadajućih podataka iz tog klastera. Provođenjem E i M koraka (iteracija), polako se formira rješenje i računalo samo zaključuje da se radi o tri skupine i koje bi to skupine mogle biti.

Poznat primjer provođenja ovog algoritma u praksi jest istraživanje vezano uz veličinu i koncentraciju crvenih krvnih zrnaca u pacijenata koji boluju od anemije. Proveo se eksperiment gdje su se koristile dvije skupine: kontrolirana (pacijenti koji ne boluju od anemije) i skupina pacijenata koja boluje od anemije. Postoji gušći set podataka o pacijentima koji nemaju anemiju te postoji više raštrkan (raspršen) set podataka o pacijentima koji boluju od anemije. Na taj način je moguće predvidjeti anomaliju u krvnim zrncima kod pacijenata i testirati ih koristeći saznanja iz eksperimenta. Moramo napomenuti kako program ne razumije koji pacijenti imaju anemiju, a koji nemaju, ali je uspio prepoznati dvije skupine podataka i to je krajnji cilj upotrebe ove tehnike.

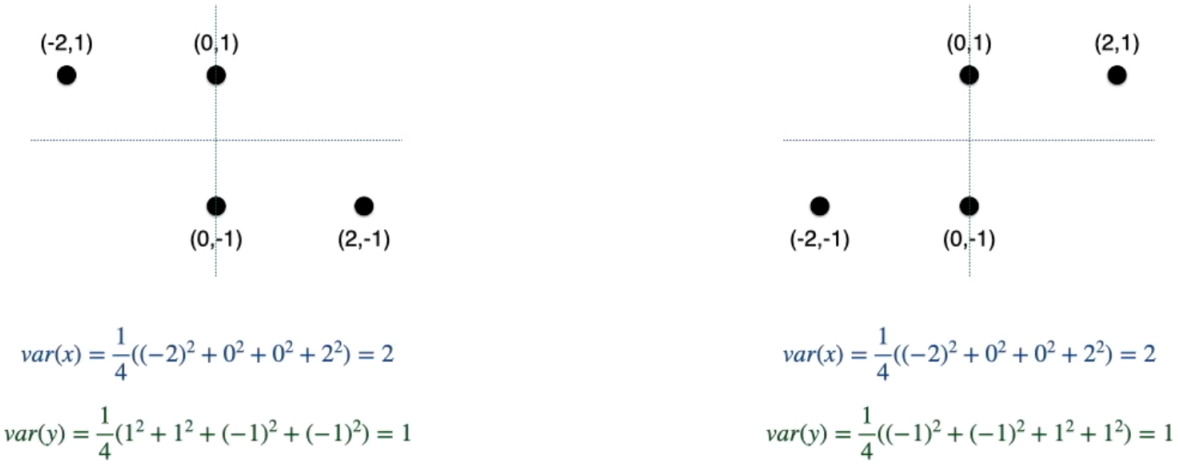
Za izradu jednog takvog modela, poslužiti će poznate činjenice. Poznato je od ranije

da se u jednodimenzionalnoj situaciji koriste neki određeni parametri i kreće se od nekih pretpostavki. Jedna od njih je da se distribucija podataka može opisati normalnom distribucijom $N(\mu, \sigma)$. Laički gledano, ritvikmath [31] tvrdi da μ označava središnju vrijednost, te se σ odnosi na širinu cijele distribucije. Za prijenos svega u dvije dimenzije, potrebno je uvesti i nove varijable. Distribucija se u dvije dimenzije opisuje središtem(μ) koje predstavlja točku oko koje se nalazi najviše podataka iz pripadne klase. Osim toga, potrebno je "odrediti i varijancu po x osi i varijancu po y osi. One označavaju raspršenost podataka u vodoravnom i okomitom pravcu. Nedostaje još samo jedan podatak, a on se naziva kovarijanca koja govori o tome u kojem smjeru su rasprostranjeni podatci. Podatke o kovarijanci te o x i y varijancama se ubacuju u Σ, matricu koja je u ovom slučaju reda veličine 2x2, a u koju se na dijagonalu upisuju prvo x-varijanca, zatim y-varijanca, a ostali se podatci popune s kovarijancama" [32].

Formula kojom se opisuje jednodimenzionalni problem je ustvari Gaussova krivulja koja se definira formulom $f(x) = N(\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$, pri čemu je μ središnja vrijednost, te σ čini tzv. "širinu" zvonkolike distribucije. Uvrste li se izračunate vrijednosti za μ i σ, funkcija koja se dobije opisuje normalnu distribuciju unesenih podataka. Takav sustav moguće je prenijeti i u dvije dimenzije, ali tu je potrebno dodati još neke varijable. Ranije je spomenuta matrica koja se naziva matricom kovarijance. Autor [32] objašnjava da se ona označava grčkim slovom Σ te će to biti nxn matrica prilikom čega je n broj dimenzija modela.

Za dvije dimenzije vrijedi da je Σ matrica 2x2 u kojoj se nalaze x varijanca (na mjestu 1,1), y varijanca (na mjestu 2,2) te vrijednost kovarijance za trenutni model. Uz pretpostavku da postoji set podataka za koji je potrebno odrediti model. Za početak je potrebno izračunati tzv. "centar mase" (navodi autor [32]) te ce se tako dobiti točka koja će reprezentirati sredinu pojedinog klastera. Točku je moguće izračunati na način da se za sve x koordinate svake točke izračuna aritmetička sredina te se isto učini i s y koordinatama svake točke. Koordinate koje se dobiju kao rezultat čine "centar mase" ili središnju vrijednost tog klastera.

Nadalje, za daljnje računanje koristi se cijeli sustav točaka i pomiće se na način da se središte mase nalazi u središtu koordinatnog sustava (0,0). Nakon toga moguće je izračunati x i y varijance. Računaju se na način da se izračuna aritmetička sredina kvadrata x koordinate svake točke te isto učinimo za y varijancu računanjem aritmetičke sredine kvadrata y koordinate svake točke. Ti nam podatci govore o tome koliko je klaster ili set podataka raspršen po x i y osi. Neki setovi podataka imaju veće raspršenje po x osi, neki po y osi, a neki imaju podjednako raspršenje koje također može biti veće ili manje. Međutim, dva modela mogu također imati iste varijance, a podatci mogu biti različito raspršeni. Takav je primjer prikazan na slici 54. Iz tog razloga računa se kovarijanca koja će odrediti "smjer" raspršenosti podataka. Na istoj slici (slici 54) moguće je primijetiti kako će se lijevi set podataka protezati na drugačiji način nego desni, a oba seta imaju iste varijance. Za izračun kovarijanci ovih setova podataka, računa se produkt x i y koordinate svake točke iz seta te se uzima aritmetička sredina svih izračunatih vrijednosti. Vrijednost koja se dobije tim postupkom je kovarijanca i govori o smjeru raspršenosti podataka. Izračunate vrijednosti kovarijanci vidljive su na slici 55. Autor [29] nam donosi i formulu funkcije koja se koristi za opis distribucije u sustavu s dvije dimenzije: $f(x) = \frac{1}{2\pi\sqrt{|\Sigma|}} exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))$



Slika 54: Dva seta podataka s istom varijancom [32]

Slika 55: Izračunata kovarijanca [32]

Općenito govoreći, moguće je na prvi pogled pretpostaviti kakvu će kovarijancu imati koji set podataka. U slučaju da se podatci rasprostiru uz negativnu dijagonalu (od gornjeg lijevog kuta do donjeg desnog kuta), zaključuje se da će takav set podataka imati i negativnu kovarijancu. U slučaju da su podatci u setu poprilično centrirani, odnosno da su ravnomjerno raspršeni u vodoravnom ili okomitom smjeru, donosi se zaključak da će kovarijanca takvog seta podataka biti vrlo mala ili će iznositi nula. Na samom kraju postoji situacija u kojoj se podatci rasprostiru uzduž pozitivne dijagonale (od donjeg lijevog kuta do gornjeg desnog kuta) je moguće izvući zaključak da će takav set podataka imati pozitivnu kovarijancu.

6.2. Programsko rješenje

Praktičan primjer provest će se koristeći navedene algoritme i metode na vlastitom setu podataka. Primjer sa [21] bit će modificiran da odgovara za konkretan slučaj. Primjer je pisan da odgovara za set podataka s četiri varijable i tri klastera. Istražit će se jedan interesantan

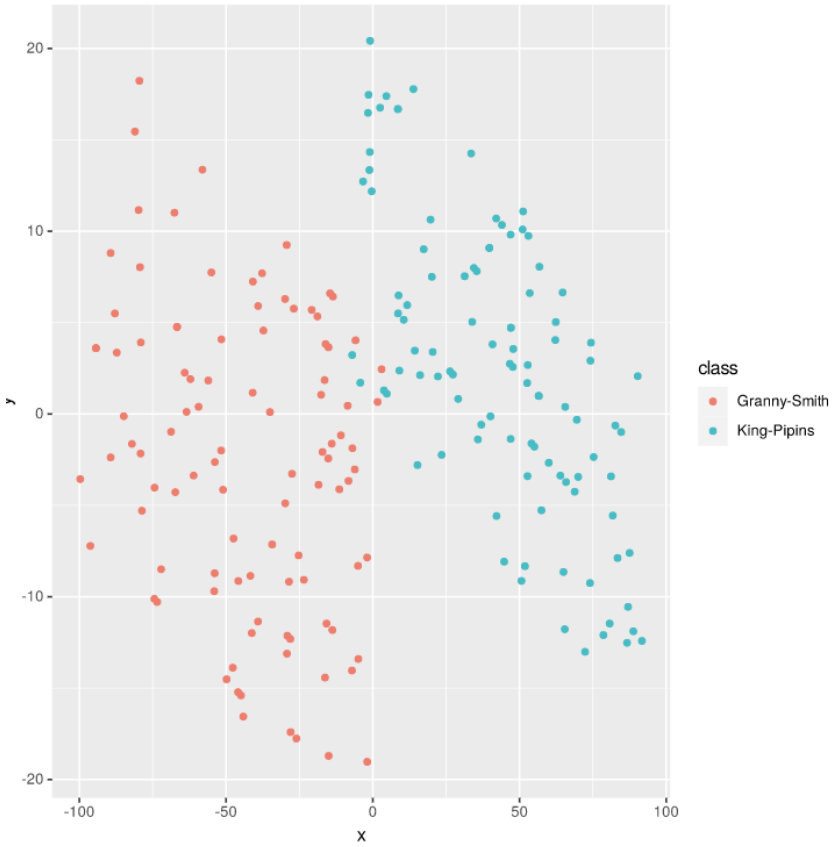
primjer koji je moguće pronaći u svakodnevnom životu. Riječ je o razlikovanju sorte jabuka na temelju njihove dvije osobine: visina baze i volumen same jabuke. Visina baze govori koliko je jabuka visoka, a volumen govori o tome koliko jabuka sadrži jestivog dijela (tzv. mesa). Tako će se koristiti navedene dvije varijable i na temelju njihovih vrijednosti će se donositi zaključak o pripadnosti u dva klastera.

Provedeno je kućno istraživanje mjerenjem volumena i visina jabuka koje pripadaju određenoj sorti. Istraživanje je provedeno na uzorku od osam jabuka (četiri jabuke jedne sorte te četiri jabuke druge sorte). Iz tog istraživanje izvlači referentni interval za svaku varijablu te se nakon svega kreira CSV datoteka koja će umnožiti broj mjerenja. Vrijednost svakog mjerenja sadrži nasumičnu vrijednost iz referentnog intervala za pojedinu varijablu. Tako se dobije CSV datoteka s 50 uzoraka jabuka jedne i 50 uzoraka mjerenaj jabuka druge sorte. Koristit će se tehnika opisanu u poglavlju 5.1 ovog rada te, za potrebe crtanja, koristit će se tehnika opisana u poglavlju 5.2 ovog rada. Drugim riječima, iz pripremljene će se CSV datoteke izvući sve činjenice o jabukama (volumen jabuke, visina baze i vrsta jabuke) te će biti korišten R server (rserve) za grafički prikaz rezultata obrade. Prikupljeni podatci odgovaraju dvijema sortama jabuka: *Granny Smith* i *King of the Pippins*. Iako su ove dvije sorte po izgledu znatno različite, ipak imaju nekih dodirnih točaka, a na kojima će se testirati točnost algoritma. Većina definiranih predikata su zapravo formule koje su ranije navedene za računanje funkcije normalne distribucije.

U ovom se primjeru SWISH smatra "programom domaćin", odnosno, sam algoritam i izračun se odvija preko njega. R server se koristi za prikaz početnog (stvarnog) stanja podatkovnog seta te rezultata provođenja algoritma. Na taj se način koriste Prolog-ove temporarne varijable kako ne bi bilo nepotrebnog zauzimanja memorije prilikom izvođenja programa. Samom SWISH-u se zadaje željeni broj iteracija te ih on toliko izvršava. Na kraju je moguće usporebom zaključiti preciznost izvođenja samog algoritma na ponuđenom setu podataka na način da se usporedi početno stanje i ono koje je računalo naučilo.

Cijeli program se pokreće korištenjem predikata em/1 koji prikuplja sve potrebne podatke koji nam služe za procjenu i maksimizaciju ispravnog odgovora. U zagradu mu se predaje željeni broj iteracija algoritma. S obzirom da se kreće od nasumičnog mjesta, jedna iteracija vrlo vjerojatno neće biti dovoljna za ostvarenje maksimalnog mogućeg ishoda. Pretpostavlja se da preciznost algoritma eksponencijalno raste do 15. iteracije te svakom sljedećom iteracijom rješenje ne postaje znatno bolje od prethodnog te povećanje postaje sve više i više zanemarivo. Prikaz na slici 56 prikazuje korištene podatke u koordinatnom sustavu.

Navodi se program da prepozna na nasumičnom setu podataka, a bazirano na ova dva seta podataka, da se radi o dva različita klastera te da prepozna koje bi bilo optimalno središte koje obuhvaća najveći broj podataka unutar distribucije. Za to se koristi predikat kojim se definira iteriranje kroz EM algoritam. Svaka iteracija poziva jednom e_korak/4 i prosljeđuje mu trenutne parametre koji su ili proslijeđeni ili ostali od prošlog m koraka te poziva jednom m_korak/2 kojem također prosljeđuje parametre potrebne za izračun novog težišta. Bazirano na unesenom broju koraka iteracije i trenutnom broju iteracije, program rekurzivno poziva isti predikat.

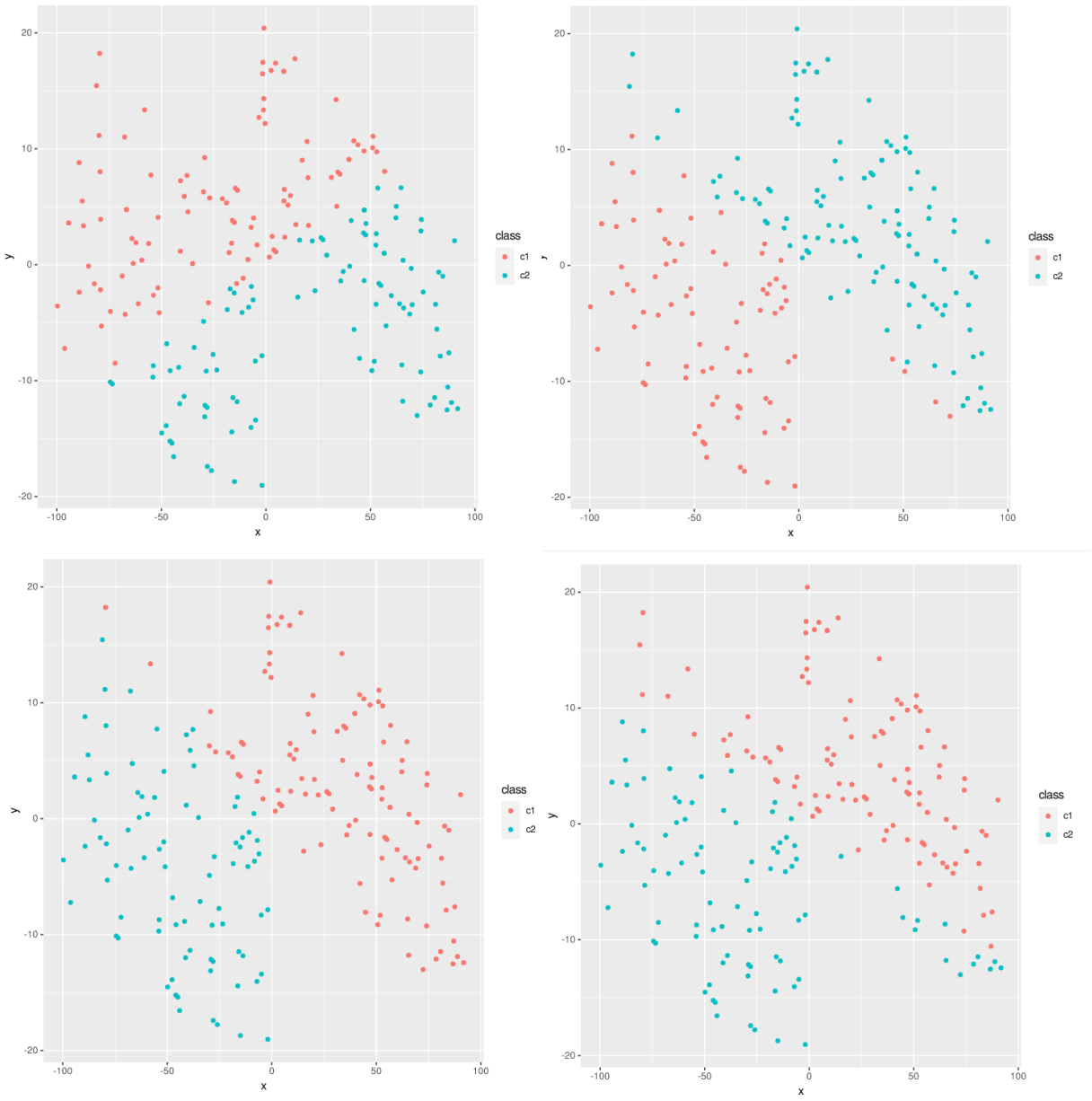


Slika 56: Distribucija podataka o jabukama [Autorski rad]

Većina predikata koji se koriste su zapravo izračuni formula kao što su središte, varijanca, određivanje maksimuma (za odluku o tome kojem klasteru pripada pojedini podatak), izračunavanje vjeroajtnosti te funkcije za dvodimenzionalnu distribuciju. Također, koristi se i par predikata koji služe da se mijenja pripadnost klasteru ili da ga se izbaci iz pojedinog podatka. To je jednostavan rad s listom te takvi i gore navedeni predikati neće biti posebice objašnjavani.

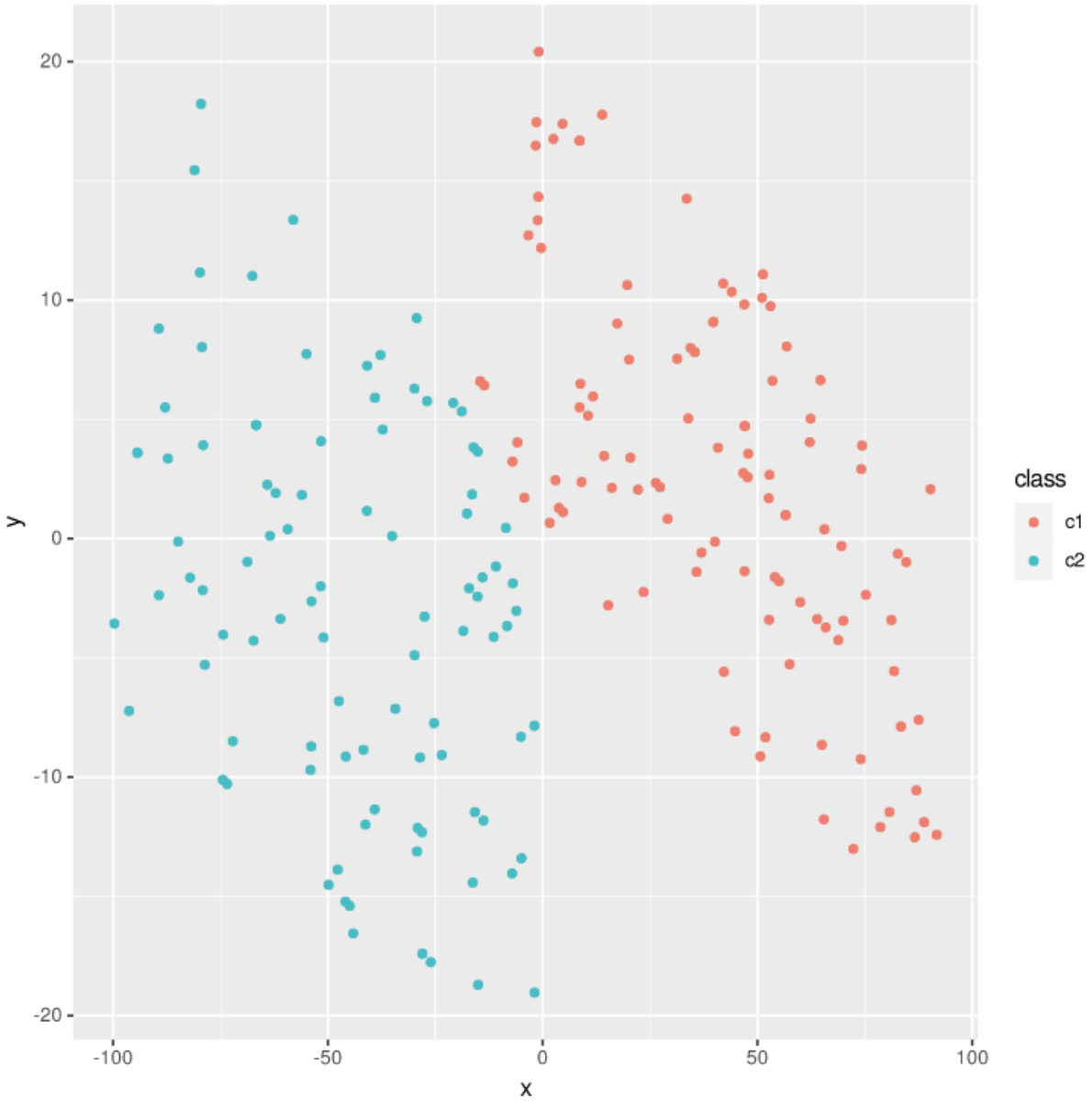
E korak kreće tako da se prikupe podatci koji su izračunati i pokušava se odrediti kojem klasteru bi pripadao koji podatak. On vraća onaj težišni dio u kojem bi, primjerice, jedan podatak pripadao 60% u jedan, a 40% u drugi klaster. Taj dio si algoritam provodi kao *soft clustering*, međutim to služi samom programu za izračun te ipak na kraju prikaže rezultat korištenjem tehnike *hard clusteringa*. Za podatak se određuje kojem klasteru pripada te na temelju toga pokušavamo odrediti nove, bolje parametre kojima bi se objedinili podatci. Izračunavaju se pripadnosti za jedan klaster, te pripadnost za drugi klaster i odabire se maksimum od te dvije vrijednosti kako bi se odredio klaster.

Pokrene li se više puta program s jednom iteracijom, moguće je pratiti kako se uistinu kreće s nasumičnom pozicijom na koordinatnom sustavu za svaku distribuciju. Na taj način nastaju početni položaji kao što je vidljivo i na slici 57. Zato je ranije spomenuto da najvjerojatnije jedna iteracija neće biti dovoljna da aktivni program shvati o koja se dva klastera radi. Moguće je zanemariti što boje možda nisu iste kao u početnom setu podataka jer program ne razumije da se radi o točno toj vrsti jabuke, bitno je samo da nauči da se radi o dvije različita klastera podataka, ali ne i koja.



Slika 57: Prva iteracija iz četiri različita vremena početka [Autorski rad]

Pokrene li se algoritam s, primjerice, 10 iteracija, vidljivo je kako rješenje već više liči na početni graf s podatcima. To je zato što u svakom trenutku program pokušava sve više doseći do distribucije koja liči onoj u početnoj klasifikaciji podataka. Također, moguće je uvidjeti i pogreške koje se događaju. Model je uspješno prepoznao da se radi o dvije vrste nečega iz promatranih podataka i uspješno je prepoznao središte oko kojeg se nalaze podatci iste klase. Međutim, model pojedine podatke pogrešno klasificira. Usporedbom slike 58 (krajnjeg rješenja) i slike 56 (stvarne klasifikacije) evidentno je da se problem dogodi prilikom vrijednosti koje preklapaju dva klastera podataka. Kada bi se podatci još više preklapali, pogreška bi vjerojatno bila veća (ovisno o tome koliko se distribucije podataka međusobno preklapaju).



Slika 58: Provedeno 10 iteracija - krajnje rješenje [Autorski rad]

7. Zaključak

R i SWISH svoje međusobno djelovanje mogu postići na nekoliko različitih i korisnih načina. Najčešća primjena ovog spoja je u području podatkovne znanosti i analize podataka. Široki spektar različitih korisnih biblioteka iz R-a omogućuje prikaz postupaka analize podataka dok SWISH (Prolog) omogućuje točnost i istinitost podataka koji se obrađuju kao i *sandbox* okruženje. Također, Prologov *backtracking* sustav pomaže prilikom korištenja algoritama koji snažno ovise o *backtrackingu* kao primjerice algoritam koji je obrađen u praktičnom primjeru. Rad je obradio tri različita načina na koje se mogu koristiti R i SWISH međusobno od kojih su neki izravniji pristupi, dok neki drugi zahtijevaju malo posredovanja kako bi se izveli.

Primjer s CSV datotekom je vrlo jednostavan, međutim ne ostavlja dojam da se odvija nešto naročito korisno i veliko tom metodom. Također, na taj način ne postoji izravan način mijenjanja podataka s obzirom na to da je potrebno proći neko određeno vrijeme nakon kojeg se mogu koristiti novi podatci koji su upisani u datoteku. Drugim riječima, korištenje vanjskog *cloud* sustava ne funkcionira bez neke određene latencije koja ipak ne smeta u slučaju da se podatci ne mijenjaju često ili da se koriste stalni podatci prilikom obrade. Posao je također zamoran jer je potrebno obraditi podatke u jednom okruženju, izvesti ih, a zatim i *uploadati*/ažurirati na poslužitelju. Uzevši u obzir latenciju učitavanja ažuriranih podataka, u slučaju da su hitno potrebne promjene, jednostavnije je pronaći novo mjesto za *upload* podataka, nego da se čekaju novoučitani podatci.

Malo izravniji način je također obrađen je korištenjem rserve, odnosno R servera unutar SWISH okruženja. Jednostavno je moguće poslati R upit unutar Prolog upita i rezultat će vratiti R server. Korištenje je vrlo jednostavno i intuitivno, međutim takvo korištenje sa sobom nosi i nekoliko problema zbog kojih neki drugi načini djeluju bolje i prigodnije za situaciju. Sama činjenica ovisnosti o serveru nije neko idealno stanje. Server ne mora biti uvijek dostupan, a postoji i mogućnost latencije odgovora. Čak i da je server dostupan >99% vremena, u tih <1% može biti hitno potrebna obrada podataka. U slučaju da je potrebno koristiti neke pojedinosti iz novijih verzija R-a od onih koje se nalaze na serveru, to neće biti moguće izvesti dok se na serveru ne ažurira radna R verzija. Velika prednost leži u samoj činjenici da R ima široki spektar različitih paketa koje je moguće koristiti u obradi podataka. Jednostavno je potrebno označiti koji se paketi koriste zato što su svi već instalirani na poslužitelju i provesti željenu komandu.

Treći od načina koji su obrađeni je korištenje rolog paketa unutar R razvojnog okruženja RStudio. Primjer više prikazuje korištenje Prolog-a u R-u, ali je dokumentiran na SWISH stranici tako da je i taj način obrađen. Kao što je već više puta naglašena velika prednost R-a kao programskog jezika, tako je važno ponoviti da je olakotna okolnost što se radi o paketu koji se može instalirati i koristiti. Međutim, problemi nastaju kada verzija R-a nije dovoljno nova za korištenje takvog paketa. Dokumentacija je također jedan od razloga zašto ovo ne bi bio prvi i omiljeni način uz pomoć kojeg se ujedinjuju ova dva programska jezika. Naime, dokumentacija je u doba pisanja ovog rada nepotpuna, nedovoljno dokumentirana i na ponekim mjestima netočna. Sam način korištenja paketa je poprilično drugačiji od onog koje dolijuje uobičajenim R paketima. S druge je pak strane moguće koristiti Prolog kao sustav nakon obrade podataka

u R-u ili pak kao sustav koji će prikupljati podatke koji se obrađuju naknadno u R-u. Poprimanje varijabli i tipova varijabli među programskim jezicima je jednostavna nakon provođenja nekoliko primjera. Također, jedna od prednosti ovog pristupa je i korištenje vanjske datoteke kao baze znanja za obradu i prikaz podataka. Tako je, primjerice, moguće koristiti podatke koji se uistinu smatraju istinitima jer sve što se ne nalazi u bazi podataka se smatra lažnim (Prologov pristup zatvorenog svijeta). To znači da su podatci koji se obrađuju i prikazuju istiniti i da se slučajno neće koristiti lažni podatci sve dok je sve istinito zapisano u bazu znanja i dok su pravila dobro definirana.

Obrađen je i jedan praktičan primjer iz grane podatkovne znanosti pod nazivom strojno učenje korištenjem EM algoritma za klasteriranje. Na taj se način uspio izraditi model za realnu situaciju iz stvarnoga života za prepoznavanje sorte jabuka. Praktičnih primjera na koje je moguće izvesti isti rezultat beskonačan je broj jer će uvijek biti potreba za nekom vrstom prepoznavanja objekata (u našem slučaju jabuka) na temelju n dimenzija (svojstava). U slučaju da se programu ponudi novi podatak, on nam može reći s kojom sigurnošću može tvrditi da pripada jednom od m kategorija (klastera) koje su mu određene te smješta podatak na vrlo vjerojatno ispravno mjesto. Ovakvim postupkom mogu se izvesti različite vrste učenja na raznim područjima poslovanja i rada od medicine, preko uzgoja kultura pa sve do naprednijih računalnih vidova koji su ipak kompleksniji modeli, ali počivaju na sličnim osnovama.

Ova dva, na prvi pogled vrlo različita programska jezika ipak povezuje zajedničko korištenje. Zajedno u suradnji mogu izvesti više nego svaki od njih pojedinačno jer koriste svaki svoje prednosti te pokrivaju nedostatke jedan drugoga. U doba u kojem se podatci čuvaju i pohranjuju u velikim količinama, podatkovna znanost se najpovoljnije razvija, a to je i područje u kojem ova dva programska jezika mogu uvelike pripomoći. U budućnosti je moguće doživjeti veću popularnost ovog spoja te da će se navedeni načini međusobne suradnje poboljšati najviše što mogu te da će se češće čuti kako se projekti koriste upravo ovim dvijema tehnologijama u međusobnoj suradnji.

Popis literature

[1] Software Engineering. „How to use multiple programming languages together in the same program?” (2. kolovoza 2022.), adresa: https://softwareengineering.stackexchange.com/questions/267167/how-to-use-multiple-programming-languages-together-in-the-same-program#267169.

[2] ——, „How to use two different programs with two different languages interact?” (2. kolovoza 2022.), adresa: https://softwareengineering.stackexchange.com/questions/117552/how-to-have-two-different-programs-with-two-different-languages-interact.

[3] Wikipedia. „Foreign function interface.” (20. kolovoza 2022.), adresa: https://en.wikipedia.org/wiki/Foreign_function_interface.

[4] Software Engineering. „Why are multiple programming languages used in the development of one product or piece of software?” (2. kolovoza 2022.), adresa: https://softwareengineering.stackexchange.com/questions/370135/why-are-multiple-programming-languages-used-in-the-development-of-one-product-or/370146#370146.

[5] Andra, „Top 5 Statistical Programming Languages In Demand 2022,” *Dataresident*, 2022.

[6] P. Teetor, *R Cookbook*. O'Reilly Media, Inc., 2019.

[7] M. Patwal, *A Short Introduction to R Programming*. 2020.

[8] R Project. „What is R?” (28. srpnja 2022.), adresa: https://www.r-project.org/about.html.

[9] N. Tomić, *Upoznavanje sa sintaksom jezika R i njegova primjena u osnovnoj statističkoj i grafičkoj analizi podataka*. 2021.

[10] Tutorialspoint. „R-Data Types.” (13. kolovoza 2022.), adresa: https://www.tutorialspoint.com/r/r_data_types.htm.

[11] Statistical tools for high-throughput data analysis. „R Built-in Data Sets.” (4. kolovoza 2022.), adresa: http://www.sthda.com/english/wiki/r-built-in-data-sets#most-used-r-built-in-data-sets.

[12] K. Lathiya. „head in R: The Complete Guide.” (29. svibnja 2022.), adresa: https://r-lang.com/head-in-r/.

[13] Tutorialspoint. „R-Variables.” (15. kolovoza 2022.), adresa: https://www.tutorialspoint.com/r/r_variables.htm.

[14] P. Blackburn, J. Bos i K. Striegnitz, *Learn Prolog Now!* 2001.

[15] Wblog. „Logika prvog reda." (29. srpnja 2022.), adresa: https://wblog.wiki/bs/Predicate_logic.

[16] W. H. Paloski, L. L. Oddete, A. J. Krever i A. K. West. „Use of a FORTH-based PROLOG for real-time expert systems. 1:Spacelab life sciences experiment application." (), adresa: https://ntrs.nasa.gov/citations/19930073450.

[17] F. Imam. „Programming Languages Used for Logic Programming." (8. rujna 2020.), adresa: https://www.finsliqblog.com/programming-languages/programming-languages-used-for-logic-programming/.

[18] SWI Prolog. „SWI-Prolog's features." (11. kolovoza 2022.), adresa: https://www.swi-prolog.org/features.html.

[19] V. Zorov. „What is the difference between Prolog and SWI-Prolog." (10. kolovoza 2022.), adresa: https://www.quora.com/What-is-the-difference-between-Prolog-and-SWI-Prolog?share=1.

[20] V. Sekovanić. „Vježbe 1 - Uvod u Prolog." (30. srpnja 2022.), adresa: https://elfarchive2021.foi.hr/mod/hvp/view.php?id=47307#h5pbookid=180&section=top&chapter=h5p-interactive-book-chapter-1345bd21-8232-45b4-81c7-d03182d52330.

[21] „SWISH documentation." (), adresa: https://www.swi-prolog.org/pldoc/doc_for?object=manual.

[22] I. C. Education. „Data Science." (15. svibnja 2020.), adresa: https://www.ibm.com/cloud/learn/data-science-introduction#toc-data-scien-JzUB5DBI.

[23] T. Mester, „What is Data Science," *data36*, 2020.

[24] I. C. Education. „Data Science." (15. srpnja 2020.), adresa: https://www.ibm.com/cloud/learn/machine-learning.

[25] RForge.net. „Rserve - Binary R server." (17. kolovoza 2022.), adresa: https://rforge.net/Rserve/.

[26] SWI Prolog. „library(quasi quotations): Define Quasi Quotation syntax." (21. kolovoza 2022.), adresa: https://www.swi-prolog.org/pldoc/man?section=quasiquotations.

[27] M. Gondan. „rolog: Prolog queries from R." (23. kolovoza 2022.), adresa: https://cran.r-project.org/web/packages/rolog/vignettes/rolog.html.

[28] GeeksforGeeks. „ML | Expectation-Maximization Algorithm." (28. kolovoza 2022.), adresa: https://www.geeksforgeeks.org/ml-expectation-maximization-algorithm/.

[29] J. McGonagle, G. Pilling, A. Dobre i 5. others. „Gaussian Mixture Model." (1. rujna 2022.), adresa: https://brilliant.org/wiki/gaussian-mixture-model/.

[30] J. Hui. „Machine Learning —Expectation-Maximization Algorithm (EM)." (3. rujna 2022.).

[31] ritvikmath. „Gaussian Mixture Model." (2. rujna 2022.), adresa: https://www.youtube.com/watch?v=EWd1xRkyEog.

[32] Serrano.Academy. „The covariance matrix.” (8. rujna 2022.), adresa: https://www.youtube.com/watch?v=WBlnwvjfMtQ.

Popis slika

1. Osnovne operacije u R-u [Autorski rad] 4
2. Korištenje seta podataka iris [Autorski rad] 6
3. Set podataka iris [Autorski rad] 6
4. Stupčasti dijagram [Autorski rad] 7
5. Brkata kutija [Autorski rad] 7
6. Histogram [Autorski rad] 7
7. Dodjela vrijednosti varijablama u R-u [Autorski rad] 8
8. Monitor trenutnih varijabli u RStudiju [Autorski rad] 8
9. Instalacija i korištenje paketa ggplot2 unutar RStudija [Autorski rad] 9
10. Izrada i prikaz podatkovnog okvira studenti [Autorski rad] 10
11. Operacije nad podatkovnim okvirom studenti [Autorski rad] 10
12. Brzi plot dijagram nad podatkovnim okvirom studenti [Autorski rad] 11
13. Izdvajanje podataka iz podatkovnog okvira studenti [Autorski rad] 11
14. Ispis elementa prvog reda i prvog stupca [Autorski rad] 12
15. Prosjek i suma nad podatcima u podatkovnom okviru [Autorski rad] 12
16. Podskup podataka iz podatkovnog okvira studenti [Autorski rad] 13
17. Korištenje funkcije order i ispis sortiranih podataka [Autorski rad] 13
18. Brisanje stupaca u R-u [Autorski rad] 14
19. Brisanje reda i vrijednosti u R-u [Autorski rad] 14
20. Činjenice i pravilo u bazi znanja Prologa [Autorski rad] 17
21. Upiti na bazu znanja sa slike 9. [Autorski rad] 18
22. Ažurirana baza znanja [Autorski rad] 18
23. Primjeri novog upita [Autorski rad] 18
24. Tipovi podataka u Prologu [20] 19

25. Izmijenjena pravila za upis kolegija [Autorski rad] 20
26. Pokušaj upisa studenta na kolegije [Autorski rad] 20
27. Status kolegija korištenjem selekcije [Autorski rad] 21
28. Primjer upita za status na kolegiju [Autorski rad] 21
29. Obvezni i izborni kolegiji u listi [Autorski rad] 22
30. Upiti nad listama kolegija [Autorski rad] 22
31. Izgled SWISH sučelja [Autorski rad] 23
32. Područja koja čine podatkovnu znanost [23] 24
33. Izvoz vektora iz RStudija [Autorski rad] 27
34. Sirovi prikaz CSV datoteke na Githubu [Autorski rad] 28
35. Baza znanja primjera sa CSV datotekama [Autorski rad] 28
36. Upiti nad bazom znanja nakon uvođenja CSV datoteke [Autorski rad] 29
37. Provjera statusa R servera kroz SWISH [Autorski rad] 30
38. Ispis podatkovnog seta iris unutar SWISH-a [Autorski rad] 31
39. Duljina latica u SWISH-u [Autorski rad] 31
40. Tablica u SWISH-u [Autorski rad] 32
41. Usporedba duljine i širine latice cvijeta iris[Autorski rad] 32
42. Duljina latice u odnosu na vrstu cvijeta u SWISH-u [Autorski rad] 33
43. Mjerenje iskorištenosti resursa i vremena u SWISH-u [Autorski rad] 34
44. Preuzimanje slike grafa u SVG formatu [Autorski rad] 34
45. Preuzimanje slike grafa u PDF formatu [Autorski rad] 34
46. Preuzimanje podataka podatkovnog seta u CSV formatu [Autorski rad] 35
47. Baza znanja korištena u rolog primjeru [Autorski rad] 36
48. Korištenje funckija once() i findall() [Autorski rad] 37
49. Dohvaćanje dva kolegija preko funkcije query() [Autorski rad] 37
50. Postupak dohvaćanja i pospremanja podataka [Autorski rad] 39
51. Prikaz dohvaćenih podataka u tabličnom prikazu [Autorski rad] 39
52. Brzi plot dijagram broja studenata na smjerovima (rolog) [Autorski rad] 40
53. EM u jednoj dimenziji i dva klastera [30] 42
54. Dva seta podataka s istom varijancom [32] 44

55. Izračunata kovarijanca [32] 44
56. Distribucija podataka o jabukama [Autorski rad] 46
57. Prva iteracija iz četiri različita vremena početka [Autorski rad] 47
58. Provedeno 10 iteracija - krajnje rješenje [Autorski rad] 48