Primjena programskog jezika R u rješavanju problema iz operacijskih istraživanja

Jurman, Ivan

Sveučilište u Rijeci, Fakultet informatike i digitalnih tehnologija

Sveučilišni diplomski studij Informatika – nastavnički smjer

Ivan Jurman

Primjena programskog jezika R u rješavanju problema iz operacijskih istraživanja

Diplomski rad

Mentor: doc. dr. sc. Martina Holenko Dlab

Rijeka, rujan 2023.

Sveučilište u Rijeci
FIDIT Fakultet informatike i digitalnih tehnologija
www.inf.uniri.hr

Rijeka, 5. lipnja 2023.

Zadatak za diplomski rad

Pristupnik: Ivan Jurman

Naziv diplomskog rada: Primjena programskog jezika R u rješavanju problema iz operacijskih istraživanja

Naziv diplomskog rada na eng. jeziku: Application of the R programming language in solving operational research problems

Sadržaj zadatka:

Zadatak je prikazati potencijalnu primjenu programskom jeziku R u rješavanju problema iz područja operacijskih istraživanja. U radu je potrebno ukratko predstaviti mogućnosti programskog jezika R, a zatim ih demonstrirati na praktičnim primjerima. U primjerima je potrebno zastupiti različite metode operacijskih istraživanja, a rad strukturirati na način da može poslužiti kao priručnik za rješavanje problema iz operacijskih istraživanja uz pomoć programskog jezika R.

Mentorica:

Doc. dr. sc. Martina Holenko Dlab

Voditeljica za diplomske radove:

Prof. dr. sc. Ana Meštrović

Zadatak preuzet: 5. lipnja 2023.

(potpis pristupnika)

Adresa: Radmile Matejčić 2
51000 Rijeka, Hrvatska
Tel: **+385(0)51 584 700**
E-mail: **ured@inf.uniri.hr**
OIB: **64218323816**
IBAN: **HR1524020061400006966**
uniri

Sažetak

U ovom radu govori se o operacijskim istraživanjima i programskom jeziku R. Cilj ovog rada je predstavljanje mogućnosti programskog jezika R za rješavanje problema iz operacijskih istraživanja, konkretnije za rješavanje problema linearnog programiranja, cjelobrojnog linearnog programiranja, redova čekanja i problema iz teorije zaliha. U radu je najprije ukratko predstavljen programski jezik R, a zatim su navedeni problemi iz područja operacijskih istraživanja koji su objašnjeni u teoriji i kako se rješavaju u programskom jeziku R uz nekoliko pratećih primjera. Rad je osmišljen tako da predstavi mogućnosti jezika R za rješavanje problema iz operacijskih istraživanja te kao takav može poslužiti kao priručnik za korištenje istog u tu svrhu.

Ključne riječi: programski jezik R, operacijska istraživanja, linearno programiranje, redovi čekanja, teorija zaliha

Sadržaj

1. Uvod

U ovom radu se govori o operacijskim istraživanjima i programskom jeziku R. Cilj ovog rada je predstavljanje mogućnosti programskog jezika R za rješavanje problema iz operacijskih istraživanja. Konkretnije, u ovom radu predstaviti će se mogućnosti programskog jezika R za rješavanje problema linearnog programiranja, cjelobrojnog linearnog programiranja, redova čekanja i problema iz teorije zaliha. Motivacija pisanja ovog rada je sažimanje instrukcija za rješavanje problema iz operacijskih istraživanja te na taj način osim predstavljanja mogućnosti jezika R za rješavanje istih, pružanje studentima ili bilo kome s interesom za operacijska istraživanja skup instrukcija odnosno priručnik za rješavanje problema iz operacijskih istraživanja koji su uključeni u ovom radu.

U radu će najprije biti ukratko predstavljen programski jezik R, a zatim navedeni problemi iz područja operacijskih istraživanja u teoriji i implementacija u programskom jeziku R uz prateće primjere. Rad je osmišljen tako da predstavi mogućnosti jezika R za rješavanje problema iz operacijskih istraživanja te kao takav može poslužiti kao priručnik za korištenje istog u tu svrhu.

Termin operacijska istraživanja prvi put se spominje 1937. godine kada je profesor Blackett objavio prve radove s vojnim primjenama operacijskih istraživanja. Tek nakon 2. svjetskog rata počinje primjena u drugim domenama, najprije u Engleskoj, a zatim u drugim zemljama. Za operacijska istraživanja 1950. godine se počinju koristiti računala, a razdoblje od 1960.-1970. godine se u svijetu smatra „zlatnim dobom“ operacijskih istraživanja te se 1967. godine operacijska istraživanja uvode u nastavne programe [1]. Operacijska istraživanja se bave proučavanjem i upravljanjem organizacijskim, tehničkim i drugim sustavima s ciljem nalaženja optimalnih rješenja koja služe za donošenje odluka i praćenje njihovog provođenja. Predstavljaju primjenu matematičkih, statističkih, ekonometrijskih i drugih znanstvenih metoda na složene probleme. Predstavljaju i primjenu znanstvenih metoda i tehnika za optimalno rješavanje problema funkcioniranja jednog sustava. To je znanstvena disciplina koja primjenom kvantitativnih i kvalitativnih metoda proučava probleme koji se javljaju pri izvođenju nekih operacija i nalazi načine najboljeg unapređenja tih operacija, s time da je operacija svaka djelatnost koja ima određeni cilj [1].

Razne tehnike i metode operacijskih istraživanja su linearno programiranje, transportni problem, problem raspoređivanja, dinamičko programiranje, teorija redova čekanja, mrežno

planiranje, teorija grafova, teorija igara, teorija zaliha, ciljno programiranje, višekriterijsko programiranje, nelinerano programiranje i simulacije [1], no u ovom ćemo se radu baviti samo problema linearnog programiranja, cjelobrojnog linearnog programiranja, redova čekanja i problema iz teorije zaliha.

Za rješavanje problema neki od poznatih alata su Gurobi, Lindo i Cplex no to su plaćeni i vrlo skupi alati, ali pružaju besplatnu licencu za akademske korisnike. Od ostalih alata, za linearno programiranje i miješano cjelobrojno linearno programiranje koriste se besplatni alati kao što su GNU linear programming kit (GLPK), COIN-OR, CBC, lp_solve i MINOS. Mogu se koristiti i razni paketi u jeziku Python, kao i paketi u jeziku R [2].

2. Programski jezik R

R je programski jezik i okruženje za statističko računanje i grafiku. To je GNU projekt koji je sličan okruženju i programskom jeziku S koje su razvili John Chambers i kolege u Bell Laboratories (bivši AT&T, sada Lucent Technologies). Programski jezik R se može smatrati različitom implementacijom jezika S. Postoje neke važne razlike, ali mnogo koda napisanog za jezik S radi jeziku R bez potrebe da se prilagođava [3].

Slika 1 Logo programskog jezika R

Jezik R pruža širok izbor statističkih tehnika, odnosno za linearno i nelinearno modeliranje, klasične statističke testove, analize vremenskih nizova, klasifikacija, klasteriranje itd. Isto tako pruža i niz grafičkih tehnika te je vrlo proširiv [3].

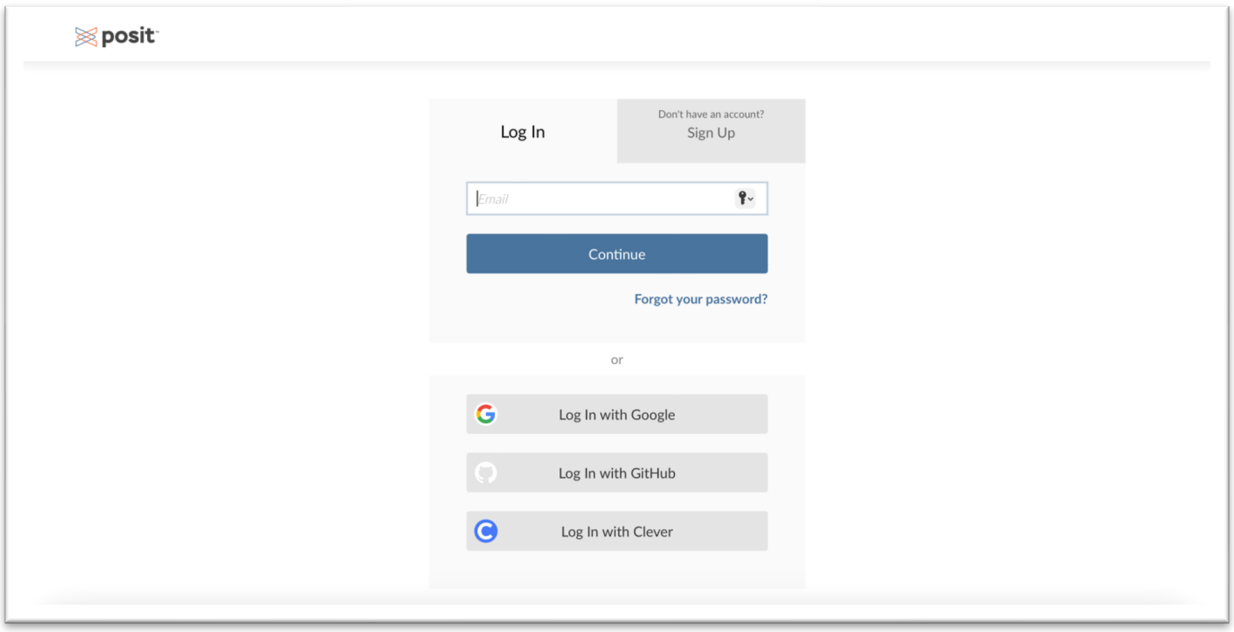
Jedna od prednosti jezika R je lakoća kojom se mogu proizvesti dobro osmišljeni grafikoni kvalitete za objavljivanje, uključujući matematičke simbole i formule svugdje gdje je to potrebno. Velika je pozornost posvećena zadanim postavkama za manje izbore dizajna u grafici uz to da korisnik zadržava potpunu kontrolu [3].

Sadašnja verzija jezika R rezultat je zajedničkog napora s doprinosima iz cijelog svijeta. R su prvobitno napisali Robert Gentleman i Ross Ihaka iz Zavoda za statistiku Sveučilišta u Aucklandu. Još od sredine 1997. godine postoji središnja skupina zvana R Core Team koja doprinosi pisanju R izvornog koda, a trenutni članovi su [4]:

- Douglas Bates
- John Chambers
- Peter Dalgaard
- Robert Gentleman
- Kurt Hornik
- Ross Ihaka
- Tomas Kalibera
- Michael Lawrence
- Friedrich Leisch
- Uwe Ligges
- Thomas Lumley
- Martin Maechler
- Sebastian Meyer
- Paul Murrell
- Martyn Plummer
- Brian Ripley
- Deepayan Sarkar
- Duncan Temple Lang
- Luke Tierney
- Simon Urbanek

Za rad u R-u jedan od populatnijih okruženja je RStudio [5] koji također ima i online varijantu zvanu RStudio Cloud u kojoj ćemo raditi u sklopu ovog rada. RStudio Cloud je potpuno besplatan, zahtjeva konekciju na Internet te pošto je to online alat dostupan je svugdje, a pristupa se putem preglednika. Prednost RStudia Cloud je ta što se napredak automatski sprema i čak kad ugasimo računalo, a zatim se prijavimo na nekom drugom računalu možemo odmah nastaviti od kud smo stali. RStudio Cloud je dostupan na sljedećem linku: https://posit.cloud

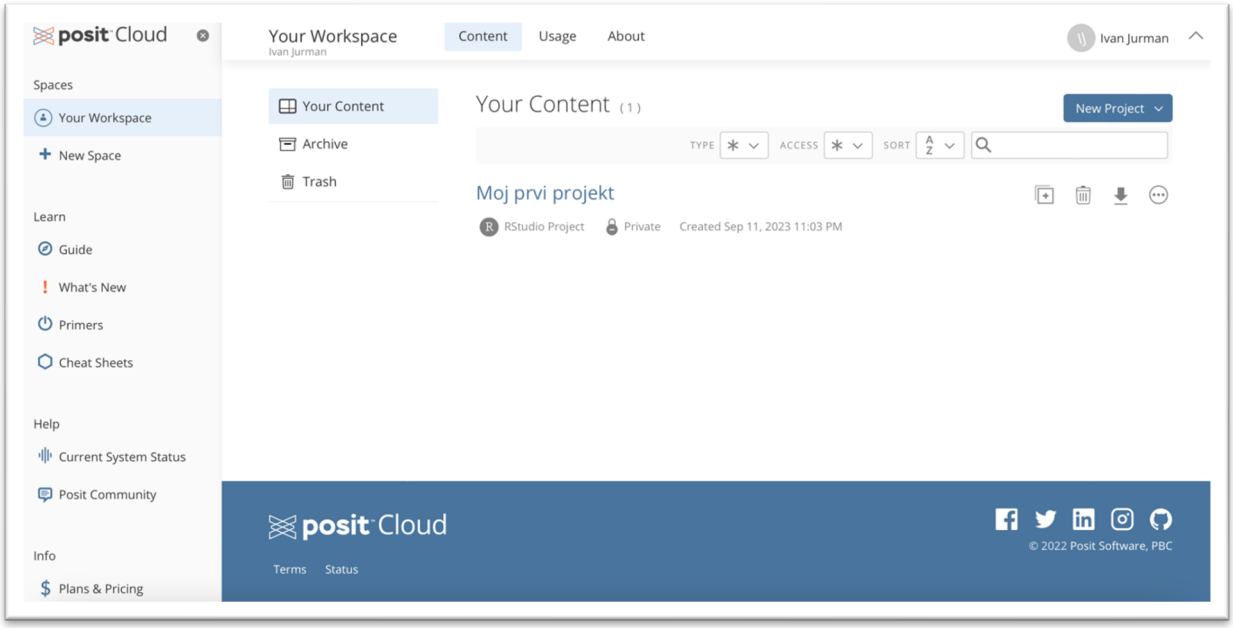
Da bismo koristili RStudio Cloud potrebno je prijaviti se, ako nemamo otvoren račun potrebno je registrirati se, to je moguće učiniti preko e-maila, Google, GitHub ili Clever računom. Izgled stranice za prijavu, odnosno registraciju je prikazan na slici 2.



Slika 2 Prijava u RStudio Cloud

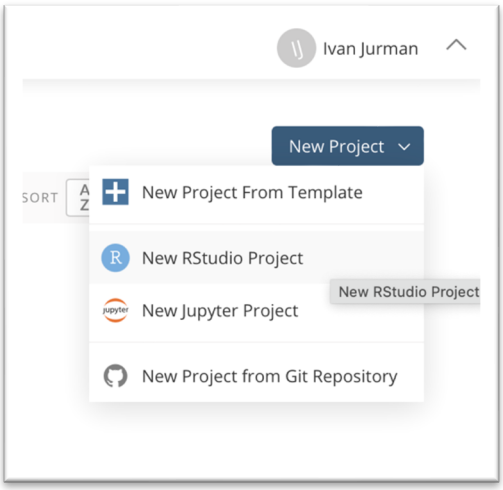
Nakon što se prijavimo u alat otvara se stranica sa slike 3 koja nam nudi pregled postojećih projekata koje smo kreirali, u ovom slučaju vidimo da je naveden samo jedan projekt pod nazivom „Moj prvi projekt“. Budući da se alat koristi u oblaku, ima ograničenje od 25 projekata po računu, stoga ima mogučnost arhiviranja projekata koji se više ne koriste. Kada arhiviramo neki projekt on se trajno arhivira te tada se ne ubraja u broj projekata računa, arhivirane projekte je moguće pregledati pod karticom „Archive“ te ih je moguće

obnoviti bilo kada. Isto tako projekt je moguće i izbrsati, nakon čega se čuva 30 dana u kartici „Trash“, ukoliko se korisnik predomisli, u suprotnom se trajno briše.



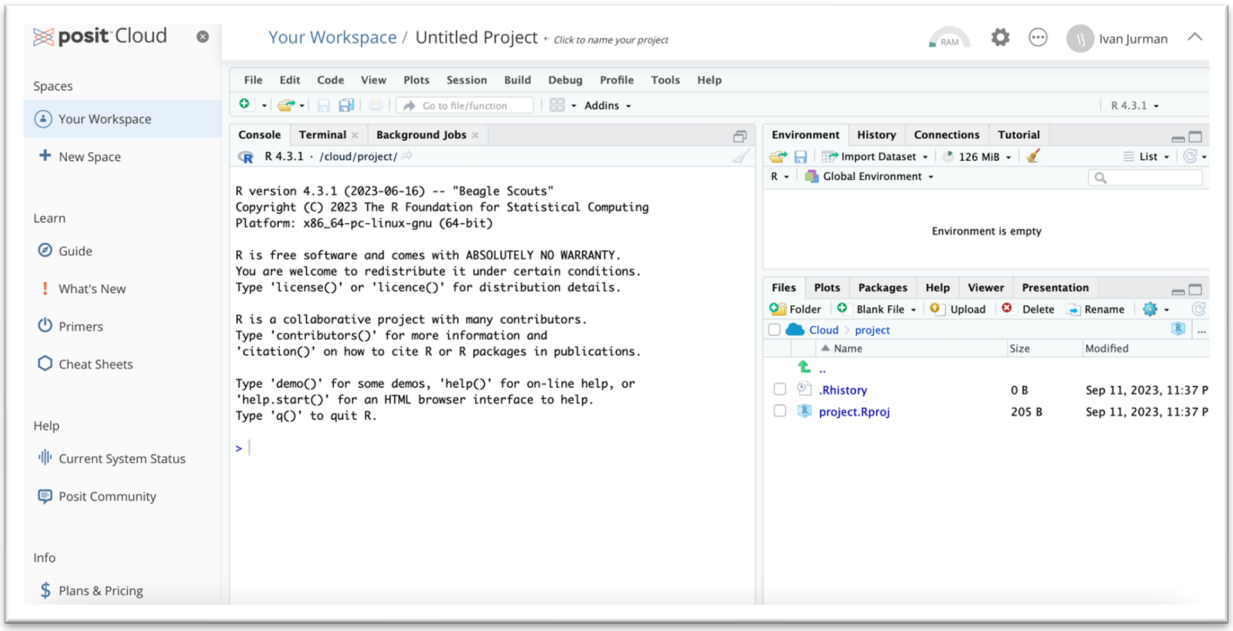
Slika 3 Pregled projekata

Novi projekt stvaramo klikom na gumb „New project“ u gornjem desnom kutu. Otvara nam se sljedeći izbornik sa slike 4, pa odabiremo opciju „New RStudio project“.



Slika 4 Izbornik za stvaranje novog projekta

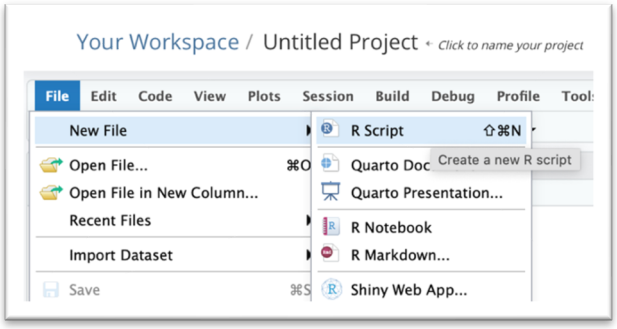
Nakon što stvorimo novi projekt, otvara se radno sučelje koje je moguće vidjeti na slici 5. U gornjem dijelu prozora vidljiv je naziv projekta, koji se u slučaju kad stvorimo novi projekt naziva „Untitled Project“ te da bi promjenili naziv samo kliknemo na naziv.



Slika 5 Radno sučelje RStudia Cloud

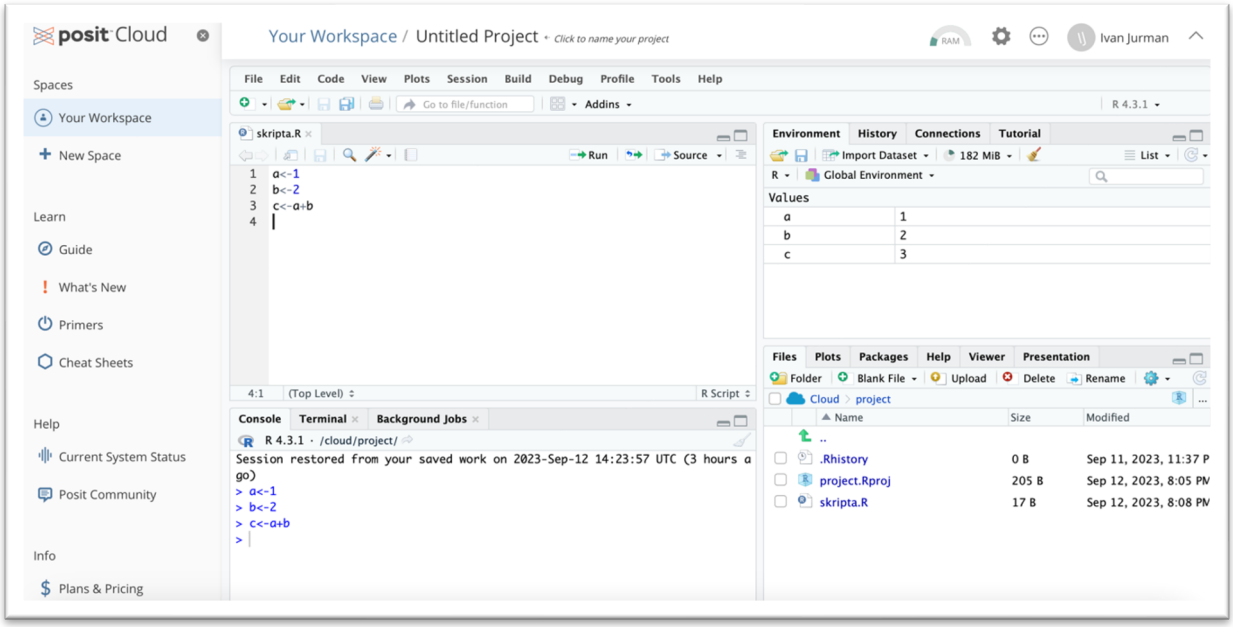
Kod stvaranja novog projekta u početku je otvorena samo konzola, u konzoli možemo pisati naredbe i pokretati ih. S desne strane pod karticom „Environment“ imamo pregled svih varijabli i njihovih vrijednosti. Ispod imamo karticu „Files“ gdje su nam prikazane sve datoteke u projektu, zatim imamo karticu „Plots“ gdje se prikazuju grafovi. Kartica „Packages“ prikazuje sve instalirane pakete, dok u kartici „Help“ možemo pronaći pomoć oko sintakse i sl. Također jedan od načina za pomoć je da označimo mišem naredbu i pritisnemo tipku F1.

Želimo li stvoriiti novu R datoteku u koju ćemo zapisati naš kod to činimo klikom na file/new file/R Script na alatnoj traci, što možemo vidjeti na slici 6.



Slika 6 Stvaranje nove R datoteke

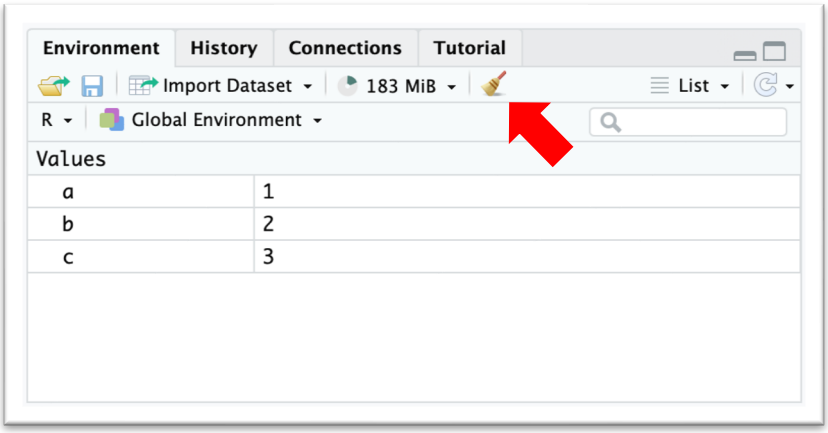
Kad stvorimo novu R datoteku otvara se u središnjem dijelu sučelja u obliku kartice, možemo imati otvoreno više R datoteka odjednom, a pregledavamo ih kao i kartice u pregledniku. Na početku se datoteka zove „Untitled1“, no prilikom spremanja možemo dodijeliti željeni naziv. U datoteku pišemo naredbe, pokretati ih možemo odmah klikom na ctrl+Enter ili kasnije tako da označimo dio koda koji želimo pokrenuti i klikom na ctrl+Enter, također moguće je označiti liniju i klikom na ctrl+Enter pokretati liniju-po-liniju.



Slika 7 Sučelje nakon stvaranja nove R datoteke, upisa i pokretanja tri naredbe

Na slici 7 možemo vidjeti tri unesene naredbe u R datoteku koju smo spremili pod nazivom “skripta”, kao što vidimo datoteka “skripta.R” je navedena i na kartici “Files” u donjem desnom uglu sučelja. Pod karticom “Console” vidimo da su sve tri naredbe pokrenute, tu bi se također prikazali i rezultati što ćemo vidjeti na primjerima kasnije u radu, a pod karticom “Environment” možemo vidjeti varijable a, b i c sa pridruženim vrijednostima.

Budući da se varijable spremaju, ukoliko želimo skriptu pokrenuti iznova ili želimo pokrenuti neku drugu skriptu preporuča se brisanje memorije kako nebi došlo do grešaka u računanju u slučaju da postoje varijable s istim nazivom. Da bi obrisali memoriju potrebno je kliknuti na ikonu metlice pod karticom "Environment", položaj metlice je prikazan na slici 8.



Slika 8 Brisanje memorije

3. Problemi iz područja operacijskih istraživanja

U ovom poglavlju bit će objašnjeni problemi linearnog programiranja, cjelobrojnog linearnog programiranja, problemi redova čekanja i teorija zaliha.

3.1. Linearno programiranje

Linearno programiranje je metoda kojom se pokušava postići najbolji ishod odnosno minimalna ili maksimalna vrijednost funkcije cilja u nekom matematičkom modelu čiji su uvjeti iskazani linearnim uvjetima. Tehniku linearnog programiranja je prvi razvio Leonid Kantorovich 1939. godine za vrijeme II. svjetskog rata kako bi planirao troškove i zaradu i na način kako bi smanjio troškove vojske i povećao gubitke neprijatelja [6].

Za rješavanje problema linearnog programiranja potrebni su nam sljedeći podaci:

Resurs	**Količina resursa *i* korištena za jedinicu aktivnosti**				Količina dostupnih resursa
	Aktivnost				
	1	2	…	n	
1	a_{11}	a_{12}	…	a_{1n}	b_1
2	a_{21}	a_{22}	…	a_{2n}	b_2
…	…	…	…	…	…
m	a_{m1}	a_{m2}	…	a_{mn}	b_m
Prinos Z po jedinici aktivnosti	c_1	c_2	…	c_n	

Tablica 1 Podaci potrebni za formulaciju linearnog programiranja

Po definiciji želimo pronaći minimum ili maksimum sljedeće funkcije koju označavamo sa Z i zovemo funkcijom cilja. Za pronalaženja maksimuma, odnosno kod maksimizacije to činimo na sljedeći način:

$$\max Z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

uz ograničenja

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2$$
$$\dots$$
$$a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m$$

uz uvjet nenegativnosti

$$x_1, x_2, \dots x_n \geq 0$$

Isto možemo zapisati i skraćeno:

$$Z = \sum_{j=1}^{n} c_j x_j$$

uz ograničenja

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i \ \ (i = 1,2, \dots, m)$$
$$x_j \geq 0 \ \ (i = 1,2, \dots, n)$$

gdje su x_j razina aktivnosti j odnosno varijable odlučivanja, c_j povećanje vrijednosti Z pri jediničnom porastu razine aktivnosti j odnosno x_j, b_i dostupna količina resursa i te a_{ij} količina resursa i korištena za aktivnost j [7].

Osim navedenog postoje i drugi oblici linearnog programiranja:

1. Minimizacija funkcije cilja:
$$\min Z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

2. Ograničenja sa oznakom „veći ili jednak od“ nejednakosti:
$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \geq b_i \qquad \text{za neke vrijednosti } i$$

3. Jednadžba kao ograničenje:
$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n = b_i \qquad \text{za neke vrijednosti } i$$

4. Izostavljanje uvjeta nenegativnosti za neke varijable koje su neograničene.

3.1.1. Implementacija u R-u

Neka je zadan matematički model linearnog problema s dvije varijable odlučivanja:

$$\max Z = c_1x_1 + c_2x_2$$

uz ograničenja

$$a_{11}x_1 + a_{12}x_2 \leq b_1$$
$$a_{21}x_1 + a_{22}x_2 \leq b_2$$
$$x_1, x_2 \geq 0$$

Da bi mogli koristiti funkcije rješavanje linearnog programiranja u jeziku R, potrebno je najprije instalirati i u kodu pozvati paket ”lpSolve”, a to činimo na sljedeći način:

```
install.packages("lpSolve")
library(lpSolve)
```

Koeficijente funkcije cilja (objective function) zapisujemo kao matricu redak (vektor) pomoću sljedeće naredbe:

```
f.obj <- c(c1, c2)
```

Koeficijente iz ograničenja (constraints) koji se nalaze uz varijable odlučivanja zapisujemo redom kao matricu pomocću sljedeće naredbe gdje je vrijednost parametra *nrow* broj redaka, a *byrow=TRUE* označava da se koeficijenti upisuju u matricu u horizontalnom smjeru slijeva na desno, redak po redak na sljedeći način:

```
f.con <- matrix(c(a11,a12, a21, a22), nrow=2, byrow=TRUE)
```

Zatim je potrebno unijeti i matricu redak sa smjerom (direction) znakova nejednakosti ili znakova jednakosti iz ograničenja, i to po redu na sljedeći način:

```
f.dir <- c("<=", "<=")
```

Unosimo matricu redak sa slobodnim koeficijentima koji se nalaze s desne strane znakova nejednakosti u oraničenjima (right-hand side):

```
f.rhs <- c(b1, b2)
```

I za kraj pokrećemo naredbu za odredivanje optimalne vrijednosti funkcije cilja Z pri čemu navodimo da se radi o funkciji maksimizacije:

```
lp("max“, f.obj, f.com, f.dir, f.rhs)
```

A za odredivanje optimalnih vrijednosti varijabli x1 i x2 pokrećemo sljedeću naredbu:

```
lp("max“, f.obj, f.com, f.dir, f.rhs)$solution
```

Napomena: uvjete nenegativnosti u kodu ne definiramo jer će ih R automatski uključiti [8].

3.1.2. Primjeri

Primjer 1: problem proizvodnje

Zamislimo da imamo problem proizvodnje koji želimo riješiti. Dakle, poduzeće proizvodi 2 vrste proizvoda u tvornici koja se sastoji od 3 proizvodna odjela: rezanje, miješanje i pakiranje. U svakom odjelu oprema se može upotrebljavati 8 sati dnevno.

- Proizvod P1 se prvo reže, a zatim pakira. Za svaku tonu tog proizvoda utroši se 1/2 sata za rezanje i 1/3 sata za pakiranje.
- Proizvod P2 se prvo se miješa, a zatim pakira. Za svaku tonu tog proizvoda utroši se 1 sat za miješanje i 2/3 sata za pakiranje.
- Proizvodi P1 i P2 se mogu se prodati po cijenama od 40 N.J, odnosno 30 N.J. po toni.

Odredite plan proizvodnje kako bi zarada bila najveća moguća [9].

Problem možemo raspisati u sljedeću tablicu:

Resurs	**Količina resursa *i* korištena za jedinicu aktivnosti**		Količina dostupnih resursa
	Aktivnost		
	P1	P2	
Rezanje	1/2	0	8
Miješanje	0	1	8
Pakiranje	1/3	2/3	8
Prinos Z po jedinici aktivnosti	40	30	

Tablica 2 Poznati parametri za primjer 1

Matematički model za navedeni problem je sljedeći:

$$\max Z = 40x_1 + 30x_2$$

uz ograničenja

$$\text{Rezanje:} \quad \frac{1}{2}x_1 \leq 8$$

$$\text{Miješanje:} \quad x_2 \leq 8$$

$$\text{Pakiranje:} \quad \frac{1}{3}x_1 + \frac{2}{3}x_2 \leq 8$$

$$x_1, x_2 \geq 0$$

Kod u jeziku R za rješavanje ovog problema će izgledati ovako:

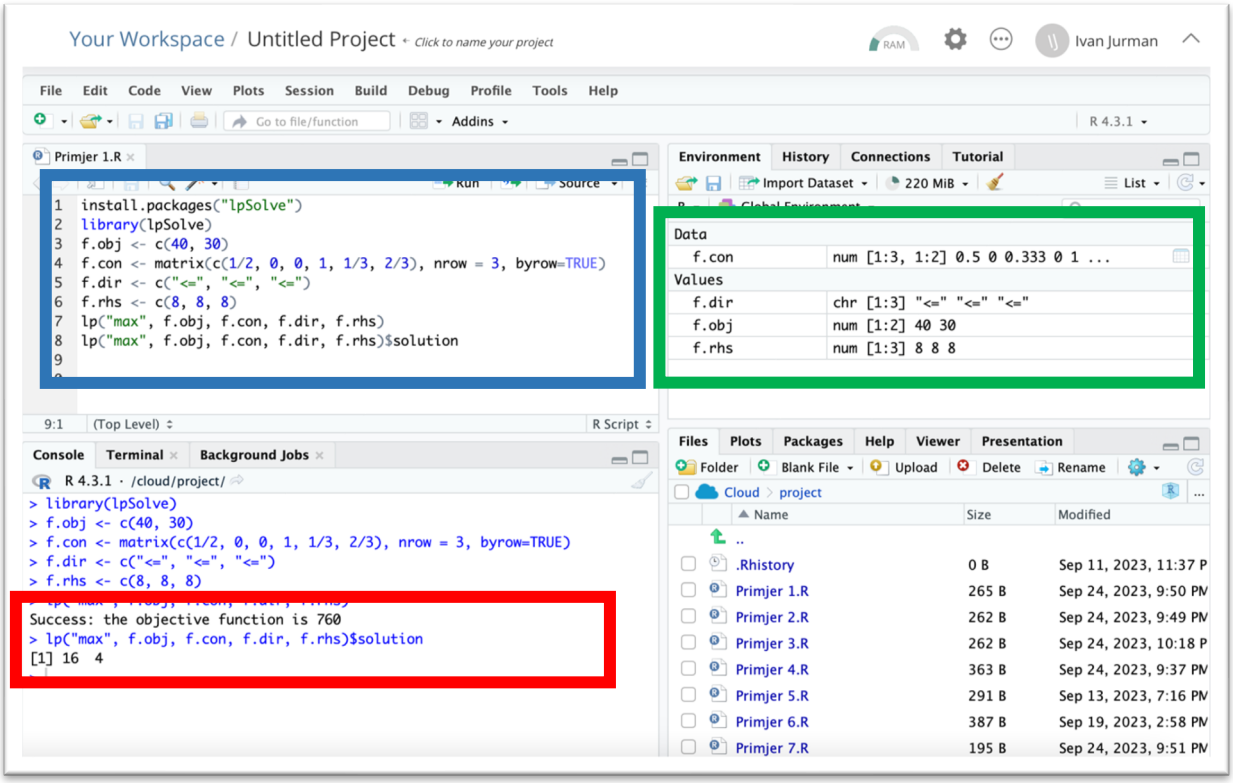
```
# instalacija i pozivanje "lpSolve" paketa
install.packages("lpSolve")
library(lpSolve)

# koeficijenti funkcije cilja          max Z = 40 x1 + 30 x2
f.obj <- c(40, 30)

# koeficijenti uz varijable odlučivanja iz ograničenja:
# rezanje: 1/2 x1 + 0 x2 <= 8
# miješanje: 0 x1 + 1 x2 <= 8
# pakiranje: 1/3 x1 + 2/3 x2 <= 8
f.con <- matrix(c(1/2, 0, 0, 1, 1/3, 2/3), nrow=3, byrow=TRUE)

# znakovi (ne)jednakosti
f.dir <- c("<=", "<=", "<=")

# slobodni koeficijenti iz ograničenja
f.rhs <- c(8, 8, 8)

# optimalna vrijednost funkcije cilja Z
lp("max", f.obj, f.con, f.dir, f.rhs)

# optimalna vrijednost varijabli x1 i x2
lp("max", f.obj, f.con, f.dir, f.rhs)$solution
```

Posljednje dvije naredbe nam daju sljedeći rezultat, odnosno optimalno rješenje problema biti će prikazano na sljedeći način:

```
Success: the objective function is 760
[1] 16  4
```

U prvom redu je prikazana vrijednosti funkcije cilja $Z = 760$, a u drugom vrijednosti varijabli odlučivanja $x_1 = 16$ i $x_2 = 4$ [8].

Na slici 9 je prikazano kako dolazimo do rješenja u alatu RStudio Cloud. Plavim okvirom je označen kod programa (namjerno su izostavljeni komentari), crvenim okvirom je označen izlaz, odnosno rješenje, a zelenim okvirom je označena lista varijabli s vrijednostima.



Slika 9 Prikaz rješenja primjera 1 u RStudiu

Primjer 2: problem proizvodnje betona

Zamislimo da proizvođač betona proizvodi tri vrste betona. Svaka vreća visokokvalitetnog betona sadrži 10 kg šljunka i 5 kg cementa, svaka vreća srednjekvalitetnog betona sadrži 11 kg šljunka i 4 kg cementa, dok svaka vreća niskokvalitetnog betona sadrži 12 kg šljunka i 3 kg cementa. U skladištu postoji 1920 kg šljunka i 780 kg cementa. Proizvođač ostvaruje zaradu od 1,20€ po vreći visokokvalitetnog, 1.10€ po vreći srednjekvalitetnog i 1,00€ po vreći niskokvalitetnog betona i želi odrediti koliko vreća kojeg betona treba proizvesti iz dostupnih sirovina da bi ostvario najveću zaradu [10].

Problem možemo raspisati u sljedeću tablicu:

Resurs	**Količina resursa *i* korištena za jedinicu aktivnosti**			Količina dostupnih resursa
	Aktivnost			
	V	S	N	
Šljunak	10	11	12	1920
Cement	5	4	3	780
Prinos Z po jedinici aktivnosti	1.2	1.1	1	

Tablica 3 Poznati parametri za primjer 2

Matematički model za navedeni problem je sljedeći:

$$\max Z = 1.2x_1 + 1.1x_2 + x_3$$

uz ograničenja

$$\text{Šljunak:} \quad 10x_1 + 11x_2 + 12x_3 \leq 1920$$

$$\text{Cement} \quad 5x_1 + 4x_2 + 3x_3 \leq 780$$

$$x_1, x_2, x_3 \geq 0$$

Kod u jeziku R za rješavanje ovog problema će izgledati ovako:

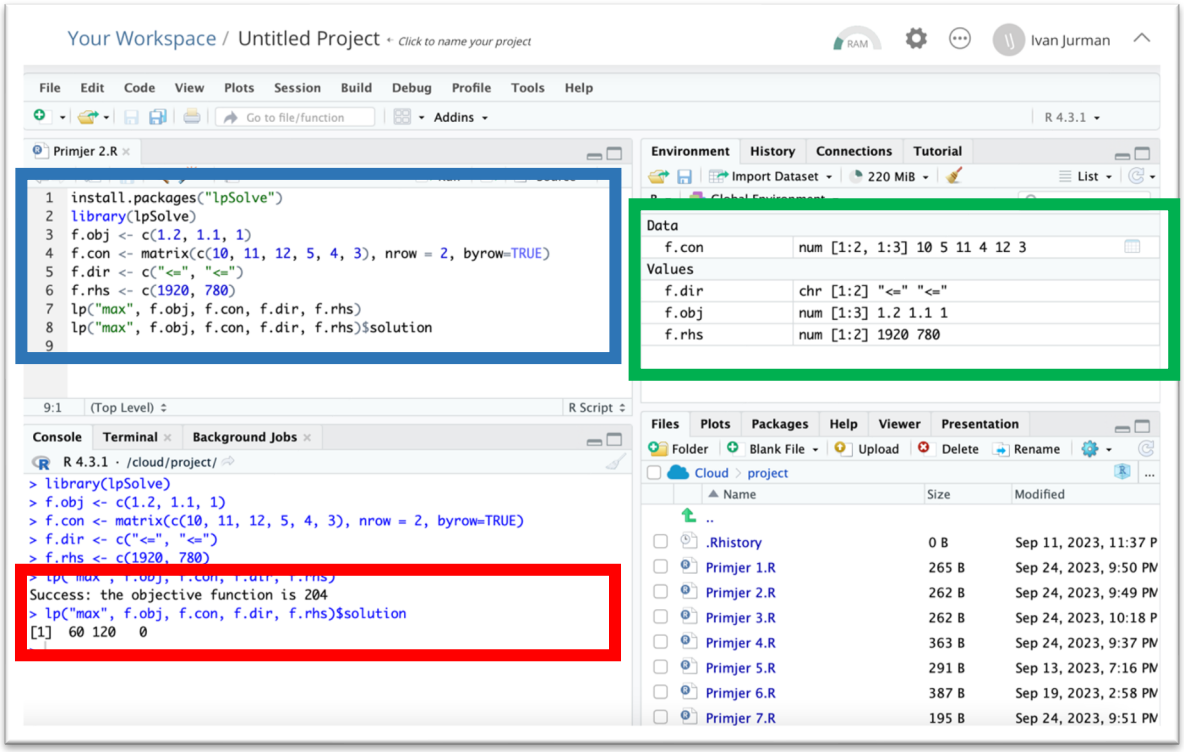
```
# instalacija i pozivanje "lpSolve" paketa
install.packages("lpSolve")
library(lpSolve)

# koeficijenti funkcije cilja          max Z = 1.2 x1 + 1.1 x2 + x3
f.obj <- c(1.2, 1.1, 1)

# koeficijenti uz varijable odlučivanja iz ograničenja:
# šljunak: 10 x1 + 11 x2 + 12 x3 <= 1920
# cement: 5 x1 + 4 x2 + 3 x3 <= 780
f.con <- matrix(c(10, 11, 12, 5, 4, 3), nrow=2, byrow=TRUE)

# znakovi (ne)jednakosti
f.dir <- c("<=", "<=")

# slobodni koeficijenti iz ograničenja
f.rhs <- c(1920, 780)

# optimalna vrijednost funkcije cilja Z
lp("max", f.obj, f.con, f.dir, f.rhs)
# optimalna vrijednost varijabli x1, x2 i x3
lp("max", f.obj, f.con, f.dir, f.rhs)$solution
```

Posljednje dvije naredbe nam daju sljedeći rezultat, odnosno optimalno rješenje problema biti će prikazano na sljedeći način:

```
Success: the objective function is 204
[1] 60  120  0
```

U prvom redu je prikazana vrijednosti funkcije cilja $Z = 204$, a u drugom vrijednosti varijabli odlučivanja $x_1 = 60,\ x_2 = 120$ i $x_3 = 0$.

Na slici 10 je prikazano kako dolazimo do rješenja u alatu RStudio Cloud. Plavim okvirom je označen kod programa (namjerno su izostavljeni komentari), crvenim okvirom je označen izlaz, odnosno rješenje, a zelenim okvirom je označena lista varijabli s vrijednostima.



Slika 10 Prikaz rješenja primjera 2 u RStudiu

Primjer 3: problem prehrane

Za prehranu pasa se koriste dvije vrte hrane. Kilogram hrane H1 sadrži 10 grama masti, 5 grama proteina i 2 grama ugljikohidrata te košta 0.6€. Kilogram hrane H2 sadrži 4 grama masti, 5 grama proteina i 6 grana ugljikohidrata te košta 1€. Svaki pas mora dnevno u hrani dobiti najmanje 20 grama masti, 20 grama proteina i 12 grama ugljikohidrata. Potrebno je odrediti koliko koje hrane treba dnevno davati svakom pojedinom psu a da pritom troškovi prehrane uz navedene uvjete budu minimalni. [10].

Problem možemo raspisati u sljedeću tablicu:

Resurs	**Količina resursa *i* korištena za jedinicu aktivnosti**		Količina dostupnih resursa
	Aktivnost		
	H1	H2	
Masti	10	4	20
Proteini	5	5	20
Ugljikohidrati	2	6	12
Prinos Z po jedinici aktivnosti	0.6	1	

Tablica 4 Poznati parametri za primjer 3

Matematički model za navedeni problem je sljedeći:

$$\min Z = 0.6x_1 + 1x_2$$

uz ograničenja

$$\text{Masti:} \quad 10x_1 + 4x_2 \geq 20$$

$$\text{Proteini:} \quad 5x_1 + 5x_2 \geq 20$$

$$\text{Ugljikohidrati:} \quad 2x_1 + 6x_2 \geq 12$$

$$x_1, x_2 \geq 0$$

Kod u jeziku R za rješavanje ovog problema će izgledati ovako:

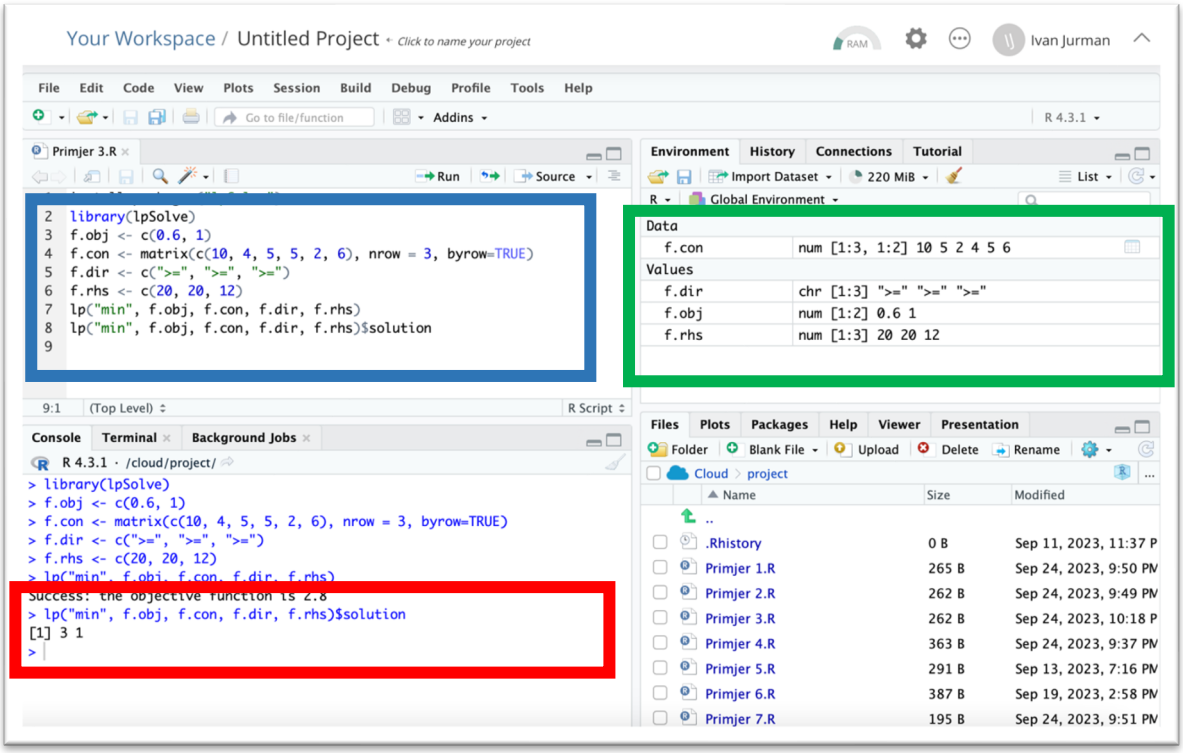
```
# instalacija i pozivanje "lpSolve" paketa
install.packages("lpSolve")
library(lpSolve)

# koeficijenti funkcije cilja           min Z = 0.6 x1 + 1 x2
f.obj <- c(0.6, 1)

# koeficijenti uz varijable odlučivanja iz ograničenja:
# masti: 10 x1 + 4 x2  >= 20
# proteini: 5x1 + 5x2 >= 20
# ugljikohidrati: 2 x1 + 6 x2 >= 12
f.con <- matrix(c(10, 4, 5, 5, 2, 6), nrow=3, byrow=TRUE)

# znakovi (ne)jednakosti
f.dir <- c(">=", ">=", ">=")

# slobodni koeficijenti iz ograničenja
f.rhs <- c(20, 20, 12)

# optimalna vrijednost funkcije cilja Z
lp("min", f.obj, f.con, f.dir, f.rhs)

# optimalna vrijednost varijabli x1 i x2
lp("min", f.obj, f.con, f.dir, f.rhs)$solution
```

Posljednje dvije naredbe nam daju sljedeći rezultat, odnosno optimalno rješenje problema biti će prikazano na sljedeći način:

```
Success: the objective function is 2.8
[1]  3  1
```

U prvom redu je prikazana vrijednosti funkcije cilja $Z = 2.8$, a u drugom vrijednosti varijabli odlučivanja $x_1 = 3, \text{i}\, x_2 = 1$.

Na slici 11 je prikazano kako dolazimo do rješenja u alatu RStudio Cloud. Plavim okvirom je označen kod programa (namjerno su izostavljeni komentari), crvenim okvirom je označen izlaz, odnosno rješenje, a zelenim okvirom je označena lista varijabli s vrijednostima.



Slika 11 Prikaz rješenja primjera 3 u RStudiu

3.2. Cjelobrojno linearno programiranje

U mnogim primjenama linearnog programiranja pojedine varijable mogu imati samo cjelobrojne vrijednosti, takve probleme nazivamo problemima iz domene cjelobrojnog linearnog programiranja [11]. Takvi problemi su vrlo teški za rješavanje. Trenutno ne postoji niti jedan generalni algoritam koji efikasno rješava sve cjelobrojne probleme linearnog programiranja [6].

U domeni cjelobrojnog programiranja razlikujemo tri vrste problema:

- **Čisto cjelobrojno programiranje** – vrijednosti svih varijabli odlučivanja su cijeli brojevi
- **Mješovito cjelobrojno programiranje (MILP)** – vrijednosti samo nekih varijabli odlučivanja su cijeli brojevi
- **Binarno programiranje** – vrijednosti varijabli su binarne, odnosno 0 ili 1

Budući da je cjelobrojno također linearno programiranje, problem se definira na isti način, samo što je potrebno uz uvjet nenegativnosti dodati i uvjet da varijable pripadaju cijelim brojevima. Pa tako ćemo problem maksimizacije čistog cjelobrojnog programiranja raspisati na sljedeći način:

$$\max Z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

uz ograničenja

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2$$
$$\dots$$
$$a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m$$

uz uvjet nenegativnosti

$$x_1, x_2, \dots x_n \geq 0$$

i uvjet cjelobrojnosti

$$x_1, x_2, \dots x_n \in \mathbb{Z}$$

Isto možemo zapisati i skraćeno:

$$Z = \sum_{j=1}^{n} c_j x_j$$

uz ograničenja

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \ \ (i = 1,2, \dots, m)$$

$$x_j \geq 0 \ \ (j = 1,2, \dots, n)$$

$$x_j \in \mathbb{Z} \ \ (j = 1,2, \dots, n)$$

gdje su x_j razina aktivnosti j odnosno varijable odlučivanja, c_j povećanje vrijednosti Z pri jediničnom porastu razine aktivnosti j odnosno x_j, b_i dostupna količina resursa i te a_{ij} količina resursa i korištena za aktivnost j [7]. Kao što možemo primijetiti dodajemo uvjet cjelobrojnosti.

U slučaju problema binarnog linearnog programiranja umjesto uvjeta nenegativnosti, pisat ćemo uvjet da varijable moraju pripadati skupu $\{0,1\}$. Pa tako umjesto $x_1, x_2, \dots x_n \geq 0$ pišemo uvjet $x_1, x_2, \dots x_n \in \{0,1\}$.

Dok u slučaju mješovitog cjelobrojnog linearnog programiranja (MILP) ne moraju sve varijable biti cjelobrojne, već samo neke, kao što neke mogu biti i binarne, pa tako dodajemo uvjete po potrebi i to samo za određene varijable.

3.2.1. Implementacija u R-u

Neka je zadan matematički model čistog cjelobrojnog linearnog problema s dvije varijable odlučivanja:

$$\max Z = c_1 x_1 + c_2 x_2$$

uz ograničenja:

$$a_{11} x_1 + a_{12} x_2 \leq b_1$$
$$a_{21} x_1 + a_{22} x_2 \leq b_2$$
$$a_{31} x_1 + a_{32} x_2 \leq b_3$$
$$x_1, x_2 \geq 0$$
$$x_1, x_2 \in \mathbb{Z}$$

Budući da je to i dalje problem linearnog programiranja, ovaj dio koda se ne razlikuje:

```
install.packages("lpSolve")
library(lpSolve)
f.obj <- c(c1, c2)
f.con <- matrix(c(a11,a12, a21, a22, a31, a32), nrow=3, byrow=TRUE)
f.dir <- c("<=", "<=", "<=")
f.rhs <- c(b1, b2, b3)
```

Razlikuju se samo pozivi funkcije za računanje rješenja, pa tako u slučaju čistog cjelobrojnog linearnog programiranja dodajemo *all.int=TRUE* na sljedeći način:

```
lp("max“, f.obj, f.com, f.dir, f.rhs, all.int=TRUE)
lp("max“, f.obj, f.com, f.dir, f.rhs, all.int=TRUE)$solution
```

U slučaju binarnog linearnog programiranja dodajemo *all.bin=TRUE* na sljedeći način:

```
lp("max“, f.obj, f.com, f.dir, f.rhs, all.bin=TRUE)
lp("max“, f.obj, f.com, f.dir, f.rhs, all.bin=TRUE)$solution
```

Dok u slučaju mješovitom cjelobrojnog linearnog programiranja, odnosno kada su samo neke varijeble cjelobrojne ili binarne, umjesto *all.int=TRUE* pišemo int.vec=vektor s rednim brojevima varijabli za koje tražimo cjelobrojnost npr. int.vec=c(2,3) te umjesto *all.bin=TRUE* pišemo binary.vec=vektor s rednim brojevima varijabli za koje tražimo binarnost npr. binary.vec=c(2,3) [11]. Recimo da imamo pet varijabli od kojih prva i druga trebaju bit cjelobrojne, a treća i četvrta binarne, to ćemo učiniti na sljedeći način:

```
lp("max“, f.obj, f.com, f.dir, f.rhs, int.vec=c(1,2), binary.vec=c(3,4))
lp("max“, f.obj, f.com, f.dir, f.rhs, int.vec=c(1,2), binary.vec=c(3,4))$solution
```

3.2.2. Primjeri

Primjer 4: problem investiranja

Zamislimo da imamo sljedeći problem i želimo pomoći prijatelju koji je broker. On razmatra razmatra 4 investicije. Očekivana dobit za investiciju 1 je $16000, za investiciju 2 $22000, za investiciju 3 $12000, a za investiciju 4 $8000. Broker trenutno ima na raspolaganju $14000 za investiranje, a za spomenute investicije mu je redom potrebno: $5000, $7000, $4000 i $3000.

Odredite koje će investicije omogućiti brokeru najveću dobit ako mora poštivati sljedeće uvjete:

1) može odabrati najviše 2 investicije,
2) ako odabere investiciju 2, onda mora odabrati i investiciju 1,
3) ako odabere investiciju 2, onda ne može odabrati investiciju 4 [11].

Problem možemo raspisati u sljedeću tablicu:

Resurs	**Količina resursa *i* korištena za jedinicu aktivnosti**				Količina dostupnih resursa
	Aktivnost				
	Investicija 1	Investicija 2	Investicija 3	Investicija 4	
Iznos za investiranje	5000	7000	4000	3000	14000
Uvjet 1)	1	1	1	1	2
Uvjet 2)	-1	1	0	0	0
Uvjet 3)	0	1	0	1	1
Prinos Z po jedinici aktivnosti	16000	22000	12000	8000	

Tablica 5 Poznati parametri za primjer 4

Matematički model za navedeni problem je sljedeći:

$$\max Z = 16000x_1 + 22000x_2 + 12000x_3 + 8000x_4$$

uz ograničenja

$$\text{Iznos za investiranje: } 5000x_1 + 7000x_2 + 4000x_3 + 3000x_4 \leq 14000$$

$$\text{Uvjet 1: } x_1 + x_2 + x_3 + x_4 \leq 2$$

$$\text{Uvjet 2: } x_2 - x_1 \leq 0$$

$$\text{Uvjet 3: } x_2 + x_4 \leq 1$$

$$x_1, x_2, x_3, x_4 \in \{0,1\}$$

Kod u jeziku R za rješavanje ovog problema će izgledati ovako:

```
# instalacija i pozivanje "lpSolve" paketa
install.packages("lpSolve")
library(lpSolve)

# koeficijenti funkcije cilja          max Z = 16000x1+22000x2+12000x3+8000x4
f.obj <- c(16000, 22000, 12000, 8000)
```

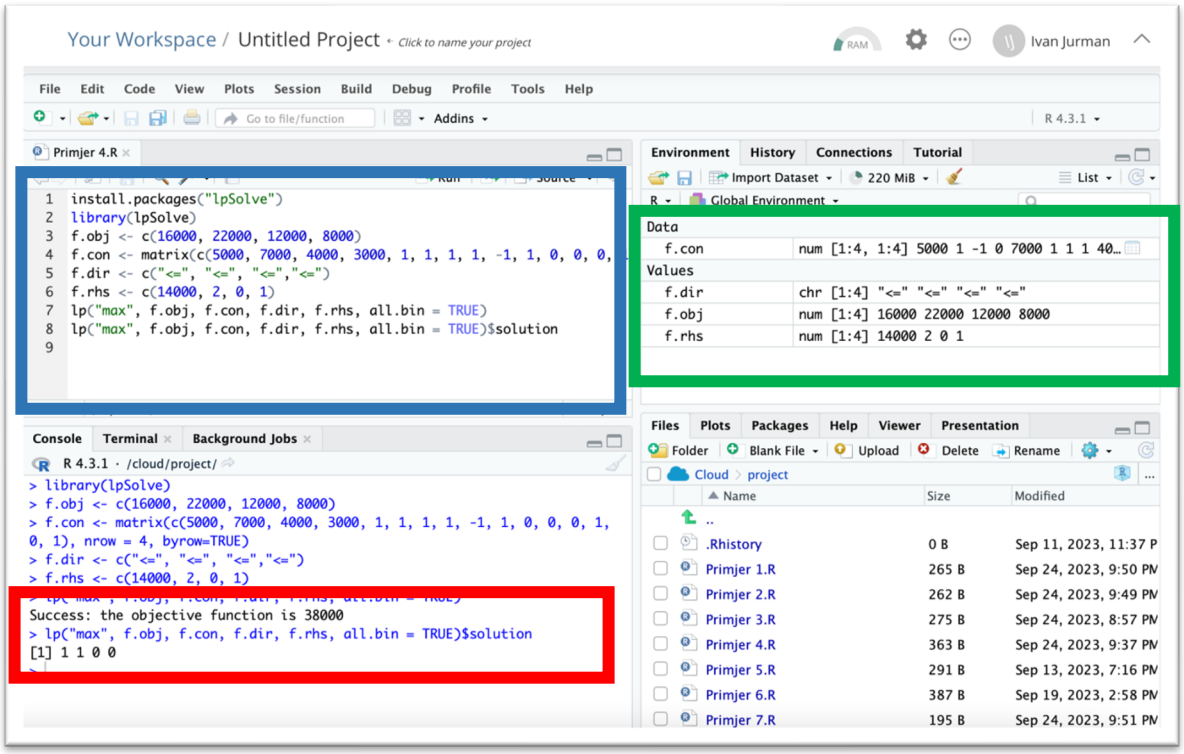
```
# koeficijenti uz varijable odlučivanja iz ograničenja:
# iznos za investiranje: 5000x1+7000x2+4000x3+3000x4<=14000
# uvjet 1: x1+x2+x3+x4<=2
# uvjet 2: x2-x1<=0
# uvjet 3: x2+x4<=1
f.con <- matrix(c(5000, 7000, 4000, 3000, 1, 1, 1, 1, -1, 1, 0, 0, 0, 1, 0, 1), nrow=4,
byrow=TRUE)

# znakovi (ne)jednakosti
f.dir <- c("<=", "<=", "<=","<=")

# slobodni koeficijenti iz ograničenja
f.rhs <- c(14000, 2, 0, 1)

# optimalna vrijednost funkcije cilja Z
lp("max", f.obj, f.con, f.dir, f.rhs, all.bin=TRUE)

# optimalna vrijednost varijabli x1 i x2
lp("max", f.obj, f.con, f.dir, f.rhs, all.bin=TRUE)$solution
```

Posljednje dvije naredbe nam daju sljedeći rezultat, odnosno optimalno rješenje problema biti će prikazano na sljedeći način:

```
Success: the objective function is 38000
[1] 1  1  0  0
```

U prvom redu je prikazana vrijednosti funkcije cilja $Z = 38000$, a u drugom vrijednosti varijabli odlučivanja $x_1 = 1, x_2 = 1, x_3 = 0$ i $x_4 = 0$ što znači da će broker investirati u prvu i drugu investiciju.

Na slici 12 je prikazano kako dolazimo do rješenja u alatu RStudio Cloud. Plavim okvirom je označen kod programa (namjerno su izostavljeni komentari), crvenim okvirom je označen izlaz, odnosno rješenje, a zelenim okvirom je označena lista varijabli s vrijednostima.



Slika 12 Prikaz rješenja primjera 4 u RStudiu

Primjer 5: problem otvaranja salona za uljepšavanje

Lanac salona za uljepšavanje želi proširiti djelatnost otvarajući nove kozmetičke i frizerske salone. Da bi se otvorio jedan kozmetički salon potrebno je uložiti 30000 kn i zaposliti 5 novih radnika, a za frizerski salon potrebno je uložiti 50000 kn i zaposliti 9 novih radnika. Očekivani prihod od kozmetičkog salona je 40000 kn godišnje, a od frizerskog salona je 80000 kn.

Poduzeće ima na raspolaganju 850000 kn, a ugovorom je utvrđeno da se ne zapošljava više od 45 novih radnika godišnje. Odredite optimalan plan širenja poduzeća koje će osigurati maksimalni godišnji prihod [9].

Problem možemo raspisati u sljedeću tablicu:

Resurs	**Količina resursa *i* korištena za jedinicu aktivnosti**		Količina dostupnih resursa
	Aktivnost		
	Kozmetički salon	Frizerski salon	
Iznos investicije	30000	50000	850000
Broj novih radnika	5	9	45
Prinos Z po jedinici aktivnosti	40000	80000	

Tablica 6 Poznati parametri za primjer 5

Matematički model za navedeni problem je sljedeći:

$$\max Z = 40000x_1 + 80000x_2$$

uz ograničenja

Iznos za investiranje: $30000x_1 + 50000x_2 \leq 850000$

Broj novih radnika: $5x_1 + 9x_2 \leq 45$

$$x_1, x_2 \in \mathbb{Z}$$

Kod u jeziku R za rješavanje ovog problema će izgledati ovako:

```
# instalacija i pozivanje "lpSolve" paketa
install.packages("lpSolve")
library(lpSolve)

# koeficijenti funkcije cilja          max Z = 40000x1+80000x2
f.obj <- c(40000, 80000)

# koeficijenti uz varijable odlučivanja iz ograničenja:
# Iznos za investiranje: 30000x1+50000x2<=850000
```

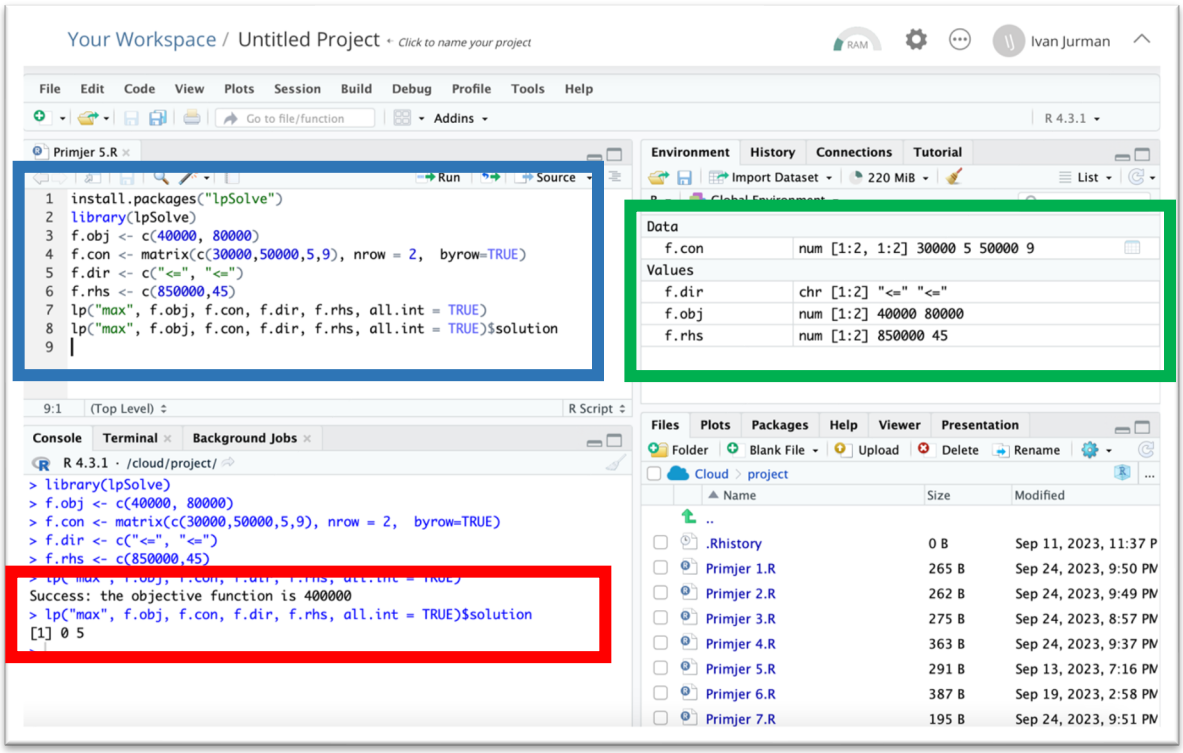
```
# Broj novih radnika: 5x1+9x2<=45
f.con <- matrix(c(30000, 50000, 5, 9), nrow=2,  byrow=TRUE)

# znakovi (ne)jednakosti
f.dir <- c("<=", "<=")

# slobodni koeficijenti iz ograničenja
f.rhs <- c(850000, 45)

# optimalna vrijednost funkcije cilja Z
lp("max", f.obj, f.con, f.dir, f.rhs, all.int=TRUE)

# optimalna vrijednost varijabli x1 i x2
lp("max", f.obj, f.con, f.dir, f.rhs, all.int=TRUE)$solution
```

Posljednje dvije naredbe nam daju sljedeći rezultat, odnosno optimalno rješenje problema biti će prikazano na sljedeći način:

```
Success: the objective function is 400000
[1] 0  5
```

U prvom redu je prikazana vrijednosti funkcije cilja $Z = 400000$, a u drugom vrijednosti varijabli odlučivanja $x_1 = 0, x_2 = 5$ što znači da će lanac salona za uljepšavanje otvoriti samo 5 frizerskih salona.

Na slici 13 je prikazano kako dolazimo do rješenja u alatu RStudio Cloud.. Plavim okvirom je označen kod programa (namjerno su izostavljeni komentari), crvenim okvirom je označen izlaz, odnosno rješenje, a zelenim okvirom je označena lista varijabli s vrijednostima.



Slika 13 Prikaz rješenja primjera 5 u RStudiu

Primjer 6: problem slastičarne

Slastičarna peče kolače i torte, u ponudi ima 3 vrste kolača i tortu. Ukupna zarada po torti je 20€, po kolaču A je zarada 23€, po kolaču B je zarada 20€ i po kolaču C je zarada 19€. Slastičarna ima u skladištu 100 jaja, 1 kilu čokolade, 1 litru mlijeka, 800 grama maslaca, 10 vanilin šećera i 9 praška za pecivo kojima uskoro istječe rok te ih treba potrošiti. Za tortu je potrebno 18 jaja, 200 grama čokolade, 200 mililitara mlijeka, 200 grama maslaca, 2 vanilin šećera i 2 praška za pecivo. Za kolač A je potrebno 16 jaja, 500 grama čokolade, 150 mililitara mlijeka, 150 grama maslaca, 4 vanilin šećera i 3 praška za pecivo. Za kolač B je potrebno 8 jaja, 50 grama čokolade, 180 mililitara mlijeka, 100 grama maslaca, 2 vanilin šećera i 1 praška za pecivo. Za kolač C je potrebno 12 jaja, 100 mililitara mlijeka, 80 grama maslaca, 1 vanilin šećera i 1 praška za pecivo. Potrebno je odrediti koliko kojih čega treba ispeći, a da pritom zarada od prodaje bude maksimalna.

Problem možemo raspisati u sljedeću tablicu:

Resurs	**Količina resursa *i* korištena za jedinicu aktivnosti**				Količina dostupnih resursa
	Aktivnost				
	Torta	Kolač A	Kolač B	Kolač C	
Jaja	18	16	8	12	100
Čokolada	0.2	0.5	0.05	0	1
Mlijeko	0.2	0.15	0.18	0.1	1
Maslac	0.2	0.15	0.1	0.08	0.8
Vanilin šećer	2	4	2	1	10
Prašak za pecivo	2	3	1	1	9
Prinos Z po jedinici aktivnosti	20	23	20	19	

Tablica 7 Poznati parametri za primjer 6

Matematički model za navedeni problem je sljedeći:

$$\max Z = 20x_1 + 23x_2 + 20x_3 + 19x_4$$

uz ograničenja

$$\text{Jaja:} \quad 18x_1 + 16x_2 + 8x_3 + 12x_4 \leq 100$$

$$\text{Čokolada:} \quad 0.2x_1 + 0.5x_2 + 0.05x_3 \leq 1$$

$$\text{Mlijeko:} \quad 0.2x_1 + 0.15x_2 + 0.18x_3 + 0.1x_4 \leq 1$$

$$\text{Maslac:} \quad 0.2x_1 + 0.15x_2 + 0.1x_3 + 0.08x_4 \leq 0.8$$

$$\text{Vanilin šećer:} \quad 2x_1 + 4x_2 + 2x_3 + x_4 \leq 10$$

$$\text{Prašak za pecivo:} \quad 2x_1 + 3x_2 + x_3 + x_4 \leq 9$$

$$x_1, x_2, x_3, x_4 \geq 0$$

$$x_1 \in \mathbb{Z}$$

Kod u jeziku R za rješavanje ovog problema će izgledati ovako:

```
# instalacija i pozivanje "lpSolve" paketa
install.packages("lpSolve")
library(lpSolve)
```

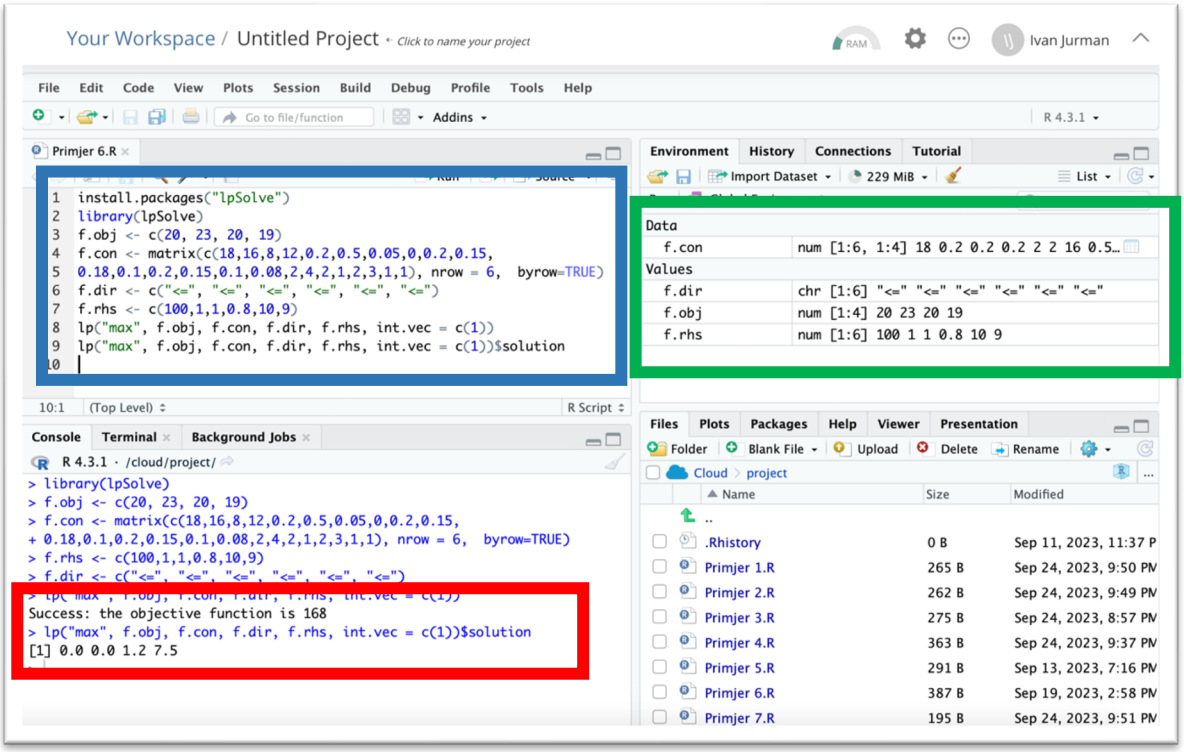
```
# koeficijenti funkcije cilja          max Z = 20x1+23x2+20x3+19x4
f.obj <- c(20, 23, 20, 19)

# koeficijenti uz varijable odlučivanja iz ograničenja:
# Jaja: 18x1+16x2+8x3+12x4<=50
# Čokolada: 0.2x1+0.5x2+0.05x3<=3
# Mlijeko: 0.2x1+0.15x2+0.18x3+0.1x4<=10
# Maslac: 0.2x1+0.15x2+0.1x3+0.08x4<=2
# Vanilin šećer: 2x1+4x2+2x3+x4<=30
# Prašak za pecivo: 2x1+3x2+x3+x4<=20
f.con <- matrix(c(18, 16, 8, 12, 0.2, 0.5, 0.05, 0, 0.2, 0.15, 0.18, 0.1, 0.2, 0.15, 0.1,
0.08, 2, 4, 2, 1, 2, 3, 1, 1), nrow=6,  byrow=TRUE)

# znakovi (ne)jednakosti
f.dir <- c("<=", "<=", "<=", "<=", "<=", "<=")

# slobodni koeficijenti iz ograničenja
f.rhs <- c(100, 1, 1, 0.8, 10, 9)

# optimalna vrijednost funkcije cilja Z
lp("max", f.obj, f.con, f.dir, f.rhs, int.vec=c(1))

# optimalna vrijednost varijabli x1 i x2
lp("max", f.obj, f.con, f.dir, f.rhs, int.vec=c(1))$solution
```

Posljednje dvije naredbe nam daju sljedeći rezultat, odnosno optimalno rješenje problema biti će prikazano na sljedeći način:

```
Success: the objective function is 167.5
[1] 0.00   0.00   1.2   7.5
```

U prvom redu je prikazana vrijednosti funkcije cilja $Z = 167.5$, a u drugom vrijednosti varijabli odlučivanja $x_1 = 0, x_2 = 0, x_3 = 1.2$ i $x_4 = 7.5$ što znači da će slastičarnica kako bi najbolje iskoristila namirnice pred istekom roka ispeći 1.25 doza kolača B i 7.5 doza kolača C.

Na slici 14 je prikazano kako dolazimo do rješenja u alatu RStudio Cloud. Plavim okvirom je označen kod programa (namjerno su izostavljeni komentari), crvenim okvirom je označen izlaz, odnosno rješenje, a zelenim okvirom je označena lista varijabli s vrijednostima.



Slika 14 Prikaz rješenja primjera 6 u RStudiu

3.3. Redovi čekanja

U svakodnevnom životu susrećemo se s raznim oblicima čekanja u redu. Neki od primjera su čekanje u redu u banci ili trgovini, čekanje dolaska vlaka, da računalo izvrši zadatak, čekanje da predstavnik službe za korisnike odgovori na poziv i sl. [12].

Teorija redova čekanja je grana matematike koja proučava kako se redovi formiraju, kako funkcioniraju te ima li problema u njihovom funkcioniranju. Ispituje svaku sastavnicu čekanja u redu, uključujući proces dolaska, proces usluge, broj poslužitelja, broj klijenata te broj mjesta u sustavu. Kao grana operacijskih istraživanja, teorija redova čekanja može pomoći u donošenju poslovnih odluka o tome kako izgraditi učinkovitije i isplativije sustave tijeka rada. Cilj teorije redova čekanja je dizajniranje uravnoteženih sustava koji služe korisnicima brzo i učinkovito, ali ne koštaju previše da bi bili održivi. Uključuje analizu dolazaka u objekt i analizu procesa koji se trenutno odvijaju za njihovo usluživanje. Konačni rezultat je skup zaključaka koji imaju za cilj identificirati sve nedostatke u sustavu i predložiti kako se oni mogu poboljšati [12].

Postoje jednokanalni i višekanalni redovi čekanja, jednokanalni redovi čekanja mogu biti neograničene ili ograničene duljine. Za probleme redova čekanja računamo sljedeće parametre:

λ – intenzitet toka dolazaka

μ – intenzitet posluživanja

ρ – stupanj opterećenja

L – prosječan broj klijenata u sustavu posluživanja

L_Q – procječan broj klijenata u redu čekanja

L_S – prosječan broj klijenata u kanalu

W – prosječno vrijeme čekanja u sustavu posluživanja

W_Q – prosječno vrijeme čekanja u redu

W_S – prosječno vrijeme posluživanja

r_P – relativna propusna moć redova čekanja

A_P – apsolutna propusna moć redova čekanja

p_n – vjerojatnost da će u redu čekanja biti n klijenata

Za izračun parametara jednokanalnog reda čekanja neograničene duljine reda koristimo sljedeće formule:

$$L = \frac{\rho}{1-\rho} \qquad L_Q = \frac{\rho^2}{1-\rho} = \rho * L \qquad L_S = L - L_Q = \rho = \frac{\lambda}{\mu} \qquad W = \frac{L}{\lambda} \qquad W_Q = \frac{L}{\mu}$$

$$W_S = W - W_Q = \frac{1}{\mu} \qquad p_0 = 1 - \rho. \qquad p_n = \rho^n * (1 - \rho)$$

uz uvjet da je koeficijent iskorištenja manji od 1 jer inače sustav nije stabilan.

$$\rho = \frac{\lambda}{\mu} \leq 1$$

Za izračun parametara jednokanalnog reda čekanja s ograničenom duljinom reda koristimo sljedeće formule:

$$\rho = \frac{\lambda}{\mu} \qquad L_Q = \frac{\rho^2 * (k * \rho^{k+1} - (k+1) * \rho^k + 1)}{(1-\rho) * (1-\rho^{k+2})} = \rho * L \qquad L_S = \frac{\rho - \rho^{k+2}}{1-\rho^{k+2}} \qquad L = L_Q + L_S$$

$$W = \frac{L}{\lambda} \qquad W_Q = \frac{L_Q}{\lambda} \qquad W_S = W - W_Q \qquad r_P = \frac{1-\rho^{k+1}}{1-\rho^{k+2}} \qquad A_P = \lambda * r_P$$

$$p_{ot} = \rho^{k+1} * \frac{1-\rho}{1-\rho^{k+2}} \qquad p_{pt} + r_P = 1 \qquad p_{ot} = p_{k+1} \qquad p_0 = \frac{1-\rho}{1-\rho^{k+2}} \qquad p_n = \rho^n * p_0$$

gdje je p_{ot} vjerojatnost otkazivanja.

Za izračun parametara višekanalnih redova čekanja s neograničenom duljinom reda potrebno je znati broj kanala S, a koristimo sljedeće formule:

$$\mu_n = \begin{cases} n \cdot \mu & ,n = 0,\ldots,S \\ S \cdot \mu & ,n = S+1, S+1,\ldots \end{cases} \qquad \rho_s = \frac{\lambda}{S \cdot \mu} < 1 \qquad p_0 = \left[1 + \rho + \frac{\rho^2}{2!} + \ldots + \frac{\rho^s}{S!} + \frac{\rho^{S+1}}{S!(S-\rho)}\right]^{-1}$$

$$p_n = \begin{cases} \frac{\rho^n}{n!} \cdot p_0 & ,1 \leq n \leq S \\ \frac{\rho^n}{S! S^{n-S}} \cdot p_0 & ,n > S \end{cases} \qquad L_Q = \frac{\rho^{S+1}}{(S-1)!(S-\rho)^2} \cdot p_0 \qquad L = L_Q + \rho \qquad L_S = \rho = L - L_Q$$

$$W_Q = \frac{L_Q}{\lambda} = \frac{\rho^{S+1}}{\lambda(S-1)!(S-\rho)^2} \cdot p_0 \qquad W = W_Q + \frac{1}{\mu} = \frac{L}{\lambda} \qquad W_s = W - W_Q$$

Za izračun parametara višekanalnih redova čekanja s ograničenom duljinom reda potrebno također znati i maksimalni broj stranaka u redu odnosno *k,* a koristimo sljedeće formule:

$$\lambda_n = \begin{cases} \lambda & ,n = 0,\ldots,k-1 \\ 0 & ,n = k,k+1,\ldots \end{cases} \quad \mu_n = \begin{cases} n\cdot\mu & ,n = 0,\ldots,S \\ S\cdot\mu & ,n = S+1,S+1,\ldots \end{cases} \quad \rho_s = \frac{\lambda}{S\cdot\mu} < 1 \qquad \frac{\rho(1-p_{otk})}{S}\cdot 100 \quad (\%)$$

$$p_0 = \left[1+\rho+\frac{\rho^2}{2!}+\ldots+\frac{\rho^s}{S!}+\frac{\rho^s}{S!}\cdot\frac{\frac{\rho}{S}-\left(\frac{\rho}{S}\right)^{k+1}}{1-\frac{\rho}{S}}\right]^{-1} \qquad p_0 = \left[\sum_{n=0}^{S}\frac{\rho^n}{n!}+\frac{\rho^{S+1}}{S!}\cdot\frac{1-\left(\frac{\rho}{S}\right)^k}{S-\rho}\right]^{-1}$$

$$p_0 = \left[\sum_{n=0}^{S}\frac{\rho^n}{n!}+k\cdot\frac{\rho^S}{S!}\right]^{-1} \quad \rho_s = 1 \quad p_{otk} = p_s + k = \frac{\rho^{S+k}}{S^k\cdot S!}\cdot p_0 \quad r_p = p_{usl} = 1-p_{otk} \quad A_p = \lambda\cdot r_p$$

$$L_Q = \frac{\rho^{S+1}}{S\cdot S!}\cdot p_0\cdot\frac{1-(k+1)\left(\frac{\rho}{S}\right)^k+k\left(\frac{\rho}{S}\right)^{k+1}}{1-\left(\frac{\rho}{S}\right)^2} \qquad L_S = \rho\cdot r_p = \frac{\lambda\cdot r_p}{\mu} = \frac{A_p}{\mu} \qquad L = L_Q + L_S$$

$$W_Q = \frac{L_Q}{\lambda} \qquad W = \frac{L}{\lambda} = W_Q + \frac{1}{\mu}\cdot r_p \qquad W_s = W - W_Q$$

3.3.1. Implementacija u R-u

Zamislimo da imamo neki problem iz redova čekanja, potrebno nam je znati tri parametra, koje smo u kodu naveli na sljedeći način te je potrebno samo zamijenit sa konkretnim vrijednostima [13]:

LAMBDA - intenzitet dolazaka

MU - intenzitet opsluživanja

N - broj klijenata u sustavu

Da bi mogli u jeziku R rješavati probleme iz redova čekanja potrebno je najprije instalirati i pokrenuti ”queueing” paket.

```
#naredbe za instalaciju i pokretanje
install.packages("queueing")
library(queueing)
```

Zamislimo da radimo s jednokanalnim redom čekanja bez ograničenja duljine reda, prema Kendallovoj notaciji, radi se o modelu MM1 pa koristimo podfunkciju *NewInput.MM1*. Da bi izlistali sve izračunate parametre odjednom to činimo sljedećom naredbom:

```
QueueingModel(NewInput.MM1(lambda=LAMBDA, mu=MU))
```

Zanima li nas npr. samo koeficijent iskorištenja odnosno faktor usluživanja ispisat ćemo ga sljedećom naredbom:

```
QueueingModel(NewInput.MM1(lambda=LAMBDA, mu=MU))$RO
```

Zanima li nas prosječan broj stranaka u sustavu opsluživanja ispisat ćemo ga sljedećom naredbom:

```
QueueingModel(NewInput.MM1(lambda=LAMBDA, mu=MU))$L
```

Zanima li nas prosječno vrijeme čekanja u sustavu opsluživanja ispisat ćemo ga sljedećom naredbom:

```
QueueingModel(NewInput.MM1(lambda=LAMBDA, mu=MU))$W
```

Dok zanima li nas vjerojatnost da će u sustavu reda čekanja biti *n* stranaka ispisat ćemo je sljedećom naredbom:

```
QueueingModel(NewInput.MM1(lambda=LAMBDA, mu=MU, n=N))$Pn
```

Analogno se može odrediti vrijednost ostalih važnijih parametara modela, npr.:

- Lq: prosječan broj stranaka u redu čekanja
- Wq: prosječno vrijeme čekanja u redu
- Throughput: koeficijent propusnosti sustava (u R-u samo apsolutna propusnost) [11]

Ako želimo izračunati parametre za jednokanalne redove s ograničenom duljinom, koristit ćemo model MM1K, tada među parametre u funkciji dodajemo vrijednost *k* koja označava ograničenje reda, a to činimo na sljedeći način:

```
QueueingModel(NewInput.MM1K(lambda=LAMBDA, mu=MU, k=K))
```

Dok ako želimo izračunati parametre za višekanalne redove s neograničenom duljinom, koristit ćemo model MMC, tada među parametre u funkciji dodajemo vrijednost *c* koja označava broj kanala, a to činimo na sljedeći način:

```
QueueingModel(NewInput.MMC(lambda=LAMBDA, mu=MU, c=C))
```

A za računanje višekanalnih redove s ograničenom duljinom koristit ćemo model MMCK, tada među parametre u funkciji također dodajemo i vrijednost *k* koja označava ograničenje reda, a to činimo na sljedeći način:

```
QueueingModel(NewInput.MMCK(lambda=LAMBDA, mu=MU, c=C, k=K))
```

3.3.2. Primjeri

Primjer 7: problem autoservisa

U svrhu primjera, zamislimo da imamo sljedeći scenarij. U autoservis dolazi u prosjeku 7 automobila na sat. Autoservis može popraviti 10 automobila na sat. Izračunajte prosječan broj vozila u redu čekanja i u sustavu, prosječno vrijeme čekanja u redu i u sustavu i prosječno vrijeme trajanja popravka. Izračunajte i vjerojatnost da u servisu neće biti reda čekanja u trenutku dolaska te vj. da će u redu biti 2 automobila [13].

To ćemo postići sljedećim kodom, komentarima je naznačeno što se računa:

```
#naredbe za instalaciju i pokretanje
install.packages("queueing")
library(queueing)

#ispiši prosječan broj stranaka u sustavu opsluživanja
QueueingModel(NewInput.MM1(lambda=7, mu=10))$L

#ispiži prosječno vrijeme čekanja u sustavu opsluživanja
QueueingModel(NewInput.MM1(lambda=7, mu=10))$W
```

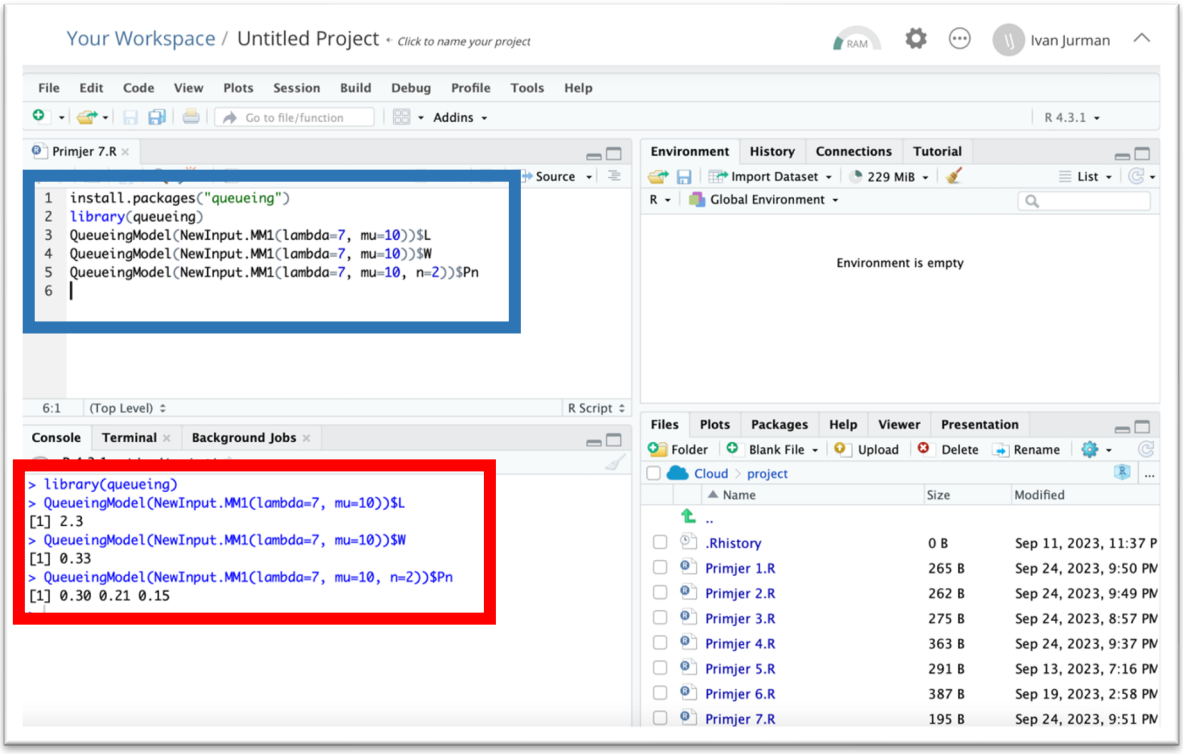
```
#Vjerojatnost da će u sustavu reda čekanja biti 0, 1, 2 stranke #respektivno
QueueingModel(NewInput.MM1(lambda=7, mu=10, n=2))$Pn
```

Po izvršenju koda, dobivamo sljedeće izlaze, odnosno rezultate:

```
> #ispiši prosječan broj stranaka u sustavu opsluživanja
> QueueingModel(NewInput.MM1(lambda=7, mu=10))$L
[1] 2.3

> #ispiši prosječno vrijeme čekanja u sustavu opsluživanja
> QueueingModel(NewInput.MM1(lambda=7, mu=10))$W
[1] 0.33

> #Vjerojatnost da će u sustavu reda čekanja biti 0, 1, 2 stranke #respektivno
> QueueingModel(NewInput.MM1(lambda=7, mu=10, n=2))$Pn
[1] 0.30   0.21   0.15
```

Dakle prosječan broj stranaka u sustavu opsluživanja je 2.3, prosječno vrijeme čekanja u sustavu opsluživanja je 0.33 sati, vjerojatnost da u sustavu neće biti stranki je 30%, da će u sustavu biti jedna stranka je 21% i da će biti dvije stranke je 15%.

Na slici 15 je prikazano kako dolazimo do rješenja u alatu RStudio Cloud. Plavim okvirom je označen kod programa (namjerno su izostavljeni komentari), crvenim okvirom je označen izlaz, odnosno rješenje.



Slika 15 Prikaz rješenja primjera 7 u RStudiu

Primjer 8: problem printanja

Na uređaj za printanje u jednom satu u prosjeku pristiže 12 zahtjeva za printanje. Proces printanja, klamanja papira i obrade po zahtjevu traje u prosjeku 4 minute. Radi učinkovitosti printera zahtjevi ne mogu čekati dulje od 10 minuta na popisu zahtjeva u uređaju. Na uslugu printanja mogu čekati do 2 zahtjeva. Želimo izračunati koeficijent iskorištenja, prosječan broj zahtjeva za printanje u sustavu, prosječno vrijeme čekanja u sustavu, vjerojatnost da u sustavu reda čekanja neće biti zahtjeva za printanje i prosječno vrijeme čekanja u redu na printanje.

To ćemo postići sljedećim kodom, komentarima je naznačeno što se računa:

```
#naredbe za instalaciju i pokretanje
install.packages("queueing")
library(queueing)
```

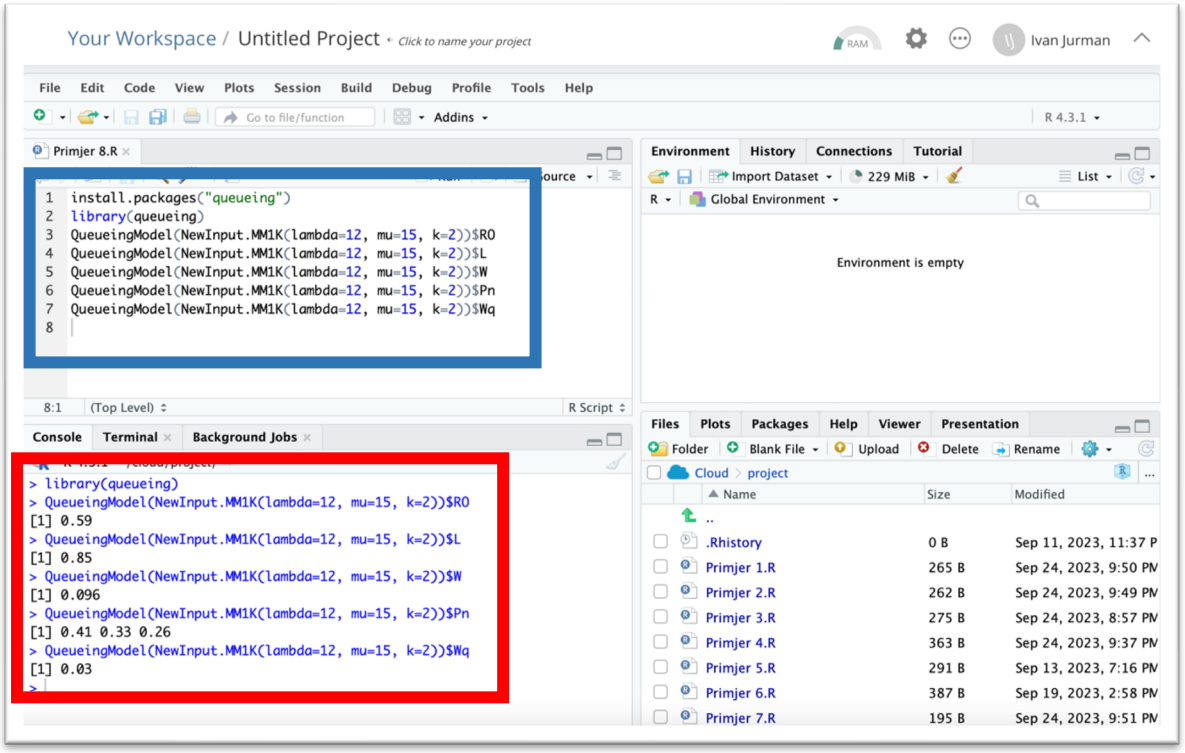
```
#ispiši koeficijent iskorištenja (faktor usluživanja)
QueueingModel(NewInput.MM1K(lambda=12, mu=15, k=2)) $RO

#ispiši prosječan broj zahtjeva za printanje u sustavu
QueueingModel(NewInput.MM1K(lambda=12, mu=15, k=2))$L

#ispiži prosječno vrijeme čekanja na printanje u sustavu
QueueingModel(NewInput.MM1K(lambda=12, mu=15, k=2))$W

#Vjerojatnost da će u sustavu reda čekanja biti 0 zahtjeva za printanje respektivno
QueueingModel(NewInput.MM1K(lambda=12, mu=15, k=2))$Pn

#ispiži prosječno vrijeme čekanja u redu za printanje
QueueingModel(NewInput.MM1K(lambda=12, mu=15, k=2))$Wq
```

Po izvršenju koda, dobivamo sljedeće izlaze, odnosno rezultate:

```
> # ispiši koeficijent iskorištenja (faktor usluživanja)
> QueueingModel(NewInput.MM1K(lambda=12, mu=15, k=2))$RO
[1] 0.59

> # ispiši prosječan broj zahtjeva za printanje u sustavu
> QueueingModel(NewInput.MM1K(lambda=12, mu=15, k=2))$L
[1] 0.85

> # ispiži prosječno vrijeme čekanja na printanje u sustavu
> QueueingModel(NewInput.MM1K(lambda=12, mu=15, k=2))$W
[1] 0.096

> # Vjerojatnost da će u sustavu reda čekanja biti 0 zahtjeva za printanje respektivno
> QueueingModel(NewInput.MM1K(lambda=12, mu=15, k=2))$Pn
[1] 0.41   0.33   0.26

> #ispiži prosječno vrijeme čekanja u redu za printanje
> QueueingModel(NewInput.MM1K(lambda=12, mu=15, k=2))$Wq
[1] 0.03
```

Dakle koeficijent iskorištenja je 59%, prosječan broj zahtjeva za printanje u sustavu je 0.85, prosječno vrijeme čekanja na printanje u sustavu je 0.096 sati, vjerojatnost da u sustavu neće biti stranki je 41% i prosječno vrijeme čekanja u redu za printanje je 0.03 sati.

Na slici 16 je prikazano kako dolazimo do rješenja u alatu RStudio Cloud. Plavim okvirom je označen kod programa (namjerno su izostavljeni komentari), crvenim okvirom je označen izlaz, odnosno rješenje.



Slika 16 Prikaz rješenja primjera 8 u RStudiu

Primjer 9: problem stanice

U svrhu primjera, zamislimo da imamo sustav sa 4 stanice opsluživanja u kojem je intenzitet dolazaka 18, a intenzitet opsluživanja 6 stranaka u jedinici vremena. Odredite koeficijent iskorištenja (faktor usluživanja), vjerojatnosti da će u sustavu reda čekanja biti 0, 1, 2 i 3 stranki, prosječan broj stranaka u redu čekanja ,prosječan broj stranaka u sustavu opsluživanja, prosječno vrijeme čekanja u redu i prosječno vrijeme čekanja u sustavu opsluživanja. [13].

Iz primjera primjećujemo da se radi o višekanalnom redu čekanja, pa ćemo koristiti MMC model. Rješenje ćemo postići sljedećim kodom, komentarima je naznačeno što se računa:

```
#naredbe za instalaciju i pokretanje
install.packages("queueing")
library(queueing)

#ispiši koeficijent iskorištenja (faktor usluživanja)
QueueingModel(NewInput.MMC(lambda=18, mu=6, c=4))$RO

#Vjerojatnost da će u sustavu reda čekanja biti 0, 1, 2, 3 #stranke respektivno
QueueingModel(NewInput.MMC(lambda=18, mu=6, n=3, c=4))$Pn

#ispiši prosječan broj stranaka u redu čekanja
QueueingModel(NewInput.MMC(lambda=18, mu=6, c=4))$Lq

#ispiši prosječan broj stranaka u sustavu opsluživanja
QueueingModel(NewInput.MMC(lambda=18, mu=6, c=4))$L

#ispiši prosječno vrijeme čekanja u redu
QueueingModel(NewInput.MMC(lambda=18, mu=6, c=4))$Wq

#ispiši prosječno vrijeme čekanja u sustavu opsluživanja
QueueingModel(NewInput.MMC(lambda=18, mu=6, c=4))$W
```

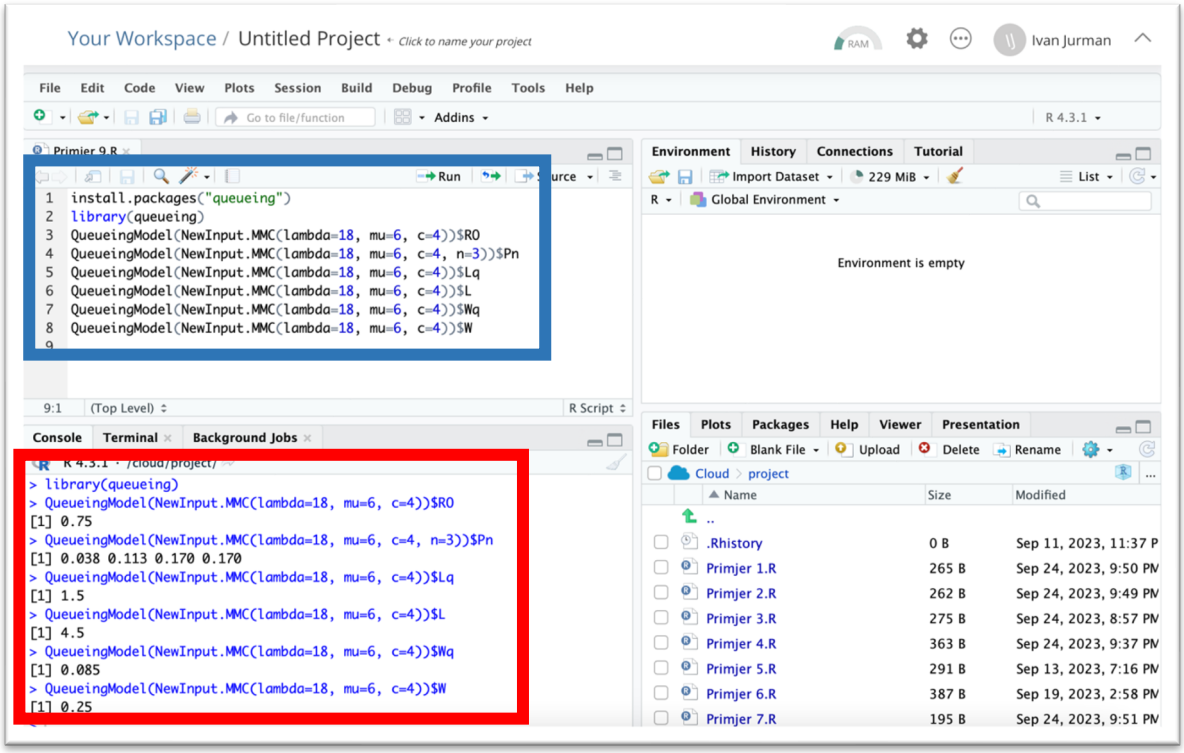
Po izvršenju koda, dobivamo sljedeće izlaze, odnosno rezultate:

```
> #ispiši koeficijent iskorištenja (faktor usluživanja)
> QueueingModel(NewInput.MMC(lambda=18, mu=6, c=4))$RO
> [1] 0.75

> #Vjerojatnost da će u sustavu reda čekanja biti 0, 1, 2, 3 stranke respektivno
> QueueingModel(NewInput.MMC(lambda=18, mu=6, n=3, c=4))$Pn
> [1] 0.038.  0.113   0.170   0.170

> #ispiši prosječan broj stranaka u redu čekanja
```

```
> QueueingModel(NewInput.MMC(lambda=18, mu=6, c=4))$Lq
> [1] 1.5

> #ispiši prosječan broj stranaka u sustavu opsluživanja
> QueueingModel(NewInput.MMC(lambda=18, mu=6, c=4))$L
> [1] 4.5

> #ispiši prosječno vrijeme čekanja u redu
> QueueingModel(NewInput.MMC(lambda=18, mu=6, c=4))$Wq
> [1] 0.085

> #ispiši prosječno vrijeme čekanja u sustavu opsluživanja
> QueueingModel(NewInput.MMC(lambda=18, mu=6, c=4))$W
> [1] 0.25
```

Dakle koeficijent iskorištenja je 75%, vjerojatnost da u sustavu neće biti stranki je 3.8%, da će u sustavu biti jedna stranka je 11.3% i da će biti dvije stranke je 17% i da će biti tri stranke je 17%. Prosječan broj stranaka u redu čekanja je 1.5, prosječan broj stranaka u sustavu opsluživanja je 4,5, prosječno vrijeme čekanja u redu je 0.085 jedinica vremena i prosječno vrijeme čekanja u sustavu opsluživanja je 0.25 jedinica vremena,

Na slici 17 je prikazano kako dolazimo do rješenja u alatu RStudio Cloud. Plavim okvirom je označen kod programa (namjerno su izostavljeni komentari), crvenim okvirom je označen izlaz, odnosno rješenje.



Slika 17 Prikaz rješenja primjera 9 u RStudiu

3.4. Teorija zaliha

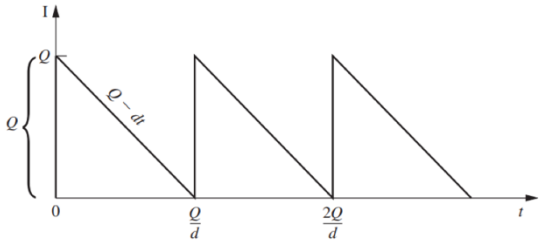
Teorija zaliha bavi se problemom optimizacije ukupnih troškova zaliha. Osnovna pitanja na koja teorija zaliha pokušava odgovoriti jesu „koliko naručiti?“, „kada naručiti?“ i „kada revidirati zalihe?“. Razlozi za držanje zaliha su npr. smanjenje vremena reagiranja na zahtjeve kupca, ostaci od sezonskih zaliha za zadovoljenje potražnje u sljedećoj sezoni, nesigurna procjena ponude i potražnje, ekonomija razmjera, posebne ponude za robu kojoj se vrijednost smanjuje s vremenom [14].

Osnovni ciljevi optimizacije su sljedeći:

- minimizacija troškova držanja zaliha, u oznaci: h
- minimizacija troškova naručivanja, u oznaci: K
- minimizacija troškova manjka, odnosno nezadovoljene potražnje, u oznaci: p
    - dijeli se na nezadovoljenu potražnju sa zaostatkom (takva se može nadoknaditi u sljedećem periodu bez novčanog gubitka) i na nezadovoljenu potražnju bez zaostatka (vodi ka gubitku u promatranom periodu)
- minimizacija ili maksimizacija likvidacijske vrijednosti viškova ovisno o dva slučaja
    - slučaj prodaje viškova, najčešće s popustom pri čemu je likvidacijska vrijednost pozitivna, odnosno jest prihod
    - slučaj zbrinjavanja viškova zbog nemogućnosti prodaje pri čemu je likvidacijska vrijednost negativna odnosno je trošak

Osnovni model ekonomične količine nabave:

- zalihe ($\boldsymbol{I}$) koje se troše tokom vremena obnavljaju se sa jednokratnom isporukom stanovite količine $\boldsymbol{Q}$
- poznata je i konstantna stopa potražnje ($\boldsymbol{d}$ komada u jedinici vremena)
- količina $\boldsymbol{Q}$ potrebna za nadomjestak zaliha dolazi instantno, čim se zalihe istroše (vrijeme isporuke, u oznaci t - jest konstantno)
- planirani manjak u osnovnom modelu nije dozvoljen



Slika 18 Osnovni model ekonomične količine nabave

Matematički model osnovne ekonomične količine nabave:

$Q*$ predstavlja jednokratnu isporuku količine robe kojom se minimiziraju ukupni troškovi, tj. ekonomičnu količinu nabave, a formula glasi:

$$Q^* = \sqrt{\frac{2dK}{h}}$$

gdje je *d* potražnja.

Odgovarajuće optimalno vrijeme ciklusa naručivanja određuje se formulom:

$$t^* = \sqrt{\frac{2K}{dh}}$$

Optimalni ukupni troškovi nabave u promatranom periodu, u smislu minimizacije troškova upravljanja zaliha, određuju se jednadžbom:

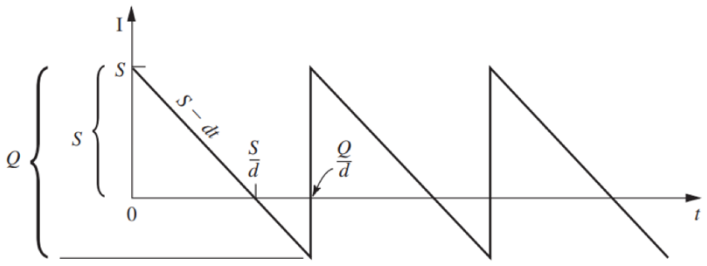
$$TC^* = \frac{Q^*}{2}h + \frac{d}{Q^*}K$$

gdje prvi dio jednadžbe predstavlja ukupan trošak držanja zaliha za period, a drugi dio ukupan trošak naručivanja za period.

Model ekonomične količine nabave s planiranim manjkom:

Iste pretpostavke kao kod bazičnog modela, uz uključenje dodatnih parametara:

- trošak manjka (shortage, underage ili penalty cost), p - razina zaliha nakon dodavanja $\boldsymbol{Q}$ količine proizvoda, S
- manjak kod zaliha neposredno prije dodavanja jednokratne isporuke stanovite količine (batch-a ili lot-a), $Q - S$



Slika 19 Model ekonomične količine nabave s planiranim manjkom

Krajnja matematička formulacija za model ekonomične količine nabave s planiranim manjkom je:

$$Q^* = \sqrt{\frac{2dK}{h}} + \sqrt{\frac{p}{p+h}} \qquad S^* = \sqrt{\frac{2dK}{h}} + \sqrt{\frac{p+h}{p}}$$

gdje $Q*$ predstavlja jednokratnu isporuku količine robe kojom se minimiziraju ukupni troškovi, tj. ekonomičnu količinu nabave, dok $S*$ predstavlja razinu zaliha nakon dodavanja optimalne naručene količine.

Odgovarajuće optimalno vrijeme ciklusa naručivanja određuje se formulom:

$$t^* = \frac{Q^*}{d}\sqrt{\frac{2K}{dh}}\sqrt{\frac{p+h}{p}}$$

Optimalni troškovi nabave u promatranom periodu, u smislu minimizacije troškova upravljanja zaliha:

$$TC = \frac{dK}{Q^*} + \frac{hS^{*2}}{2Q^*} + \frac{p(Q^* - S^*)^2 d}{2Q^*}$$

Maksimalni manjak uz optimalne prethodne rezultate određen je jednadžbom:

$$Q^* - S^* = \sqrt{\frac{2dK}{p}} \sqrt{\frac{h}{p+h'}}$$

Dok je dio vremena u kojem ne postoji manjak zaliha određen jednadžbom:

$$\frac{S^*/d}{Q^*/d} = \frac{p}{p+h}$$

3.4.1. Implementacija u R-u

Kod rješavanja problema iz teorije zaliha, potrebno je znati sljedeće parametre:

- Stopu potražnje **d**
- Troškove naručivanja **k**
- Troškove držanja zaliha **h**

Da bi mogli koristiti funkciju za računanje potrebno je najprije instalirati i pokrenuti paket *SCperf,* što činimo na sljedeći način:

```
install.packages("SCperf")
library(SCperf)
```

Nakon što smo instalirali potreban paket izračun je vrlo jednostavan, obavlja se pozivom funkcije *EOQ* u koju unosimo već nam poznate parametre. U slučaju osnovnog modela ekonomične količine nabave to su parametri *d, k* i *h,* to činimo na sljedeći način:

```
EOQ(d,k,h)
```

Dok u slučaju modela ekonomične količine nabave s planiranim manjkom potrebno je znati i parametar *p,* odnosno trošak manjka. Poziv funkcije je identičan osnovnom modelu samo što dodajemo i parametar *p*, a to činimo na sljedeći način:

```
EOQ(d,k,h,p)
```

3.4.2. Primjer

U svrhu primjera zamislimo da se trgovačko društvo X bavi preprodajom kiselog kupusa kao pakiranih glavica kupusa u salamuri. Naručuje ih od drugog trgovačkog društva proizvođača po cijeni od 10 kn za pakiranje, a preprodaje po 30 kn. ABC su izračunali da im je prosječna mjesečna potražnja kupusom 1500 paketa, te da su troškovi držanja glavica kupusa 1.5% njihove nabavne cijene. Svako izdavanje narudžbe prema dobavljaču ABC košta 15 kuna [14].

Želimo izračunati:

- ekonomičnu količinu nabave
- optimalno vrijeme ciklusa naručivanja
- ukupan trošak zaliha uz ekonomičnu količinu nabave

Dakle iz primjera možemo iščitati potrebne nam parametre koji glase:

d = 1500
K = 15
h = 10*1.5% = 0.15

Rješenje ćemo dobiti sljedećim kodom:

```
#naredbe za instalaciju i pokretanje
install.packages("SCperf")
library(SCperf)

#ispiši parametre osnovnog modela ekonomične količine nabave
EOQ(1500, 15, 0.15)
```

Dobiveni izlaz funkcije, odnosno rješenje izračuna je sljedeće:

```
> #ispiši parametre osnovnog modela ekonomične količine nabave
> EOQ(1500, 15, 0.15)
      Q           T          TVC
547.72        0.37       82.16
```

Odnosno optimalna količina nabave iznosi 547, optimalno vrijeme obnove zaliha je svakih 11 dana (0.37*30 dana) i s takvom optimalnom politikom naručivanja trgovačko društvo X će bilježiti minimalne ukupne mjesečne troškove zaliha od 82.16 kuna.

Da bi pokazali i primjer računanja modela ekonomične količine nabave s planiranim manjkom zamislimo da trgovačko društvo X nadalje preprodaje glave kupusa od dobavljača isključivo velikom trgovačkom lancu Y, kojemu X mora isplatiti penale u slučaju nemogućnosti isporuke robe, u visini od 10% prodajne cijene glave kupusa [14].

Želimo izračunati:

- ekonomičnu količinu nabave (EOQ)
- maksimalni manjak obzirom na (EOQ)
- optimalno vrijeme ciklusa naručivanja
- ukupan trošak zaliha uz ekonomičnu količinu nabave

Dakle svi su parametri isti kao i u prvom dijelu primjera, samo što dodatno imamo još i trošak manjka koji glasi:

$$p = 30.00*10\% = 3$$

U ovom slučaju jedina razlika je što u funkciju dodajemo parametar *p*, na sljedeći način:

```
#naredbe za instalaciju i pokretanje
install.packages("SCperf")
library(SCperf)
```

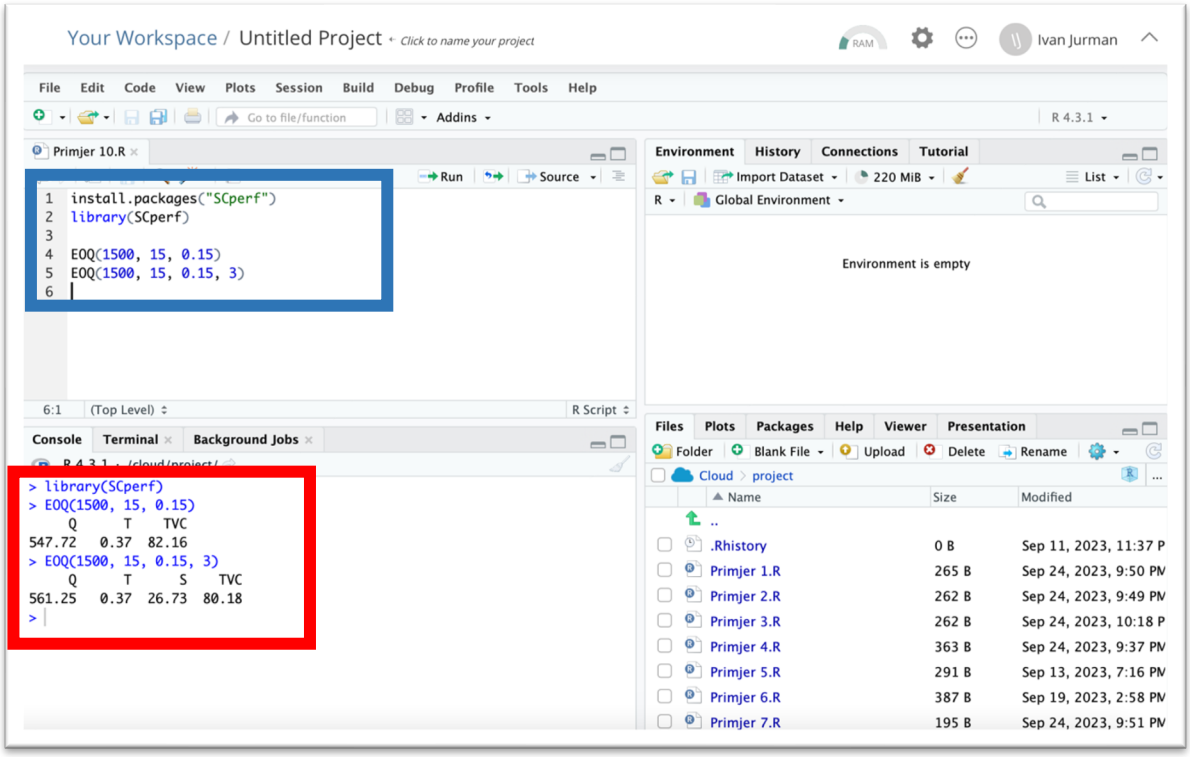
```
#ispiši parametre modela ekonomične količine nabave s planiranim manjkom
EOQ(1500, 15, 0.15, 3)
```

Dobiveni izlaz funkcije, odnosno rješenje izračuna je sljedeće:

```
> #ispiši parametre modela ekonomične količine nabave s planiranim manjkom
> EOQ(1500, 15, 0.15, 3)
      Q         T         S       TVC
561.25      0.37     26.73     80.18
```

Odnosno optimalna količina nabave iznosi 561, optimalno vrijeme obnove zaliha je svakih 11 dana (0.37*30 dana) i s takvom optimalnom politikom naručivanja trgovačko društvo X će bilježiti minimalne ukupne mjesečne troškove zaliha od 80.18 kuna, uz maksimalni dozvoljeni manjak od 27.

Na slici 20 je prikazano kako dolazimo do rješenja u alatu RStudio Cloud. Plavim okvirom je označen kod programa (namjerno su izostavljeni komentari), crvenim okvirom je označen izlaz, odnosno rješenje.



Slika 20 Prikaz rješenja primjera 10 u RStudiu

4. Zaključak

Budući da programski jezik R ima puno raznih mogućnosti te neke od njih su i rješavanje problema iz operacijskih istraživanja. Ovim radom smo pokrili probleme linearnog programiranja, redova čekanja i teorija zaliha. Budući da je sam jezik prilično jednostavan, rješavanje takvih problema u jeziku R je zaista jednostavno i lako za korištenje te kao takav smatram da je odlična alternativa kod rješavanja problema iz operacijskih istraživanja. Smatram da se ovim radom pokrilo dovoljno primjera iz različitih mogućih scenarija za svaki problem, pa tako iz linearnog programiranja imamo tri primjera s različitim brojem varijabli i uvjeta kako bi prikazali kako se program prilagođava tim različitim scenarijima, kod cjelobrojnog linearnog programiranja imamo primjer za sva tri moguća scenarija, odnosno čistog cjelobrojnog, binarnog i mješovitog cjelobrojnog linearnog programiranja. Isto tako iz teorije redova čekanja pokrili smo jednokanalne redove čekanja s ograničenom i neograničenom duljinom reda kao i višekanalni red čekanja s neograničenom duljinom reda. Također uključen je i primjer iz teorije zaliha koji uključuje osnovni model ekonomične količine nabave kao i model ekonomične količine nabave s planiranim manjkom. Budući da su primjeri jednostavni, lako razumljivi i mogu poslužiti kao šablona za gotovo svaki scenarij siguran sam da bi ovaj rad poslužio kao dobar priručnik svima koji žele riješiti neki od pokrivenih problema iz operacijskih istraživanja u jeziku R, pa bili oni studenti, netko treći sa zanimanjem za operacijska istraživanja ili netko tko jednostavno želi optimizirati vlastiti obrt i sl.

Literatura

[1] Martina Holenko Dlab, „Uvod u operacijska istraživanja“, Fakultet informatike i digitalnih tehnologija, Sveučilište u Rijeci, 2022. dostupno na webu: https://moodle.srce.hr/2021-2022/pluginfile.php/5327724/mod_resource/content/3/1_Uvod_u_OI.pdf [posljednji pristup: 15. rujna 2023.]

[2] „What are the free softwares available for solving operation research problems?“, Quora, dostupno na webu: https://www.quora.com/What-are-the-free-softwares-available-for-solving-operation-research-problems [posljednji pristup: 18. rujna 2023.]

[3] „What is R“, dostupno na webu: https://www.r-project.org/about.html [posljednji pristup: 1. rujna 2023.]

[4] „Contributors“, dostupno na webu: https://www.r-project.org/contributors.html [posljednji pristup: 1. rujna 2023.]

[5] Justin Chia, „5 Best R IDE & Editors (Ranked & Reviewed for 2023)“, https://justjooz.com/best-r-ides/ [posljednji pristup: 11. rujna 2023.]

[6] Rebeka Čordaš, „Linearno programiranje i primjene“, diplomski rad, Sveučilište J.J.Strossmayera u Osijeku, dostupno na webu: http://www.mathos.unios.hr/~mdjumic/uploads/diplomski/ČOR03.pdf [posljednji pristup: 2. rujna 2023.]

[7] Fredrick S. Hilier, Gerald J. Lieberman, „Introduction to operations research“, 9. izdanje, ISBN 978-0-07-337629-5

[8] Martina Holenko Dlab, Dino Pitoski, „Rješavanje problema linearnog programiranja u R-u“, Fakultet informatike i digitalnih tehnologija, Sveučilište u Rijeci, 2022. dostupno na webu: https://moodle.srce.hr/2021-2022/pluginfile.php/6174829/mod_resource/content/0/Linearno_programiranje_R.pdf [posljednji pristup: 2. rujna 2023.]

[9] Martina Holenko Dlab, „Uvod u linearno programiranje“, Fakultet informatike i digitalnih tehnologija, Sveučilište u Rijeci, 2022 dostupno na webu: https://moodle.srce.hr/2021-2022/pluginfile.php/5327740/mod_resource/content/3/2_Uvod_u_linearno_programiranje.pdf [posljednji pristup: 2. rujna 2023.]

[10] „Linearno programiranje“, Fakultet kemijskog inženjerstva i tehnologije, Sveučilište u Zagrebu, dostupno na webu: http://matematika.fkit.hr/staro/izborna/referati/Daniela%20Petkovicek%20-%20Linearno%20programiranje.pdf [posljednji pristup: 12. rujna 2023.]

[11] Martina Holenko Dlab, „Cjelobrojno programiranje“, Fakultet informatike i digitalnih tehnologija, Sveučilište u Rijeci, 2022 dostupno na webu: https://moodle.srce.hr/2021-2022/pluginfile.php/6248459/mod_resource/content/1/2_Cjelobrojno_programiranje_1.pdf [posljednji pristup: 2. rujna 2023.]

[12] Tea Crnobrnja, „Teorija redova čekanja“, diplomski rad, Sveučilište J.J.Strossmayera u Osijeku, dostupno na webu: http://www.mathos.unios.hr/~mdjumic/uploads/diplomski/CRN16.pdf [posljednji pristup: 5. rujna 2023.]

[13] Martina Holenko Dlab, Dino Pitoski, „Primjeri rješavanja problema redova čekanja alatom R („Quewing“ paket)“, Fakultet informatike i digitalnih tehnologija, Sveučilište u Rijeci, 2022 dostupno na webu: https://moodle.srce.hr/2021-2022/pluginfile.php/6305367/mod_resource/content/1/Uputstva_redovi__ekanja_R__queueing_package_%20%281%29.pdf [posljednji pristup: 6. rujna 2023.]

[14] Dino Pitoski, „Teorija zaliha“, Fakultet informatike i digitalnih tehnologija, Sveučilište u Rijeci, 2022 dostupno na webu: https://moodle.srce.hr/2021-2022/pluginfile.php/6343102/mod_resource/content/1/Teorija%20zaliha.pdf [posljednji pristup: 6. rujna 2023.]

Popis slika

Popis tablica