SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br 3955.

PROGRAMSKA KNJIŽNICA ZA PODRŠKU KORIŠTENJA STATISTIČKOG ALATA UNUTAR SIMULACIJE TRGOVANJA ELEKTRIČNOM ENERGIJOM

Filip Sakač

Zagreb, lipanj 2015.

Sadržaj

Uvod 1
1. Power TAC 2
1.1. Programski agenti 3
1.2. Uvođenje statističkih alata u programske agente 4
2. Programski jezik R 6
2.1. Korištenje jezika R unutar Jave 8
2.1.1. Programska knjižnica RCaller 9
2.1.2. Programska knjižnica rJava 10
2.1.3. Programska knjižnica Renjin 11
2.2. Usporedba performansi postojećih rješenja 13
3. Implementacija knjižnice za korištenje jezika R unutar Jave 17
3.1. Prijenos podataka između jezika R i Jave 18
3.1.1. Prijenos podataka u *R* 18
3.1.2. Prijenos podataka u *Javu* 20
3.2. Mogućnosti i ograničenja knjižnice RWrapper 21
3.3. Primjer upotrebe R skripte u PowerTAC agentu 22
Zaključak 24
Literatura 25
Sažetak 26
Summary 27
Prilog 1 – funkcija Power TAC agenta koja koristi R skriptu 28
Prilog 2 – Upute za instalaciju 29

Uvod

Zbog razvoja tehnologija i sve većeg udjela obnovljivih izvora energije pojavila se potreba za naprednim elektroenergetskim mrežama (engl. *smart grid*). Takve elektroenergetske mreže trebat će i nove metode određivanja cijena i trgovanja električnom energijom, koje će poticati potrošnju energije iz obnovljivih izvora i prilagodbu korisnikove potrošnje stvarnom stanju tržišta električnom energijom. U svrhu istraživanja i simuliranja takvog tržišta električnom energijom razvijeno je natjecanje Power TAC.

Power Trading Agent Competition (*Power TAC*) je organiziran kao natjecanje u kojem se inteligentni programski agenti natječu za profit na liberaliziranom tržištu električne energije zasnovanom na naprednoj elektroenergetskoj mreži. Za rad agenta ključna je statistička obrada podatka kojih dobije od Power TAC poslužitelja. Na temelju tih podataka agent radi predviđanja o potrošnji svojih korisnika i cijenama na tržištu, te donosi odluke o svojim potezima. Zbog toga je agentu potrebno omogućiti korištenje naprednih statističkih alata.

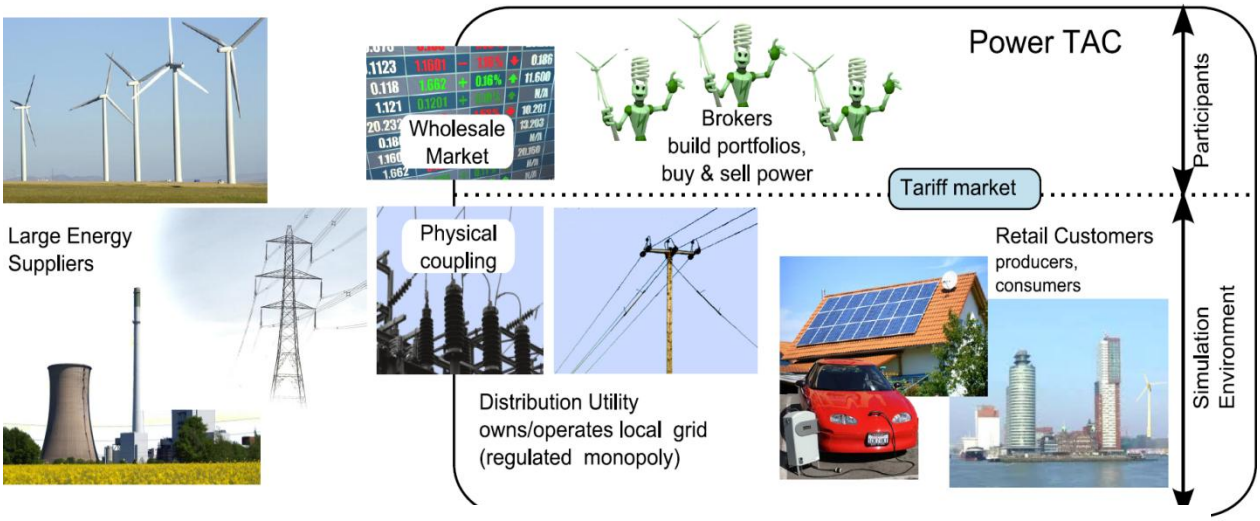
Tema ovog rada je povezivanje jezika *R*, koji je jedan od najpopularnijih jezika za statističku obradu podataka, s inteligentnim agentom iz natjecanja Power TAC pisanim u programskom jeziku *Java*. Zbog popularnosti jezika *R* i *Jave* već su razvijena mnoga rješenja koja omogućuju njihovu zajedničku upotrebu. U ovom radu će se usporediti performanse nekih od tih rješenja, te pobliže opisati *Renjin*, rješenje koje je korišteno u izradi rada. Rad će također opisati razvijenu programsku knjižnicu čiji je cilj olakšati programeru upotrebu *Renjina*. Knjižnica programeru omogućuje pozivanje skripti pisanih u jeziku *R*, te automatski obavlja prijenos i konverziju podataka iz *Jave* u *R* i obrnuto.

U prvom poglavlju dan je kratki uvod u natjecanje Power TAC i programske agente koji se u njemu natječu. Također će biti objašnjena motivacija za uvođenje statističkih alata u programske agente. U drugom poglavlju bit će opisan jezik *R,* i usporedit će se različita rješenja za povezivanje jezika *R* i *Jave*. Tema trećeg poglavlja je razvoj programske knjižnice za jednostavno korištenje skripti iz jezika *R* u agentu.

1. Power TAC

Power Trading Agent Competition (Power TAC) nastao je kao projekt šest sveučilišta iz Europe i Sjeverne Amerike među kojima je i Fakultet elektrotehnike i računarstva Sveučilišta u Zagrebu. Cilj projekta je simulirati tržište električne energije koje je liberalizirano. Pokušaj uvođenja takvog tržišta u Kaliforniji 2000. godine doveo je do velike energetske krize [1], zbog toga je postalo jasno da su potrebna dodatna istraživanja i simulacije kako bi se utvrdili zakonski okviri takvog tržišta. Power TAC je jedan ekonomičan način simuliranja takvih tržišta [2].

Glavni dijelovi Power TAC simulacije prikazani su na slici 1. Simulacija se sastoji od dva tržišta: veleprodajnog (engl. w*holesale market*) i tarifnog (engl. *tariff market*). Na veleprodajnom tržištu energiju prodaju veliki proizvođači energije, a na tarifnom tržištu brokeri nude kupcima tarife s različitim cijenama električne energije. Distributer električne energije je tvrtka koja posjeduje infrastrukturu elektroenergetske mreže. Glavni akteri na tržištima su inteligentni programski agenti koji rade kao brokeri. Oni kupuju i prodaju električnu energiju na veleprodajnom tržištu, i nude tarife na tarifnom tržištu.



Slika 1 Glavni dijelovi Power TAC simulacije [2]

1.1. Programski agenti

Programski agenti rade kao brokeri u natjecanju Power TAC i imaju za cilj ostvariti što veću zaradu jer je na kraju simulacije broker s najviše novca proglašen pobjednikom. Tijekom svakog vremenskog odsječka broker može:

- ponuditi nove tarife na tarifnom tržištu,
- zamijeniti postojeće tarife novima,
- mijenjati cijene korisnicima čija tarifa to dopušta,
- stvarati i prihvaćati ponude na veleprodajnom tržištu, te
- ponuditi usluge balansiranja.

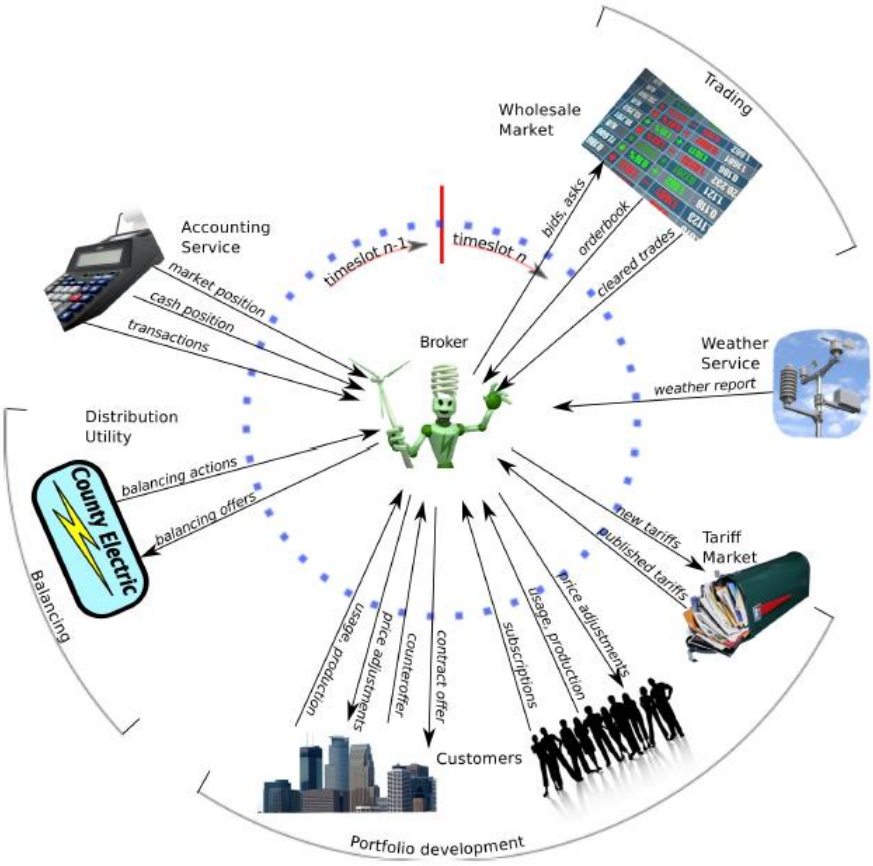
Broker svaki vremenski odsječak od Power TAC poslužitelja dobiva informacije o vremenskoj prognozi, stanju ponuda na veleprodajnom tržištu, potrošnji njegovih pretplatnika na tarifnom tržištu, njegovoj ukupnoj ponudi i potražnji, te njegovom rangu u natjecanju.

Za brokera je, osim pametnog trgovanja na tržištima, vrlo važno uravnotežiti njegovu sveukupnu ponudu i potražnju jer svaki višak ili manjak energije nekog brokera će pokriti distributer električne energije, po cijenama koje su vrlo nepogodne za brokera. Navedeno uravnoteženje broker može postići kupovinom i prodajom na veleprodajnom tržištu, podešavanjem cijena na tarifama, otkupljivanjem vlastitih proizvodnih kapaciteta ili korištenjem nekog sustava pohrane.

Pošto je potrošnja električne energije korisnika promjenjiva, kao i proizvodnja energije iz obnovljivih izvora, da bi broker osigurao potrošačima potrebnu količinu energije i postigao balans svoje ponude i potražnje, potrebne su mu kvalitetne metode predviđanja potrošnje i proizvodnje električne energije.

1.2. Uvođenje statističkih alata u programske agente

Programski agent u natjecanju Power TAC stalno razmjenjuje poruke s poslužiteljem. Različite vrste poruka koje agent prima od poslužitelja i akcije koje agent može poduzeti prikazane su na slici 2.



Slika 2 Poruke kojima agent komunicira s Power TAC okolinom [2]

Na slici se vidi da agent raspolaže s velikom količinom podataka, na temelju kojih mora donositi odluke. Postavlja se pitanje kako obraditi te podatke. Prva mogućnost je da agenti obrađuju podatke koristeći programski jezik *Java* jer je to jezik u kojima su oni napisani. U ovom radu istražuje se mogućnost korištenja jezika *R* za statističku obradu podataka agenta.

Java je programski jezik široke namjene, što znači da se može koristiti za pisanje programa za različite domene primjene. *Java* dolazi s klasom *Math* u kojoj su implementirane neke osnovne matematičke funkcije, ali nema funkcija za statistiku. Zbog toga se za statistiku u *Javi* moraju koristiti vanjske knjižnice, ili se

željene funkcije moraju samostalno implementirati. Primjeri knjižnica za statističku obradu podataka u Javi *su Apache Commons Math* [3] i *JSAT* [4].

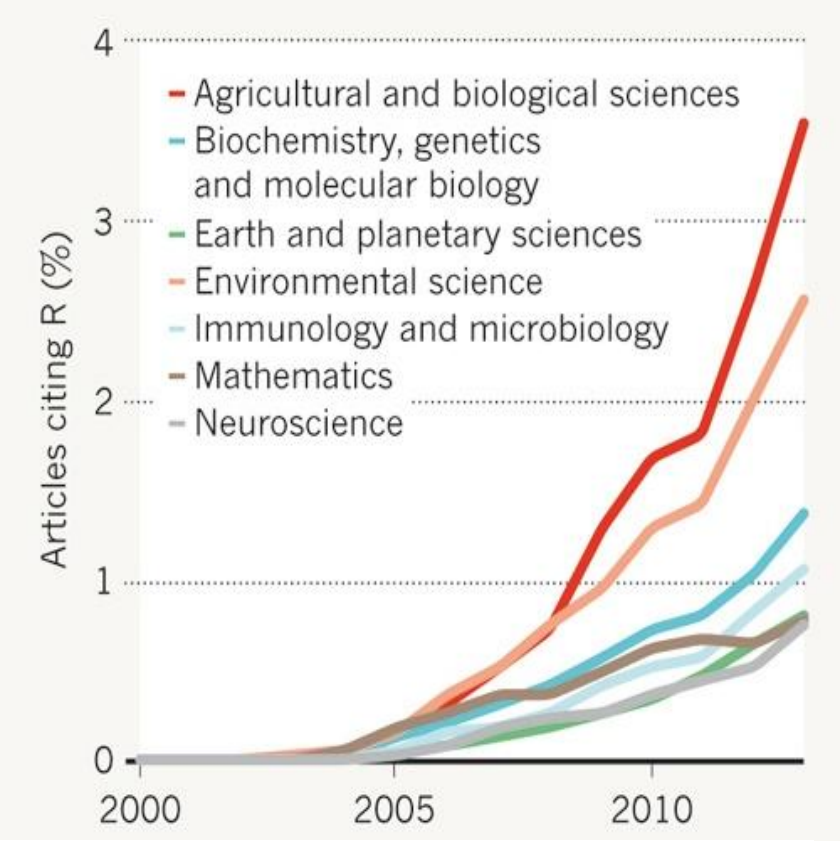
Prednost korištenja *Java* knjižnica za statistiku je jednostavnost njihovog integriranja s Power TAC agentom koji je također pisan u *Javi*. Druga prednost korištenja *Java* knjižnica je da ne zahtijevaju stvaranje vanjskih procesa i komunikaciju s njima, što loše utječe na performanse.

Problem korištenja *Java* knjižnica za statističku obradu podataka je manji broj implementiranih statističkih operacija koje je moguće pronaći u usporedbi s jezikom *R*. Također pisanje vlastitih funkcija za obradu podataka je lakše u jeziku koji je za to specijaliziran, nego u jeziku opće namjene. Ostale prednosti koje korištenje jezika *R* sa sobom donosi opisane su u drugom poglavlju ovog rada.

Konačno rješenje je kompromis između performansi i lakoće integracije u programskog agenta koje nudi *Java,* i količine implementiranih statističkih metoda koje nudi jezik *R.*

2. Programski jezik R

R je programski jezik namijenjen za statističku obradu i grafički prikaz podataka. Razvili su ga 1993. godine Ross Ihaka i Robert Gentleman sa Sveučilišta Auckland u Novom Zelandu, a danas ga održava *R* Development Core Team. *R* je besplatan i otvorenog koda te radi na svim važnijim operacijskim sustavima [5]. *R* je danas jedan od najpopularnijih alata za statističku obradu podataka [6]. Na slici 3 prikazan je graf postotka znanstvenih radova u kojima se spominje *R* po različitim granama znanosti. Na grafu se jasno vidi trend porasta u svim granama znanosti.



Slika 3 Graf citiranja jezika R u znanstvenim radovima [7]

R je interpretirani programski jezik. Korisnik ga najčešće koristi preko komandne linije ili tako da pokreće već napisane skripte, također postoje i grafička sučelja za rad u jeziku *R*. *R* podržava više programskih paradigmi uključujući proceduralnu, objektno orijentiranu i funkcionalnu paradigmu. Sintaksa jezika *R* vrlo je slična sintaksi jezika *S* koji je poslužio kao inspiracija u stvaranju jezika *R*.

Osnovne strukture podataka koje *R* podržava su vektori, matrice, polja, liste i okviri podataka, a oni se sastoje od elementarnih dijelova koji su: brojke (engl. *numeric*) koje predstavljaju decimalne brojeve, cijeli brojevi (engl. *integer*), kompleksni brojevi (engl. *complex*), logičke vrijednosti (engl. *logical*) i znakovi (engl. *character*).

R dolazi s puno implementiranih funkcija za statističku obradu podataka i stvaranje grafova, ali jedna od najvećih prednosti jezika R je njegova proširivost putem CRAN (*Comprehensive R Archive Network*) paketa. CRAN pakete izrađuju i dijele korisnici jezika *R* a sadrže funkcije i objekte korisne u nekoj domeni. Pakete je moguće preuzeti s jedne od stanica u CRAN mreži koje su održavane od strane fakulteta diljem svijeta. U toj mreži trenutno postoji čak 6695 paketa, i taj broj neprekidno raste [8].

Jedan od glavnih razloga popularnosti jezika *R* sigurno je njegova mogućnost stvaranja kvalitetnih grafova. *R* omogućava korisnicima crtanje raznih vrsta grafova, a njegove mogućnosti se mogu proširiti korištenjem paketa kao što su *Lattice* i *ggplot2.*

2.1. Korištenje jezika R unutar Jave

Java je objektno orijentirani programski jezik opće namjene. Moto *Jave* je "napiši jednom, pokreni svugdje" (engl. *write once, run anywhere*) što znači da je prenosivost jedan od glavnih ciljeva *Jave*. *Java* postiže prenosivost pomoću svojeg virtualnog stroja (engl. *java virtual machine*) koji pokreće Javine programe. Zbog toga je *Java* implementirana na mnogim platformama i danas je jedan od najkorištenijih programskih jezika [9].

Popularnost *Jave* dovela je i do potrebe za statističkom obradom podataka u *Javi*. Iako postoje programske knjižnice za statistiku u *Javi*, nema ih toliko koliko ih ima za *R*, koji je ipak specijalno namijenjen za statistiku i koriste ga mnogi stručnjaci iz raznih znanstvenih područja. Zato u mnogim programskim jezicima postoje načini za upotrebu jezika *R*. Neki primjeri su *rpy2* za *Python* i *R.NET* za *C#*. Također postoje i obrnuta rješenja, paketi koji omogućuju komunikaciju iz jezika *R* prema drugim programskim jezicima kao što su na primjer *rJava* i *rPython*.

U tablici 1 su sažeto opisane prednosti i nedostatci pojedinih rješenja za korištenje jezika *R* iz konteksta *Jave*, a u nastavku ovog poglavlja je detaljnija analiza svakog od tih rješenja.

Tablica 1 Prednosti i mane rješenja za povezivanje jezika R i Jave

	Prednosti	Nedostatci
RCaller	Jednostavno postavljanje i korištenje, u potpunosti kompatibilan s jezikom *R*	Jako loše performanse, potrebno je imati instaliran *R*
rJava	Najbolje performanse, u potpunosti kompatibilan s jezikom *R*, callback metode	Komplicirano postavljanje, potrebno je imati instaliran *R*
Renjin	Nije potrebno imati *R* instaliran, jednostavno postavljanje i korištenje, dobre performanse	Nije u potpunosti kompatibilan s jezikom *R,* ne podržava sve CRAN pakete

2.1.1. Programska knjižnica RCaller

RCaller je programska knjižnica za *Javu* koja omogućuje pozivanje naredbi jezika *R* iz *Jave*. *RCaller* je otvorenog koda. Glavna prednost *RCallera* je jednostavnost korištenja. *RCaller* automatski pretvara strukture podataka iz *Jave* kao što su polja primitivnih tipova podataka i jednostavni objekti (bez funkcija) u *R* kod. Podatke i naredbe *RCaller* šalje vanjskom *R* procesu, a rezultati tog procesa vraćeni su u XML obliku koje je onda moguće parsirati i koristiti u *Javi* [10].

Za korištenje *RCallera* potrebno je imati instaliran *R* na računalu i `.jar` datoteku koju je moguće preuzeti sa stranice ili ju sam stvoriti iz koda kojeg je moguće naći na stranici *GitHub*. Nakon toga potrebno je tu datoteku uključiti u svoj projekt. Pokretanje *R* naredbi radi se tako da se prvo instancira novi objekt klase `RCaller` i pozove njegova metoda `setRscriptExecutable` koja kao parametar prima niz znakova (engl. *string*) koji predstavlja put do izvršne datoteke kojom se pokreće *R*. Nakon toga se instancira novi objekt klase `Rcode`. `Rcode` objekt ima metode za dodavanje varijabli iz *Jave* koje automatski pretvara u *R* kod, i metodu za dodavanje naredbi jezika *R* u obliku niza znakova. Kada se sav željeni kod i sve potrebne varijable dodaju tada se poziva metoda `setRCode` objekta `Rcaller`, nakon toga kod je moguće pokrenuti na dva načina: metoda `runAndReturnResult` kod svakog poziva pokreće novi *R* proces, dok metoda `runAndReturnResultOnline` sve pozive izvršava na istom *R* procesu pa je manje vremenski zahtjevna. Obje metode kao parametar primaju ime varijable koju žele dobiti natrag kao rezultat. Ako se koristi online metoda, na kraju programa potrebno je pozvati metodu `stopStreamConsumers` da bi se završio *R* proces.

Za dobivanje rezultata potrebno je od objekta klase `RCaller` dobiti parser i zatim pozvati njegovu odgovarajuću metodu ovisno o tipu podatka u koji je potrebno pretvoriti rezultat (na primjer ako se rezultat zove *a*, i on je matrica decimalnih brojeva dimenzija dva puta dva, koristi se metoda `getAsDoublematrix("a",2,2)`).

Glavni nedostatak *RCallera* su njegove performanse, koje su zbog pretvorbe tipova podataka preko XML-a dosta spore u usporedbi s drugim sličnim rješenjima.

2.1.2. Programska knjižnica rJava

rJava je CRAN paket kome je glavna zadaća omogućiti korištenje *Jave* unutar jezika *R*, ali također omogućuje i suprotni smjer preko *JRI* (Java/R interface) [11]. Instalacija i upotreba *rJave* je nešto kompliciranija, prvo je potrebno imati instaliran *R* na računalu, zatim ga pokrenuti i u njemu izvršiti naredbu `install.packages("rJava")`. Kada R instalira CRAN paket potrebno je pokrenuti naredbu `system.file("jri",package="rJava")` koja će vratiti put do mape gdje je instaliran *JRI*. U toj mapi nalazit će se tri `.jar` datoteke koje je potrebno dodati u *Java* projekt u kojem se želi koristiti *R*, i run skripta koja služi za pokretanje programa koji koriste *rJavu*.

Osnovna klasa u *rJavi* je `Rengine`, njezin konstruktor prima tri argumenta, u prvi argument se obično proslijede argumenti koje *Java* program dobije od konzole, drugi argument je logička vrijednost koja određuje dali se odmah koristi glavna *R* petlja, obično to nije potrebno pa se postavi na false, treći argument prima objekt koji implementira `RmainLoopCallbacks` sučelje.

`RmainLoopCallbacks` sučelje sadrži osam funkcija koje se zovu kada se u *R* dogode određeni događaji, na primjer kada *R* dobije podatke na svoj ulaz. Za jednostavne primjene te funkcije ne moraju ništa raditi.

Nakon što se instancira `Rengine` objekt potrebno je pozvati njegovu metodu `waitForR` u kojoj se čeka dok se ne pokrene *R* proces. Ta metoda vraća istinitu logičku vrijednost ako je proces uspješno pokrenut. Nakon toga mogu se koristiti metode `Rengine` objekta `assign` i `eval`. `Assign` metoda služi za prijenos podatka iz *Jave* u *R*, prima dva argumenta: prvi je niz znakova koji predstavlja ime koje će varijabla imati u jeziku *R*, a drugi mora biti polje nekog primitivnog tipa koje sadrži podatke koji će se prenijeti u *R*. `eval` metoda prima niz znakova koji predstavljaju naredbu jezika *R*, opcionalno može se dodati i drugi parametar koji je logička vrijednost i određuje hoće li se prenositi rezultat naredbe natrag u *Javu*.

Funkcija `eval` vraća objekt tipa `REXP` koji predstavlja sve strukture podataka iz jezika *R*. `REXP` objekt ima različite metode kojima se može pretvoriti u jednodimenzionalna ili dvodimenzionalna *Java* polja. `REXP` objekt ima i `toString` metodu koja ispisuje podatak onako kako ga ispisuje *R*.

Programe u kojima se koristi *rJava* ne mogu se pokrenuti kao ostali *Java* programi. Prije pokretanja programa potrebno je pravilno postaviti varijable okoline, za tu svrhu koristi se run skripta koja dolazi s instalacijom *rJave*. U run skripti potrebno je podesiti classpath varijablu na korijenski direktorij projekta.

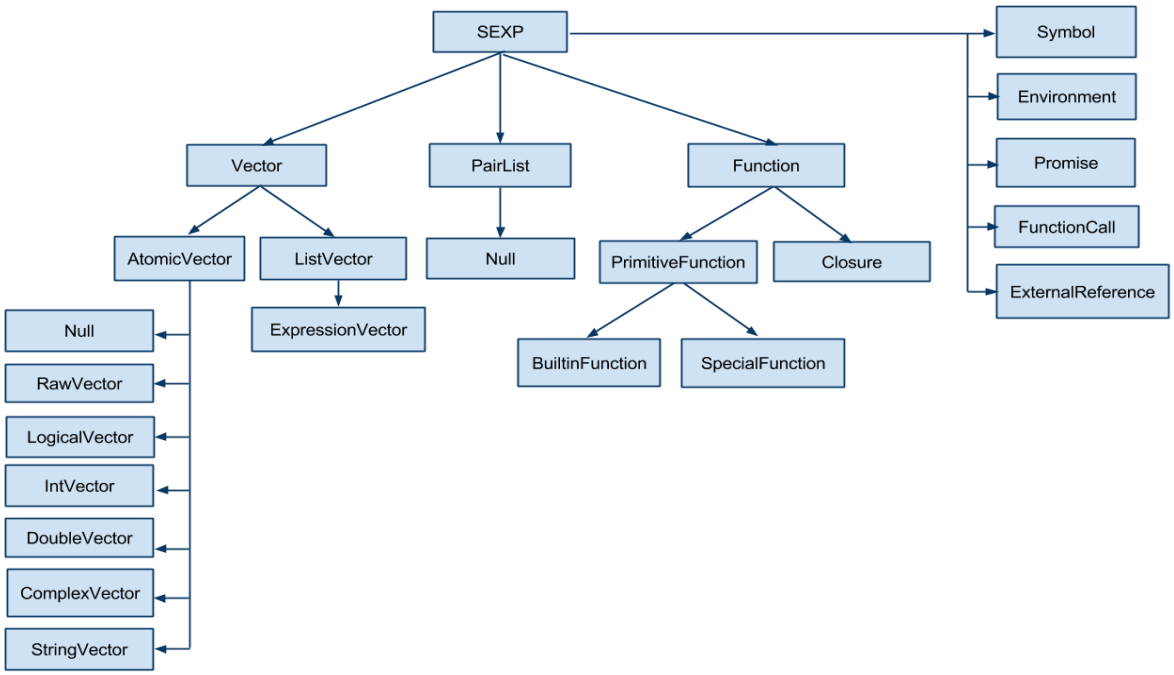
rJava ima bolje performanse od *Rcallera*, i nudi više mogućnosti s callback metodama, ali podešavanje i pokretanje programa je dosta zahtjevnije. Također *rJava* omogućuje i obrnuti smjer, korištenje objekata iz *Jave* u programu pisanom jezikom *R*.

2.1.3. Programska knjižnica Renjin

Renjin je interpreter jezika *R* u potpunosti implementiran u *Javi* [12]. Za razliku od ostalih rješenja za upotrebu *Renjina* nije potrebno imati instaliran *R* na računalu. *Renjin* je otvorenog koda i razvija ga Nizozemska tvrtka *BeDataDriven*. *Renjin* i *R* nisu sasvim kompatibilni, postići sto postotnu kompatibilnost je krajnji cilj *Renjina*.

Za upotrebu *Renjina* potrebno je u projekt dodati *Renjin script engine* `.jar` datoteku. Renjinu se najčešće pristupa preko *Java* sučelja za skriptne programske jezike [13]. U programu je prvo potrebno instancirati objekt klase `ScriptEngineManager` i zatim pozvati njegovu metodu `getEngineByName` s nizom znakova *Renjin* kao argumentom. Ta metoda vraća objekt klase `ScriptEngine` koji omogućuje pokretanje naredbi i skripti jezika *R*.

Glavna metoda za korištenje naredbi jezika *R* je `eval`, toj metodi se predaju naredbe u obliku niza znakova, ili datoteke u kojoj su naredbe spremljene. Naredba `eval` vraća rezultate u obliku *Javine* natklase `Object`. Povratne vrijednosti funkcije `eval` se uvijek mogu pretvoriti u `SEXP` objekt. Klasa `SEXP` je natklasa za sve tipove podatka u *Renjinu*. *Renjin* ima klase za sve tipove podataka jezika *R*, njihova hijerarhija je prikazana na slici 4.



Slika 4 Hijerarhija tipova podataka u Renjinu [12]

Iz `SEXP` objekta se podaci dobivaju tako da ga se pretvori u odgovarajući objekt iz *Renjinove* hijerarhije ovisno o tipu podataka kojeg vraća pokrenuta *R* skripta. Tip podatka je moguće odrediti pozivom metode `getTypeName`. Nakon pretvaranja se pozivom metode `getElement` može pristupiti pojedinačnim elementima objekta po njihovom indeksu, ili se koristi metoda `toArray` koja pretvara *Renjinov* objekt u *Java* polje. Višedimenzionalna polja jezika *R* ne mogu se automatski pretvoriti u odgovarajuća polja za *Javu*.

Objekt klase `ScriptEngine` također nudi metode `get` i `put` koje se koriste za prijenos podataka iz *R* okoline u *Javu* i obrnuto. Metoda `get` kao argument traži niz znakova koji predstavlja ime varijable iz *R* okoline, a povratna vrijednost je tipa *Javine* nadklase Object iz kojeg se podaci dobivaju na već opisan način. `put` metoda prenosi podatke u *R*, i prima dva argumenta. Prvi argument je ime koje će podatak imati u *R* okolini, a drugi je sam podatak. Podatak mora biti primitivnog tipa, omotač primitivnog tipa (engl. *wrapper class*), niz znakova ili polje nekog od tih tipova da bi ga *Renjin* automatski pretvorio u odgovarajući tip jezika *R*. Ako objekt koji se prenosi u *R* nije tipa kojeg će *Renjin* automatski pretvoriti, on će ipak biti prenesen kao `externalptr`.

Objekt tipa `externalptr` može se u skriptama pisanim u jeziku *R* koristiti jednakom sintaksom kao da je *R* objekt. Za to je u *R* skripti potrebno naredbom `import` uključiti odgovarajuću *Java* kalsu. Nakon toga moguće je pozivati metode `extrnalptr` objekta jednakom sintaksom kao da je *R* objekt.

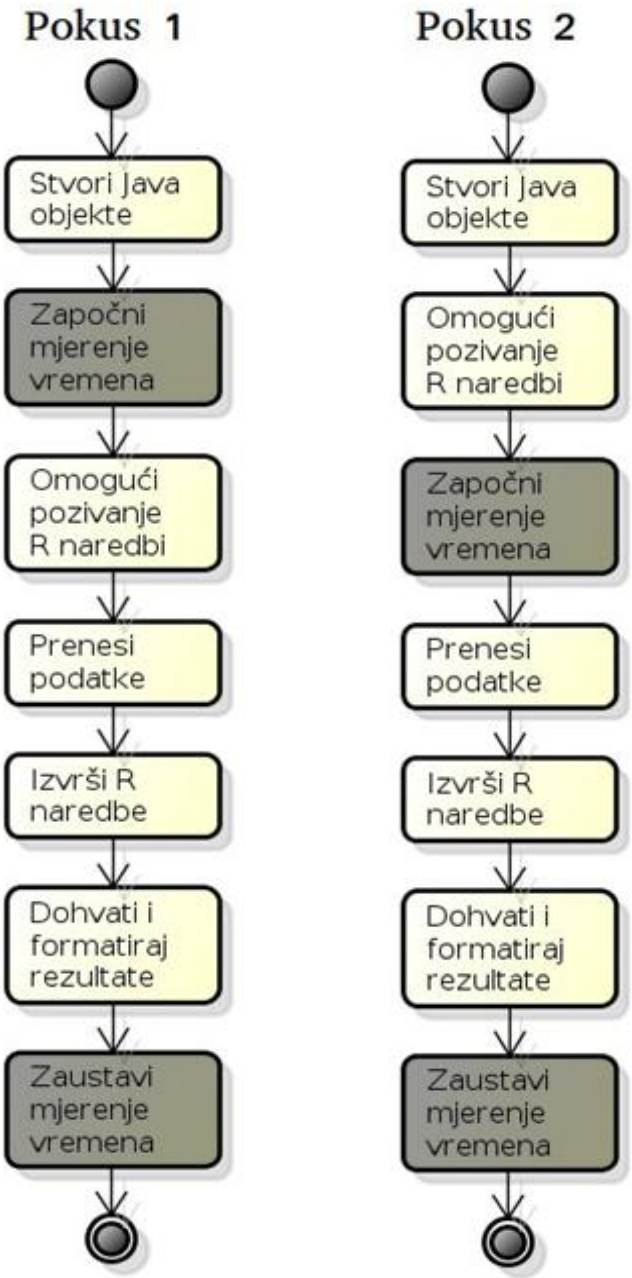
Glavna prednost *Renjina* je upravo njegova prenosivost. *Renjin* je u cijelosti napisan u *Javi*, pa se može koristiti svugdje gdje i *Java* bez bilo kakvih dodatnih instalacija i podešavanja. Zbog toga je *Renjin* odabran kao način povezivanja jezika *R* i *Jave* u ovom radu. *Renjinove* performanse su također vrlo dobre jer ne treba komunicirati s vanjskim *R* procesom, a zbog implementacije u Javi se neke naredbe kao što su programske petlje izvršavaju brže nego u originalnom *R* interpreteru. Prednost mu je i lakše korištenje objekata iz *Jave* u kodu pisanom u jeziku *R*.

Mogući nedostatak *Renjina* je taj da nije potpuno kompatibilan s jezikom *R*, pa se u *Renjinu* ne mogu koristiti svi CRAN paketi. Na službenim stranicama *Renjina* postoji popis svih CRAN paketa i stanja njihove kompatibilnosti s *Renjinom* [16].

2.2. Usporedba performansi postojećih rješenja

U sklopu ovog rada provedena je obavljena je usporedba performansi razmatranih rješenja za povezivanje jezika *R* i *Jave*. Testirano je vrijeme potrebno za prijenos podataka iz *Jave* u *R*, obradu naredbama jezika *R*, te prijenos i transformacija podataka natrag u podatkovne strukture koje se koriste u *Javi*. U dijagramu aktivnosti na slici 5 prikazana su dva načina na koji su provedeni testovi.

Pokusi se razlikuju samo u trenutku kada mjerenje vremena započinje. U prvoj aktivnosti *Stvori Java objekte* stvara se polje brojeva od 1 do 1000 u decimalnom zapisu, i prazno polje u kojem će biti pohranjen rezultat. U aktivnosti *Omogući pozivanje R naredbi* izvršavaju se potrebne naredbe i stvaraju se potrebni objekti za korištenje rješenja koje se testira (na primjer u slučaju *Renjina* `ScriptEngineManager` i `ScriptEngine`). U obradi podataka se svaki element polja stvorenog u *Javi* kvadrira i pohranjuje u novu varijablu u jeziku *R*. Mjerenje vremena zaustavlja se kad su obrađeni podatci spremni za daljnju upotrebu u *Javi*.



Slika 5 Dijagrami aktivnosti testiranja performansi rješenja za povezivanje jezika R i Jave

Pokusi su izvršeni na prijenosnom računalu s dva gigabajta ram memorije i *AMD Brazos C50* procesorom. Na računalu je korišten operacijski sustav *Ubuntu 14.04 LTS*. *RCaller* i *Renjin* su pokrenuti iz razvojne okoline *Eclipse*, dok je *rJava* pokrenuta sa skriptom za pokretanje koja s njom dolazi. *RCaller* je testiran u normalnom i online načinu rada. U normalnom načinu rada *RCaller* za svako korištenje jezika *R* stvara novi proces, dok se u online načinu uvijek koristi isti proces.

Tablica 2 Rezultati prvog pokusa

	RCaller	RCaller Online	rJava	Renjin
Prosjek / [ms]	2121.854	2402.509	1302.281	3307.187
Medijan / [ms]	2099.428	2402.377	1296.005	3264.671
Std. devijacija / [ms]	61.638	6.640	16.722	103.188
Maksimum / [ms]	2197.480	2410.575	1323.726	3491.035
Minimum / [ms]	2067.200	2392.587	1281.399	3251.062

U tablici 2 vidi se vrijeme u milisekundama potrebno da se obrade podaci iz *Jave* korištenjem naredbi jezika *R*, ako prije toga nemamo pripremljene objekte koji nam to omogućuju. Ovo vrijeme uključuje i pokretanje procesa *R* kod *RCallera* i *rJave*. Vidljivo je da najbolje rezultate postiže *rJava*, a najgore *Renjin*. *RCaller* je u ovom slučaju brži ako se koristi u normalnom načinu rada.

Tablica 3 Rezultati drugog pokusa

	RCaller	RCaller Online	rJava	Renjin
Prosjek / [ms]	1974.564	451.437	8.476	13.183
Medijan / [ms]	1983.007	450.068	8.489	13.207
Std. devijacija / [ms]	64.601	21.786	0.043	0.084
Maksimum / [ms]	2053.135	486.736	8.511	13.274
Minimum / [ms]	1881.236	431.587	8.400	13.052

U tablici 3 prikazani su rezultati drugog pokusa. Prikazano je vrijeme u milisekundama potrebno da se obrade podatci ako su već stvoreni objekti koji su za to potrebni. Ponovo je vidljivo da najbolje performanse postiže *rJava*, ali ovog puta ju slijedi *Renjin* sa samo malo lošijim rezultatima. *RCaller* je mnogo lošiji u oba načina rada, ali ovog puta su performanse drastično bolje ako se koristi u

online načinu rada. Vidi se da kod *RCallera* u normalnom načinu nema velike razlike između prvog i drugog pokusa jer se u oba slučaja za svaku naredbu pokreće novi proces. U online načinu rada *RCaller* stvara samo jedan *R* proces na kojem izvršava sve naredbe, pa kad je jednom proces pokrenut se naredbe puno brže izvršavaju nego u normalnom načinu rada.

Za upotrebu u programskom agentu bitniji su rezultati drugog pokusa, jer je važno da kad je agent jednom pokrenut i pripremljen brzo može obraditi podatke i dalje ih koristiti. Samo vrijeme pokretanja i pripreme za korištenje alata *R* nije toliko važno.

Slični pokus proveden je i u dokumentu koji opisuje *Rcaller* [10]. U njemu je uspoređen *RCaller* u oba načina rada s rješenjima *rJava* i *Rserve*. *Rserve* je također jedno rješenje povezivanja *Jave* s jezikom *R* koje nije promatrano u ovom radu. Pokus je proveden na način sličan drugom pokusu opisanom u ovom radu, dakle prenose se podaci u *R* i natrag, ali su svi potrebni objekti već pripremljeni. Rezultati tog pokusa prikazani su u tablici 4.

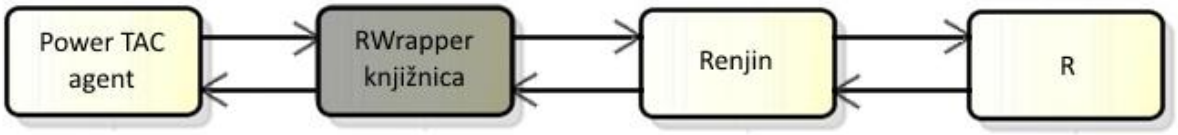
Tablica 4 rezultati pokusa provedenog u radu koji opisuje RCaller [10]

	RCaller	RCaller Online	Rserve	rJava
Min / [ms]	557.00	257.00	0.00	0.00
Max / [ms]	643.00	296.00	14.00	9.00
Prosjek / [ms]	569.90	266.96	1.21	1.28
Medijan / [ms]	565.00	263.00	1.00	1.00
Std. devijacija / [ms]	14.92	9.63	1.76	1.39
MAD	4.45	5.93	0.00	1.48

Rezultati pokusa provedenog u radu koji opisuje *RCaller* [10] pokazuju slične odnose između performansi pojedinih rješenja kao i pokusi provedeni u ovom radu, ali su prosječna vremena nekoliko puta manja. Razlog tome može biti razlika u računalima na kojima su pokusi provedeni, kao i sami detalji pokusa.

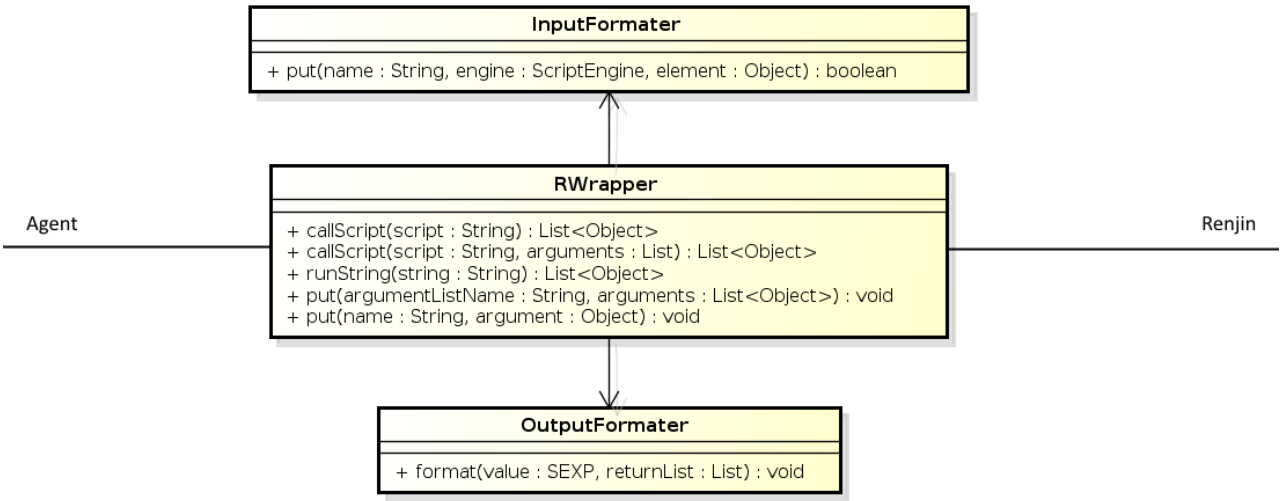
3. Implementacija knjižnice za korištenje jezika R unutar Jave

Cilj ovog rada je izraditi programsku knjižnicu koja će omogućiti transparentno korištenje jezika *R* unutar programskog agenta koji sudjeluje u natjecanju Power TAC. Da bi se to ostvarilo potreban je bio način izvršavanja naredbi i skripti jezika *R* unutar programskog jezika Jave, što je omogućeno korištenjem *Renjina*. Programska knjižnica *RWrapper* koja je izrađena u sklopu ovog rada ima za cilj sakriti način na koji se *Renjin* koristi od programera koji razvija Power TAC agenta. Na slici 6 prikazana je uloga i položaj *RWrapper* knjižnice u odnosu na ostale komponente sustava Power TAC agenta.



Slika 6 Položaj knjižnice RWrapper u sustavu Power TAC agenta

Knjižnica automatski pretvara *Renjinove* `SEXP` objekte u odgovarajuće *Java* objekte, te omogućuje korisniku transparentno slanje višedimenzionalnih polja i listi u *R*. Dijagram klasa knjižnice RWrapper s prikazanim javnim metodama nalazi se na slici 7.



Slika 7 Dijagram klasa knjižnice RWrapper

Glavna klasa koja korisniku služi kao sučelje za korištenje *R* skripti i naredbi je `RWrapper`. Sve metode klase `RWrapper` su javne i statičke. Korisnik ne mora sam stvarati objekte klasa `ScriptEngineManager` i `ScriptEngine`, jer su oni privatne varijable unutar klase `RWrapper`. Za pozivanje skripti koristi se metoda `callScript` koja ima dvije verzije. Ako korisnik samo želi pozvati skriptu bez da koristi neke podatke iz *Jave,* onda je metodu `callScript` dovoljno pozvati samo s jednim argumentom, nizom znakova koji pokazuje na lokaciju skripte u datotečnom sustavu. Ako korisnik želi poslati neke podatke iz *Jave* i obraditi ih u *R* skripti, onda mora pozvati `callScript` s dva argumenta, lokacijom skripte u datotečnom sustavu, i listom objekata koje želi koristiti u *R* skripti. Podacima poslanim iz *Jave* su u *R* skripti pristupa kroz varijablu `input`. Varijabla `input` je tipa liste jezika *R*, i u njoj se nalaze svi podaci koje je korisnik poslao iz *Jave* oblikovani tako da ih se može koristiti u skripti. Povratni tip metode `callScript` je također lista objekata i sadrži rezultate zadnje naredbe provedene u *R* skripti. Korisnik može iz te liste odabrati objekt i pretvoriti ga u odgovarajuću klasu.

Metoda `runString` služi za izvršavanje *R* naredbi zadanih kao niz znakova. Kao rezultat ta metoda također vraća listu objekata kao i metoda `callScript`. Metoda `put` služi za prijenos podataka iz *Jave* u *R*, i ima dvije verzije, prva kao argumente prima niz znakova kojima korisnik zadaje ime koje će lista imati kao varijabla jezika *R*, a drugi argument je lista objekata s podacima koje korisnik želi prenijeti u *R*. Ako korisnik želi prenijeti samo jedan objekt, onda metodu `put` može pozvati s nizom znakova imena varijable i jednim objektom kojeg želi prenijeti u *R*. Metoda `get` služi za dobivanje neke varijable korištene u *R* jeziku, kao argument prima niz znakova s imenom te varijable u jeziku *R*, a rezultat vraća kao listu *Java* objekata.

3.1. Prijenos podataka između jezika R i Jave

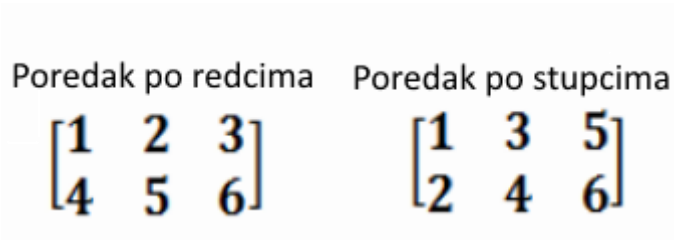
3.1.1. Prijenos podataka u *R*

Na slici 4 su prikazane strukture podataka korištene u *Renjinu*. *Renjin* bi trebao automatski pretvarati sva jednodimenzionalna polja iz *Jave* koja se sastoje od primitivnih tipova podataka ili njihovih omotača u svoje odgovarajuće strukture. U većini slučajeva Renjin uspješno provodi tu konverziju, no u verziji *Renjina*

korištenoj u izradi ovog rada (verzija 0.7.0-RC7) javljale su se greške kod polja podataka tipova `int` i `boolean`. U RWrapperu se zato sva polja tih tipova koja dođu na ulaz prvo pretvaraju u polja svojih omotača (`Integer` i `Boolean`).

Za prijenos podataka iz *Jave* u *R* koristi se `InputFormater`. `InputFormater` prvo provjerava može li *Renjin* sam automatski pretvoriti objekt koji se želi prenijeti u *R*. Ako može, onda jednostavno pozove metodu `put` od objekta `ScriptEngine`. Zatim se provjerava jeli ulazni objekt višedimenzionalno polje.

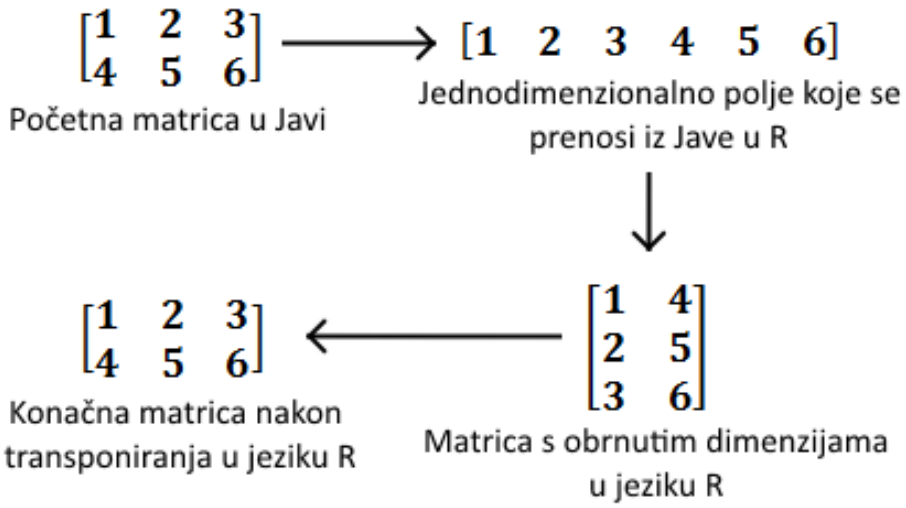
Višedimenzionalna polja u jeziku *R* su zapravo jednodimenzionalna polja kojima sa atributom `dim` postavljenim na željene dimenzije. Razlika između jezika *R* i *Jave* je da su u jeziku *R* polja spremaju u poretku po stupcima (engl. *column major order*), a u *Javi* iako nije eksplicitno zadano se na dvodimenzionalna polja većinom gleda kao da su spremljena u poretku po redcima (engl. *row major order*) kao u jeziku *C*. Na slici 8 je prikazana razlika u redoslijedu popunjavanja matrice u poretku po redcima i poretku po stupcima.



Slika 8 Razlike između poretka po redcima i poretka po stupcima

Kad prenosi dvodimenzionalno polje iz *Jave* u *R* knjižnica *RWrapper* najprije stvori jednodimenzionalno polje u koje spremi sve članove dvodimenzionalnog polja jedan za drugim, te ga potom prenese u *R*. Zatim *RWrapper* odredi dimenzije početnog polja. Da bi se riješio problem spremanja polja u različitim redoslijedima *RWraper* postavi dimenzije polja koja je prenio u *R* obrnuto nego što su bile u originalnoj matrici (na primjer ako je originalna matrica imala dva retka i tri stupca postavi dimenzije prenesenog polja na tri retka i dva stupca). Tu matricu *R* popunjava u poretku po stupcima, a stupci u toj matrici su duljine redaka u originalnoj matrici, tako da je dobivena matrica jednaka

transponiranoj originalnoj matrici. Nakon transponiranja su početna i dobivena matrica jednake. Postupak je prikazan na slici 9.



Slika 9 Postupak prijenosa dvodimenzionalnih matrica

Za trodimenzionalna polja postupak je sličan, prvo se svi članovi trodimenzionalnog polja spreme u jednodimenzionalno polje, i to polje se prenese u *R*. Zatim se tom polju dimenzije postave u suprotnom redoslijedu od originalnog polja i pozove se *R* funkcija `aperm` tako da se transponiraju sve dvodimenzionalne matrice u trodimenzionalnom polju.

Ako je ulazni objekt lista, `InputFormater` za svaki element liste rekurzivno zove svoju funkciju `put`. Kad prenese sve elemente pojedinačno od njih stvori novu listu u *R* okolini . Tako će bilo kakva konstrukcija načinjena od lista u *Javi* biti ispravno prenesena u *R*.

Podacima prenesenim iz *Jave* se u skriptama pisanim u jeziku *R* pristupa preko varijable `input` koja je tipa liste.

3.1.2. Prijenos podataka u *Javu*

Kod prijenosa podataka iz jezika *R* u *Javu* koristi se klasa `OutputFormater`. *Renjin* rezultate naredbi jezika *R* vraća u obliku `SEXP` objekata. *RWrapper* za svaki `SEXP` objekt zove metodu `format` iz klase `OutputFormater`. Metoda `format` pretvara `SEXP` objekte u odgovarajuće strukture podataka u *Javi*.

Metoda `format` prvo provjerava dali je ulazni `SEXP` objekt tipa `Vector`. Ako objekt nije vektor, korisniku će se vratiti `SEXP` objekt bez ikakvog formatiranja. Ako ulazni objekt je vektor, provjerava se dali je lista ili polje. U slučaju da je podatak lista funkcija `format` će biti rekurzivno pozvana za svaki element liste.

Ako je podatak polje, prvo se provjerava njegova duljina. Ako je duljina polja jedan, poziva se metoda `getElementAsObject` i rezultat te metode se vrati korisniku. Ako je duljina polja veća od jedan potrebno je utvrditi dimenzije polja. Metoda `format` može pretvoriti jednodimenzionalna, dvodimenzionalna i trodimenzionalna *R* polja u odgovarajuće objekte jezika *Java*. Polja koja imaju više od tri dimenzije biti će vraćena korisniku kao `SEXP` objekt. Dvodimenzionalna i trodimenzionalna polja *Java* polja koja vraća metoda `format` su zapisana u poretku po redcima.

3.2. Mogućnosti i ograničenja knjižnice RWrapper

Knjižnica *RWrapper* programeru koji radi na Power TAC agentu olakšava rad tako što mu omogućuje pozivanje skripti pisanih u jeziku *R*. *RWrapper* također obavlja automatsku pretvorbu podataka koji se prenose između jezika *R* i *Jave*. Koristeći *RWrapper* korisnik može sa samo jednom naredbom izvršiti *R* skriptu koja će mu obraditi podatke iz *Jave* i dobiti natrag rezultate koji su odmah spremni za daljnje korištenje u *Javi*.

RWrapper može konvertirati sve primitivne tipove podataka u *Javi*, omotače primitivnih tipova i nizove znakova u odgovarajuće strukture jezika *R*. Također *RWrapper* će pretvoriti polja i liste sačinjene od navedenih tipova u odgovarajuće strukture jezika *R*. Liste koje se prenose mogu u sebi sadržavati podatke različitih tipova i druge liste.

Polja koja se prenose mogu biti jednodimenzionalna, dvodimenzionalna ili trodimenzionalna. U *Javi* je moguće stvoriti dvodimenzionalno polje koje sastoji od polja koja nemaju jednak broj članova (primjer koda za stvaranje takvog polja prikazan je u ispisu 1), takva polja nisu podržana.

```
String[][] trokut = new String [4][];
trokut[0] = new String [] {"a"};
trokut[1] = new String [] {"a","b"};
trokut[2] = new String [] {"a","b","c"};
trokut[3] = new String [] {"a","b","c","d"};
```

Ispis 1 primjer matrice s nepravilnim dimenzijama

Ako korisnik pokuša prenijeti podatak ne podržanog tipa iz *Jave* u *R* knjižnica će mu podići iznimku, dok u obrnutom smjeru ako korisnik šalje podatak iz jezika *R* kojeg knjižnica ne podržava on će se prenijeti kao objekt klase `SEXP` kojeg će korisnik moći sam dalje obrađivati.

RWrapper koristi *Renjin* za povezivanje *Jave* s jezikom *R*, zbog toga je ograničen brojem CRAN paketa koji za njega postoje. Također u originalnom interpreteru *R* jezika CRAN paketi se jednostavno instaliraju naredbom `install.packages`, u *Renjinu* je potrebno ručno preuzeti CRAN paket s *Renjinove* stranice [14] i dodati ga u svoj projekt.

3.3. Primjer upotrebe R skripte u PowerTAC agentu

U ovom poglavlju biti će prikazan jednostavan primjer uporabe jezika *R* unutar Power TAC agenta. Skripta koja je korištena u ovom primjeru nalazi se u ispisu 2. Skripta iz *Jave* dobiva polja zapisa temperatura i brzine vjetra u zadnjih nekoliko krugova natjecanja. U prve dvije naredbe ulazni podaci iz varijable `input` se spremaju u nove varijable jezika *R*. Zatim se te varijable ispišu na konzolu. U zadnjoj naredbi stvara se lista koja sadrži medijane dobivenih podataka.

```
temperatures <- input[[1]]
windSpeeds <- input[[2]]
cat("Temperature\n",temperatures,"\n")
cat("Brzine vjetra\n",windSpeeds,"\n")
list(median(temperatures),median(windSpeeds))
```

Ispis 2 R skripta korištena u primjeru

Power TAC agent će svaki puta kada dobije izvještaj o vremenu od poslužitelja pohraniti temperaturu i brzinu vjetra iz tog izvještaja u listu. Kada se u listi skupi deset izvještaja agent će pozvati *R* skriptu iz ispisa 2 kojoj će predati podatke o temperaturi i brzini vjetra. Kao rezultat skripta vraća rezultat svoje

zadnje naredbe, u ovom slučaju to je lista s medijanima temperature i brzine vjetra. Na kraju agent ispiše rezultate na konzolu i isprazni liste. Funkcija agenta koja obrađuje izvještaje i poziva *R* skriptu može se naći u prilogu ovog rada.

Rezultat jednog poziva skripte kojeg agent ispisuje na konzolu nalazi se u ispisu 3. Prva četiri reda ispisana su naredbama jezika *R*, a zadnja dva reda ispisana su iz *Jave*. U ovom primjeru vidi se da su podaci dobiveni od poslužitelja obrađeni u skripti pisanoj u jeziku *R*, i nakon toga su rezultati obrade vraćeni u *Javu* i ispisani na konzolu.

```
Temperature
 5.9 6.1 6.1 6.1 6.1 6.2 9 11.3 13.4 14.3
Brzine vjetra
 2 2 1 2 2 1 1 3 3 4
Medijan temperatura: 6.15
Medijan brzina vjetra: 2.0
```

Ispis 3 Izlaz Power TAC agenta

Zaključak

Zbog napretka tehnologija i ekoloških pritiska elektroenergetske mreže u svijetu prelaze iz tradicionalnog oblika u modernije elektroenergetske mreže. S njima se također mijenja i tržište električne energije tako da se udaljava od monopola i postaje sve više liberalizirano. Da bi se izbjegli problemi kao što je velika kriza električne energije u Kaliforniji 2000. godine potrebno je istražiti takva tržišta i odrediti pravilne zakonske okvire.

Natjecanje Power TAC nastalo je kao jedno ekonomično rješenje ispitivanja i simuliranja tržišta električnom energijom. U tom natjecanju sudjeluju inteligentni programski agenti koji kao posrednici preprodaju električnu energiju s ciljem da ostvare što veći profit.

Programski agenti u natjecanju Power TAC primaju poruke od poslužitelja s raznim podacima o tržištu i okolini na temelju kojih moraju donositi odluke. U ovom radu predlaže se da se ti podaci obrađuju u specijalizirano programskom jeziku za statističku obradu *R,* a ne u *Javi* u kojoj je napisan kod samog programskog agenta.

Uspoređena su različita rješenja koja omogućuju korištenje jezika *R* iz *Jave*, te je od njih odabran *Renjin* za implementaciju knjižnice *RWrapper*. *Renjin* nudi dobar kompromis između performansi i lakoće kojom se *Java* može koristiti u programskom agentu, i količine funkcija za statističku obradu koje su implementirane u jeziku *R.*

Knjižnica *RWrapper* koja je izrađena u sklopu ovog rada rješava probleme prijenosa različitih tipova podataka između *Jave* i jezika *R*. Knjižnica se jednostavno integrira u programskog agenta i pruža programeru jednostavno sučelje za korištenje skripti pisanih u jeziku *R* za obradu podataka.

U budućnosti bi se u knjižnicu *RWrapper* moglo uvesti i podršku za korištenje knjižnice *rJava* kao poveznice između *Jave* i jezika *R.* Tako bi se korisnicima koji na svojem računalu već imaju instaliran *R* omogućilo korištenje svih njegovih paketa i postizanje još boljih performansi. Korisnici koji nemaju instaliran *R* i dalje bi imali opciju korištenja *Renjina*.

Literatura

[1] Weare, C., *The California Electricity Crisis: Causes and Policy Options*. San Francisco: Public Policy Institute of California, 2003

[2] Ketter, W., Collins, J., Reddy, P., Weerdt, M. The 2015 Power Trading Agent Competition, Department of Decision and Information Sciences ,Rotterdam School of Management, 2015.

[3] Commons Math: The Apache Commons Mathematics Library, http://commons.apache.org/proper/commons-math/userguide/stat.html, lipanj 2015.

[4] Java Statistical Analysis Tool, https://github.com/EdwardRaff/JSAT/tree/master, lipanj 2015.

[5] Frequently Asked Questions on R, http://cran.r-project.org/doc/FAQ/R-FAQ.html, travanj 2015.

[6] Bob Muenchen, R #1 by Wide Margin in Latest Kdnuggets Poll, http://r4stats.com/2015/05/27/r-1-by-wide-margin-in-latest-kdnuggets-poll/, svibanj 2015.

[7] Nature.com - Programming tools: Adventures with R, http://www.nature.com/news/programming-tools-adventures-with-r-1.16609, svibanj 2015.

[8] Contributed packages, http://cran.at.r-project.org/, svibanj 2015.

[9] LangPop.com – Programing language popularity, http://www.langpop.com, svibanj 2015.

[10] Satman, M., H., RCaller: A Software Library for Calling R from Java, Istanbul University, Department of Econometrics**,** 2014.

[11] rJava - Low-level R to Java interface, https://www.rforge.net/rJava/index.html, travanj 2015.

[12] Renjin developer documentation, http://docs.renjin.org/en/latest/, travanj 2015.

[13] Java Scripting Programmers Guide, http://docs.oracle.com/javase/6/docs/technotes/guides/scripting/programmer_guide/, svibanj 2015.

[14] Renjin CRAN builds, http://packages.renjin.org/, svibanj 2015.

[15] Eclipse - The Eclipse Foundation open source community website, https://eclipse.org/, svibanj 2015.

Sažetak

Power TAC je natjecanje u kojem se inteligentni programski agenti natječu u ostvarivanju profita na liberaliziranom tržištu električne energije. Programski agenti na temelju podataka o tržištima i vremenskoj prognozi moraju donositi odluke o svojim potezima da bi mogli svojim korisnicima osigurati potrebnu energiju i ostvariti profit na tržištu.

Za obradu podataka Power TAC agentu potrebno je omogućiti korištenje naprednih statističkih alata. Jedan od danas najpopularnijih jezika za statističku obradu podataka je *R*. Glavna prednost korištenja jezika *R* za obradu podataka je broj funkcija koje su implementirane u njemu i njegovim programskim knjižnicama.

Uz ovaj rad razvijena je knjižnica *RWrapper* koja olakšava programeru koji radi na Power TAC agentu upotrebu jezika *R* za obradu svojih podataka. *RWrapper* za povezivanje *Jave* i jezika *R* koristi *Renjin* zbog njegovih performansi i jednostavne integracije s agentom. Knjižnica za korisnika rješava mnoge probleme do kojih dolazi prilikom prijenosa podataka između ta dva programska jezika i korisniku pruža jednostavno sučelje za pozivanje naredbi i skripti pisanih u jeziku *R*.

Summary

Power TAC is a competition in which inteligent software agents compete in making profit on a free electricity market. Agents need to make decisions based on informations about markets and the weather report in order to secure electricity for their customers and to make a profit.

To analyze data, Power TAC agents need the ability to use advanced tools for statistical data analysis. One of the most popular programming languages used for data analysis today is *R*. The main advantage of using *R* for data analysis is the number of functions implemented in the language and its libraries.

In this thesis the Java library *RWrapper* has been developed. *RWrapper* makes it easier for programmers working on Power TAC agents to use *R* for their data analysis. *RWrapper* uses *Renjin* for connecting *Java* and *R* because of its speed and easy integration with Power TAC agents. The library solves many problems which occur when data is transferred between these two programming languages, and offers a simple interface to the user for running *R* scripts and commands.

Prilog 1 – funkcija Power TAC agenta koja koristi R skriptu

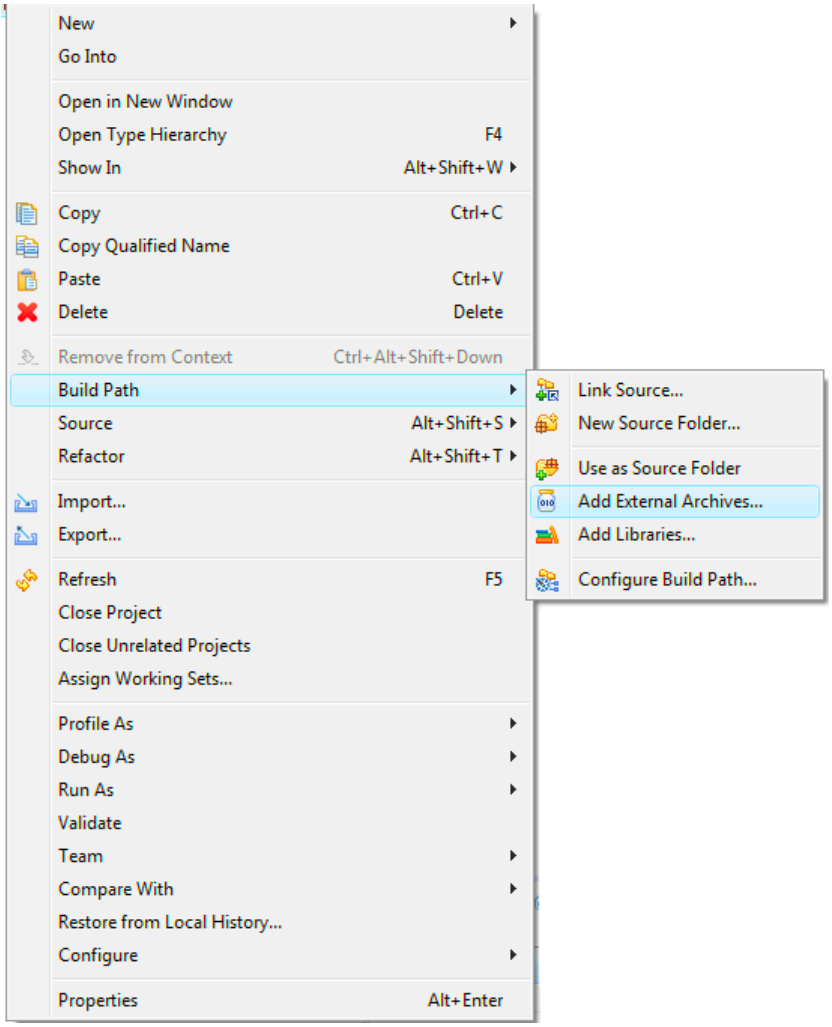
```
private List<Double> temperature = new LinkedList<Double>();
private List<Double> windSpeed = new LinkedList<Double>();

/**
 * Receives a new WeatherReport.
 */
  public synchronized void handleMessage(WeatherReport report) {
      temperature.add(report.getTemperature());
      windSpeed.add(report.getWindSpeed());
      if (temperature.size() >= 10) {
          List<Object> arguments = new LinkedList<Object>();
          arguments.add(temperature.toArray(new Double[10]));
          arguments.add(windSpeed.toArray(new Double[10]));
          try {
              List<Object> results;
              results = RWrapper.callScript("scripts/SampleScript", arguments);
              System.out.println("Medijan temperatura: "+ results.get(0));
              System.out.println("Medijan brzina vjetra: "+ results.get(1));
          } catch (FileNotFoundException e) {
              e.printStackTrace();
          } catch (ScriptException e) {
              e.printStackTrace();
          }
          temperature = new LinkedList<Double>();
          windSpeed = new LinkedList<Double>();
          System.out.println();
      }
  }
```

Prilog 2 – Upute za instalaciju

Instalacija knjižnice *RWrapper* svodi se na dodavanje `.jar` datoteke u *Java* projekt. U ovom poglavlju biti će pokazan primjer instalacije knjižnice u programskom okruženju *Eclipse* [15].

Prvo je potrebno pokrenuti *Eclipse* i u kartici *Package Explorer* desnim klikom miša odabrati projekt u kojem se želi koristiti knjižnicu (u ovom primjeru projekt se zove *Moj Projekt*). Ako korisnik nema otvorenu karticu *Package Explorer*, istu može uključiti u izborniku *Window* pod opcijom *Show View*. U padajućem izborniku koji se pojavi kada se odabere projekt potrebno je doći do opcije *Build Path* i unutar nje odabrati opciju *Add External Archives*. Na slici 10 je prikazan način kako doći do opcije *Add External Archives* u padajućem izborniku.



Slika 10 Opcija Add External Archives u padajućem izborniku

Kad se odabere opcija *Add External Archives* pojavit će se prozor u kojem je potrebno odabrati knjižnicu *RWrapper.jar* u svom datotečnom sustavu i pritisnuti tipku *open*. Nakon toga *Eclipse* će dodati knjižnicu u vaš projekt. U klasi u kojoj želite koristiti knjižnicu potrebno je na početku koda dodati naredbu `import hr.unizg.ztel.rwrapper.*;`. Sada se u klasi mogu koristiti statičke metode klase `RWrapper` preko kojih se može koristiti jezik *R*.