UNIVERZA V LJUBLJANI

FAKULTETA ZA DRUŽBENE VEDE

Uroš Gjerek

Razvoj orodja za grafično predstavitev podatkov o tržnem deležu

Diplomsko delo

Ljubljana, 2016

UNIVERZA V LJUBLJANI

FAKULTETA ZA DRUŽBENE VEDE

Uroš Gjerek

Mentor: izr. prof. dr. Aleš Žiberna

Razvoj orodja za grafično predstavitev podatkov o tržnem deležu

Diplomsko delo

Ljubljana, 2016

Razvoj orodja za grafično predstavitev o tržnem deležu

Analiza podatkov je proces, v katerem iz podatkov dobimo uporabne informacije. Sestavljen je iz več medsebojno povezanih korakov, kot so zbiranje, priprava, obdelava, analiziranje podatkov in vizualna predstavitev le-teh. V praksi se za različne korake uporabljajo različna programska orodja. Nekatera za zbiranje podatkov, druga za njihovo obdelavo in analiziranje, tretja za vizualno predstavitev podatkov. Tak »razbit« način dela predstavlja težavo zaradi velike možnosti napak. Če napaka naredimo v predhodnem koraku, se to odrazi v vseh kasnejših korakih analize. Poleg velike možnosti napak je »razbit« način dela je tudi časovno neučinkovit. Rešitev predstavlja avtomatizacija procesa analize podatkov. Združila bi namreč korake priprave, obdelave, analize in vizualne predstavitve podatkov v enotno aplikacijo. Ker bi se zmanjšalo število človeških interakcij, bi se zmanjšala možnost napak, povečala pa bi se hitrost analize podatkov. V svoji nalogi sem ustvaril spletno aplikacijo za avtomatizacijo podatkov o tržnem deležu, ki je v obliki dashboard-a ali armaturne plošče. Za izdelavo aplikacije sem uporabil programski jezik R in paket shiny.

Ključne besede: Analiza podatkov, Grafična predstavitev podatkov, Dashboard, Shiny, programski jezik R.

Development of software for graphical presentation of market share

Data analysis is the process of acquiring useful information from data. This process consists of several interconnected steps such as data collection, preparation, processing, analysis and visual presentation of data. In practice various steps are supported by different software tools. Some tools are used for data collection, others for data processing and analysis, and still others for data visualisation. This »fragmented« approach to work with data is a problem, because it significantly raises the chance of making an error. If an error is made in one of the earlier steps this is then reflected in all the following steps of the process. In addition to the bigger chance of an error, the »fragmented« approach to work is also time consuming. The solution to the problem can be the automation of the process of data analysis. The automation aims to merge steps of data collection, processing, analysis and visualisation in a single application. As the number of the human interventions is reduced, the chance of an error is lowered and the time required reduced. In my thesis I developed an internet application to automate the market share data analysis. The application is designed as a dashboard and was built in R programming language and Shiny R package.

Keywords: Data analysis, Data visualisation, Dashboard, Shiny, R programming language.

Kazalo

Kazalo slik

Kazalo tabel

1 Uvod

Cilj diplomskega dela je izdelati spletno aplikacijo, ki bo omogočala avtomatizacijo analiz podatkov o tržnem deležu. Aplikacija bi omogočala nalaganje anketnih podatkov, analizo podatkov in grafični prikaz rezultatov (v obliki interaktivnih oken oz. dashboard-ov) glede na izbrane parametre analiz. Spletna aplikacija bo postala enotno orodje za analizo podatkov o tržnem deležu in bo združevala funkcionalnosti dashboardov in sistemov za poročanje.

Analiza podatkov je proces, ki je razdeljen na več korakov (priprava / čiščenje podatkov, analize podatkov, vizualizacija podatkov) (Kabacoff, 2011), za vsakega se uporabljajo različna orodja. Za pripravo / čiščenje in analizo podatkov se najpogosteje uporablja statistična orodja, kot so npr. IBM SPSS Statistics, za vizualni prikaz podatkov se uporabljajo preglednice, kot je npr. Microsoft Excel. Pri kontinuiranih raziskavah se za shranjevanje podatkov uporabljajo še podatkovne baze (Microsoft Access, MySQL-ite ...). Največja težava tako razbitega načina analiziranja je velikem številu napak, ki se pojavijo kot posledica človeškega dejavnika. Pri veliki količini ročnega dela (poganjanju SPSS sintaks, kopiranju podatkov v excel ali access) je velika verjetnost napak. Spletna aplikacija bi omogočila analizo podatkov s čim manjšim številom človeških interakcij, na ta način pa bi zmanjšali število napak.

Avtomatizacija analize bi zmanjšala število zanjo porabljenih ur, kar bi posledično vplivalo tudi na povečano delovno učinkovitost.

Za izdelavo avtomatizacije bom uporabil programski jezik R (R Core Team 2016). Čeprav je bil R prvotno namenjen statističnim izračunom, je z dodatnimi paketi mogoče zelo razširiti funkcionalnost programskega jezika. Pri izdelavi avtomatizacije bom uporabil različne pakete, od katerih sta najpomembnejša shiny (Chang in drugi 2016) in shinydashboard (Chang 2015). Paket shiny omogoča izdelavo spletnih aplikacij s programom R.

2 Teoretični pregled

2.1 Dashboard

2.1.1 Kaj je dashboard ali armaturna plošča

Dashboard ali armaturna plošča je dobila ime po analogiji iz armaturne plošče avta. Tako kot se na armaturni plošča avta preko števcev, opozorilnih lučk in prikazovalnikov informacij prikazujejo trenutne informacije, ki so potrebne za varno vožnjo, dashboard služi podobnemu namenu in prikazuje informacije, ki jih lahko uporabimo za nadaljnje odločitve.

V literaturi ni enotne definicije, kaj dashboard ali armaturna plošča je. Dashboard ali armaturna plošča je vizualni prikaz pomembnih informacij, ki so potrebne za dosego enega ali več ciljev. Informacije so prikazane in urejene na enem zaslonu tako, da jih je mogoče vse zajeti z enim pogledom (Few 2006).

Dashboard je poslovno orodje, ki prikazuje indikatorje uspešnosti (PE - performance indicators), ključne indikatorje uspeha (KPI - key performance indikators) ali katerokoli drugo za poslovne uporabnike zanimivo informacijo. Podatki na dashboardih so po pridobitvi iz enega ali več podatkovnih virov pogosto prikazani v realnem času. Dashboardi so interaktivni in omogočajo poglede na podatke iz več različnih zornih kotov. Sestavljeni so iz različnih orodij za vizualizacijo podatkov, kot so grafikoni, različni števci in zemljevidi. Različni sektorji istega podjetja imajo različne poslovne koristi od dashboardov. Tako se lahko rudar s pomočjo dashborarda lažje odloči, kje bo vrtal, CEO velikega podjetja pa se bo na podlagi podatkov, ki jih dobi iz dashboarda, odločil kam bo investiral denar (Hetherington 2009).

Prvi dashboard sistemi so se pojavili že v osemdesetih letih prejšnjega stoletja. Takrat so se imenovali EIS (Executive Information Systems) – informacijski sistemi za vodilne delavce. Namen teh sistemov je bil prikazati vodilnim delavcem ključne finančne podatke podjetja. Vendar se v praksi ti sistemi niso uveljavili, ker podatkovna skladišča še niso bila razvita do te mere, da bi omogočala zadostno natančnost izračunov. Baze podatkov so bile netočne in podatki so bili zapisani v različnih med seboj nepovezanih bazah. Tehnološki razvoj na področju podatkovnih skladišč, OLAP procesov in poslovne inteligence v 90 letih prejšnjega stoletja ter pojav KPI (key performance indicator) – ključnih kazalnikov uspeha na področju managmenta in upravljanja s podjetji - so vplivali na razvoj dashboard sistemov. Odločilno pa je na njihov razvoj vplival škandal Enron iz leta 2001. Od takrat naprej zakonodajalci

zahtevajo, da večja podjetja uporabljajo sisteme, ki v vsakem trenutku kažejo dejansko stanje podjetja (Few 2006).

Informacije na dashboardu so ponavadi prikazane vizualno s kombinacijo grafičnih elementov in besedila, vendar je poudarek na grafičnih prikazih, ker imajo le-ti večjo sporočilno vrednost kot besedilni. Informacije so prikazane tako, da jih končni uporabnik najlažje zajamejo in iz njih najhitreje izluščijo relevantne informacije (Few 2006).

- **Prikaz relevantnih informacij:** na dashboardu so prikazane relevantne informacije, ki so potrebne za dosego cilja. Te informacije so pogosto iz različnih virov podatkov, ki jih je treba pred prikazom na dashboardu še ustrezno pripraviti. Pogosto se prikazujejo KPI (ključni indikatorji uspeha) ali pa kaki drugi kazalci uspešnosti poslovanja.
- **Velikost dashboarda mora ustrezati velikosti računalniškega zaslona:** podatki, prikazani na dashboardu, ne smejo zavzeti več površine, kot jo ima računalniški monitor. Uporabnik mora zajeti vse informacije naenkrat, na prvi pogled. Če je dashboard prevelik, da bi ustrezal velikosti monitorja, se mora uporabnik premikati po zaslonu in ne more zajeti vseh informacij na prvi pogled. Glavni namen dashboarda je, da imamo najpomembnejše informacije na voljo takoj, da jih lahko absorbiramo hitro in brez truda in da jih lahko na hitro izluščimo.
- **Spletni brskalnik kot medij za prikaz dashboarda:** spletni brskalnik je trenutno najustreznejši medij za prikaz, seveda pa ni edini. Izbira medija za prikaz je odvisna od načina delovanja dashboarda. Za aplikacije, ki se morajo osveževati v realnem času, spletni brskalnik ni najustreznejša izbira.
- **Dashboard se uporabljajo za spremljanje informacij na prvi pogled:** čeprav lahko na dashboradu prikažemo različne vrste podatkov, je skupni imenovalec vseh prikazov to, da so podani v agregirani obliki. Vseh podatkov, ki so potrebni za dosego cilja, ne moremo spremljati na prvi pogled v izvorni obliki, zato jih moramo agregirati. Dashboard mora pokazati na katere informacije moramo biti pozorni in katere informacije zahtevajo določene aktivnosti. Dashboardu ni potrebno podati vseh podrobnosti, vendar pa mora pokazati, da se nekaj dogaja na področju, ki ga opazujemo. To je tudi primarni namen dashboarda. V primeru, da potrebujemo dodatne informacije lahko naredimo dodatna orodja, ki bodo omogočala dodatne analize podatkov.
- **Dashboardi imajo majhen, zgoščen, jedrnat in intuitiven način prikaza informacij:** dashboard ima prikaz, ki jasno, enostavno in hitro prenesejo sporočilnost. Ne sme zasedati

dosti prostora in mora ustrezati velikosti računalniškega zaslona. Če grafični elementi, ki so videti kot števec goriva, prometni semafor ali termometer, ustrezajo za prikaz informacij, potem jih lahko uporabimo. Če pa imamo kakšne druge grafične elemente, ki bi ustrezneje prikazali informacije, pa raje uporabimo le-te.

- **Dashboard je narejen po meri:** informacije na dashboardu so posebej prilagojene zahtevam določenim uporabnikom, določeni skupini ali funkciji. Če podatki ne bi bili posebej prilagojeni, ne bi služili svojemu namenu.

2.1.2 Kategorizacija dashboardov

Dashboarde lahko kategoriziramo po različnih kriterijih. Kategorizacija (Few 2006) vsebuje naslednje kriterije (vloga dasboarda, tip podatkov, domena podatkov, tip meritve, frekvenca posodobitev, interaktivnost in mehanizem prikazovanja. V tabeli 2.1 so predstavljene kategorije dasboardov glede na različne kriterije razvrščanja.

Tabela 2.1: Različne kategorizacije dashboardov

Kategorizacija	Vrednosti
Vloga v podjetju	Strateška Analitična Operativna
Tip podatkov	Kvantitativni ne-kvantitativni
Domena podatkov	Prodaja Finance Marketing Proizvodnja Kadrovska služba
Frekvenca posodobitev	Mesečno Tedensko Dnevno Po urah V realnem času
Interaktivnost	Statični prikaz Interaktivni prikaz (vrtilna tabela, filtri…)
Mehanizmi prikazovanja	Primarno grafični Primarno tekstovni Integracija grafičnih in tekstovnih

Vir: Few (2006, 30–31).

Glede na vlogo, ki jo imajo dashboardi v poslovnem procesu, jih delimo na tri skupine: strateški, analitični in operativni (Few 2006) .

2.1.2.1 Strateški dashboard

Namenjeni so managmentu za strateške odločitve. Zagotavljajo hiter vpogled v trenutno poslovno stanje. Poudarek je na sumarnih indikatorjih uspešnosti, ki lahko vključujejo tudi napoved za prihodnost. Čeprav so lahko v ta tip dashobardov vključeni tudi elementi primerjav glede na preteklo obdobje, ni priporočljivo, da vsebujejo preveč podrobnosti, ki lahko zmedejo končnega uporabnika in odvrnejo njegovo pozornost od ključnih indikatorjev.

Uporabljajo se najbolj enostavni načini prikaza podatkov. Ker se ti dashboardi uporabljajo za dolgoročno oz. strateško načrtovanje, se podatki ne osvežujejo takoj, temveč v izbranem časovnem intervalu. Prikazani podatki so med sabo neodvisni, zato ne omogočajo nadaljnjih podrobnejših analiz.

2.1.2.2 Analitični dashboard

Dashboardi, ki podpirajo analizo podatkov, zahtevajo drugačen oblikovalski pristop. Pogosto se zahteva večja »globina« analiz, ki vsebuje različne primerjave in zgodovino podatkov. Leti se osvežujejo na izbrano časovno obdobje, zato ni potrebe po osveževanju v realnem času.

Večja globina analiz pomeni, da je mogoče delati različne vzročne analize med podatki. Ne zanima nas samo, da se je določen indikator znižal, zanima nas tudi, zakaj je do tega prišlo in kako bi to lahko popravili.

Analitični dashboard je kot nadzorna naprava, ki analitiku pove, da je prišlo do spremembe pri izbranem indikatorju in mu omogoča bolj poglobljeno analizo vzrokov.

2.1.2.3 Operativni dashboard

Dashboardi, ki se uporabljajo za spremljanje poslovanja, so drugače oblikovani kot strateški ali analitični. Karakteristika, ki najbolj vpliva na videz, je dinamičnost. Operativni dashboardi se uporabljajo za nadzor nad procesi, ki se spreminjajo iz sekunde v sekundo. Če v robotski roki, ki pritrjuje vrata na avto zmanjka vijakov, ne moremo čakati na to informacijo do naslednjega dne, temveč jo potrebujemo takoj, da lahko na podlagi te informacije sprožimo druge aktivnosti.

Videz operativnih dashbardov je tako kot pri strateških čim bolj preprost. Zapleten videz bi v stresnih situacijah, ki zahtevajo takojšen odgovor, dodatno zmedel uporabnike in obstajala bi

velika verjetnost napake. V nasprotju s strateškimi dashboardi mora imeti operativni možnost, da takoj obvesti uporabnika, ko določen poslovni proces ne teče znotraj postavljenih okvirjev. Operativni dashboardi morajo posredovati natančno informacijo, da se takoj ve, na katerem mestu je prišlo do odklona pri poslovnem procesu.

2.1.3 Uporaba dashboardov

Dashboardi so uporabni za različna opravila. Lahko jih uporablja meteorolog za spremljanje vremena, analitik obveščevalne službe za spremljanje klepeta potencialno sumljivih oseb ali finančni analitik, ki išče priložnosti za investicije (Few 2006). Skupni imenovalec vseh dashboardov je prikaz trenutnega stanja ključnih informacij v obliki kvantitativnih rezultatov. S pomočjo dashboardov se nadzirajo ključne informacije, ki so potrebne za uspešno opravljanje delovnih nalog.

V tabeli 2.2 so prikazane mere, ki se pogosto uporabljajo v dashboardih glede na poslovna področja.

Tabela 2.2: Pogoste mere merjenja v dashboardih

Kategorija	Mere
Prodaja	Rezervacije Izdani računi Načrt prodaje Število naročil Znesek naročil Prodajne cene
Marketing	Tržni delež Uspeh kampanj Demografska struktura strank
Finance	Promet Stroški Dobiček
Tehnična služba	Število klicev Število uspešno rešenih zadev Zadovoljstvo strank Dolžina klicev

Kategorija	Mere
Distribucijski centri	Število dni od prejetja do izdaje naročila Zaostanki Zaloge
Proizvodjna	Število proizvedenih enot Čas, potreben za proizvodnjo ene enote Število napak
Kadrovska služba	Zadovoljstvo zaposlenih Fluktuacija zaposlenih Število prostih delovnih mest
IT	Čas nedelovanje sistemov Uporaba sistemov Število popravljenih napak (hroščev)
Spletne storitve	Število obiskovalcev Število ogledov strani Čas, porabljen za ogled strani

Vir: Few (2006, 33–34).

Prikazane mere so v dashboardih podane v sumarni obliki (vsota, povprečje, mere deviacij, korelacije). Sumarni pregled podatkov je uporaben v primerih, kjer je treba nadzorovati več merjenih indikatorjev naenkrat.

2.2 Tržni delež

2.2.1 Pomen in izračun

Tržni delež je indikator konkurenčnosti, ki kaže položaj podjetja (produkta / storitve) glede na izbrano konkurenco. Tržni delež, podprt s prodajnimi podatki, pomaga managerjem oceniti dogajanje na trgu.

Ocenijo lahko ne samo dogajanje na trgu kot celoti, temveč tudi kaj se dogaja med samimi konkurenti. Prodajni podatki kažejo rast prodaje, vendar če nimamo podatkov o tržnih deležih, ne moremo vedeti, kaj se dogaja s kategorijo in z našim izdelkom / storitvijo znotraj nje. Izguba tržnega deleža nakazuje na potencialne dolgoročne težave, ki zahtevajo strateške spremembe.

Obstajata dva različna načina izračuna tržnih deležev. Izračunamo glede na vrednost v denarni enoti ali pa glede na volumen, količino prodanih izdelkov.

Tržni delež količine izračunamo glede na število izdelkov, ki jih proda določeno podjetje glede na število vseh izdelkov, ki jih prodajo vsa podjetja znotraj izbrane panoge ali kategorije.

$$TD\#_{\text{števila izdelkov}}\% = \frac{\#\ \text{Število izdelkov}_{\ 1}}{\sum_1^n \#\ \text{Število izdelkov vseh podjetij}}$$

Tržni delež denarne enote se od tržnega deleža količine razlikuje tako, da upošteva ceno izdelkov. Izračunamo ga tako, da vsoto cene izdelkov, ki jih proda določeno podjetje, delimo s celotno vsoto cen vseh izdelkov, ki jih prodajo vsa podjetja znotraj izbrane panoge ali kategorije (Farris in drugi 2010).

$$TD\%_{\text{denarne enote}}\% = \frac{€\ \text{Prodaja izdelkov}_{\ 1}}{\sum_1^n €\ \text{Vsota prodaje cene izdelkov vseh podjetij}}$$

V industriji kot je FMCG (vsakdanji izdelki široke potrošnje), kjer imamo veliko število izdelkov manjše vrednosti in kjer se dosti izdelkov razdeli zastonj zaradi akcijskih ponudb, se za izračun tržnega deleža uporablja denarna enota.

Buzzell in drugi (1975) so velikost tržnega deleža povezovali z velikostjo dobička. V študiji Profit Impact of Market Strategies (PIMS), ki jo je izvedel Marketing Science Institute, so ugotovili, da ima podjetje s 40-odstotnim tržnim deležem dvakrat večji delež dobička (ROI)

kot podjetje, ki ima 10-odstotni tržni delež. Kot razloge dobičkonosnosti tržnega deleža so navedli:

- **ekonomijo obsega:** podjetja z velikim tržnim deležem imajo velike zmogljivosti pri naročilih, proizvodnji, marketingu in pri drugih stroškovnih predpostavkah. To jim omogoča, da ustvarijo proizvod z manjšimi stroški in imajo posledično večji dobiček.
- **tržna moč:** podjetja z večjim tržnim deležem imajo večjo pogajalsko moč na trg, s katero, si lahko zagotovijo ugodnejše pogoje poslovanja. Le-ti se kažejo v nižji ceni surovin, ki jih potrebujejo za svojo proizvodnjo in v višji ceni njihovih produktov.
- **kvaliteta managementa:** dobri managerji so sposobni doseči visoke tržne deleže. Sposobni managerji lahko obvladujejo stroške, dosežejo veliko produktivnost pri zaposlenih itd. In ko podjetje doseže vodilni položaj, mogoče tudi z razvojem nove kategorije, je ta položaj veliko lažje obraniti, kot pa ga je bilo doseči.

Strateške posledice te študije so bile, da naj podjetja težijo k doseganju čim večjega tržnega deleža, da bodo lahko izkoristile prednosti ekonomije obsega in tržne moči (Yannopoulos 2010).

Pojavile pa so se tudi študije, ki so dvomile v povezavo med tržnim deležem in dobičkonosnostjo. Jacobsen in Aaker (1985) sta preizkusila učinek tržnega deleža na PIMS bazi. Ugotovila sta, da povezava med tržnim deležem in dobičkonosnostjo sicer obstaja, vendar je le navidezna in izgine, ko upoštevamo dejavnike, kot so lojalnost strank, distribucijski sistem in učinkovitost oglaševanja.

2.2.2 Indikator prevladujočega položaja

Tržni delež je indikator, na podlagi katerega se ugotavlja prevladujoč položaj in posledično tudi zloraba prevladujočega položaja. Šteje se, da ima podjetje prevladujoč položaj, če je njegov tržni delež na trgu Republike Slovenije višji od 40 odstotkov.

Prevladujoč položaj lahko uživa tudi dvoje ali več podjetij skupaj, pri čemer se šteje, da imajo ta podjetja prevladujoč položaj, če je njihov tržni delež na trgu Republike Slovenije višji od 60 odstotkov (Zakon o preprečevanju omejevanja konkurence 2008).

2.2.3 Kategorija

Čeprav je tržni delež relativno preprost indikator, ki se izračuna iz predpostavke »mi / (mi + preostali), se hitro postavi vprašanje, kdo so preostali. Težava se tudi pojavi z definiranjem

kategorije, v kateri se bo računal tržni delež. Če kategorijo definiramo preširoko, lahko zgubimo fokus. Če pa je kategorija definirana preozko, lahko zamudimo priložnosti ali pa ne opazimo nevarnosti (Farris in drugi 2010). V nalogi bom tržni delež izračunal iz izdelkov široke potrošnje, iz kategorij hrana (hrana in pijača), kozmetika (kozmetike in drugi izdelki za higieno) in čistila (sredstva za pranje, čiščenje in osveževanje doma in perila).

2.3 R, izbrani paketi in R-studio

2.3.1 R

R je programski jezik in programsko okolje za statistične analize in vizualizacijo podatkov. V nasprotju s komercialnimi statističnimi paketi, kot so SPSS, SAS, Stata je R prosto dostopen pod GNU General Public Licence. V praksi to pomeni, da je vsa (izvorna) koda paketa javna in prosto dostopna, da jo lahko kdorkoli pregleda in skopira ali prilagodi in jo ponovno uporabi.

R je zgrajen modularno. Ob inštalaciji programa dobimo osnovne pakete, ki nam omogočajo osnovne operacije in analize. Ena od glavnih prednosti R-a je njegova razširljivost. Funkcionalnost programa je mogoče razširiti s številnimi paketi, ki so prosto dostopni preko CRAN (Comprehensive R Archive Network) spletne strani.[1] Ko je paket objavljen na CRAN, si ga lahko vsak z dostopom do spleta prenese na svoj računalnik. Odprto-kodni razvoj pomeni večjo agilnost pri razvoju novih paketov oz. razširitev. Kakor hitro se pojavi nova statistična metoda, se najde posameznik, ki napiše paket za novo metodo. V določenih primerih pa razširljivost pomeni tudi slabost. Pakete pogosto pišejo posamezniki, ki po osnovni izobrazbi niso programerji. Ker kode ne napišejo optimalno, se to odraža v hitrosti delovanja programa (Field in drugi 2012). Ena od slabosti je neusklajenost pri poimenovanju in uporabi privzetih vrednosti. Nekatere funkcije manjkajoče vrednosti pri analizah samodejno odstranijo iz izračunov, pri drugih pa je treba manjkajoče vrednosti posebej izločiti z posebnimi argumenti.

Programski jezik R ima naslednje funkcije (R Core Team 2016):

- R je dobro razvit, enostaven in učinkovit programski jezik, ki vključuje pogojne stavke, zanke, uporabniško določene rekurzivne funkcije
- R ima učinkovit sistem za pripravo in shranjevanje podatkov

[1] Poleg CRAN spletne strani, obstajajo še druge na katerih so objavljeni paketi za R (Bioconductor, Omegahat, MRAN in GitHub).

- R zagotavlja niz operaterjev za računanje nad polji, seznami, vektorji in matricami
- R zagotavlja veliko, koherentno in integrirano zbirko orodij za analizo podatkov
- R zagotavlja grafične zmogljivsoti za analizo podatkov in za prikaz le-teh

S pojmom računalniško okolje okarakteriziramo R kot načrtovan in skladen sistem za obdelavo podatkov, ki je po potrebi razširljiv z drugimi programskimi jeziki (C, C++, Fortran). Razširljivost s programskimi jeziki C, C++, Fortran se uporabi pri računsko zahtevnih izračunih.

2.3.2 Shiny

Shiny je paket za programski jezik R, ki omogoča izdelavo interaktivnih spletnih aplikacij (Chang in drugi 2016). Omogoča izgradnjo kompleksnih interaktivnih spletnih aplikacij v programskem jeziku R. S pomočjo paketa shiny in programskega jezika R se naredi uporabniški vmesnik, ki skrbi za videz aplikacije, in server, ki poskrbi za analitski del aplikacije. Shiny paket prevede R kodo uporabniškega vmesnika v HTML, CSS in JavaScript, ki je potreben za prikaz aplikacije na spletu. (Zev 2016).

Paket shiny ima naslednje lastnosti (Chang in drugi 2016):

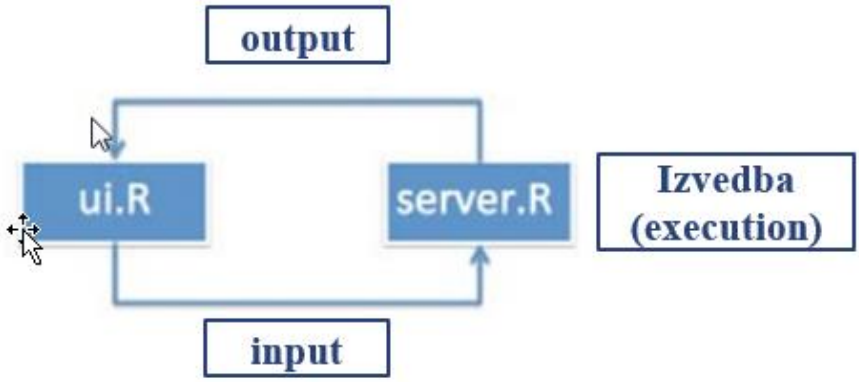
- z R kodo omogoča enostavno izgradnjo spletnih aplikacij, brez poznavanja JavaScript jezika, HTML-ja in CSS-ja
- shiny aplikacije so avtomatsko "žive" na isti način, kot so "živi" izračuni v preglednicah. Rezultat se spremeni takoj, ko se spremenijo vhodni podatki brez potrebe po osveževanju spletne strani
- uporabniški vmesnik v shiny aplikaciji je lahko napisan v celoti v R jeziku ali pa je neposredno v HTML kodi z dodatki CSS-ja in JavaScript-e
- uporabniški vmesnik temelji na priljubljenem programskem ogrodju Bootstrap3
- vnaprej narejeni izhodni gradniki za prikaz grafov, tabel in izhodnih R objektov
- za hitro dvosmerno komunikacijo med spletnim brskalnikom in R-om uporablja paket httpuv
- uporablja se reaktivni programski model, s katerim zaobidemo zapleteno programiranje dogodkov

Vsaka shiny aplikacija je sestavljena iz dveh delov (Beely 2016): iz spletne strani, ki prikazuje aplikacijo uporabniku, in iz serverja, ki opravlja analitske operacije. Ta delitev se kaže v R kodi aplikacije. Shiny aplikacija je sestavljena iz dveh datotek: iz datoteke ui.R in

datoteke server.R. Novejše verzije paketa shiny omogočajo, da se obe datoteki združita v eno, ki se imenuje app.R. Tudi v združeni datoteki sta funkciji za uporabniški vmesnik in server jasno ločeni.

R koda iz datoteke ui.R se prevede v HTML stran, ki jo vidi uporabnik aplikacije, server pa je odgovoren za logiko izvajanja aplikacije. To so nizi navodil, ki povedo spletni strani, kaj je treba pokazati, ko uporabnik na njej spreminja vhodne parametre (Hillebrand in Nierhof 2015).

Slika 2.1: Odnos med uporabniškim vmesnikom (ui.R) in serverjem (server.R)



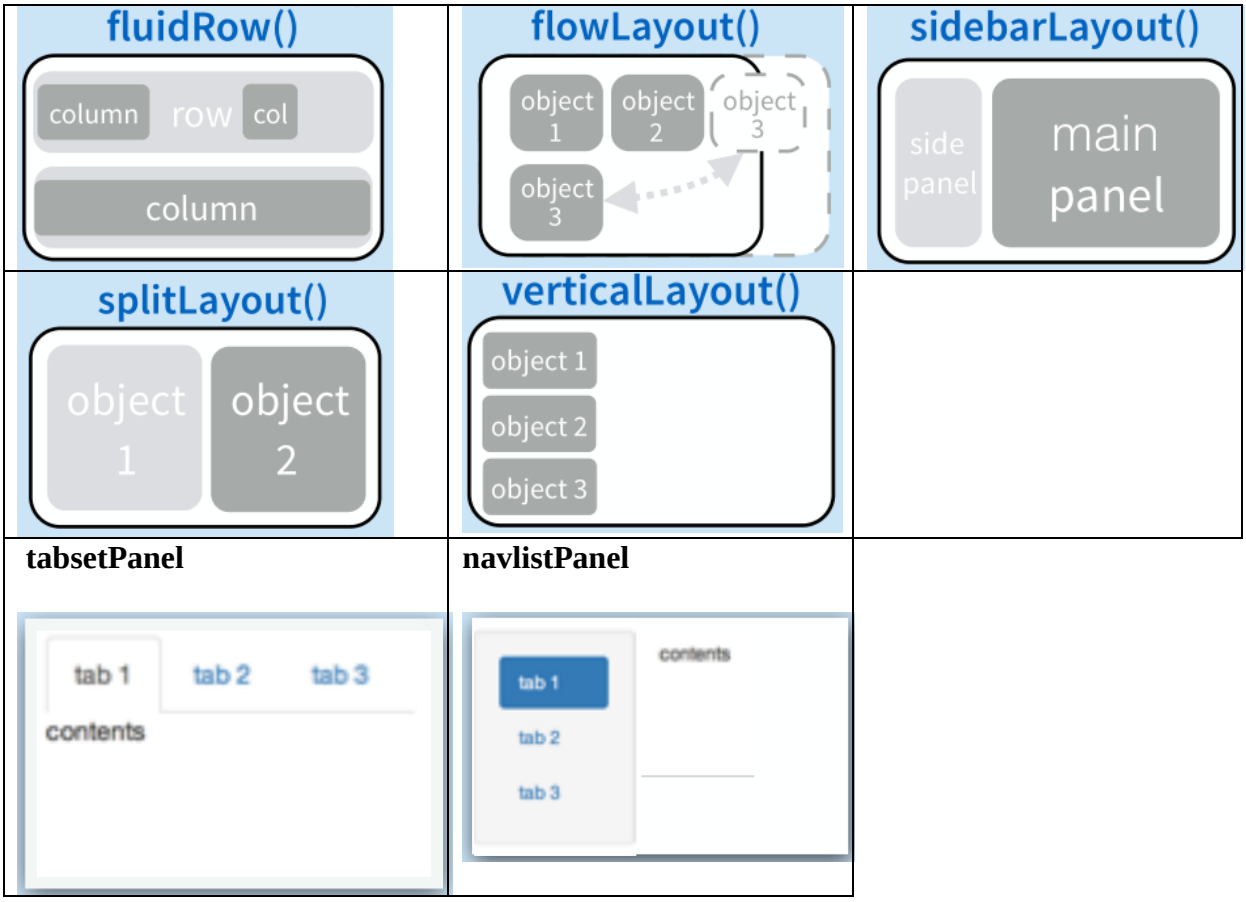
Vir: Hillebrand in Nierhof (2015).

Uporabniški vmesnik (ui.R)

Shiny za videz uporabniškega vmesnika uporablja Twitter Bootstrap, ki je priljubljeno programsko okolje za izgradnjo spletnih strani. Twitter Bootstrap je zbirka orodij za izdelavo spletnih strani in spletnih aplikacij. Vsebuje HTML in CSS oblikovne predloge za videz strani, spletne obrazce, gumbe, navigacijo in druge vizualne elemente (Beely 2016). Preko teh vnaprej definiranih vizualnih elementov se zagotavlja minimalna stopnja vizualne podobe. Ko zaženemo aplikacijo, se R koda pretvori v HTML, CSS in JavaScript kodo. Seveda pa lahko kodo, s katero generiramo spletne strani, napšemo tudi na roke. Tak način omogoča večjo fleksibilnost, vendar zahteva poglobljeno znanje HTML-ja, CSS-ja in JavaScript-e.

Shiny vsebuje številne vnaprej pripravljene predloge za postavitev spletne strani. Najpogostejše uporabljena predloga je sidebarLayout, ki omogoča, da imamo na levi strani vhodne gradnike, na desni strani pa rezultatsko okno, v katerem se rezultati spreminjajo glede na izbrane parametre na vhodnih gradnikih. Poleg sidebarLayout postavitve obstajajo še fluidRow, flowLayout, splitLayout, verticalLayout, tabsetPanel, navlistPanel, navbarPage.

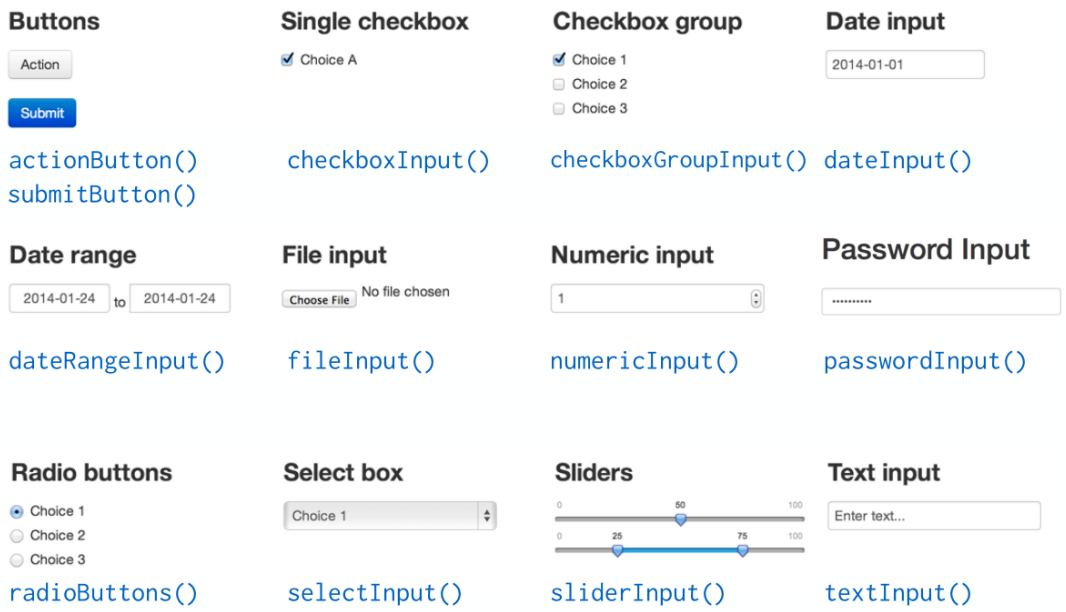
Slika 2.2: Možnosti postavitve strani (layout)



Vir: Chang (2015).

Vnosni (input) gradniki, so tisti del uporabniškega vmesnika, s katerim zagotovimo interaktivnost aplikacije. Shiny ima veliko število različnih gradnikov, preko katerih uporabnik upravlja z aplikacijo. Uporabnik preko vnosnega gradnika izbere poljubne možnosti. Te se posredujejo serverju, ki izvede analitske postopke v skladu z izbiro na vnosnih gradnikih. Server rezultate obdelave posreduje uporabniškemu vmesniku, ki objavi rezultat na spletni strani.

Slika 2.3: Vhodni (input) gradniki uporabniškega vmesnika



Vir: Hillebrand in Nierhof (2015).

Poleg vhodnih gradnikov uporabniškega vmesnika obstajajo tudi izhodni gradniki uporabniškega vmesnika. Izhodni gradniki so rezultati, ki se prikazujejo v uporabniškem vmesniku. Ti rezultati so najpogosteje grafični prikazi, slike, tabele ali besedilo.

Server (server.R)

Server je del aplikacije, na katerem se izvedejo vse analitske operacije. Vse obdelave podatkov in izrisi rezultatov se opravijo na serverju, prikažejo pa na uporabniškem vmesniku v vnaprej pripravljenih izhodnih gradnikih. Server izvede analitske operacije glede na parametre, ki jih dobi iz vhodnih gradnikov. Server je sposoben prebrati spreminjajoče se parametre iz vhodnih gradnikov in jih uporabiti v analitskih operacijah. Vsakič, ko se spremenijo vhodni parametri, server ponovno zažene analize z novimi parametri in vrne ustrezne rezultate. Ta koncept se imenuje reaktivnost. Koncept reaktivnosti je v določenih primerih lahko moteč. Če imamo večje število vhodnih gradnikov in vsi so vključeni v določen proces, bi to pomenilo, da se ta proces zažene ob vsaki spremembi vhodnega gradnika. Za take primere imamo submitButton, ki prepreči izvajanje programske kode. Šele ob kliku na submitButton se zaženejo vse reaktivne funkcije.

Ker se izračuni na serverju opravljajo dinamično, server uporablja posebne funkcije. Te so vezane na vhodne gradnike. Znotraj paketa shiny obstajajo naslednje funkcije za pripravo različnih prikazov:

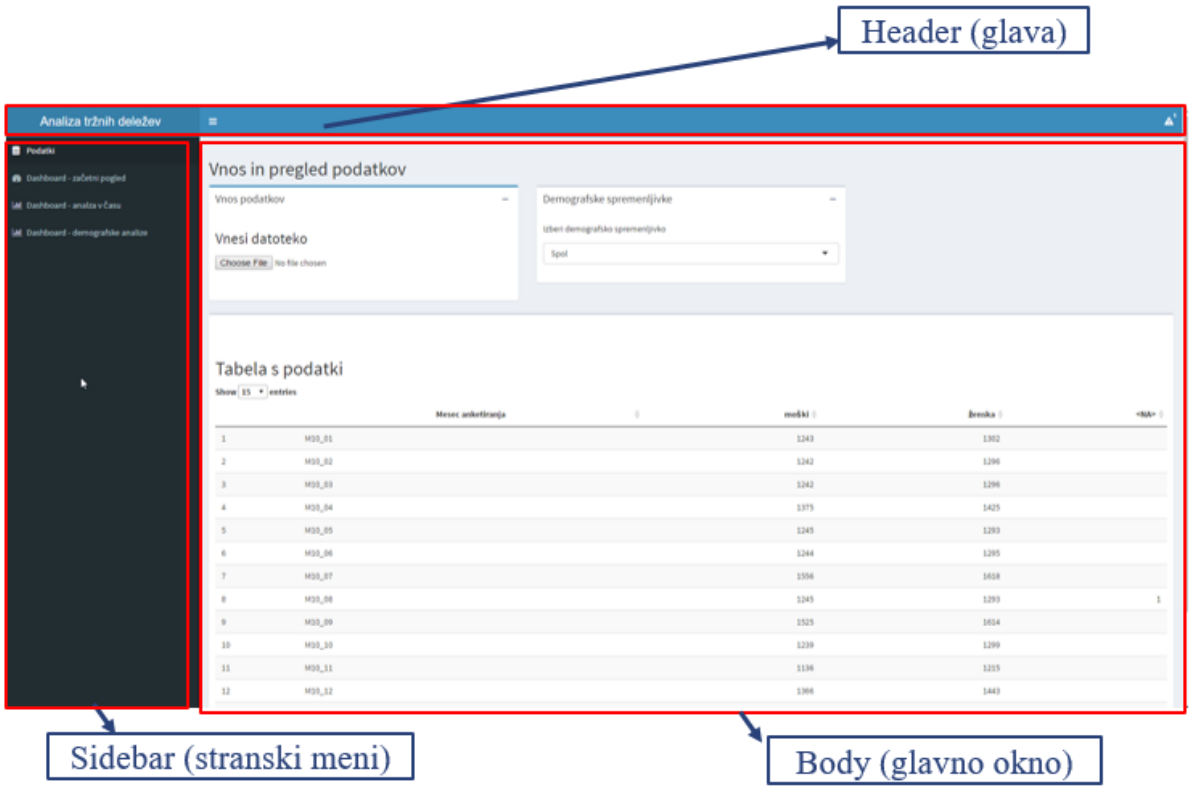
- **renderDataTable**: za prikaz dinamičnih tabel
- **renderImage**: za prikaz slik
- **renderPlot**: za prikaz grafov
- **renderPrint**: za prikaz izpisov iz rezultata analiz
- **renderTable**: za prikaz statičnih tabel
- **renderText**: za prikaz besedila
- **renderUI**: za prikaz dinamično generiranih elementov uporabniškega vmesnika

2.3.3 Paket shinydashboard

Paket shinydashboard omogoča izgradnjo dashboard-ov znotraj paketa shiny in programskega okolja R (Chang 2015). Shinydashoboard uporablja temo AdminLTE, ki je narejena na osnovi Bootstrap3 zbirke za izdelavo spletnih strani in spletnih aplikacij. Čeprav že sam paket shiny omogoča izgradnjo dashboard-ov, je prednost paketa shinydashboard, predvsem v poenotenju in izboljšanju vizualne podobe (Hillebrand in Nierhof 2015). Spletne aplikacija, narejena s paketom shinydashboard, je sestavljena iz treh delov:

- Glava (header)
- Stranski meni (sidebar)
- Glavni del (body)

Slika 2.4: Primer izgleda aplikacije narejene s paketom shinydashboard



V glavi dashboarda imamo lahko poleg imena same aplikacije še spustne (drop-down) menije, ki vsebujejo menije za sporočanje, obveščanje in naloge.

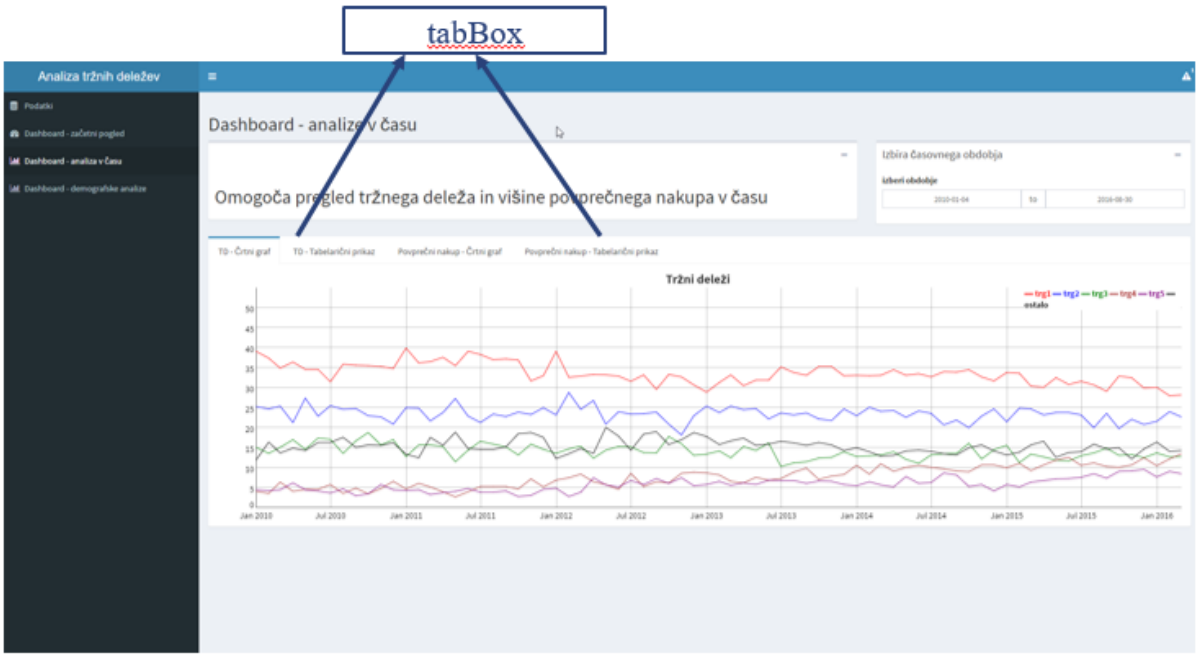
Stranski meni (sidebar) in glavni del (body) aplikacije sta vsebinsko povezana. Stranski meni se uporablja za navigacijo po glavnem delu aplikacije. Odvisno od izbire na stranskem meniju, se odpre izbran del glavne aplikacije. Poleg menijev, ki se uporabljajo za navigacijo po spletni aplikaciji, lahko stranski meni vsebuje vse vhodne gradnike, ki so mogoči v shiny paketu (Hillebrand in Nierhof 2015).

Glavni del spletne aplikacije se uporablja za prikaz izbrane vsebine. Osnovni gradnik glavnega dela je t.i. škatla (Box), ki lahko vsebuje vse vhodne in izhodne gradnike iz paketa shiny. Škatle zagotavljajo strukturo, ki je potreba za prikaz različnih vsebin.

Poleg vseh gradnikov iz paketa shiny, paket shinydashboard vsebuje še tri gradnike, ki so pogosto uporabljeni pri izgradnji dashboardov. Ti dodatni gradniki pripomorejo k bolj strukturiranem videzu aplikacije in jo naredijo prijaznejšo do uporabnika.

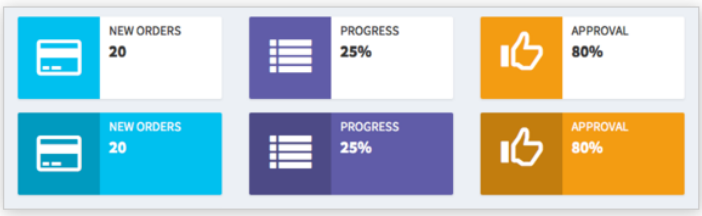
- »**tabBox**«: se uporabi, ko želimo prikazati več različnih vsebin znotraj ene škatle (Box). Ta ima več različnih zavihkov (tab), na katerih so predstavljene različne vsebine. Na vsak zavihek lahko postavimo različne vhodne in izhodne gradnike paketa shiny.

Slika 2.5: tabBox gradnik



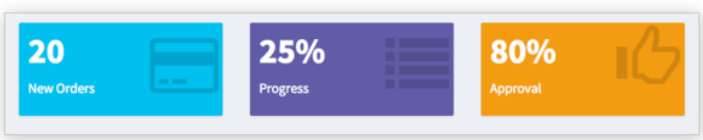
- »**infoBox**« **in** »**valueBox**«: ker je primarni namen dashboarda agregacija podatkov, se kot končni rezultat pogosto prikaže samo ena številka, ki je agregirana vrednost enega ali več indikatorjev (Hillebrand in Nierhof 2015). Za grafični prikaz številke in kratkega teksta v povezavi z ikono uporabimo infoBox ali valueBox gradnik. InfoBox in valueBox sta po vsebini ekvivalentna, razlikujeta se le po grafičnem videzu.

Slika 2.6: infoBox



Vir: Hillebrand in Nierhof (2015).

Slika 2.7: valueBox



Vir: Hillebrand in Nierhof (2015).

2.3.4 Paketi readr, dplyr in tidyr

Skupni imenovalec vseh treh zgoraj omenjenih paketov je, da se pogosto uporabljajo skupaj pri pripravi oz. obdelavi podatkov. Paket readr se uporablja za branje tekstovnih datotek, paketa dplyr in tidyr pa za pripravo podatkov.

2.3.4.1 Paket readr

Readr je paket za hitro prebiranje in shranjevanje tekstovnih datotek, v katerih so podatki urejeni v tabelarični obliki in ločeni z izbranim ločilom (Wichkham in drugi 2016). Tabelarična oblika pomeni, da so v vrsticah običajno enote merjenja, v stolpcih pa spremenljivke. Vrstice običajno pomenijo enote merjenja, stolpci merjene spremenljivke. Čeprav so funkcije za prebiranje tekstovnih datotek vključene med osnovne funkcije R programa, sem raje uporabil funkcije iz paketa readr. V primerjavi s funkcijami iz osnovnega paketa, te omogočajo (privzeto po Wickham 2016):

- hitrejše branje podatkov (do deset krat, z velikostjo podatkov se povečuje prihranek)
- konsistentno poimenovanje v vseh funkcijah (col_names, col_types in ne header, colClases)
- enako delovanje vseh funkcij, ne glede na nastavljene regijske nastavitve programa R. Privzeta regijska nastavitev je ameriška, ki jo lahko spremenimo.
- detekcijo vrstice, v kateri se pojavi napaka. Če se v dokumentu v zapisu pojavi napaka (nepravilno število znakov, ki se uporabljajo kot ločilo, nepravilni format zapisa), funkcija izpiše vrstico zapisa, kjer se je le-ta pojavila.
- generiranje »tibble« podatkovnih struktur. Rezultat branja tekstovnih datotek je vedno podatkovna struktura tibble. Tibble je podatkovna struktura, zelo podobna podatkovnemu okvirju. Od podatkovne strukture se razlikuje po tem, da se format znakovnih spremenljivk ob branju nikoli ne spremeni v faktorje, ne omogoča

poimenovanja vrstic, imena spremenljivk pa se lahko razlikujejo od standardnega poimenovanja.

2.3.4.2 Paket dplyr

Dplyr je paket, ki vsebuje funkcije za učinkovito pripravo podatkov (Wickham in Francois 2016). Veliko število funkcij za pripravo podatkov je zajeto že v osnovnem paketu programa R, še večje število funkcij za pripravo podatkov pa je razpršeno po različnih paketih. Veliko število različnih funkcij, ki se uporabljajo za iste ali skoraj iste naloge, povzroča težave s kompatibilnostjo delovanja.

Dplyr je paket, ki se uporablja za pripravo podatkov, zapisanih v tabelarični obliki. Iste funkcije se uporabljajo za podatke, urejene v obliki podatkovnih struktur, za data.table, za tibble podatkovne strukture ali za podatkovne baze. Za pripravo podatkov, ki so shranjeni v podatkovnih bazah, ni treba več uporabiti namenskih paketov, temveč lahko uporabimo paket dplyr.

Poleg večje kompatibilnosti je paket dplyr hitrejši po delovanju od ostalih podobnih paketov, ker je paket napisan v C++ jeziku in ima poenoten uporabniški vmesnik.

Glavne funkcije, ki se nanašajo na obdelavo podatkov, so:

- izbira enot (ukaz filter)
- izbira spremenljivk (ukaz select)
- tvorjenje novih spremenljivk (ukaz mutate)
- agregacije spremenljivk (ukaz summarise)
- grupiranje enot (ukaz group_by)
- združevanje podatkovnih struktur (ukazi iz družine _join)

Ker lahko z znakom %>% (pipe) združimo več korakov priprave podatkov v zaporedje korakov, lahko s funkcijami iz paketa dplyr hitro in učinkovito pripravimo podatke. Programska koda, zapisana v zaporedju korakov, ki so združeni z znakom %>%, je jasna in razumljiva.

2.3.4.3 Paket tidyr

Tidyr je paket, ki vsebuje funkcije za učinkovito spreminjanje strukture podatkov, ki so v tabelarični obliki, ki pomeni, da so v vrsticah običajno enote merjenja, v stolpcih pa spremenljivke. Taka oblika pomeni, da so podatki urejeni »tidy«(Wickham 2016). V praksi

podatki pogosto niso urejeni. Lahko je ena enota merjenja zapisana v več vrsticah ali imamo v eni spremenljivki združene vrednosti več teh oz. imamo odgovore ene spremenljivke razbite v več spremenljivk. Taka oblika podatkov pomeni, da so le-ti neurejeni, »messy«.

Zaradi različnih analiz in priprav je treba podatke tudi transponirati. Pri določenih grafičnih funkcijah je potrebno, da so podatki za risanje grafov urejeni na točno določen način. Zato je treba zamenjati vrstice in stolpce v podatkovni strukturi ali pa spremeniti dimenzijo le-te. Sprememba dimenzije podatkovne strukture pomeni, da spremenimo število stolpcev in vrstic v njej. Tabelo iz široke »wide« oblike pretvorimo v dolgo »long« obliko. Prva ima večje število stolpcev in manjše število vrstic, druga pa večje število vrstic in manjše število stolpcev.

Glavne funkcije, ki jih omogoča paket tidyr so (Wickham 2016):

- združevanje stolpcev v vrstice (ukaz gather). Ukaz spremeni podatkovno strukturo tako, da stolpce združi v vrstice. Nova struktura ima manj stolpcev in več vrstic. Taka struktura podatkov omogoča enostavno agregacijo podatkov.
- združevanje vrstic v stolpce (ukaz spread). Ukaz spremeni podatkovno strukturo tako, da iz vrstic naredi stolpce. Nova struktura ima več stolpcev in manj vrstic. Taka struktura podatkov je primerna za prikaz podatkov.
- razdelitev ene spremenljivke v več spremenljivk (ukaz seperate). Ukaz razdeli vrednosti v določeni spremenljivki v dve ali več novih spremenljivk.
- združevanje več spremenljivk v eno novo spremenljivko (ukaz unite). Ukaz združi vrednosti več spremenljivk v eno novo spremenljivko.

2.3.5 Paketi ggplot2, formattable, dygraphs in DT

Predstavitev rezultatov v grafični obliki je zadnja faza v procesu analize podatkov. Rezultate analiz je treba predstaviti na vizualni način, za kar se uporabljajo paketi, kot so ggplot2, formattable, DT in dygraphs. Paket ggplot2 omogoča izdelavo različnih grafičnih prikazov rezultatov. Paket formattable se uporablja za izdelavo poljubno oblikovanih tabel. Za prikaz dinamičnih podatkovnih struktur se uporablja paket DT. Dygraph je paket, ki omogoča izdelavo interaktivnih grafov časovnih serij.

2.3.5.1 ggplot2

Paket ggplot2 se uporablja za različne grafične prikaze (Wickham 2016). Grafični prikaz temelji na teoriji »slovnice grafike« (Wilkinson 2005). Tako kot daje slovnica jeziku pravila,

slovnica grafike pomeni, da se naj grafični prikazi naredijo po določenih standardnih pravilih. Osnovni koncept risanja grafov s paketom ggplot2 je, da lahko ustvarimo vsak grafični prikaz iz osnovnih komponent, ki jih združimo v enotno plast (layer) (Teutonico 2015). Za izdelavo grafov potrebujemo:

- **podatke**: podatki, ki jih želimo prikazati, morajo biti v pravilni obliki, kar pomeni, da so podatki shranjeni v podatkovnem okvirju ali drugi podobni strukturi.
- **geometrični objekt (geom)**: geometrični objekt določa videz samega končnega grafa. Geometrični objekti so histogrami, črtni grafi, stolpčni grafi in številni drugi. Izbira geometričnega objekta je odvisna od vrste podatkov, ki jih želimo prikazati. Za prikaz ene spremenljivke se uporabijo drugačni geometrični objekti, kot pa za prikaz dveh spremenljivk.
- **statistične povzetke (stats)**: določijo, kako želimo podatke povzeti za prikaz. Kot primer: za histogram je treba podatke združit v razrede. Brez tega koraka histograma ne bomo mogli narisati.

Seveda pa v paketu ggplot2 obstajajo še številne druge funkcije, s katerimi lahko zelo natančno določimo videz grafa.

2.3.5.2 Paketi formattable, dygraphs, DT

Paketi formattable, dygraphs in DT se uporabljajo za vizualni prikaz podatkov. Ti paketi vrnejo rezultate v HTML jeziku, ki v kombinaciji s CSS in JavaScript-o omogoča interaktivnost in dinamičnost prikazov.

Paket formattable se uporablja za prikaz in izdelavo poljubno oblikovanih tabel. Omogoča izdelavo poljubno oblikovanih vektorjev in poljubno oblikovanje podatkovnih okvirov (Kun in Russel 2016). Za oblikovanje podatkovnih okvirjev se uporablja CSS.

Paket dygraphs se uporablja za izdelavo interaktivnih časovnih serij. Ta paket je R-ov vmesnik do grafične dygraphs java skripte. Omogoča različne interaktivne prikaze časovnih vrst (Vanderkam in drugi 2016).

Paket DT se uporablja za prikaz dinamičnih podatkovnih struktur. R-ov paket DT je vmesnik do java skript knjižnice DataTables. R-ove podatkovne strukture (matrice ali podatkovni okviri) se lahko prikažejo kot dinamične tabele v HTML zapisu, ki jih je mogoče filtrirati, sortirati in oblikovati na različne načine (Xie 2016).

2.3.6 RStudio

RStudio je integrirano razvojno okolje (integrated development environment – IDE) programiranje v jeziku R. Za programiranje v R-ju obstaja veliko različnih razvojnih okolij (RGui, Eclipse, Vim, Zeus..), vendar ima RStudio najboljšo integracijo s paketoma shiny in paketom shinydashboard [2].

V okolje RStudio so integrirane različne funkcionalnosti, ki olajšajo delo pri kompleksnejših projektih. Programski vmesnik je razdeljen na več delov in tako omogoča soočasno pregled nad različnimi funkcijami programa (Teutonico 2015):

- **skriptno okno**: omogoča pregled nad skriptami in vsebuje različne funkcije, ki olajšajo pisanje skript v jeziku R.
- **konzolno okno**: izpisujejo se rezultati skript, ki so bile zagnane v skriptnem oknu. Konzolno okno omogoča vnos skript in njihov zagon.
- **paketno okno**: v njem je naveden seznam vseh instaliranih paketov. Uporablja se za inštalacijo novih paketov ali za posodabljanje obstoječih.
- **okno s pomočjo**: v njem se izpisujejo zahtevane teme pomoči (paketi ali izbrane funkcije)
- **grafično okno**: izrisujejo se grafični prikazi analiz. RStudio omogoča, da se grafični prikazi shranjujejo in omogoča listanje med shranjenimi grafičnimi prikazi.
- **okno z zgodovino**: v njem se shranjujejo vsi zagnani ukazi iz skriptnega ali konzolnega okna.
- **okoljsko okno**: v njem se izpisujejo vsi objekti, ki jih uporabljamo pri analizah. Ti objekti so lahko vektorji, matrike, polja, seznami, podatkovni okvirji in uporabniško definirane funkcije.
- **brskalnik po datotekah**: to okno je brskalnik datotek. Le-ta je uporaben zlasti pri projektih, kjer imamo na enem mestu shranjene tako skripte podatkov kot tudi mape in podmape, v katerih imamo shranjene tako datoteke, ki jih uporabljamo kot vhodne, kot tudi datoteke, ki so rezultat analiz in obdelav podatkov.
- **pogledno okno**: integriran spletni brskalnik, v katerem se izpisujejo rezultati analiz, narejeni z modernimi paketi, ki rezultate izvozijo v HTML zapis. Ta je pogosto

[2] Avtorji razvojnega okolje RStudio in paketa shiny in shinydashboard so iz istega podjetja, ki se imenuje R Studio.

oblikovan s CSS in v povezavi z java skriptami omogoča dinamičnost in interaktivnost prikazov.

2.4 Metodologija razvoja orodij za avtomatizacijo poročanja

Pri metodologiji razvoja programske rešitve sem izhajal iz metodologije podatkovnega rudarjenja. Le-ta obsega naslednje korake: razumevanje problema oz. določitev ciljev, razumevanje podatkov, priprava podatkov, modeliranje, evaluacija in uvajanje ali integracija rešitve (Tsiptsis in Chorianopoulos 2009). Razvoj orodij za avtomatizacijo poročanja in metode podatkovnega rudarjenja imajo številne skupne točke.

Metodologijo razvoja orodij za avtomatizacijo poročanja sem strnil v naslednje korake:

- določitev cilja avtomatizacije
- razumevanje podatkov
- obdelava podatkov, analize in vizualizacija
- avtomatizacija procesa

2.4.1 Cilj avtomatizacije

Določitev jasnega cilja avtomatizacije je najpomembnejši korak v procesu avtomatizacije. Brez jasno določenega cilja proces avtomatizacije ne bo uspešen (Tsiptsis in Chorianopoulos 2009). Določitev cilja za sabo potegne preostale korake v razvoju avtomatizacije. Že v cilju avtomatizacije mora biti določeno, kakšen bo videz aplikacije in na kakšen način se bodo prikazovali rezultati. Cilj določa pripravo podatkov, vrsto analiz in vizualizacijo podatkov. Če se med razvojem avtomatizacije spremeni cilj raziskave, je ponavadi treba spremeniti celotni sistem priprave podatkov, analiz in vizualizacije podatkov.

2.4.2 Razumevanje podatkov

Razumevanje podatkov pomeni, da poznamo pomen podatkov, ki jih imamo na voljo, in način, kako so ti podatki zapisani v bazi. Poleg pomena samih spremenljivk moramo razumeti tudi medsebojni vpliv več spremenljivk. Podatki so zbrani na različne načine in pod različnimi pogoji. Preden se lotimo obdelave podatkov, moramo natančno poznati pomen vsake spremenljivke. Tudi manjkajoči podatki utegnejo imeti pomembno vlogo pri razumevanju podatkov. Brez razumevanja podatkov le-teh ne moremo uspešno obdelati.

2.4.3 Priprava podatkov, analiza rezultatov in vizualizacija

Priprava podatkov pomeni, da slednje ustrezno pripravimo za nadaljnje analize. Začetni podatki običajno niso v ustrezni obliki, ki bi omogočala takojšnje analize. Preden jih uporabimo v analizah, je treba podatke obdelati. Priprava se največkrat nanaša na brisanje neustreznih vrednosti in na računanje novih izvedenih spremenljivk. Samo ustrezno pripravljeni podatki se lahko uporabijo pri analizah in vizualizaciji podatkov. Pri analizah podatkov se izračunajo različne statistike, ki se jih z različnimi načini vizualizacije predstavi končnemu uporabniku.

2.4.4 Avtomatizacija procesa

Avtomatizacija procesa je zadnja stopnja v izgradnji avtomatiziranih orodij. Glede na cilje, razumevanje, priprave podatkov, analiz in vizualizacije se izberejo ustrezni načini avtomatizacije. S pomočjo različnih programskih jezikov se zgradi aplikacija z grafičnim vmesnikom, ki omogoča dinamično prikazovanje rezultatov. V aplikacijo se vgradijo podatki, proces priprave, analiziranja, vizualizacije podatkov in določena stopnja interaktivnosti. Le-ta omogoča, da si uporabnik izbere poljubne analize.

3 Izdelava spletne aplikacije

3.1 Zbiranje podatkov

3.1.1 Merski instrument

Merski instrument za zbiranje podatkov je telefonski vprašalnik, ki je sestavljen iz treh delov. V prvem anketiranca vprašamo, v katerih vseh trgovinah so prejšnji dan nakupovali vsakdanje izdelke iz široke potrošnje. V drugem delu jih za vsako trgovino, v kateri so prejšnji dan nakupovali, vprašamo, ali so kupovali izdelke iz kategorij hrane, čistila in kozmetike. V tretjem koraku pa vprašamo, koliko denarja v evrih so porabili za izdelke iz zgoraj navedenih kategorij v vsaki od trgovin, ki so jo prejšnji dan obiskali.

3.1.2 Zbiranje in priprava podatkov (čiščenje, uteževanje)

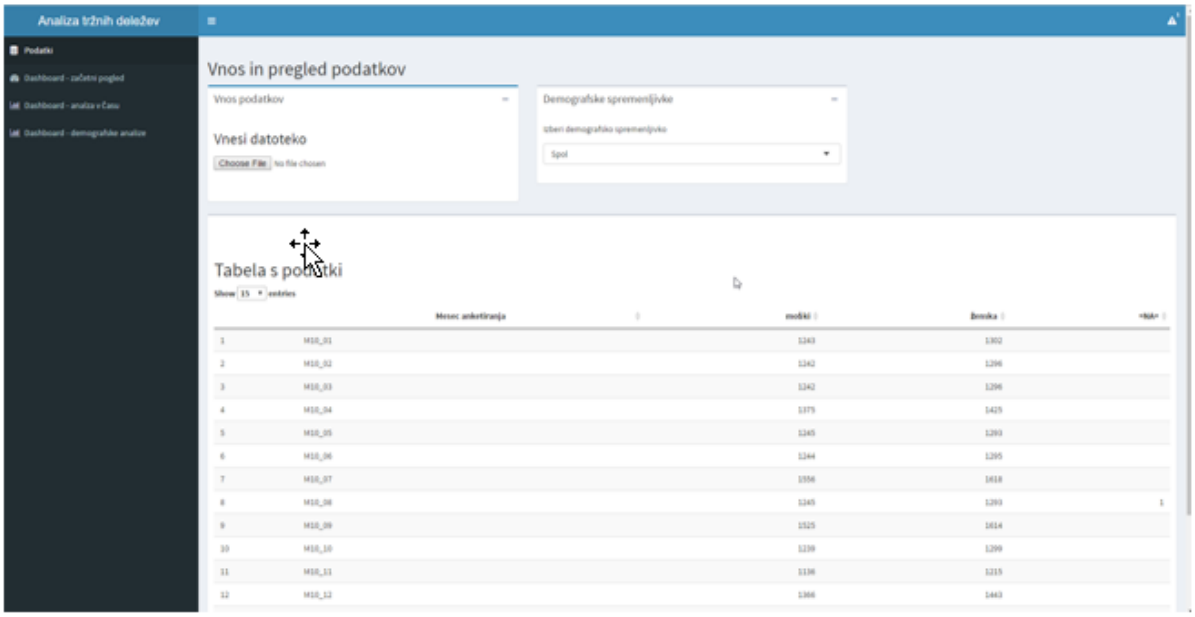
Podatki so zbrani s telefonskim anketiranjem. Vsak dan se pokliče 60 gospodinjstev. Primerne so osebe v starosti od 15 do 75 let, ki so nazadnje v gospodinjstvu imele rojstni dan. Po končanem anektiranju se podatki utežijo z rakin metodo po kombinaciji spola in starosti, izobrazbi in regiji, da dobimo reprezentativnost na ravni države. Ker se anketa izvaja po telefonu, je zaradi starostno popačenega vzorca uteževanje nujno potrebno.

Iz podatkov o porabi je treba izbrisati še vse neveljavne in manjkajoče vrednosti. Pogosto se anketiranci ne spomnijo, koliko so prejšnji dan porabili pri izbranem trgovcu na izbrani kategoriji, zato treba te vrednosti pred začetkom analiz izločiti.

3.2 Delovanje spletne aplikacije

Spletna aplikacija za pregled podatkov o tržnem deležu deluje tako, da na podlagi trenutnih podatkov in izbranih parametrov vrne zahtevane rezultate.

Slika 3.1: Začetni videz spletne aplikacije

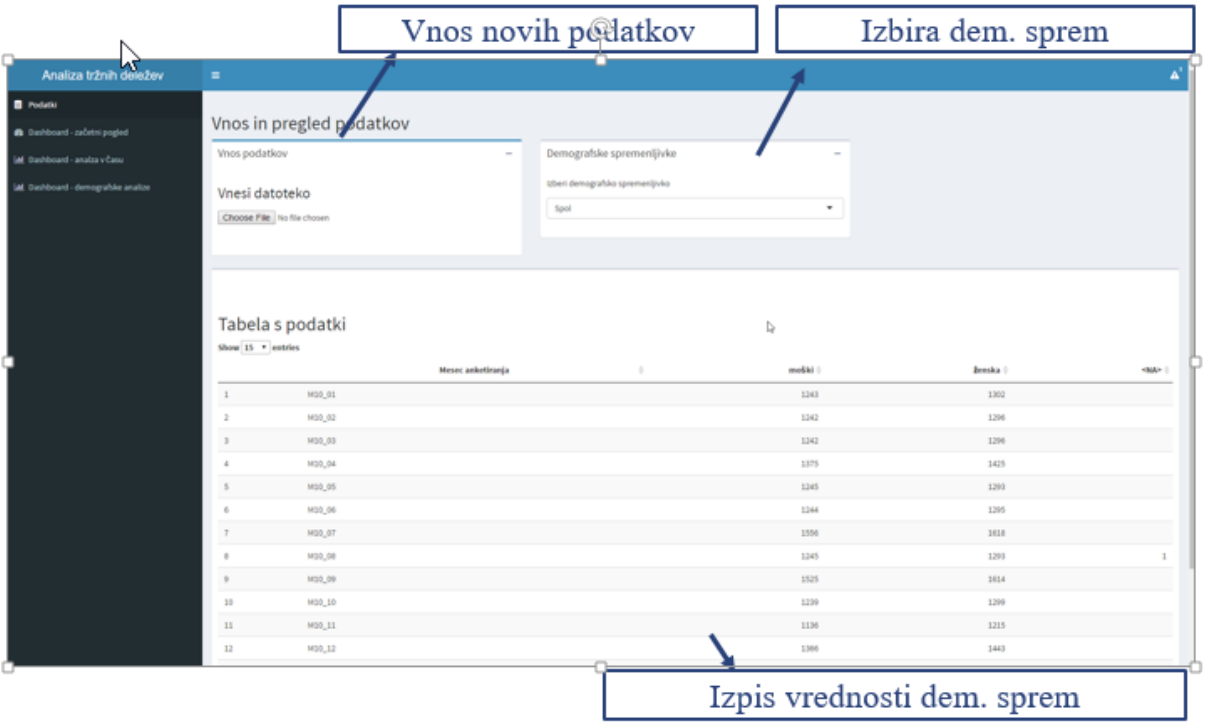


Aplikacija je razdeljena na tri funkcijske sklope:

- Import oz. zajem podatkov
- Osnovni prikaz podatkov
- Podrobnejši prikaz podatkov

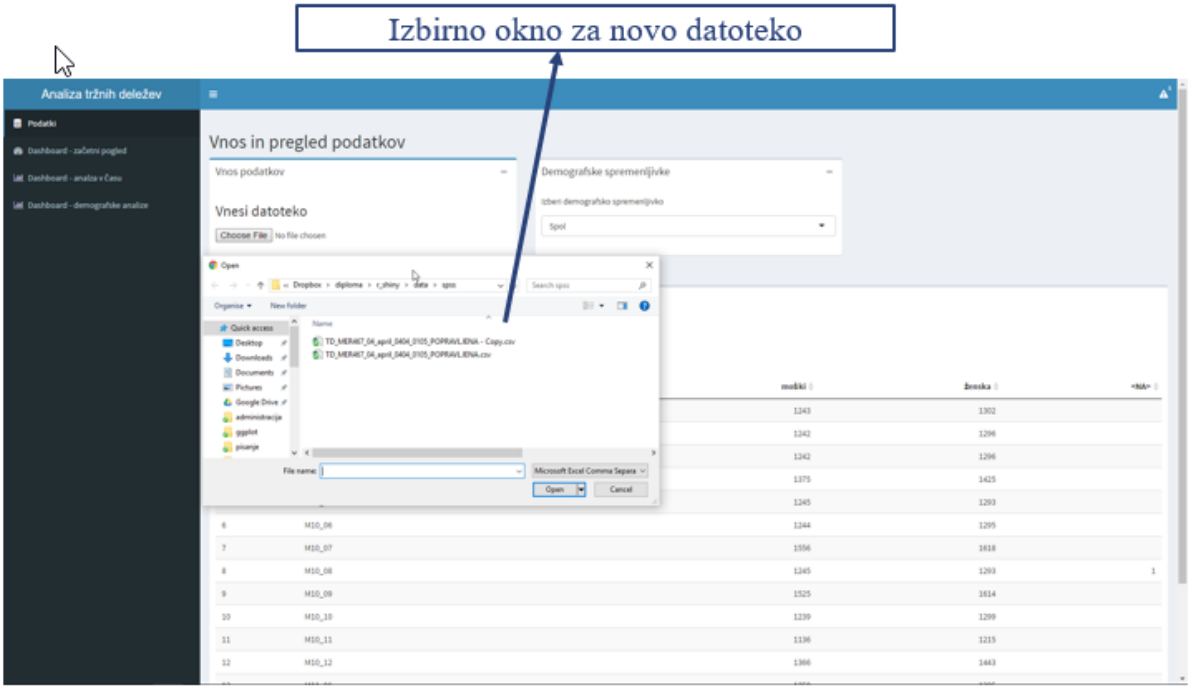
3.2.1 Zajem podatkov

Slika 3.2: Uporabniški vmesnik za vnos in pregled podatkov



Uporabniški vmesnik omogoča pregled podatkov, ki so v aplikaciji, in omogoča dodajanje novih. S klikom na meni Vnesi datoteko se odpre novo vnosno okno, ki omogoča izbiro poljubne datoteke.

Slika 3.3: Izbirno okno za vnos nove datoteke



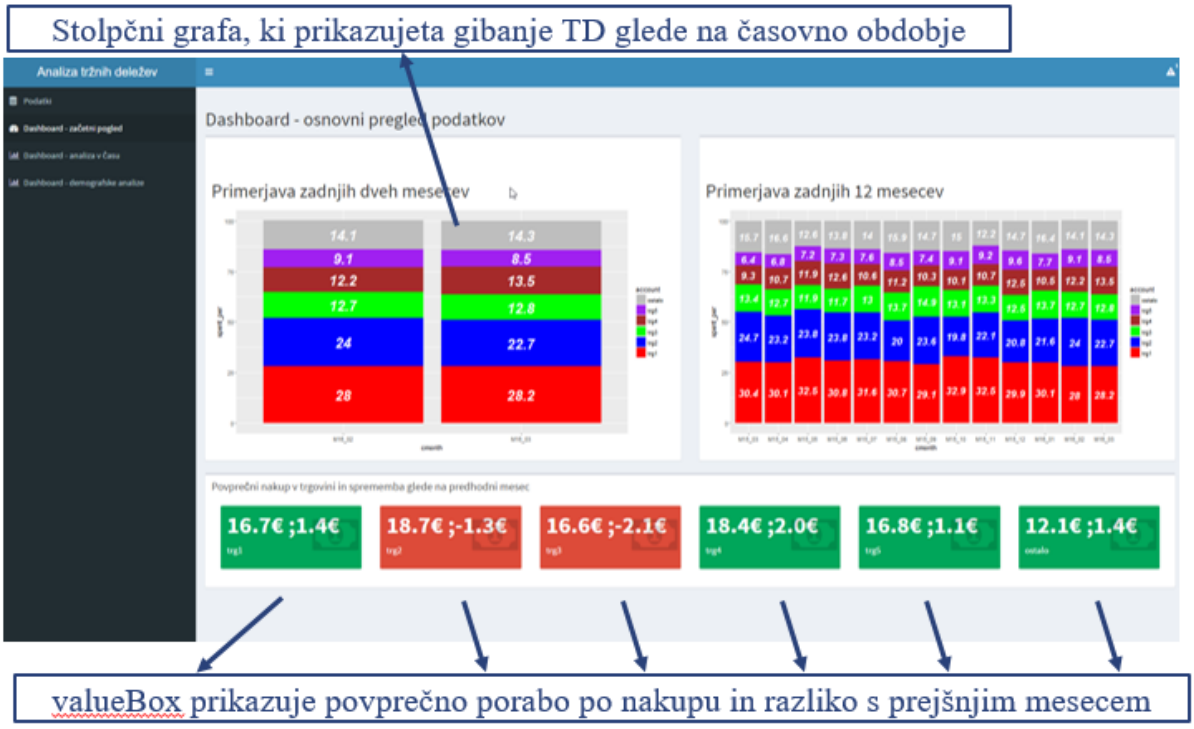
Vnos nove datoteke je povezan s čiščenjem podatkov in pripravo le-teh za analize. Novo anketno datoteko je treba ustrezno pripraviti. To pomeni, da je treba datoteko očistiti neprimernih vrednosti in jo pripraviti v tako obliko, da bo skladna z obstoječo strukturo podatkov. Podrobneje si bomo proces urejanja podatkov pogledali v podpoglavju o izgradnji spletne aplikacije.

Uporabniški vmesnik omogoča osnoven pregled vrednosti izbranih demografskih spremenljivk po mesecih anketiranja. S tem osnovnim pregledom demografskih spremenljivk lahko vidimo, ali so podatki skladni s prejšnjimi meritvami.

3.2.2 Osnovni prikaz podatkov

Na osnovnem prikazu podatkov so prikazani podatki o tržnem deležu in o povprečni porabi pri enem nakupu v izbrani trgovini. Osnovni pregled podatkov je namenjen najosnovnejšemu pregledu rezultatov. Tržni delež prikazujemo v dveh časovnih obdobjih. V levem delu grafa je prikazan TD po trgovcih glede zadnja dva meseca, za katera imamo podatke v bazi. V desnem grafu je prikazano enoletno časovno obdobje. Z enoletnim prikazom dobimo vpogled v trend gibanja podatkov. Za prikaz tržnih deležev sem uporabil naložene stolpčne grafe, v katerih, je prikazano razmerje deležev med posameznimi trgovinami in celoto. Ker se primerjajo deleži, je skupna vsota enaka 100.

Slika 3.4: Uporabniški vmesnik za osnovni prikaz podatkov



ValueBox v spodnjem delu aplikacije označuje velikost povprečnega nakupa po izbrani trgovini. Številka na levi strani pove, koliko je znašala vrednost povprečnega nakupa, številka na desni pa razliko glede na prejšnji mesec. V primeru, da je razlika manjša od -1€, se valueBox obarva v rdeče. Če je razlika v povprečni velikosti nakupa večja od -1€ in manjša od 1€, se valueBox obarva v rumeno. Če znaša razlika v povprečni porabi med trenutnim in prejšnjim mesecem več kot 1€, se valueBox obarva v zeleno barvo.

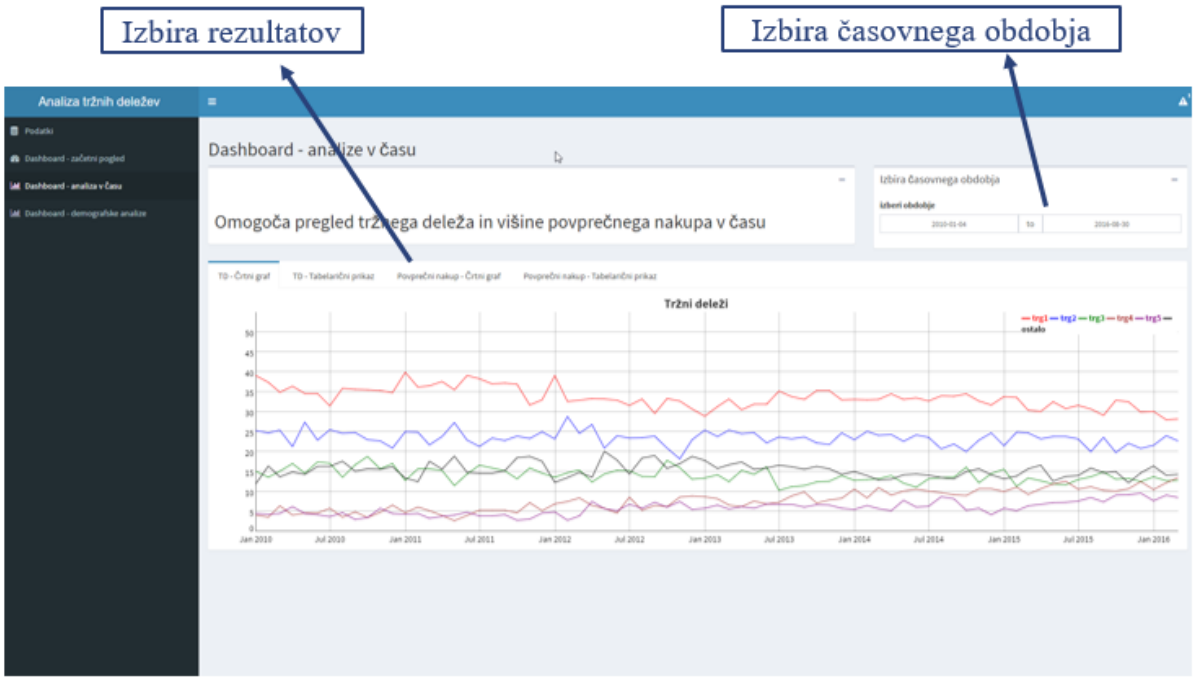
Pri tem načinu prikaza nimamo možnosti izbire časovnega obdobja. Podatki so zmeraj prikazani za zadnji mesec, za katerega imamo podatke. Pri valueBox prikazih in pri levem stolpičnem grafu se kot predhodna časovna točka uporabi predhodni mesec. Pri desnem stolpčnem grafu je časovno obdobje dolgo 12 mesecev.

Vsi zgoraj omenjeni prikazi so narejeni dinamično. V primeru posodobitve podatkov se bodo prikazi spremenili. Novejši podatki bodo spremenili izračun zadnjega meseca in prikazi se bodo samodejno osvežili.

3.2.3 Podrobnejši prikaz podatkov

Podrobnejši prikaz omogoča večjo fleksibilnost pri izbiri prikazov. Uporabnik lahko izbere različne načine prikaza in časovno obdobje, za katerega se rezultati prikazujejo. Prikazana sta dva rezultata: tržni delež in povprečna velikost nakupa v izbrani trgovini.

Slika 3.5: Uporabniški vmesnik za analize podatkov v času



Za prikaz tržnega deleža v času sem uporabi črtni graf, ki se običajno uporablja pri prikazu časovnih vrst. S črtnim grafom prikažemo spremembe izbrane spremenljivke skozi čas, obenem pa omogoča prepoznavanje vzorcev, trendov in prelomnic (Churchill in Iacobucci 2002).

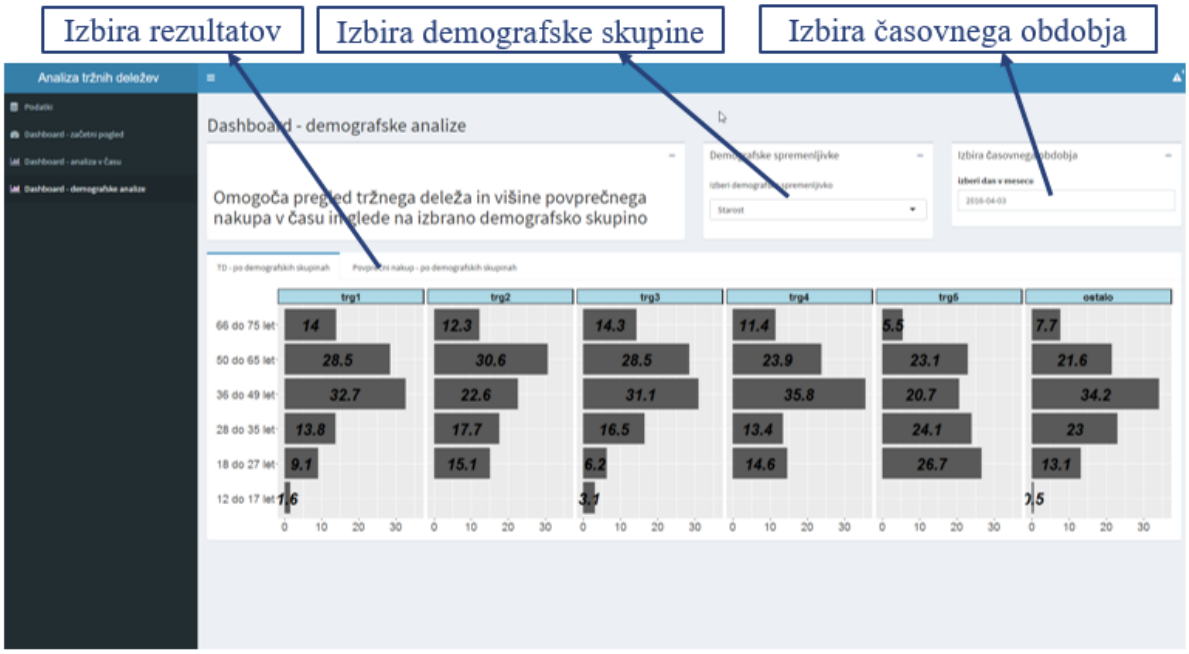
S spreminjanjem časovnega obdobja si uporabnik sam izbere časovni interval. Ker je graf narejen s paketom dygraphs, klik z miško na graf povzroči, da se v zgornjem desnem kotu grafa izpišejo vrednosti izbran mesec. Če se z miško premikamo po grafu (po različnih mesecih), se vrednosti dinamično spreminjajo.

Uporabnik ima možnost, da si izbere tabelarični prikaz podatkov. Le-ta prikaže številke v tabeli. Tabelarični prikaz je uporaben tam, kjer želimo imeti vpogled v natančne številke.

Uporabnik ima možnost, da si izbere med črtnim grafom ali tabelaričnim prikazom tržnega deleža oz. povprečne vrednosti nakupa pri izbranem trgovcu. Različni rezultati so predstavljeni na različnih zavihkih.

Dashboard – demografske analize omogoča analizo deleža porabe v izbrani trgovini in povprečno višino nakupa glede na izbrane demografske spremenljivke. Uporabnik lahko izbira med demografskimi spremenljivkami in med časovnim obdobjem. Slednje znaša en mesec in ga ni mogoče povečati. Rezultat se izpisuje v spodnjem delu okna. Tržni delež ali povprečna velikost nakupa glede na izbrano demografsko spremenljivko se izpisujeta v različnih zavihkih.

Slika 3.6: Uporabniški vmesnik za demografske analize



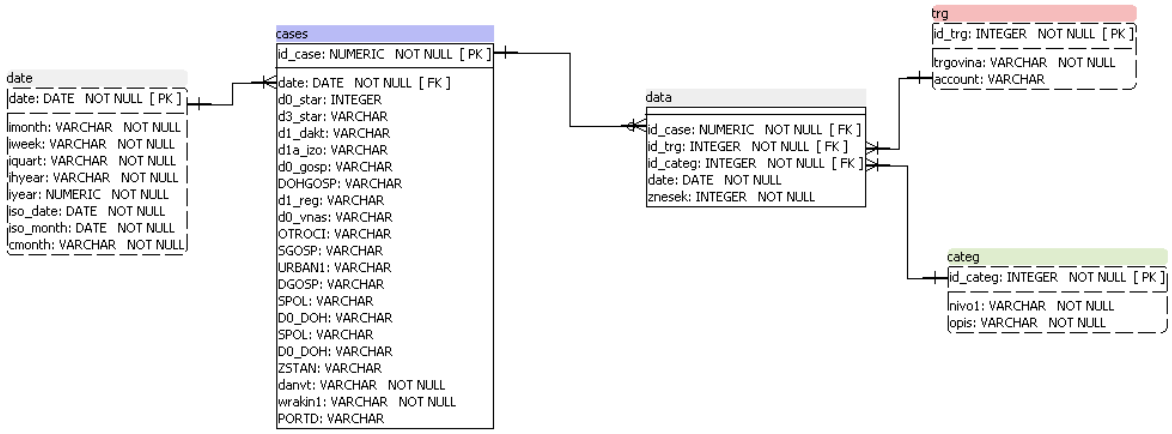
Za prikaz demografskih podatkov po trgovini sem izbral stolpčni graf. Stolpčni graf prikaže razmerje med izbranimi demografskimi skupinami.

3.3 Empirična izgradnja aplikacije

Aplikacija je bila narejena v programskem jeziku R. Paketa shiny in shinydashboard sta bila uporabljena za izgradnjo okvirja aplikacije. Za pripravo podatkov in prikaz rezultatov so bili uporabljeni različni paketi. Za pripravo podatkov so bili uporabljeni paketi, readr, tidyr in dplyr. Paket ggplot2 sem uporabo za izris določenih grafov, paket formattable za izris formatiranih (obarvanih) tabel, paket dygraphs za prikaz dinamičnih črtnih grafov, paket DT pa za prikaz dinamičnih tabel.

3.3.1 Opis strukture podatkov

Slika 3.7: Struktura podatkov



Podatki, ki so potrebni za delovanje aplikacije, so shranjeni v R-u lastni .RDS datoteki. Vsaka tabela je shranjena v lastni .RDS datoteki. Zapis vsake tabele podatkov v svojo .RDS datoteko je potreben zaradi osveževanja podatkov. Osveževanje zapisov v eni tabeli vpliva samo na točno določeno .RDS datoteko. V primeru, da bi imeli vse tabele shranjene v eni .RDS datoteki, bi ob spremembi ene same tabele morali zapisati vse tabele v novo .RDS datoteko.

Shranjevanje podatkov v .RDS datoteki sem izbral zaradi količine podatkov in učinkovitosti delovanja. Čeprav so v aplikaciji zajeti podatki od januarja 2010, količina ni tako velika, da bi zahtevala uporabo podatkovne baze. »Splošno pravilo pri izbiri načina shranjevanja podatkov za analizo s programom R je ali je mogoče shraniti podatke v spomin računalnika. Če je to mogoče, potem ni potrebe po uporabi podatkovnih baz« (Wickham in Francois 2016).

V procesu izgradnje podatkovne baze sem testiral tudi zapis podatkov v SQLite podatkovno bazo. Vendar pa zapis podatkov v podatkovno bazo SQLite[3] in R paket za delo s podatkovnimi bazami, RSQLite[4] nista izboljšala delovanje aplikacije. Sta pa povečala kompleksnost analiz in pojavile so se težave pri branju podatkov iz SQLite podatkovne baze, ki so posledica hroščev v RSQLite knjižnici.

Na sliki 3.7 so podatki predstavljeni v obliki podatkovne baze samo za lažjo predstavitev in enostavnejše razumevanje podatkov. V tabelah so zajeti naslednji podatki:

- **data**: shranjeni so samo podatki o nakupih. O anketirancu je shranjen samo identifikacijski podatek, drugi podatki pa se nanašajo na datum, trgovino in kategorijo

[3] https://sqlite.org/
[4] https://cran.r-project.org/web/packages/RSQLite/index.html

nakupa. Tabela ima transakcijski način zapisa, kar pomeni, da je ena oseba lahko v tej bazi večkrat zapisana. To se zgodi v primeru, ko oseba opravi več nakupov v enem dnevu ali v eni trgovini kupi izdelke iz več produktnih kategorij.

- **cases**: shranjeni so demografski podatki o anketirancu. Najvažnejši podatki o anketirancu so spol, starost, izobrazba, delovna aktivnost, regija, število otrok. Poleg demografskih podatkov je tu shranjena še spremenljivka, ki jo uporabljamo za uteževanje. Tabela ima »širok« način zapisa, kar pomeni, da je vsaka oseba v njej zapisana samo enkrat. Identifikacijskem podatku sledijo spremenljivke, ki se nanašajo na preostalo demografijo.
- **date**: shranjeni so podatki o datumskih spremenljivkah, ki jih potrebujemo za agregacijo časovnih podatkov. Osnovna merska enota za izvajanje analiz je mesec. Mesec, ki ga uporabljamo za prikaz analiz, ni enak koledarskemu mesecu. Zaradi specifike trgovinske branže (veliki nakupi se delajo ob vikendih, različni trgovci imajo popuste ob različnih dnevih tedna), merski mesec zajema obdobje od ponedeljka do nedelje. Vsak merski mesec je lahko dolg štiri ali pet tednov in se zato ne ujema s koledarskim mesecem. Osnovno pravilo, ki ga uporabimo pri razvrščanju tedna v mesec, je pravilo četrtka. Če četrtek v tednu pade v novi mesec, ponedeljek, torek in sredo iz prejšnjega meseca štejemo k novemu mesecu. Če je četrtek še v starem mesecu (zadnji dan meseca), petek, soboto in nedeljo iz novega meseca štejemo še k starem. Zaradi specifičnosti časovnega obdobja moramo imeti tako tabelo, ki nam olajša izračune.
- **trg**: shranjeni so podatki o trgovinah. Določeni večji trgovci imajo v svojem portfelju več blagovnih znamk trgovin. Ker se pri analizah osredotočamo samo na največje trgovce, je potrebno vse manjše blagovne znamke združiti na raven večjih trgovcev.
- **categ**: shranjeni so podatki o kategorijah. V analizah so zajete kategorije hrana in pijača, kozmetika in drugi izdelki za higieno in sredstva za pranje, čiščenje in osveževanje doma in perila.

3.3.2 Priprava podatkov

Preden anketne podatke uporabimo v analizi, jih moramo ustrezno pripraviti. Priprava se nanaša samo na nove podatke, to je na tiste, ki jih na novo vnašamo v aplikacijo. V našem primeru to pomeni, da moramo ustrezno pripraviti mesečne podatke, ki jih želimo na novo uvoziti v aplikacijo. Podatki, ki so že v aplikaciji, so v pravilni obliki za nadaljnje analize in jih ni treba pripravljati.

Priprava podatkov je nujno potreben korak, ki sledi vsakemu zbiranju le-teh. Podatki se vedno zbirajo na način, ki je najprimernejši za anketiranca. Z uporabo filtrov pri zbiranju podatkov poskrbimo, da anketiranci odgovarjajo samo na relevantna vprašanja. Npr. osebe, ki živi sama v gospodinjstvu, ne sprašujemo o dohodku gospodinjstva. V samskem gospodinjstvu je dohodek le-tega enak osebnemu dohodku. Pred začetkom izvajanja analiz je treba podatke pravilno urediti, da bodo odgovori med spremenljivkami konsistentni.

Čiščenje podatkov se uvršča v fazo priprave le-teh. Največkrat se nanaša na urejanje manjkajočih ali pa na brisanje ekstremnih vrednosti. Na vprašanje, koliko so plačali za izdelke pri izbranem trgovcu, respondenti pogosto ne vedo natančnega odgovora. Anketar kot odgovor vnese vrednost 9999, ki je določena kot vrednost za »ne vem odgovor«. Vendar pa se v praksi pogosto zgodi, da je anketar nepazljiv in kot odgovor ne vem vnese vrednosti 99 ali 999 namesto številke 9999. Prvi dve vrednosti bi pod določenimi pogoji lahko predstavljali veljavne odgovore, vendar ju upoštevamo kot manjkajoče vrednosti.

Pred začetkom priprave podatkov je treba le-te uvoziti v aplikacijo. Za uvoz podatkov sem uporabil paket readr, ki omogoča branje različnih tekstovnih datotek.

Paketa foreign in haven, ki omogočata branje SPSS datotek sta imela težave z branjem SPSS datotek. Če .sav datoteka ni popolno urejena, imata oba težave z branjem oznak (label) in vrednosti spremenljivk. Težavo predstavljajo predvsem numerične spremenljivke, ker program IBM SPSS Statistics predvideva, da številske spremenljivke nimajo oznak oz. imen vrednosti (label). Težavo predstavljajo tudi spremenljivke, katerih vrednosti so le delno poimenovane. Npr. vprašanje velikost gospodinjstva ima samo eno oznako, ki označuje osem ali več članov gospodinjstva. Druge vrednosti nimajo oznak, ker vrednosti same po sebi pomenijo število članov gospodinjstva. V izogib prevelikemu ročnemu popravljanju spremenljivk sem se odločil za izvoz .sav datoteke v tekstovno obliko (.csv), ki sem jo s funkcijo read_delim iz paketa readr uvozil v R program.

Uvožena datoteka vsebuje veliko število polj. Da bi lahko to tabelo uporabili za združevanje s tabelo cases, v kateri so demografske značilnosti anketirancev, je treba izbrati samo določene spremenljivke. Za izbris nepotrebnih spremenljivk sem uporabil funkcijo select iz paketa dplyr.

Ker pa nova tabela ne vsebuje kategorične spremenljivke, v kateri je označen starostni razred anketiranca, je treba to spremenljivko izračunati iz spremenljivke, v kateri je zapisana starost

anketiranca v numerični spremenljivki. Za to opravilo, rekodiranja vrednosti v novo spremenljivko, sem uporabil kombinacijo funkcij mutate in if_else.

Novo tabelo, ki smo jo v predhodnih korakih ustrezno pripravili, je treba združiti z obstoječo tabelo cases. Za ta korak sem uporabil funkcijo bind_rows iz paketa dplyr.

Za združitev novih podatkov s tabelo data, je treba iz uvožene datoteke izbrati spremenljivke, ki se nanašajo na porabo prejšnjega dne po trgovinah. Za ta korak sem vnovič uporabil funkcijo select iz paketa dplyr. Dobljena tabela je še zmeraj v »široki« obliki, kar pomeni, da so anketiranci zapisani v vrsticah, spremenljivke o porabi pa so stolpci.

Zaradi analiz je treba tabelo preoblikovati v transakcijsko obliko, kjer so v eni vrstici zapisane naslednje spremenljivke: identifikator osebe, datum nakupa, kategorija nakupa, trgovina nakupa in znesek. Transakcijska oblika zapisa omogoča enostavnejše čiščenje podatkov (prečistiti moramo samo eno spremenljivko, v kateri so zapisane vrednosti za nakupe po trgovinah). Transakcijski način omogoča enostavno agregacijo podatkov. Namesto seštevanja po vseh kategorijah znotraj vsake trgovine posebej, lahko uporabimo funkcije za agregacijo podatkov v kombinaciji s funkcijami za združevanje le-teh. Funkcije za združevanje omogočajo, da podatke razbijemo po različnih demografskih skupinah.

Za transformacijo tabele cases iz široke v dolgo obliko sem uporabil funkcijo spread iz paketa tidyr. Za čiščenje podatkov oz. brisanje vrednosti, ki predstavljajo manjkajoče vrednosti (99, 999, 9999), sem uporabil funkcijo filter iz paketa dplyr s katero sem izločil te vrednosti iz tabele.

Dobljeno tabelo je treba združiti z obstoječo tabelo data. Za ta korak je bila uporabljena funkcija bind_rows iz paketa dplyr.

3.3.3 Združevanje podatkov

Pred začetkom izvajanja analiz je treba podatke združiti v skupno združeno tabelo (re_month_data), ki je glavni vir za vse preostale analize, ki se nanašajo na izračun povprečne porabe in tržne deleže.

Na sliki 3.7 je predstavljena struktura podatkov. Na tabelo data, ki predstavlja potrošnjo anketirancev po trgovinah, kategorijah in datumu, sem nalepil vse preostale tabele. Podatke iz tabele cases zaradi demografskih spremenljivk in uteži, podatke iz tabele date zaradi časovnih agregacij, podatke iz tabele trg zaradi združevanja blagovnih znamk trgovin v večje trgovce in

podatke iz tabele categ zaradi opisov kategorij. Dobil sem tabelo, ki ima toliko zapisov, kot je vseh zapisov v tabeli data in tolikšno število spremenljivk, kot je število vsote unikatnih spremenljivk pri vseh združenih tabelah. Pri združevanju se tvori tudi spremenljivka spent, ki predstavlja porabo po izbrani kategoriji v izbrani trgovini. Dejanski znesek porabe, ki ga imamo zapisanega v tabeli data, pomnožimo z utežjo, da dobimo uteženo porabo.

Za združevanje tabel in tvorbo nove spremenljivke sem uporabil ukaza inner_join in mutate iz paketa dplyr.

Slika 3.8: Ukaz za združevanje tabel in tvorbo nove spremenljivke

```
# MONTH DATA (združevanje osnovnih tabel

re_month_data <- reactive({
    cases <- re_cases()
    data <- re_data()

    month_data <- data %>% select(-date) %>%
        inner_join(cases, by = "id_case") %>%
        inner_join(date, by = "date") %>%
        inner_join(trg, by="id_trg") %>%
        inner_join(categ, by="id_categ") %>%
        mutate(spent=znesek * wrakin1)

    return(month_data)

})
```

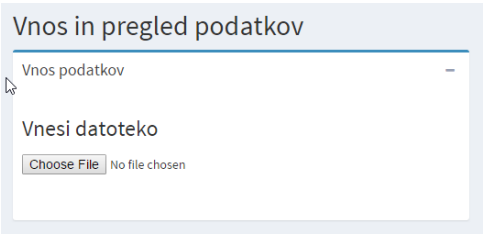
3.3.4 Izdelava aplikacije v jeziku R

V nadaljnjem poglavju so prikazani izbrani gradniki in funkcije, ki so bile uporabljene pri izdelavi aplikacije.

Vhodni gradnik za zajem podatkov (fileInput)

Vhodni gradni (fileInput) se uporablja za zajem podatkov. Datoteka se zajame tako, da se shrani v začasno datoteko na disku. Potem ko se začasna datoteka shrani na disku, se sočasno ustvari nov podatkovni okvir, v katerem je zapisana pot do začasne datoteke. Če želimo prebrati datoteko, moram uporabiti ukaz za prebiranje datotek, sklicevati pa se moramo na podatkovni okvir.

Slika 3.9: Vhodni gradnik (fileInput) in programska koda

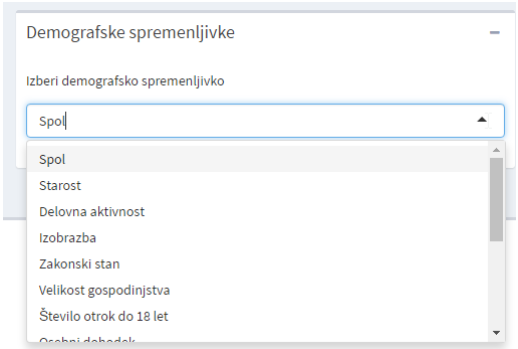



```
fileInput(inputId="in_fi",
          label=h3("Vnesi datoteko"),
          accept = c("text", ".csv")))
```

Vhodni gradnik za izbirni niz (selectInput)

Vhodni gradnik se uporablja za izbiro različnih možnosti iz spustnega menija. V aplikaciji uporabljam izbirni niz za izbiro demografskih spremenljivk. Rezultat izbirnega niza je vektor, v katerem je zapisana vrednost izbire. Vektor je lahko dolžine ena oz. več kot ena, če je možno izbrati več odgovorov.

Slika 3.10: Vhodni gradnik izbirni niz (selectInput) in programska koda




```
selectInput(inputId="si_demo", label=h5("Izberi demografsko spremenljivko"),
            choices=list("Spol"="SPOL","Starost"="d3_star","Delovna aktivnost"="d1_dakt",
                         "Izobrazba"="d1a_izo","Zakonski stan"="ZSTAN",
                         "Velikost gospodinjstva"="d0_gosp","Število otrok do 18 let"="OTROCI",
                         "Osebni dohodek"="D0_DOH","Dohodek gospodinjstva"="DOHGOSP",
                         "Regija"="d1_reg","Velikost naselja"="d0_vnas","Tip naselja"="urban1",
                         "Nakupni član gospodinjstva"="SGOSP",
                         "Največji prispevek dohodku gospodinjstva"="DGOSP",
                         "Dan anketiranja"="danvt"), selected=1)
```

Funkcija za generiranje izhodnega gradnika (valueBox)

Slika 3.11: Izhodni gradnik valueBox in programska koda

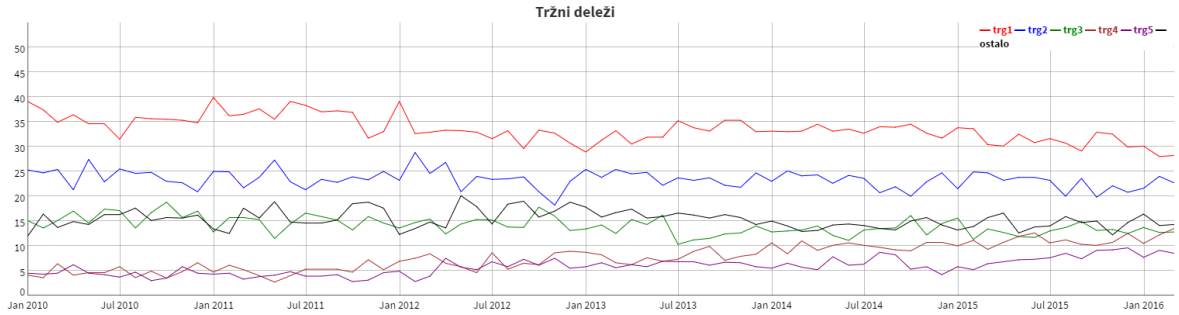

```
output$ostaloBox <- renderValueBox({
    porabax <- re_valueBox_dif()
    brand <- "ostalo"
    vr <- porabax[porabax$account==brand,3][[1]]
    razl <- porabax[porabax$account==brand,4][[1]]

    valueBox(
        paste0(sprintf("%.1f€", vr),"   ;", sprintf("%.1f€", razl)),
        "ostalo", icon = icon("money"),
        color = ifelse(razl >= 1, "green",
                       ifelse(razl >= -1,"yellow","red"))
    )
})
```

Izhodni gradnik valueBox se uporablja za prikaz različnih informacij. Ukaz renderValueBox je funkcija, ki izhodni gradnik valueBox ustvari dinamično na serverju in ga prikaže v uporabniškem vmesniku. Dinamično pomeni, se besedilo ali številke lahko spreminjajo. ValueBox je ustrezen za prikaz ključnih izračunov. S spreminjanjem oblik valueBoxa (barve, ikone) le-ta postane podoben semaforju. V našem primeru se barva valueBox-a spreminja glede na razliko v povprečni vrednosti nakupa za trenutni in pretekli mesec. Če je razlika

manjša od 1€, se obarva rdeče, če znaša med -1€ in 1€ se obarva rumeno, če pa je večja od 1€ pa je obarvan zeleno.

Funkcija za generiranje izhodnega gradnika za dinamični črtni graf (Dygrahp)

Slika 3.12: Izhodni gradnik za dinamični črti graf (dygraph) in programska koda

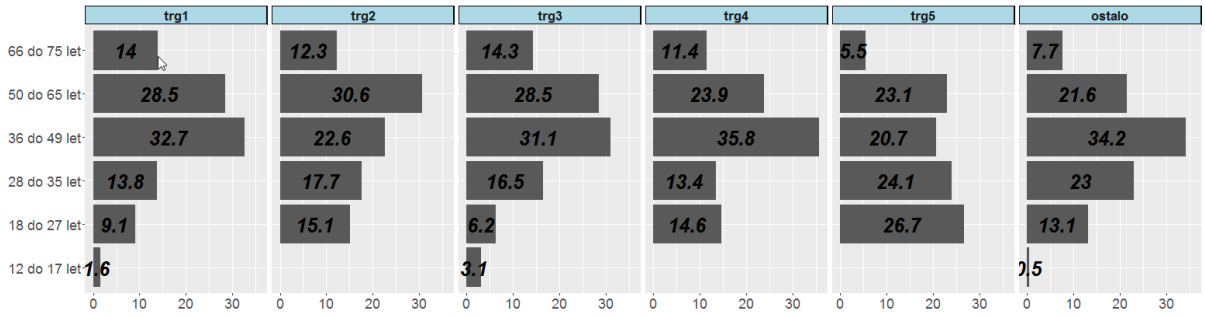



```
output$dygraph <- renderDygraph({

    dygd <- cmonth_data2() %>%
        separate(col=cmonth, into = c("year","month"), sep = "_", remove = FALSE) %>%
        mutate(year=str_replace(year,"M","")) %>%
        mutate(month=if_else(str_length(month) == 1, str_c("0",month), month)) %>%
        mutate(year=as.numeric(year) + 2000) %>%
        mutate(datum=str_c(year,month,01)) %>%
        mutate(datum=as.Date(datum, format="%Y%m%d")) %>%
        select(datum,account,spent_per) %>%
        mutate(account=factor(account,levels=c("trg1","trg2","trg3","trg4","trg5","ostalo"), ordered=TRUE)) %>%
        spread(key=account, value=spent_per)

    dygd <- data.frame(dygd)

    dygdt <- xts(dygd[,-1], order.by=dygd[,1])

    dygraph(dygdt, main = "Tržni deleži") %>%
        dyAxis("y", valueRange = c(0, 55)) %>%
        dyOptions(colors = c("red", "blue", "green", "brown", "purple", "black"))

})
```

Dinamični graf se uporablja za prikaz tržnega deleža in prikaz povprečne vrednosti nakupa. Dinamični graf se ustvari dinamično na serverju s funkcijo dygraph. Pred izrisom grafa s funkcijo dygraph, je treba na ustrezen način pripraviti podatke. V vrsticah so shranjena časovna obdobja (meseci), v stolpcih so zapisane trgovine in vrednosti glede na pripadajoč mesec. V mojem primeru je priprava obsegala izračun s funkcijami iz paketa dplyr. Dobljeno tibble podatkovno strukturo sem pretvoril v podatkovni okvir. Omenjena podatkovna struktura ne omogoča uporabe rownames funkcije. Rownames funkcija je potrebna za generiranje xts objekta (objekt za shranjevanje časovnih vrst), ki se uporabi za generiranje dinamičnih grafov v dygraphs funkciji.

Funkcija za generiranje izhodnega gradnika za graf po demografskih skupinah

Slika 3.13: Izhodni gradnik za graf po demografskih skupinah in programska koda




```
demo_ch_data <- reactive({

    izbira <- input$si_demo2

    month_data_demo <- re_month_data() %>%
    group_by_("id_case","cmonth", izbira, "account") %>%
    summarise_("st_obiskov"="max(wrakin1)", "st_trg"="n()", "zapr" = "sum(spent)") %>%
    select_("id_case",izbira, "cmonth", "account", "st_obiskov", "st_trg", "zapr") %>%
    group_by_("cmonth","account", izbira) %>%
    summarize_("total_obiski"="sum(st_obiskov)", "total_spent"="sum(zapr)", "spent_am"="round(total_spent/total_obiski,1)") %>%
    group_by_("cmonth", "account") %>%
    mutate_("xxx"="sum(total_spent)","spent_per"="round(total_spent/xxx*100,1)") %>%
    ungroup("cmonth", "account") %>%
    mutate(account=factor(account,levels=c("trg1","trg2","trg3","trg4","trg5","ostalo"), ordered=TRUE)) %>%
    select_("cmonth", "account", izbira, "spent_per", "spent_am") %>%
    rename_("dems"=izbira)

    izb_m <- input$di_demo2
    izbm_m <- as.Date(izb_m, format="%Y-%m-%d")
    txj <- date %>% filter(date == izbm_m) %>% select(cmonth) %>% inner_join(month_data_demo, by = "cmonth")

    return(txj)
})

output$demoTD2 <- renderPlot({

    ggplot(demo_ch_data(), aes(x=dems, y=spent_am)) + geom_bar(stat="identity") + facet_grid(.~account) + coord_flip() +
        theme(strip.text = element_text(face="bold", size=rel(1.5)), strip.background = element_rect(fill="lightblue", colour="black", size=1),
              axis.title.x=element_blank(), axis.title.y=element_blank(), axis.text.y=element_text(size=18), axis.text.x=element_text(size=18)) +
        geom_text(aes(y=spent_am/2, label=spent_am), size=9, fontface="bold.italic", color="black")
})
```

Graf po demografskih skupinah se uporablja za prikaz deleža porabe posamezne demografske skupine in povprečne velikosti nakupa po posameznem trgovcu. Izris grafa je razdeljen na dva dela: na pripravo podatkov in na sam izris. Za sam izris grafa se uporablja funkcija ggplot iz paketa ggplot2. Pred izrisom grafa je potrebno pripraviti podatke na ustrezen način. Graf se spreminja glede na izbran mesec in glede na izbrano demografsko skupino. Za pripravo podatkov sem uporabil različne funkcije iz paketa dplyr.

4 Zaključek

Pri svoji diplomski nalogi sem naredil spletno aplikacijo za analizo tržnega deleža. Aplikacija omogoča zajem podatkov, obdelavo le-teh in prikaz rezultatov v grafični in tabelarični obliki.

Pred avtomatizacijo je bil postopek izdelave poročila razdeljen na več med seboj ločenih korakov, ki so zaradi ročnega dela predstavljali potencialne vire napak. Priprava podatkov se je opravljena v programu SPSS Statistics, sami podatki so se shranjevali v access podatkovni bazi, za izdelavo grafičnih prikazov pa sem uporabljal Excel. Aplikacija je celoten proces združila v en korak, ki obsega vnos podatkov v aplikacijo. Možnosti za človeško napako je manj in zmanjšalo se je število ur, potrebnih za izdelavo poročil.

Kot najpomembnejšo stvar pri izdelavi avtomatizacij bi izpostavil cilj, ki ga z avtomatizacijo želimo doseči. Kot cilj so mišljeni rezultati, ki se bodo prikazovali in način na katerega se bodo prikazovali. Način je lahko statičen ali dinamičen, pri čemer dinamičen način zahteva dodatne informacije, ki jih moramo imeti zbrane pred samim začetkom izdelave aplikacije. Ker sam cilj določa vse naslednje korake izdelave aplikacije, ga ni priporočljivo spreminjati med samim procesom le-te. Sprememba ciljev potegne za sabo vse ostale korake, ki se morajo opraviti še enkrat.

Z vidika programiranja v R-u je najpomembnejša odločitev izbira dodatnih paketov, ki se bodo uporabljali pri programiranju. Izbira paketov je v prvi vrsti odvisna od funkcij, ki jih potrebujemo. Priporočljivo je, da se uporablja pakete istega avtorja ali iste skupine avtorjev. Predpostavljam, da avtor ali skupina avtorjev poskrbi, da so njihovi paketi med sabo povsem skladni. Sam najpogosteje uporabljam pakete avtorjev okoli podjetja R-Studio.

Na koncu pa bi omenil zadevo, ki mi je povzročila največ težav. V začetku sem načrtoval, da bo avtomatizacija sposobna prebrati .sav datoteko. Kljub vsem poskusom s funkcijami iz paketov foreign in haven nisem uspešno prebral .sav datoteke. Na koncu sem težavo, ki je izvirala iz nepravilno urejene .sav datoteke, rešil tako, da sem podatke iz .sav pretvoril v .csv zapis in tega potem uporabil za vnos v avtomatizacijo.

5 Literatura

1. Beely, Chris. 2016. *Web Application Development with R Using Shiny, Second Edition*. Birmingham - Mumbai: Packt Publishing.
2. Buzzell, D. Robert, Bradley T.Gale, Ralph, G.M. Sultan. 1975. *Market Share - a Key to Profability.* Boston. Harvard Business Publishing. Dostopno prek: https://hbr.org/1975/01/market-share-a-key-to-profitability (1. junij 2016).
3. Chang, Winston. 2015. shinydashboard: *Create Dashboards with 'Shiny'.* Dostopno prek: https://CRAN.R-project.org/package=shinydashboard (25. avgust 2016).
4. Chang, Winston, Joe Cheng, JJ Allaire, Yihui Xie in Jonathan McPherson. 2016. *shiny: Web Application Framework for R.* Dostopno prek: https://CRAN.R-project.org/package=shiny (25. avgust 2016).
5. Churchill, Gilbert A. Jr. in Dawn Iacobucci. 2002. *Marketing Research*. Mason: South-Western.
6. Few, Stephen. 2006. *Information Dashboard Design*. Gravenstein: O'Reilly Media.
7. Farris, Paul W., Neil T. Bendle, Phillip E. Pfeifer in David J. Reibstein. 2010. *Marketing Metrics: The Definitive Guide to Measuring Marketing Performance*. New Jersey: Pearson Education, Inc.
8. Field Andy, Jeremy Miles in Zoe Field. 2012. *Discovering Statistics Using R*. London: SAGE publications Ltd.
9. Grolemund, Garrett. 2015. *Hands-On Programming with R*. Gravenstein: O'Reilly Media.
10. Hetherington, Victoria. 2009. *The Dashboard Demystified: What is a Dashboard?* Dostopno prek: http://www.dashboardinsight.com/articles/digital-dashboards/fundamentals/the-dashboard-demystified.aspx (6. junij 2016).
11. Hillebrand, Julian in Maximilian H. Nierhof. 2015. *Mastering RStudio - Develop, Communicate, and Collaborate with R*. Birmingham - Mumbai: Packt Publishing.
12. Jacobsen, Robert in David A. Aaker. 1985. Is Market Share All That It's Cracked Up To Be? *Journal of Marketing*. 49(autumn) 11−22.
13. Kabacoff, Robert. 2011. *R in Action.* Shelter Island: Manning Publications.
14. R Core Team. 2016. R: *A language and environment for statistical computing. R Foundation for Statistical Computing,* Dostopno prek: https://www.R-project.org/ (25. avgust 2016).

15. Ren, Kun in Kenton Russell. 2016. *formattable: Create 'Formattable' Data Structures.* Dostopno prek: https://CRAN.R-project.org/package=formattable (25. avgust 2016).
16. Teutonico, Donato. 2015. *ggplot2 Essentials*. Birmingham - Mumbai: Packt Publishing.
17. Tsiptsis, Konstantinos in Antonios Chorianopoulos. 2009. *Data Mining Techniques in CRM*: Inside Customer Segmentation. Chichester: Wiley.
18. Vanderkam, Dan, JJ Allaire, Jonathan Owen, Daniel Gromer, Petr Shevtsov in Benoit Thieurmel. 2016. *dygraphs: Interface to 'Dygraphs' Interactive Time Series Charting Library*. Dostopno prek: https://CRAN.R-project.org/package=dygraphs (25. avgust 2016).
19. Wickham, Hadley. 2009. ggplot2: *Elegant Graphics for Data Analysis*. New York: Springer-Verlag.
20. --- 2016. *tidyr: Easily Tidy Data with `spread()` and `gather()` Functions.* Dostopno prek: https://CRAN.R-project.org/package=tidyr (25. avgust 2016).
21. Wickham, Hadley, Jim Hester in Romain Francois. 2016. readr: *Read Tabular Data*. Dostopno prek: https://CRAN.R-project.org/package=readr (25. avgust 2016).
22. Wickham, Hadley in Romain Francois. 2016. *dplyr: A Grammar of Data Manipulation*. Dostopno prek: https://CRAN.R-project.org/package=dplyr (25. avgust 2016).
23. Wilkinson, Leland. 2005. *The Grammar of Graphics*. New York: Springer Science+Business Media, Inc.
24. Xie, Yihui. 2016. *DT: A Wrapper of the JavaScript Library 'DataTables'*. Dostopno prek: https://CRAN.R-project.org/package=DT (25. avgust 2016).
25. Yannopoulos, Peter. 2010. The Market Share Effect: New Insights from Canadian Data. *Journal of Global Business Managment*. 6(2). Dostopno prek: http://www.jgbm.org/page/31Peter%20Yannopoulos.pdf (1. junij 2016).
26. *Zakon o preprečevanju omejevanja konkurence* (ZPOmK-1). Ur. I. RS 36/2008 (11. april 2008).
27. Zev, Ross. 2016. *R powered web applications with Shiny (a tutorial and cheet sheet with 40 example apps).* Dostopno prek: http://zevross.com/blog/2016/04/19/r-powered-web-applications-with-shiny-a-tutorial-and-cheat-sheet-with-40-example-apps/ (5. junij 2006).