Izjemno kratek uvod v R

Jure Demšar

30/5/2019

R in RStudio

R (https://www.r-project.org/) je odprto kodni programski jezik namenjen primarno statisičnim izračunom. Osnovni vmesnik za delo z Rjem je precej okoren, zato si delo lahko precej olajšamo z namestitvijo drugega razvojnega okolja. Najbolj razširjeno ter razmeroma kvalitetno razvojno okolje se imenuje RStudio (https://www.rstudio.com).

Paketi

Ker je R odprto kodni programski jezik, lahko vsak prispeva k njegovem razvoju s pomočjo dodatnih paketov. Uporaba paketov nam močno olajša delo, večina uporabnih paketov se nahaja v repozitoriju CRAN. V RStudiu lahko pakete iz repozitorija CRAN naložimo preko menuja Tools in izbire Install Packages. Če poznamo ime paketa, ki ga želimo namestiti, lahko to naredimo tudi neposredno preko ukazne vrstice, na primer ukaz `install.packages("dplyr")` namesti paket dplyr. Ko paket namestimo, ga moramo pred uporabo še uvoziti v našo skripto, to storimo s pomočjo ukaza `library()`. Spodnja koda namesti ter uvozi vse pakete, ki jih bomo potrebovali pri reševanju današnjih nalog.

```
# install packages
install.packages("dplyr")
install.packages("ggplot2")
install.packages("mcmcse")
install.packages("reshape2")
install.packages("rstan")

# load packages into workspace
library(dplyr)
library(ggplot2)
library(mcmcse)
library(rstan)
library(reshape2)
```

Nalaganje podatkov

Podatki so običajno shranjeni v obliki .csv (angl. *comma separated value*) datotek. Vsaka vrstica v datoteki predstavlja eno meritev, v vsaki vrstici lahko najdemo več atributov te meritve, atributi so ločeni s posebnim znakom, ki v vrednostih atributov ne nastopa. V R podatke iz datoteke csv naložimo s pomočjo ukaza `read.csv()`. Privzeti znak za ločevanje atributov je vejica, zaradi praktičnih razlogov (zmeda z decimalno vejico/piko, vejice v tekstovnih podatkih) se vejica običajno ne uporablja. Delilni znak (angl. *separator*) lahko nastavimo z uporabo parametra `par`. Spodnja koda v spremenljivko `basket_data` naloži podatke iz datoteke `basketball_shots.csv`, ki se nahaja v direktoriju `data`.

```
# load data
basket_data <- read.csv("../data/basketball_shots.csv", sep=";")
```

Podatki se naložijo v podatkovno strukturo imenovano podatkovni okvir (angl. *data frame*). Ta podatkovni tip si lahko predstavljate kot tabelo, kjer stolpci predstavljajo atribute, vrstice pa kot primerke/meritve. V posameznih celicah se tako nahajajo vrednosti atributa za določen primerek.

S pomočjo konstruktorja `data.frame()` lahko naredimo tudi nov podatkovni okvir, operacija `c()` v spodnjem primeru predstavlja *combine* in združi več posameznih elementov v vektor.

```
# we can create new data frames like this, c() combines elements into a vector
new_df <- data.frame(first_column=c(1, 2), second_column=c(10, 11))
```

```
##   first_column second_column
## 1            1            10
## 2            2            11
```

Do posameznih stolpcev podatkovnega okvirja dostopamo z znakom $, do števila vrstic v okvirju pa pridemo z ukazom `nrow()`.

```
# to access columns of a data frame use the $ sign
new_df$first_column
```

```
## [1] 1 2
```

```
# to get number of rows in a data frame use nrow
nrow(new_df)
```

```
## [1] 2
```

Z ukazom `rbind()` lahko združimo več podatkovnih okvirjev v enega, pogoj je, da se okvirji ujemajo v vseh stolpcih.

```
# create another data frame
new_df2 <- data.frame(first_column=c(3, 4), second_column=c(12, 13))

# we can merge data frames together (if they have same columns) using rbind
new_df3 <- rbind(new_df, new_df2)
```

Pri pripravi podatkov za Stan moramo običajno iz podatkovnega okvirja izluščiti samo določen stolpec. Ko stolpec shranimo v spremenljivko, dobimo spremenljivko tipa vektor. Za izračun dolžine vektorja ne uporabljamo ukaza `nrow()`, saj vektor nima vrstic, ampak uporabimo ukaz `length()`.

```
# we can extract a column and save it into a vector
first_column <- new_df$first_column

# to get length of a vector use the length function
length(first_column)
```

```
## [1] 2
```

Ko pripravljamo podatke za Stan, jih shranimo v podatkovno struktoro `list`. Gre za seznam spremenljivk različnega tipa, v spodnjem primeru tako naredimo seznam, ki vsebuje celo število (`n`) ter vektor realnih števil (`y`).

```
# we pass data to stan through lists
n <- nrow(new_df)
y <- new_df$first_column
stan_data <- list(n = n, y = y)
```

Osnovne operacije nad podatki

Za potrebe delavnice bomo spoznali zgolj najbolj osnovne operacije za delo s podatki. Urejanje podatkov si bomo olajšali z uporabo knjižnice `dplyr`. V naših podatkih o metih na koš nastopa več igralcev, vsak izmed njih je 60 krat vrgel na običajen koš ter 60 krat na koš z manjšim obsegom obroča. Spodnji primer

prikazuje, kako iz vseh podatkov izluščimo samo podatke o metih prvega igralca (atribut `PlayerID` je enak 1) na običajen obroč (atribut `SpecialRim` je enak 0).

```
player1_data <- filter(basket_data, PlayerID == 1 & SpecialRim == 0)
```

Za vsak met imamo 5 podatkov: * ID igralca (*PlayerID*), * ID meta (*ThrowNum*), * kot izmeta (*Angle*), * uspešnost (*Made*), * tip obroča (*SpecialRim*).

Recimo, da za nadaljno analizo potrebujemo zgolj stolpca s podatki o zaporedni številki meta ter o uspešnosti meta, ostalih stolpcev se lahko znebimo z uporabo operacije `select`. Reduciranje števila stolpcev je še posebej koristno, če operiramo z ogromnimi količinami podatkov.

```
player1_data <- select(player1_data, ThrowNum, Made)
```

Operacijo `melt` iz paketa `reshape2` bomo uporabljali predsem za pripravo podatkov pred vizualizacijami. Spodnji primer prikaže uporabo melt, ko želimo primerjati zneske, ki so jih zagonska podjetja vložila v raziskave in razvoj, marketing in administracijo.

```
##   research marketing administration
## 1 165349.2  471784.1       136897.8
## 2 162597.7  443898.5       151377.6
## 3 153441.5  407934.5       101145.6
```

Osnovna uporaba ukaza `melt` nam podatke zreducira na 2 stolpca, poimenovana `variable` in `value`, kjer vsaka vrstica predstavlja eno celico (par `variable`/`value`) iz izhodiščnega podatkovnega okvirja. Tako pripravljene podatke lahko nato v vizualizacijah grupiramo po vrednosti atributa `variable`, medtem ko vrednosti (`value`) preslikamo na želeno os grafa.

```
melt_investments <- melt(investments)
```

```
##          variable    value
## 1        research 165349.2
## 2        research 162597.7
## 3        research 153441.5
## 4       marketing 471784.1
## 5       marketing 443898.5
## 6       marketing 407934.5
## 7  administration 136897.8
## 8  administration 151377.6
## 9  administration 101145.6
```

Cev (angl. *pipe*)

V kodi boste srečali tudi cev (`pipe` operator, `%>%`), s pomočjo tega operatorja lahko kodo precej skrajšamo. Operator nam omogoča, da podatke, ki so rezultat ene operacije, prenesemo neposredno v naslednjo operacijo (kot bi naredili cev med dvema operacijama, po cevi se rezultat prve operacije prenese na vhod druge operacije). Spodnji primer prikazuje, kako lahko združimo filtriranje in selektiranje podatkov iz zgornjega primera.

```
player1_data <- basket_data %>%
  filter(PlayerID == 1 & SpecialRim == 0) %>%
  select(player1_data, ThrowNum, Made)
```