Predavanje 07 – Funkcije

Funkcije

Funkcije kot smo jih spoznali do sedaj so ukazi, ki izvršijo neko določeno nalogo. V R-ju obstajajo mnogi paketi, kot sta na primer *openxlsx* in *ggplot2*, ki razširijo delovanje R-ja z dodajanjem novih funkcij. V R-ju si lahko napišemo tudi svoje funkcije in si s tem olajšamo delo.

Definicija funkcije

Za definicijo funkcije potrebujemo:

1) Ime funkcije: Ime s katerim bomo klicali funkcijo.
2) Vhod: Katere vhodne parametre sprejme funkcija za svoje izvajanje.
3) Izhod: Kaj nam funkcija na koncu izvajanja vrne.
4) Jedro funkcije: Blok ukazov, ki se izvedejo, s katerimi izdelamo izhod funkcije.

Vhod je lahko tudi prazen. Primer je funkcija `getwd()`, ki le izpiše trenutni direktorij na konzolo. Izhod je tudi lahko prazen oz. vrne vrednost **NULL**. Izhod vedno vrne samo eno vrednost, oziroma vektor, seznam,...

Funkcijo v R-ju definiramo s privzeto besedo `function` na naslednji način:

```
imeFunkcije <- function (<parametri>) {
  #Blok kode, ki naj se izvede
  return()
}
```

Napišimo svojo prvo funkcijo, ki vrne niz "I love R!". Funkcijo moramo najprej pognati, kot ostale ukaze in jo nato lahko uporabljamo enako, kot že vrgajene funkcije.

```
mnenje <- function () {
  return("I love R!")
}
```

Poženimo funkcijo:

```
mnenje()
```

```
## [1] "I love R!"
```

Izhod funkcije lahko shranimo kot spremenljivko.

```
izhod <- mnenje()
izhod
```

```
## [1] "I love R!"
```

Dodajanje vhodnih parametrov

V prejšnjem primeru funkcija `hello()` nima vhodnih parametrov. Napišimo sedaj funkcijo, `bmi()`, ki ima dva vhodna parametra in sicer višino v m in težo v kg. Na izhodu pa nam vrne indeks telesne teže.

```
bmi <- function (visina, teza) {
  return(teza / (visina / 100) ^ 2)
}
```

Testirajmo funkcijo:

```
  bmi(170, 70)
```

```
## [1] 24.22145
```

Izhod funkcije lahko shranimo v spremenljivko:

```
  bmi_rezultat <- bmi(170, 70)
  bmi_rezultat
```

```
## [1] 24.22145
```

Poglejmo si delovanje še na podatkih iz drugega predavanja:

```
df <- read.table("./data_raw/osebe.csv", header = TRUE, sep=",", quote = "\"", dec = '.')
df
```

```
##    spol visina teza  imena
## 1     f    179   75  Micka
## 2     m    185   89  Marko
## 3     m    183   70 Gregor
## 4     m    172   80  Tomaz
## 5     f    174   58    Ana
## 6     m    185   86  Peter
## 7     f    193   73  Mojca
## 8     f    169   63  Katja
## 9     m    173   72   Anze
## 10    f    168   70   Alja
```

Ali funkcija deluje na stolpcih? Testirajmo:

```
  bmi(df$visina, df$teza)
```

```
##  [1] 23.40751 26.00438 20.90239 27.04164 19.15709 25.12783 19.59784 22.05805
##  [9] 24.05693 24.80159
```

Funkcija deluje na vseh podatkovnih tipih, za katere R lahko izračuna jedro funkcije.

V našem primeru bi klic `bmi(df$visina, df$spol)` vrnil napako:

```
  bmi(df$visina, df$spol)
```

```
## Warning in Ops.factor(teza, (visina/100)^2): '/' not meaningful for factors
```

```
##  [1] NA NA NA NA NA NA NA NA NA NA
```

V stolpcu `spol` niso številke. R ne zna deliti takih vektorjev.

Preverjanje pravilnosti vhoda

Vsaka funkcija predpostavlja, da na vhod dobi točno določen tip podatkov. V primeru, da želimo svojo kodo deliti z drugimi programerji, oziroma če želimo preprečiti tudi nadalnje napake v svojih skriptah, je priporočljivo preveriti vrednosti in tip vhoda.

Napišimo sedaj funkcijo, `hipotenuza()`, ki ima dva vhodna parametra; dolžini katet pravokotnega trikotnika. Na izhodu pa nam vrne dolžino hipotenuze $c = \sqrt{a^2 + b^2}$.

```
hipotenuza <- function (a, b) {
  c <- sqrt(a * a + b * b)
  return(c)
}
```

Testirajmo funkcijo:

```
hipotenuza(1, 1)
```

```
## [1] 1.414214
```

```
hipotenuza(3, 7)
```

```
## [1] 7.615773
```

```
hipotenuza(-3, 2) #Vrne odgovor, čeprav je nesmiselen vhod
```

```
## [1] 3.605551
```

```
hipotenuza("a", "b") #Vrne napako, ker je tip vhoda napačen
```

```
## Error in a * a: non-numeric argument to binary operator
```

Če pišemo funkcijo le za lastno uporabo predpostavimo, da bomo pazili, kaj podamo na vhod.

Napišimo sedaj še funkcijo `kateta()`, ki sprejme eno dolžino katete in dolžino hipotenuze ter vrne dolžino druge katete. Dolžino manjkajoče katete izračunamo kot $b = \sqrt{c^2 - a^2}$.

```r
kateta <- function (a, c) {
  b <- sqrt(c * c - a * a)
  return(b)
}
```

Testirajmo funkcijo:

```r
kateta(1, sqrt(2))
```

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

```r
kateta(3, 7)
```

```
## [1] 6.324555
```

```r
kateta(7, 3) #Vrne opozorilo, da funkcija ne zna narediti izračuna
```

```
## Warning in sqrt(c * c - a * a): NaNs produced
```

```
## [1] NaN
```

```r
kateta(-1, 2) #Vrne odgovor, čeprav je nesmiselen vhod
```

```
## [1] 1.732051
```

```r
kateta("a", "b") #Vrne napako, ker je tip vhoda napačen
```

```
## Error in c * c: non-numeric argument to binary operator
```

V primeru `kateta(7, 3)` dobimo le opozorilo, čeprav je izračun vrednost **NaN** (Not A Number). S to oznako so v R-ju definirane neizračunljive vrednosti, kot na primer 0/0.

Poskusimo našo funkcijo razširiti in popraviti, tako da bo zaznala take vrste težav. V prvem primeru smo funkciji kot vhod podali kateto, ki je daljša od hipotenuze, kar ni smiselno. V tem primeru lahko predpostavimo, da je prvi parameter hipotenuza in opravimo izračun.

```r
kateta <- function (a, c) {
  if (c > a) {
    b <- sqrt(c * c - a * a)
  }else{
    warning("Kateta je daljša od hipotenuze. Zamenjam vrednosti.")
    b <- sqrt(a * a - c * c)
  }
  return(b)
}
```

```r
kateta(1, sqrt(2))
```

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

```r
kateta(3, 7)
```

```
## [1] 6.324555
```

```r
kateta(7, 3) #Vrne pravilo vrednost in opozorilo
```

```
## Warning in kateta(7, 3): Kateta je daljša od hipotenuze. Zamenjam vrednosti.
```

```
## [1] 6.324555
```

V zadnjem primeru nam funkcija vrne pravilni izračun, hkrati pa si še izpišemo opozorilo s funkcijo `warning()`. Funkcija `warning()` nam opozorila izpiše na koncu izvajanja funkcije, lahko pa si jih ogledamo tudi, če v konzolo napišemo `warning()`.

Dodajmo še preverjanje negativnih vrednosti. V tem primeru, bi želeli, da se funkcija zaustavi in nam vrne napako. To lahko naredimo z ukazom `stop()`, ki zaustavi funkcijo in nam vrne željeno sporočilo napake.

```r
kateta <- function (a, c) {
  if (a <= 0 | c <= 0) {
    stop("Negativne vrednosti stranic!")
  }
  if (c > a) {
    b <- sqrt(c * c - a * a)
  }else{
    warning("Kateta je daljša od hipotenuze. Menjam vrednosti.")
    b <- sqrt(a * a - c * c)
  }
  return(b)
}
```

Testirajmo:

```r
kateta(1, sqrt(2))
```

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

```r
kateta(3, 7)
```

```
## [1] 6.324555
```

```r
kateta(7, 3) #Vrne pravilno vrednost in opozorilo
```

```
## Warning in kateta(7, 3): Kateta je daljša od hipotenuze. Menjam vrednosti.
```

```
## [1] 6.324555
```

```r
kateta(-1, 2) #Vrne napako skupaj z našim opisom napake
```

```
## Error in kateta(-1, 2): Negativne vrednosti stranic!
```

V zadnjem primeru nam R vrne napako, ki smo jo sami definirali in pripadajoči opis. Tako lažje odkrijemo, kaj je izvor napake, če se nam to pripeti v prihodnosti.

Nekatere napake R zazna že sam, tako nam klic `kateta("a", "b")` že sam vrne napako, lahko pa ga poskušamo zaznati tudi sami. Za to lahko uporabimo funkcijo `is.numeric()`.

```
kateta <- function (a, c) {
  if (!(is.numeric(a) & is.numeric(c))) {
    stop("Na vhodu morajo biti številke.")
  }
  if (a <= 0 | c <= 0) {
    stop("Negativne vrednosti stranic!")
  }
  if (c > a) {
    b <- sqrt(c * c - a * a)
  }else{
    warning("Kateta je daljša od hipotenuze. Menjam vrednosti.")
    b <- sqrt(a * a - c * c)
  }
  return(b)
}
```

Testirajmo:

```
kateta(1, sqrt(2))
```

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

```
kateta(3, 7)
```

```
## [1] 6.324555
```

```
kateta(7, 3) #Vrne pravilno vrednost in opozorilo
```

```
## Warning in kateta(7, 3): Kateta je daljša od hipotenuze. Menjam vrednosti.
```

```
## [1] 6.324555
```

```
kateta(-1, 2) #Vrne napako skupaj z našim opisom napake
```

```
## Error in kateta(-1, 2): Negativne vrednosti stranic!
```

```
kateta("a", "b") #Vrne napako, ker je tip vhoda napačen
```

```
## Error in kateta("a", "b"): Na vhodu morajo biti številke.
```

Obstaja še ena možnost za napako in sicer, če pozabimo podati vhodne parametre. V tem primeru, dobimo napako, ki nam sporoči, da ta parameter manjka.

```
kateta(1)
```

```
## Error in kateta(1): argument "c" is missing, with no default
```

Če želimo, našo funkcijo klicati tudi z manj parametri, lahko dodamo privzete vrednosti tem parametrom. Pazimo, da so vsi obvezni parametri definirani na začetku. Edina sprememba sedaj je v glavi funkcije `function (a = 1, c = 2)`.

```
kateta <- function (a = 1, c = 2) {
  if (!(is.numeric(a) & is.numeric(c))) {
    stop("Na vhodu morajo biti številke.")
  }
  if (a <= 0 | c <= 0) {
    stop("Negativne vrednosti stranic!")
  }
  if (c > a) {
    b <- sqrt(c * c - a * a)
  }else{
    warning("Kateta je daljša od hipotenuze. Menjam vrednosti.")
    b <- sqrt(a * a - c * c)
  }
  return(b)
}
```

```
kateta(0.5) #izračun za klica kateta(0.5, 2)
```

```
## [1] 1.936492
```

```
kateta() #izračun za klica kateta(0.5, 1)
```

```
## [1] 1.732051
```

Vračanje vrednosti funkcije

Funkcija lahko vrača vrednosti kjerkoli znotraj funkcije. Po klicu ukaza `return()` se izvajanje funkcije takoj zaključi. To lahko uporabimo za implementacijo svoje funkcije, shrani naše podatke v *csv* ali *xlsx* obliki. V funkciji bomo uporabili tudi ukaz `paste()`, ki združi vse parametre v en niz in med njih vrine znak podan s parametrom `sep`. Primer:

```
paste("A", "B", sep = "-")
```

```
## [1] "A-B"
```

```
paste("A", 2, 3, sep = ".")
```

```
## [1] "A.2.3"
```

Napišimo sedaj željeno funkcijo:

```
shraniPodatke <- function (podatki, datoteka, koncnica = "csv") {
  library(openxlsx)
  wd <- getwd() # prebermo delovni direktorij
  ime <- paste(wd, datoteka, sep = "/")
  celotno_ime <- paste(ime, koncnica, sep = ".")
  print(celotno_ime)
  if (koncnica == "csv") {
    write.table(podatki, file = celotno_ime, dec = ".", sep = ",", row.names = FALSE, col.names = TRUE)
    return()
  }
  if (koncnica == "xlsx") {
    write.xlsx(podatki, celotno_ime)
    return()
  }
  warning("Datoteka ni shranjena. Neznana končnica.")
  return()
}
```

```
datoteka <- "osebe"
shraniPodatke(df, datoteka)
```

```
## [1] "C:/Users/Patrik/Documents/MASTER FRI/Delavnice_FRI/repozitoriji/R-za-neprogramerje/Predavanje_0
```

```
## NULL
```

```
shraniPodatke(df, datoteka, koncnica = "xlsx")
```

```
## [1] "C:/Users/Patrik/Documents/MASTER FRI/Delavnice_FRI/repozitoriji/R-za-neprogramerje/Predavanje_0
```

```
## Error in saveWorkbook(wb, file = file, overwrite = overwrite): File already exists!
```

Območje spremenljivk

V prejšnjih funkcijah smo uporabljali spremenljivke `a`, `b` in `c`. Te spremenljivke so dostopne samo znotraj funkcije in enaka imena spremenljivk lahko uporabljamo zunaj funkcije. Prikažimo na preprostem primeru, ki uporablja dve spremenljivki `x` in `y`.

```
podvoji <- function (x) {
  print(x)
  print(y)
  y <- x * 2
  print(x)
  print(y)
  return (y)
}
x <- 7
y <- 8
print("Pred klicem")
```

```
## [1] "Pred klicem"
```

```r
print(x)
```

```
## [1] 7
```

```r
print(y)
```

```
## [1] 8
```

```r
print("Med klicem")
```

```
## [1] "Med klicem"
```

```r
x <- podvoji(x)
```

```
## [1] 7
## [1] 8
## [1] 7
## [1] 14
```

```r
print("Po klicu")
```

```
## [1] "Po klicu"
```

```r
print(x)
```

```
## [1] 14
```

```r
print(y)
```

```
## [1] 8
```

Pred klicem funkcije vidimo, da sta vrednosti `x = 7` in `y = 8`, kar je pričakovano. Takoj znotraj funkcije **podvoji** sta vrednosti še vedno enaki `x = 7` in `y = 8`. Znotraj funkcije je vredost `y` dosegljiva, čeprav je nismo podali kot vhod. Funkcija lahko dostopa tudi do vseh spremenljivk, ki so nastale v workspacu, ki je to funkcijo klical. Čeprav to deluje, se odsvetuje uporaba teh spremenljivk, ker je koda v takem primeru zelo nerazumljiva.

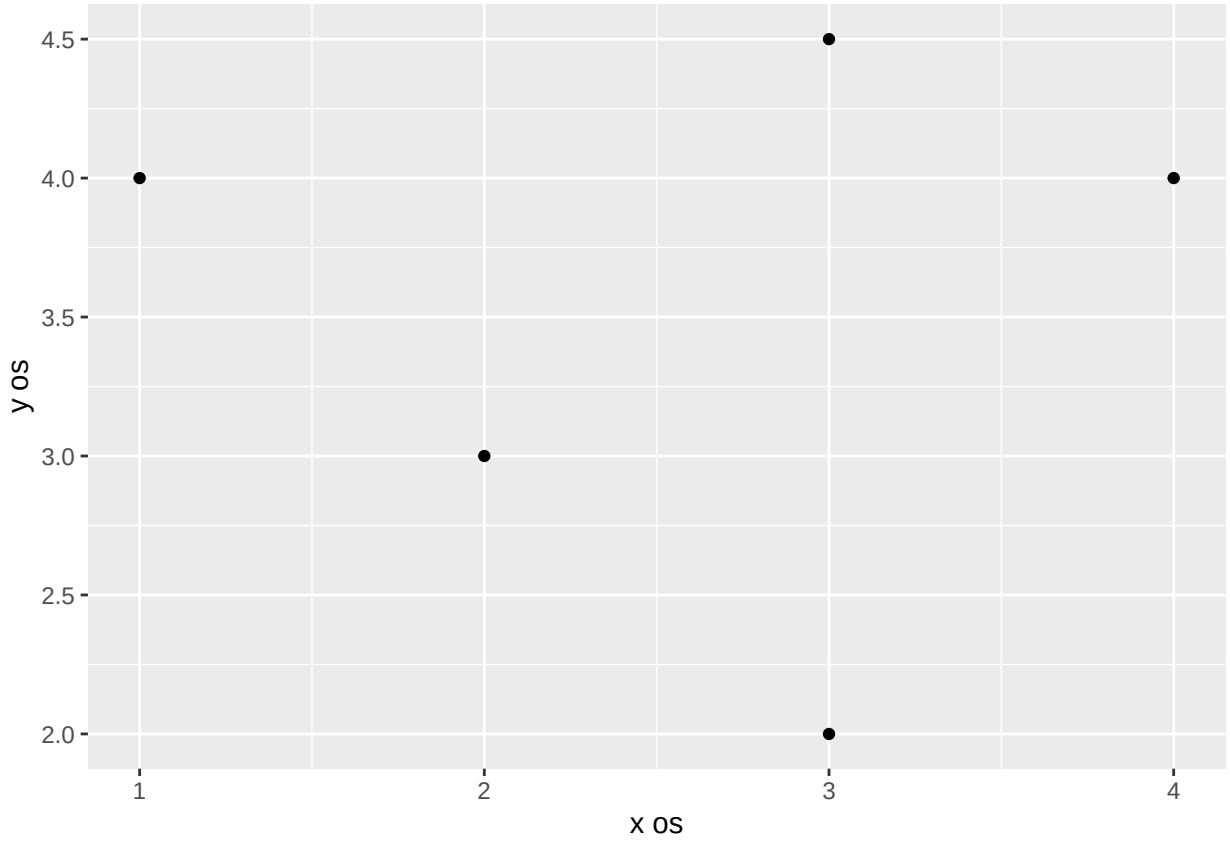
Pri drugem izpisu znotraj funkcije se je vrednost `y` spremenila na 14. Po klicu funckije (v workspace-u) pa spet velja `y = 8`. To je zato, ker ima vsaka funkcija svoje delovno območje in naredi kopijo spremenljivke `y`. Dobra lastnost tega je, da vsaka funcija deluje lokalno in z ukazi znotraj funkcije nikoli ne spremenimo vrednosti zunaj nje. Edina sprememba, ki se je na koncu zgodila, je `x = 14`, saj to vrednost funkcija dejansko vrne in jo spremenljivki `x` priredimo zunaj funkcije.

Domača naloga

1) Napišite funkcijo `razsevni_diagram(x, y, labelx, labely)`, ki zna z uporabo funkcije `ggplot()` izrisati razsevni diagram teh točk. Če label ne podamo sta privzeti imeni x in y.

Primer:

```
razsevni_diagram(c(1, 2, 3, 4, 3), c(4, 3, 2, 4, 4.5), "x os", "y os")
```



2) Dopolnite funkcijo `hipotenuza(a,b)` tako, da bo varnejša. Preverite, da je na vhodu res pozitivno število.

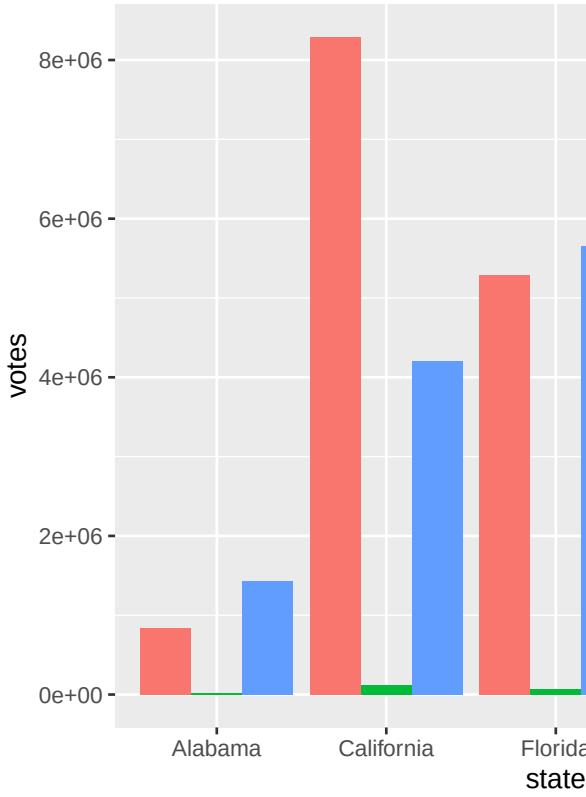
3) Napišite funkcijo `nalozi_priljubljene_pakete()`, s katero boste lahko v prihodnje lažje naložili vse pakete, ki jih uporabljate. Npr.: *ggplot2*, *openxlsx*, *tidyr*

4) Preberite podatke v mapi *data_raw* o ameriških volitvah. Podatke smo pobrali 6. novembra 2020 iz: https://www.kaggle.com/unanimad/us-election-2020?select=president_county_candidate.csv.

- Napište funkcijo `count_votes`, ki prejme kot parameter celo tabelo in ime kandidata, nato pa izpiše skupno število glasov, ki jih je ta kandidat prejel.

```
## [1] 70124380
```

- Napišite funkcijo `plot_votes`, ki za poljuben vektor imen zveznih držav izriše stolpični diagram s



številom glasov za vsako državo za stranke “REP”, “DEM” in “LIB”.

- (Težje) Napišite funkcijo `top_3`, ki prejme v vhodu tabelo in ime zvezne države in izpiše prva tri imena kandidatov ali kratice strank, glede na število glasov. Kaj naj izpiše naj določi parameter “output”, ki sprejme le niza ‘candidate’ ali ‘party’, drugače pa vrne napako.

```
##      candidate  votes
## 5 Donald Trump 764246
## 9    Joe Biden 420984
## 8 Jo Jorgensen  13095
```

- Razširite fleksibilnost funkcije `top_3` z dodatnim parametrom n, ki določa koliko vrstic naj funkcija vrne. Če bo n večji od vseh možnih vrstic za dano državo, naj vrne funkcija opozorilo in spremeni n v največje možno število vrstic. Funkcijo imenujte `top_n` in naj bo privzeta vrednost n-ja 5.

```
##        candidate  votes
## 5   Donald Trump 764246
## 9      Joe Biden 420984
## 8   Jo Jorgensen  13095
## 11    Kanye West   4156
## 7  Howie Hawkins   2969

## Warning in top_n(x, "Virginia", "candidate", 100): Prevelik ’n’, vrednost
## spremenjena v: 4

##      candidate   votes
## 4    Joe Biden 2363399
## 2 Donald Trump 1951677
## 3 Jo Jorgensen   64113
## 1    Write-ins    6842
```