

ModulA

Jana Obšteter

November 6, 2020

1 Osnove R-a

V R-u lahko ukazno kodo pišemo direktno v konzolo ali v urejevalnik, pri čemer se koda vedno izvrši v konzoli.

```
# Seštejemo števili 3 + 5  
3 + 5
```

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

Spremenljivkam vrednosti pripišemo z operatorjem `<-`, ki je značilen za programski jezik R, lahko pa uporabimo tudi `=`. Ime spremenljivke je poljubno in lahko vključuje velike in male črke, `.` in `/`. *Ne sme se začeti s številko ali* ter ne sme vključevati šumnikov.

```
# Spremenljivki stevilo pripišemo vrednosti 3  
stevilo <- 3
```

Vrstice, ki se pričnejo z `#` so komentarji in se ne izvršijo

```
# Nastavimo stevilo  
# stevilo <- 3
```

R-funkcije so vključene v različne pakete oz. knjižnice. Nekatere knjižnice z osnovnimi funkcijami so v R naložene kot privzeto, ostale pa moramo namestiti. Knjižnice namestimo z ukazom `install.packages("imePaketa")`. Namestitev paketa je enkratna, ob vsakem zagonu R-a pa moramo želene knjižnice naložiti v delovni prostor z ukazom `library(imePaketa)`.

```
# Kot primer naložimo paket 'base', ki je kot privzet vključen v R  
library(base)
```

2 Osnovni podatkovni tipi in strukture

Podatkovni tipi

R razlikuje med različnimi podatkovnimi tipi. V nadaljevanju bomo ustvarili spremenljivko vsakega izmed tipov.

Cela števila

```
# Ustvarimo spremenljivko "stevilo" z vrednostjo 10  
stevilo <- 10  
# Izpišimo "stevilo"  
stevilo
```

```
## [1] 10
```

```
# Preverimo tip spremenljivke  
class(stevilo)
```

```
## [1] "numeric"
```

R cela števila ustvari kot tip “numeric” in ne “integer”. Če želimo, da je spremenljivka eksplicitno tip “integer”, jo moramo pretvoriti.

```
# Ustvarimo spremenljivko "stevilo" z vrednostjo 10 kot "integer"  
stevilo <- as.integer(10)  
# Preverimo tip spremenljivke  
class(stevilo)
```

```
## [1] "integer"
```

Eksplicitno lahko tip “integer” ustvarimo tudi z dodatkom črke “L” na koncu celega števila.

```
# Direktno ustvarimo "integer" spremenljivko "stevilo" z vrednostjo 10  
stevilo <- 10L  
class(stevilo)
```

```
## [1] "integer"
```

Realna števila

```
# Ustvarimo spremenljivko "realnoStevilo" z vrednostjo 3.52  
realnoStevilo <- 3.52  
# Izpišimo "realnoStevilo"  
realnoStevilo
```

```
## [1] 3.52
```

```
# Preverimo tip spremenljivke  
class(realnoStevilo)
```

```
## [1] "numeric"
```

Znaki

Znake R navaja v navednicah, pri čemer je lahko znak črkovni, številčni ali kombinacija obeh.

```
# Ustvarimo spremenljivko "beseda" z vrednostjo "jagoda"  
beseda <- "jagoda"  
# Izpišimo spremenljivko "beseda"  
beseda
```

```
## [1] "jagoda"
```

```
# Preverimo tip spremenljivke  
class(beseda)
```

```
## [1] "character"
```

Logični vektor

Logične vrednosti v R-u so TRUE in FALSE.

```
# Ustvarimo spremenljivko "logicnaVrednost" z vrednostjo TRUE
logicnaVrednost <- TRUE
# Izpišimo "logicnaVrednost"
logicnaVrednost
```

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

```
# Preverimo tip spremenljivke
class(logicnaVrednost)
```

```
## [1] "logical"
```

Z logičnimi vrednostmi lahko tudi računamo, pri čemer je vrednost TRUE vredna 1, vrednost FALSE pa 0.

```
# Sesštejmo vrednosti TRUE in TRUE
TRUE + TRUE
```

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

Manjkajoča vrednost

Manjkajoče vrednosti v R-u so predstavljene kot NA.

```
# Ustvarimo spremenljivko "manjkaVrednost" z manjkajočo vrednostjo
manjkaVrednost <- NA
# Izpišimo "manjkaVrednost"
manjkaVrednost
```

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

```
# Preverimo tip spremenljivke
class(manjkaVrednost)
```

```
## [1] "logical"
```

Prazen element

Prazen element R predstavi kot NULL.

```
# Ustvarimo prazen element "prazenElement"
prazenElement <- NULL
# Izpišimo "prazenElement"
prazenElement
```

```
## NULL
```

```
# Preverimo tip spremenljivke
class(prazenElement)
```

```
## [1] "NULL"
```

Podatkovne strukture

V R-u lahko ustvarimo različne podatkovne strukture. V nadaljevanju bomo ustvarili vsako izmed struktur.

Vektor

Vektor ustvarimo s funkcijo `c()`. Vektor vključuje en podatkovni tip.

```
# Ustvarimo številski vektor "stevila" z vrednostmi 1, 10 in 100
stevila <- c(1, 10, 100)
# Izpišimo vektor
stevila

## [1] 1 10 100
```

Vektor številskega zaporedja lahko ustvarimo tudi z navedbo razpona števil.

```
# Ustvarimo številski vektor "mesece" s številskimi meseci 1 do 12
mesece <- 1:12
# Izpišimo vektor
mesece

## [1] 1 2 3 4 5 6 7 8 9 10 11 12
```

Vektorje znakov prav tako ustvarimo s funkcijo `c()`.

```
# Ustvarimo vektor znakov "letniCasi" z letnimi časi
letniCasi <- c("pomlad", "poletje", "jesen", "zima")
# Izpišimo vektor
letniCasi

## [1] "pomlad" "poletje" "jesen" "zima"
```

Elemente vektorjev izberemo z indeksom v oglatih oklepajih.

```
# Izberimo drugi element vektorja "letniCasi"
letniCasi[2]

## [1] "poletje"
```

Faktor

Faktorji so 'vektorji' kategoričnih spremenljivk. Vsebujejo določeno število vrednosti oz. ravni. Ustvarimo jih s funkcijo `factor()`.

```
# Ustvarimo faktor "letniCasi_f" s ponovljenimi vrednostmi za letne čase
# c("pomlad", "pomlad", "poletje", "poletje", "jesen", "jesen", "zime", "zima")
letniCasi_f <- factor(c("pomlad", "pomlad", "poletje", "poletje", "jesen", "jesen", "zime", "zima"))
# Izpišimo "letniCasi_f"
letniCasi_f

## [1] pomlad pomlad poletje poletje jesen jesen zime zima
## Levels: jesen poletje pomlad zima zime
```

Ko je faktor ustvarjen, ne moremo dodajati oz. ustvariti novih ravni.

```
# Poskušajmo dodati novo raven v faktor
letniCasi_f[9] <- "zjutraj"
```

```
## Warning in `[<-factor`(`*tmp*`, 9, value = "zjutraj"): invalid factor level, NA
## generated
```

```
# Ta operacija je nedovoljena, saj "zjutraj" ni ena izmed ravni faktorja (ne moremo ustvariti novih)
```

Seznam

Seznam ustvarimo s funkcijo `list()` in lahko vključujejo različne podatkovne tipe kot tudi strukture.

```
# Ustvarimo seznam "leto", ki vključuje vektorje
# "meseci" in "letniCasi" iz prejšnjega koraka in število 2020
leto <- list(meseci, letniCasi, 2020)
# Izpišimo seznam "leto"
leto
```

```
## [[1]]
## [1] 1 2 3 4 5 6 7 8 9 10 11 12
##
## [[2]]
## [1] "pomlad" "poletje" "jesen" "zima"
##
## [[3]]
## [1] 2020
```

```
# Preverimo dolžino seznama "leto"
length(leto)
```

```
## [1] 3
```

Elemente seznama izberemo z indeksom v dvojnih oglatih oklepajih.

```
# Izberemo prvi element seznama "leto"
leto[[1]]
```

```
## [1] 1 2 3 4 5 6 7 8 9 10 11 12
```

```
# Ker je prvi element vektor, izberimo še peti element znotraj tega
leto[[1]][5]
```

```
## [1] 5
```

Podatkovni okvir / tabela

Podatkovni okvir oz. tabelo ustvarimo s funkcijo `data.frame()`. Podatkovni okvir lahko v posameznih stolpcih vsebuje različne podatkovne tipe.

```
# Ustvarimo tabelo "povpTemp" z letnimi časi in povprečno temperature za vsakega izmed njih
# Predpostavimo, da je povprečna temperatura pomladi 18, poletju 24, jeseni 13 in pozimi 5
povpTemp <- data.frame(LetniCas = letniCasi, temp = c(18, 24, 13, 5))
# Izpišimo tabelo
print(povpTemp)
```

```
##   LetniCas temp
## 1   pomlad   18
## 2   poletje   24
## 3    jesen   13
## 4     zima    5
```

Prvih ali zadnjih par vrstic tabele izpišemo s funkcijo `head()` ali `tail()`.

```
# Izpišimo prvih par vrstic tabele "povpTemp"
head(povpTemp)
```

```
##   LetniCas temp
```

```
## 1    pomlad    18
## 2   poletje    24
## 3     jesen    13
## 4      zima     5
```

Privzeta vrednost funkcije head je 6 vrstic, zato izpiše celotno tabelo. Obnašanje funkcije head() lahko modificiramo s parametrom n, kjer navedemo število vrstic.

```
# Izpišimo prvi dve vrstici tabele "povpTemp"
head(povpTemp, n=2)
```

```
##    LetniCas temp
## 1    pomlad    18
## 2   poletje    24
```

Število vrstic in stolpcev preverimo s funkcijama nrow() in ncol().

```
# Preverimo število vrstic in stolpcev tabele "povpTemp"
nrow(povpTemp)
```

```
## [1] 4
```

```
ncol(povpTemp)
```

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

Imena vrstic in stolpcev preverimo s funkcijama rownames() in colnames().

```
# Preverimo imena vrstic in stolpcev tabele "povpTemp"
rownames(povpTemp)
```

```
## [1] "1" "2" "3" "4"
```

```
colnames(povpTemp)
```

```
## [1] "LetniCas" "temp"
```

Elemente tabele izberemo z indeksom vrstice/stolpca v oglatih oklepajih.

```
# Izberimo prvo vrednost v prvem stolpcu tabele "povpTemp"
povpTemp[1,1]
```

```
## [1] "pomlad"
```

```
# Izberimo prvo vrstico tabele "povpTemp"
povpTemp[1,]
```

```
##    LetniCas temp
## 1    pomlad    18
```

```
# Izberimo prvi stolpec tabele "povpTemp"
povpTemp[,1]
```

```
## [1] "pomlad" "poletje" "jesen"  "zima"
```

Stolpce tabel lahko izberemo tudi po imenu.

```
# Izberimo stolpec <LetniCasi> v tabeli "povpTemp"
povpTemp$LetniCas
```

```
## [1] "pomlad" "poletje" "jesen"  "zima"
```

Matrika

Matriko ustvarimo s funkcijo `matrix()`. Matrike ponavadi vsebujejo numerične spremenljivke, saj omogoča matrično računanje.

```
# Ustvarimo matriko "vrednosti" z vrednostmi 1:20, štirimi vrsticami in petimi stolpci
vrednosti <- matrix(1:20, nrow=4, ncol=5)
# Izpišimo matriko
vrednosti
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    5    9   13   17
## [2,]    2    6   10   14   18
## [3,]    3    7   11   15   19
## [4,]    4    8   12   16   20
```

Elemente matrike izberemo z indeksom vrstice/stolpca v oglatih oklepajih (enako kot tabele).

```
# Izberimo element v drugi vrstici tretjega stolpca matrike vrednosti
vrednosti[2, 3]
```

```
## [1] 10
```

Polje

Polje ustvarimo s funkcijo `array()`.

```
# Ustvarimo polje povpTempLeto s povprečnimi letnimi temperaturami v
# letnih časih v dveh različnih letih.
# Povprečna temperatūra je vektor vrednosti 18, 24, 13, 5, 15, 27, 12, 7; leti sta 2018 in 2019
# Dimenzije: x = letni čas (4), y = povprečna temperatura (1), z = leto (2)
povpTempLeto <- array(c(18, 24, 13, 5, 15, 27, 12, 7),
                      dim=c(4, 1, 2),
                      dimnames=list(letniCasi,
                                     "povpTemp",
                                     c(2018, 2019)))
# Izpišimo polje "povpTempLeto"
povpTempLeto
```

```
## , , 2018
##
##      povpTemp
## pomlad      18
## poletje     24
## jesen       13
## zima        5
##
## , , 2019
##
##      povpTemp
## pomlad      15
## poletje     27
## jesen       12
## zima        7
```

Dimenzije polja preverimo s funkcijo `dim()`, strukturo ps s funkcijo `str()`.

```
# Preverimo dimenzije polja "povpTempLeto"
dim(povpTempLeto)

## [1] 4 1 2

# Preverimo strukturo polja "povpTempLeto"
str(povpTempLeto)

## num [1:4, 1, 1:2] 18 24 13 5 15 27 12 7
## - attr(*, "dimnames")=List of 3
## ..$ : chr [1:4] "pomlad" "poletje" "jesen" "zima"
## ..$ : chr "povpTemp"
## ..$ : chr [1:2] "2018" "2019"
```

3 Osnovne operacije

Osnovni aritmetični operatorji

Osnovni aritmetični operatorji v R-u so + za seštevanje, - za odštevanje, * za množenje, / za deljenje.

Osnovne računske operacije lahko apliciramo na števila ali vektorje (enakih dolžin).

```
# Ustvarimo spremenljivki "a" z vrednostjo 2 in "b" z vrednostjo 11.2
a <- 2
b <- 11.2
# Ustvarimo spremenljivko "vsota" kot vsoto a in b
(vsota <- a + b)
```

```
## [1] 13.2
```

```
# Ustvarimo spremenljivko "razlike" kot vsoto a in b
(razlika <- a - b)
```

```
## [1] -9.2
```

```
# Ustvarimo spremenljivko "zmnožek" kot vsoto a in b
(zmnožek <- a * b)
```

```
## [1] 22.4
```

```
# Ustvarimo spremenljivko "kvocient" kot vsoto a in b
(kvocient <- a / b)
```

```
## [1] 0.1785714
```

Ponovimo vajo z vektorji.

```
# Ustvarimo vektorja "a" z vrednostmi c(1, 2, 3) in "b" z vrednostmi c(3.12, 5.44, 10)
a <- c(1, 2, 3)
b <- c(3.12, 5.44, 10)
# Ustvarimo spremenljivko "vsota" kot vsoto a in b
(vsota <- a + b)
```

```
## [1] 4.12 7.44 13.00
```

```
# Ustvarimo spremenljivko "razlike" kot vsoto a in b
(razlika <- a - b)
```

```
## [1] -2.12 -3.44 -7.00
```



```
# Ustvarimo spremenljivko "zmnožek" kot vsoto a in b
(zmnožek <- a * b)
```

```
## [1] 3.12 10.88 30.00
```

```
# Ustvarimo spremenljivko "kvocient" kot vsoto a in b
(kvocient <- a / b)
```

```
## [1] 0.3205128 0.3676471 0.3000000
```

R ima posebej operator za ostanek pri deljenju `%%`, celoštevilsko deljenje `%/%` in množenje matrik `%*%`.

```
# Preverimo ostanek pri deljenju 10 / 3
10 %% 3
```

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

```
# Preverimo celoštevilski kvocient deljenja 10 / 3
10 %/% 3
```

```
## [1] 3
```

```
# Ustvarimo matriko m1 z vrednostmi 1:9 in tremi vrsticami
# ter matriko m2 z vrednostmi 9:18 in tremi vrsticami
m1 <- matrix(1:9, nrow=3)
m2 <- matrix(10:18, nrow=3)
m1 %*% m2
```

```
##      [,1] [,2] [,3]
## [1,] 138 174 210
## [2,] 171 216 261
## [3,] 204 258 312
```

Primerjalni operatorji

V R-u lahko uporabimo tudi operatorje primerjav, ki vrnejo logične vrednosti. Primerjamo lahko katerikoli podatkovni tip ali strukturo. Primerjalni operatorji so `>` za večje, `<` za manjše, `==` za določanje enakosti in `!=` za določanje neenakosti. Primerjalni operatorji vrnejo logično vrednost.

```
# Ustvarimo spremenljivko a z vrednotjo 5
a <- 5
# Preverimo, ali je 5 večje od 3
a > 3
```

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

```
# Preverimo, ali je 5 enako 3
a == 3
```

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

```
# Preverimo, ali 5 ni enako 3
a != 3
```

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

```
# Preverimo, ali vektor c(1, 2, 3) vsebuje 4
a %in% c(1,2,3)
```

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

Operatorje primerjav lahko uporabimo tudi za druge podatkovne tipe. Logične vrednosti so interne zapisane kot 1 (TRUE) in 0 (FALSE), tako da jih lahko tudi seštevamo.

```
# Preverimo, koliko vrednosti a (5) je v vektorju vrednosti = c(1, 3, 5, 4, 5, 7)
vrednosti = c(1, 3, 5, 4, 5, 7)
sum(vrednosti == a)
```

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

“Operatorji” za znake

Za združevanje znakov ne moremo uporabiti enakih operacij kot za številske spremenljivke. Za združevanje znakov uporabimo funkcijo `paste()`, kjer s parametrom `sep` navedemo željeno ločilo

```
# Združimo besedi Janez in Novak s presledkom
paste("Janez", "Novak", sep=" ")
```

```
## [1] "Janez Novak"
```

Lahko uporabimo tudi funkcijo `paste0()`, ki ima kot privzeto ločilo “”.

```
# Združimo besedi Janez in Novak brez separatorja
paste0("Janez", "Novak")
```

```
## [1] "JanezNovak"
```

Osnovne vgrajene funkcije

R vključuje kopico vgrajenih funkcij za najrazličnejše pogosto uporabljene operacije. Nekaj osnovnih funkcij vključuje `sum()` za vsoto, `mean()` za povprečje, `var()` za varianco, `sd()` za standardni odklon, `length()` za dolžino elementa, `min()` za minimalno vrednost, `max()` za maksimalno vrednosti in `print()` za prikaz. Večina navedenih funkcij deluje na vektorjih, nekatere pa tudi na drugih elementih.

```
# Ustvarimo vektor "stevila" z vrednostmi 1:10
stevila <- 1:10
# Seštejmo elemente vektorja s funkcijo sum()
(sum(stevila))
```

```
## [1] 55
```

```
#Izračunajmo povprečje in varianco vektorja "stevila" s funkcijo mean() in var()
(mean(stevila))
```

```
## [1] 5.5
```

```
(var(stevila))
```

```
## [1] 9.166667
```

```
# Preverimo dolžino vektorja stevila s funkcijo length()
(length(stevila))
```

```
## [1] 10
```

```
# Preverimo minimalno in maksimalno vrednost vektorja "stevila" s funkcijama min() in max()
(min(stevila))
```

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

```
(max(stevila))
```

```
## [1] 10
```

Pri računskih operacijah lahko naletimo tudi na manjkajoče vrednosti, ki jih kot privzeto R ne odstrani.

```
# Ustvarimo vektor vrednosti <- c(1, 2, 3, 4, NA)
vrednosti <- c(1, 2, 3, 4, NA)
# Seštejmo vektor vrednosti
sum(vrednosti)
```

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

R vrne manjkajočo vrednosti, saj ne more izračunati vsote zaradi manjkajočih vrednosti. V pomoči za funkcije lahko preverimo, kako prilagoditi obnašanje funkcije. Funkcija `sum()` naprimer vključuje logični parameter `na.rm`, ki določi, ali naj bodo manjkajoče vrednosti odstranjene.

```
# Preverimo pomoč za funkcijo sum()
?sum()
# Ponovno sštejmo vektor vrednosti
# Ker želimo odstraniti manjkajočih vrednosti, nastavimo parameter na.rm na FALSE
(sum(vrednosti, na.rm = TRUE))
```

```
## [1] 10
```

4 Osnove dela z datotekami

Nastavitev delovnega imenika

V R-u lahko nastavimo delovni imenik s funkcijo `setwd()`. Ta postane privzeta pot za branje in pisanje datotek.

```
getwd()
```

```
## [1] "/home/jana/Documents/Delavnice/R/Material"
```

Na primeru bomo pogledali, kako v R prebrati tabelo, izluščiti lastnosti podatkov, preurediti tabelo in zapisati rezultat.

Branje datotek

Različne tipe datotek v R preberemo z različnimi funkcijami oz. različnimi parametri. V nadaljevanje bomo v R prebrali .csv datoteko ter pregledali vsebino tabele. Ime datoteke je `ZivljenjskaDoba.csv`, ki vsebuje podatke zbrane s strani organizacija WHO o življenjski dobi (`ZivDoba`) v izbranih državah (`Country`). Datoteka vsebuje tudi podatke o statusu države (`Status`), Umrljivosti, zaužitem alkoholu v litrih (`Alkohol`), odstotek imuniziranih 1-letnikov na hepatitis B (`HepatitisB`), število primerov ošpic na 1000 prebivalcev (`Ospice`) in indeks telesne mase (`ITM`).

```
# Preberimo datoteko ZivljenjskaDoba_mali.csv
zivDoba <- read.csv("ZivljenjskaDoba.csv")
```

Lastnosti prebranih podatkov

Najprej preverimo, ali so podatki prebrani pravilno. Ponavadi pogledamo prvih par vrstic s funkcijo `head()`, zadnjih par s funkcijo `tail()` in pa strukturo podatkov s funkcijo `str()`.

```
# Preverimo prvih par vrstic prebrane tabele s funkcijo head()
head(zivDoba)
```

```
##   Country Leto      Status ZivDoba Umrlijivost Alkohol HepatitisB Ospice  ITM
## 1 Austria 2015 Developed   81.5         65      NA        93   309 57.6
## 2 Austria 2014 Developed   81.4         66  12.32        98   117 57.1
## 3 Austria 2013 Developed   81.1         68  11.82        95     0 56.6
## 4 Austria 2012 Developed   88.0          7  12.26        92    36 56.1
## 5 Austria 2011 Developed   88.0         73  12.04        89    68 55.7
## 6 Austria 2010 Developed   84.0         75  12.10        86    52 55.2
##   Polio HIV      BDP Solanje
## 1    93 0.1 43665.9470   15.9
## 2    98 0.1 51322.6400   15.9
## 3    95 0.1   554.7153   15.7
## 4    92 0.1 48333.5727   15.7
## 5    89 0.1 51126.7414   15.7
## 6    86 0.1 46657.6290   15.4
```

```
# Preverimo zadnjih par vrstic prebrane tabele s funkcijo tail()
tail(zivDoba)
```

```
##   Country Leto      Status ZivDoba Umrlijivost Alkohol HepatitisB Ospice  ITM
## 123 Sweden 2005 Developed   85.0         66    6.5      NA    13 55.3
## 124 Sweden 2004 Developed   83.0          7    6.6      NA     5 54.9
## 125 Sweden 2003 Developed   82.0         69    6.9      NA     3 54.4
## 126 Sweden 2002 Developed   79.9         71    6.9      NA     9 53.9
## 127 Sweden 2001 Developed   79.8         73    6.6      NA     5 53.4
## 128 Sweden 2000 Developed   79.6         73    6.2      NA    59 52.8
##   Polio HIV      BDP Solanje
## 123    98 0.1 4385.353   16.0
## 124    99 0.1 42442.224   16.0
## 125    99 0.1 36961.425   15.9
## 126    99 0.1 29571.745   15.9
## 127    99 0.1 26969.245   15.9
## 128    99 0.1 29283.550   15.9
```

```
# Preverimo strukturo prebrane tabele s funkcijo str()
str(zivDoba)
```

```
## 'data.frame':   128 obs. of  13 variables:
## $ Country   : chr  "Austria" "Austria" "Austria" "Austria" ...
## $ Leto      : int   2015 2014 2013 2012 2011 2010 2009 2008 2007 2006 ...
## $ Status    : chr  "Developed" "Developed" "Developed" "Developed" ...
## $ ZivDoba   : num   81.5 81.4 81.1 88 88 84 82 84 81 79.8 ...
## $ Umrlijivost: int    65 66 68 7 73 75 77 76 8 81 ...
## $ Alkohol   : num   NA 12.3 11.8 12.3 12 ...
## $ HepatitisB: int    93 98 95 92 89 86 83 83 85 83 ...
## $ Ospice    : int   309 117 0 36 68 52 49 448 20 23 ...
## $ ITM       : num   57.6 57.1 56.6 56.1 55.7 55.2 54.7 54.2 53.7 53.2 ...
## $ Polio     : int    93 98 95 92 89 86 83 83 85 83 ...
## $ HIV       : num   0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 ...
## $ BDP       : num  43666 51323 555 48334 51127 ...
## $ Solanje   : num   15.9 15.9 15.7 15.7 15.7 15.4 15.3 15.1 15.2 15 ...
```

S funkcijami `as.podatkovniTip()` lahko spremenimo podatkovni tip stolpcev.

```
# Vidimo, da je leto kodirano kot številka spremenljivka (integer).
# S funkcijo as.factor() jo spremenimo v faktor (kategorično spremenljivko).
zivDoba$Leto <- as.factor(zivDoba$Leto)
```

Preverimo lahko tudi število prebranih stolpcev in vrstic s funkcijama ncol() in nrow().

```
# Preverimo število vrstic in število stolpcev
nrow(zivDoba)
```

```
## [1] 128
```

```
ncol(zivDoba)
```

```
## [1] 13
```

S funkcijo colnames(objekt)[indeksStolpca] lahko preimenujemo stolpce.

```
# Preimenujmo stolpec "Country" v "Drzava"
colnames(zivDoba)[1] <- "Drzava"
```

Vrstice tabele izberemo z indeksom v oglatih oklepajih. Stolpce tabele lahko prav tako izberemo z indeksom v oglatih oklepajih ali znakom \$ in imenom stolpca. Če v indeksu ne navedemo številke stolpca (ali vrstice), R izbere vse stolpce (vrstice).

```
# Izberimo tretjo vrstico tabele zivDoba z indeksom
zivDoba[3,]
```

```
##      Drzava Leto      Status ZivDoba Umrljivost Alkohol HepatitisB Ospice  ITM
## 3 Austria 2013 Developed    81.1          68    11.82          95      0 56.6
##      Polio HIV      BDP Solanje
## 3      95 0.1 554.7153    15.7
```

```
# Izberimo tretji stolpec tabele zivDoba z indeksom [vrstica, stolpec]
zivDoba[, 3]
```

```
##      [1] "Developed" "Developed" "Developed" "Developed" "Developed"
##      [6] "Developed" "Developed" "Developed" "Developed" "Developed"
##     [11] "Developed" "Developed" "Developed" "Developed" "Developed"
##     [16] "Developed" "Developing" "Developing" "Developing" "Developing"
##     [21] "Developing" "Developing" "Developing" "Developing" "Developing"
##     [26] "Developing" "Developing" "Developing" "Developing" "Developing"
##     [31] "Developing" "Developing" "Developing" "Developing" "Developing"
##     [36] "Developing" "Developing" "Developing" "Developing" "Developing"
##     [41] "Developing" "Developing" "Developing" "Developing" "Developing"
##     [46] "Developing" "Developing" "Developing" "Developing" "Developing"
##     [51] "Developing" "Developing" "Developing" "Developing" "Developing"
##     [56] "Developing" "Developing" "Developing" "Developing" "Developing"
##     [61] "Developing" "Developing" "Developing" "Developing" "Developing"
##     [66] "Developing" "Developing" "Developing" "Developing" "Developing"
##     [71] "Developing" "Developing" "Developing" "Developing" "Developing"
##     [76] "Developing" "Developing" "Developing" "Developing" "Developing"
##     [81] "Developed" "Developed" "Developed" "Developed" "Developed"
##     [86] "Developed" "Developed" "Developed" "Developed" "Developed"
##     [91] "Developed" "Developed" "Developed" "Developed" "Developed"
##     [96] "Developed" "Developing" "Developing" "Developing" "Developing"
##    [101] "Developing" "Developing" "Developing" "Developing" "Developing"
##    [106] "Developing" "Developing" "Developing" "Developing" "Developing"
##    [111] "Developing" "Developing" "Developed" "Developed" "Developed"
##    [116] "Developed" "Developed" "Developed" "Developed" "Developed"
```

```
## [121] "Developed" "Developed" "Developed" "Developed" "Developed"
## [126] "Developed" "Developed" "Developed"
```

```
# Izberimo tretji stolpcev tabele zivDoba z imenom (Status)
zivDoba$Status
```

```
## [1] "Developed" "Developed" "Developed" "Developed" "Developed"
## [6] "Developed" "Developed" "Developed" "Developed" "Developed"
## [11] "Developed" "Developed" "Developed" "Developed" "Developed"
## [16] "Developed" "Developing" "Developing" "Developing" "Developing"
## [21] "Developing" "Developing" "Developing" "Developing" "Developing"
## [26] "Developing" "Developing" "Developing" "Developing" "Developing"
## [31] "Developing" "Developing" "Developing" "Developing" "Developing"
## [36] "Developing" "Developing" "Developing" "Developing" "Developing"
## [41] "Developing" "Developing" "Developing" "Developing" "Developing"
## [46] "Developing" "Developing" "Developing" "Developing" "Developing"
## [51] "Developing" "Developing" "Developing" "Developing" "Developing"
## [56] "Developing" "Developing" "Developing" "Developing" "Developing"
## [61] "Developing" "Developing" "Developing" "Developing" "Developing"
## [66] "Developing" "Developing" "Developing" "Developing" "Developing"
## [71] "Developing" "Developing" "Developing" "Developing" "Developing"
## [76] "Developing" "Developing" "Developing" "Developing" "Developing"
## [81] "Developed" "Developed" "Developed" "Developed" "Developed"
## [86] "Developed" "Developed" "Developed" "Developed" "Developed"
## [91] "Developed" "Developed" "Developed" "Developed" "Developed"
## [96] "Developed" "Developing" "Developing" "Developing" "Developing"
## [101] "Developing" "Developing" "Developing" "Developing" "Developing"
## [106] "Developing" "Developing" "Developing" "Developing" "Developing"
## [111] "Developing" "Developing" "Developed" "Developed" "Developed"
## [116] "Developed" "Developed" "Developed" "Developed" "Developed"
## [121] "Developed" "Developed" "Developed" "Developed" "Developed"
## [126] "Developed" "Developed" "Developed"
```

S funkcijo `table()` preverimo zastopanost posamezne spremenljivke znotraj stolpcev, pri čemer je funkcija bolj uporabna za kategorične spremenljivke.

```
# Preverimo, koliko podatkov (vrstic) imamo po državi s funkcijo table()
table(zivDoba$Drzava)
```

```
##
## Austria Brazil Chile Egypt France Lithuania Peru Sweden
## 16 16 16 16 16 16 16 16
```

Za preverjanje stolpcev v numeričnih stolpcev je bolje uporabiti funkcijo `summary()`. Ta vrne minimalno (in maksimalno) vrednost spremenljivke, prvi (in tretji) kvantil in povprečje.

```
# Preverimo vsebnost stolpca ZivDoba s funkcijo summary
summary(zivDoba$ZivDoba)
```

```
## Min. 1st Qu. Median Mean 3rd Qu. Max.
## 68.60 72.80 77.95 77.13 81.10 89.00
```

Za preverjanje parametrov podatkov lahko uporabimo tudi funkcije kot npr. `mean()` za povprečje, `sd()` za standardni odklon, `var()` za varianco itd.

```
# Preverimo povprečje življenjske dobe preko vseh držav
mean(zivDoba$ZivDoba)
```

```
## [1] 77.12734
```

```
# Preverimo varianco življenjske dobe
var(zivDoba$ZivDoba)
```

```
## [1] 25.82011
```

```
# Preverimo minimalno umrljivost v podatkih
min(zivDoba$Umrlijivost)
```

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

```
# Preverimo maksimalno umrljivost v podatkih
max(zivDoba$Umrlijivost)
```

```
## [1] 229
```

Osnovne operacije na tabelah

V tabeli lahko uporabimo operatorje primerjav, da izberemo samo določene vrstice ali vrednosti.

```
# Izberimo vse vrstice tabele, v katerih je vrednost spremenljivke ZivDoba večja od 85
zivDoba[zivDoba$ZivDoba > 85,]
```

##	Drzava	Leto	Status	ZivDoba	Umrlijivost	Alkohol	HepatitisB	Ospice	ITM
## 4	Austria	2012	Developed	88	7	12.26	92	36	56.1
## 5	Austria	2011	Developed	88	73	12.04	89	68	55.7
## 72	France	2008	Developing	89	88	11.90	47	604	59.1
## 73	France	2007	Developing	89	89	12.20	42	39	58.6
## 74	France	2006	Developing	86	92	12.40	39	40	58.0
## 121	Sweden	2007	Developed	89	63	6.90	NA	1	56.1
## 122	Sweden	2006	Developed	88	64	6.50	NA	19	55.7
##	Polio	HIV	BDP	Solanje					
## 4	92	0.1	48333.573	15.7					
## 5	89	0.1	51126.741	15.7					
## 72	98	0.1	45413.657	16.1					
## 73	99	0.1	416.584	16.1					
## 74	99	0.1	36544.585	16.1					
## 121	98	0.1	53324.379	15.8					
## 122	98	0.1	46256.472	15.9					

```
# Izberimo vse vrstice, v katerih je vrednost stolpca "Drzava" enaka "France"
zivDoba[zivDoba$Drzava == "France",]
```

##	Drzava	Leto	Status	ZivDoba	Umrlijivost	Alkohol	HepatitisB	Ospice	ITM
## 65	France	2015	Developing	82.4	78	NA	86	157	62.5
## 66	France	2014	Developing	82.2	79	11.50	83	267	62.0
## 67	France	2013	Developing	82.0	81	11.10	74	272	61.6
## 68	France	2012	Developing	81.5	83	11.50	78	0	61.1
## 69	France	2011	Developing	81.7	83	11.80	74	14949	6.6
## 70	France	2010	Developing	81.3	86	11.70	65	5048	6.1
## 71	France	2009	Developing	81.1	88	11.80	51	1541	59.6
## 72	France	2008	Developing	89.0	88	11.90	47	604	59.1
## 73	France	2007	Developing	89.0	89	12.20	42	39	58.6
## 74	France	2006	Developing	86.0	92	12.40	39	40	58.0
## 75	France	2005	Developing	81.0	93	12.20	35	36	57.5
## 76	France	2004	Developing	82.0	94	13.18	35	4448	57.0
## 77	France	2003	Developing	79.3	99	13.49	28	0	56.4

```
## 78 France 2002 Developing 79.2 11 13.78 29 5185 55.8
## 79 France 2001 Developing 79.0 13 13.89 28 0 55.2
## 80 France 2000 Developing 78.8 13 13.63 26 10000 54.6
## Polio HIV BDP Solanje
## 65 98 0.1 36526.7711 16.3
## 66 98 0.1 42955.2429 16.2
## 67 99 0.1 42554.1225 16.2
## 68 99 0.1 4838.2444 16.1
## 69 99 0.1 4381.2880 16.1
## 70 99 0.1 473.3428 16.0
## 71 98 0.1 41631.1314 16.0
## 72 98 0.1 45413.6571 16.1
## 73 99 0.1 416.5840 16.1
## 74 99 0.1 36544.5853 16.1
## 75 98 0.1 34879.7263 15.5
## 76 99 0.1 33874.7426 15.5
## 77 96 0.1 29691.1816 15.4
## 78 97 0.1 24275.2426 15.5
## 79 98 0.1 22527.3177 15.6
## 80 98 0.1 22465.6418 15.7
```

Tabelam lahko tudi dodajamo in odstranjujemo vrstice in stolpce. Stolpce in vrstice odstranjujemo z znakom - (minus) ter navedbo indeksa.

```
# Odstranimo stolpec "Polio" z indeksom iz tabele zivDoba
# novo ustvarjeno tabelo poimenujemo zivDoba_nov
zivDoba_nov <- zivDoba[, -10]
# Preverimo imena stolpcev nove tabele zivDoba_nov s funkcijo colnames()
colnames(zivDoba_nov)
```

```
## [1] "Drzava" "Leto" "Status" "ZivDoba" "Umrljivost"
## [6] "Alkohol" "HepatitisB" "Ospice" "ITM" "HIV"
## [11] "BDP" "Solanje"
```

Izbrane vrstice in/ali stolpce lahko tudi zapišemo v nov objekt oz. tabelo.

```
# Ustvarimo vektor 'ospiceFR', v katerega zberemo podatke o letu in pojavnosti ošpic v Franciji
ospiceFR <- zivDoba[zivDoba$Drzava == "France", c("Leto", "Ospice")]
head(ospiceFR)
```

```
## Leto Ospice
## 65 2015 157
## 66 2014 267
## 67 2013 272
## 68 2012 0
## 69 2011 14949
## 70 2010 5048
```

```
# Preimenuj drugi stolpceviz 'Ospice' v 'ospiceFR'
colnames(ospiceFR)[2] <- 'ospiceFR'
head(ospiceFR)
```

```
## Leto ospiceFR
## 65 2015 157
## 66 2014 267
## 67 2013 272
## 68 2012 0
```



```
## 69 2011      14949
## 70 2010      5048

# Ustvarimo vektor 'ospiceBR', v katerega zberemo podatke o letu in pojavnosti ošpic v Braziliji
ospiceBR <- zivDoba[zivDoba$Drzava == "Brazil", c("Leto", "Ospice")]
# Preimenuj drugi stolpceviz 'Ospice' v 'ospiceBR'
colnames(ospiceBR)[2] <- 'ospiceBR'

# Ustvarimo vektor 'ospiceAU' in 'ospiceEG'
# Vanju zberemo podatke o pojavnosti ošpic v Avstriji in Egiptu
ospiceAU <- zivDoba$Ospice[zivDoba$Drzava == "Austria"]
ospiceEG <- zivDoba$Ospice[zivDoba$Drzava == "Egypt"]
```

Tabelam lahko tudi dodajamo vrstice ali stolpce z navedbo indeksa vrstice/stolpca ali imena stolpca ter vsebino.

```
# Ustvarimo nov stolpec "Celina" v tabeli zivDoba in mu pripišimo vrednost "Evropa"
zivDoba$Celina <- "Evropa"
# Preverimo tabelo
head(zivDoba)
```

```
##      Drzava Leto      Status ZivDoba Umrljivost Alkohol HepatitisB Ospice  ITM
## 1 Austria 2015 Developed    81.5          65      NA          93    309 57.6
## 2 Austria 2014 Developed    81.4          66  12.32          98    117 57.1
## 3 Austria 2013 Developed    81.1          68  11.82          95      0 56.6
## 4 Austria 2012 Developed    88.0           7  12.26          92     36 56.1
## 5 Austria 2011 Developed    88.0          73  12.04          89     68 55.7
## 6 Austria 2010 Developed    84.0          75  12.10          86     52 55.2
##      Polio HIV          BDP Solanje Celina
## 1      93 0.1 43665.9470    15.9 Evropa
## 2      98 0.1 51322.6400    15.9 Evropa
## 3      95 0.1   554.7153    15.7 Evropa
## 4      92 0.1 48333.5727    15.7 Evropa
## 5      89 0.1 51126.7414    15.7 Evropa
## 6      86 0.1 46657.6290    15.4 Evropa
```

V tabelah lahko tudi spreminjamo vrednost. Stolpce, vrstica ali polja, katerim želimo spremeniti vrednost, lahko izberemo z indeksom, imenom stolpca ali pa operatorji primerjav (če želimo spremeniti le del vrednosti, ki ustrezajo nekemu pogoju).

```
# Vse države v tabeli zivDob ne ležijo v Evropi, in sicer "Egypt", "Brazil", "Chile" in "Peru"
# Naprej v vsrticah z vrednostjo "Egypt" spremenimo vrednost v stolpcu "Celina" na "Afrika"
#
zivDoba$Celina[zivDoba$Drzava == "Egypt"] <- "Afrika"

#"Brazil", "Chile" in "Peru" ležijo v Južni Ameriki
# Tem vrsticam spremenimo vrednost v stolpcu "Celina" na JuznaAmerika
zivDoba$Celina[zivDoba$Drzava %in% c("Brazil", "Peru", "Chile")] <- "JuznaAmerika"

# Prevririmo zastopanost vrednosti v stolpcu "Celina"
table(zivDoba$Celina)
```

```
##
##      Afrika      Evropa JuznaAmerika
##      16          64          48
```

Pisanje podatkov

S funkcijo `write.csv()` ali `write.table()` zapišemo podatke v datoteko.

```
# S funkcijo write.csv() zapišemo popravljeno tabelo zivDoba v datoteko
# ZivljenjskaDoba_popravljena.csv
# Shranimo podatke brez navednic in brez imena vrstic
write.csv(zivDoba, "ZivljenjskaDoba_popravljena.csv", quote=FALSE, row.names=FALSE)
```

5 Preurejanje podatkov

S funkcijo `cbind()` lahko povežemo vektorje v novo tabelo.

```
# S funkcijo cbind() združimo zgoraj ustvarjene vektorje 'ospiceFR', 'ospiceAU' in 'ospiceEG'
# v novo tabelo 'ospice'
ospice <- cbind(ospiceAU, ospiceEG)
# Preverimo novonastalo tabelo
head(ospice)
```

```
##      ospiceAU ospiceEG
## [1,]      309      5432
## [2,]      117      1314
## [3,]         0       405
## [4,]        36       245
## [5,]        68        26
## [6,]        52        16
```

S funkcijo `merge()` lahko združujemo tabele na podlagi skupnega ključa.

```
# S funkcijo merge() združimo zgoraj ustvarjene vektorje 'ospiceFR' in 'ospiceBR'
# na podlagi spremenljivke 'Leto' v novo tabelo 'ospiceMerged'
ospiceMerged <- merge(ospiceFR, ospiceBR, by="Leto")
# Preverimo novonastalo tabelo
head(ospiceMerged)
```

```
##   Leto ospiceFR ospiceBR
## 1 2000    10000         36
## 2 2001         0          1
## 3 2002     5185          1
## 4 2003         0          2
## 5 2004     4448          0
## 6 2005         36          6
```

S funkcijo `melt()` iz paketa `reshape` lahko "vrtimo" tabele: spreminjamo razporeditev podatkov v stolpce in vrstice.

```
# Recimo, da želimo imeti en stolpec z vsemi vrednostmi za hepatitisB, polio in HIV za vse države,
# v drugem stolpcu pa bi imeli spremenljivko, za katero bolezen gre
# Takšno organizacijo podatkov ponavadi potrebujemo za grafično prikazovanje podatkov
# Najprej izberemo željene stolpce celotne tabele zivDoba - Drzava, Leto, HepatitisB, HIV in Polio
bolezni <- zivDoba[, c("Drzava", "Leto", "HepatitisB", "HIV", "Polio")]
# Preverimo novonastalo tabelo
head(bolezni)
```

```
##      Drzava Leto HepatitisB HIV Polio
## 1 Austria 2015          93 0.1    93
```

```
## 2 Austria 2014      98 0.1   98
## 3 Austria 2013      95 0.1   95
## 4 Austria 2012      92 0.1   92
## 5 Austria 2011      89 0.1   89
## 6 Austria 2010      86 0.1   86
```

```
# S funkcijo melt() iz paketa reshape preuredimo tabelo,
# tako da sta naša ključa Leto in Drzava (stolpca bosta ostala nespremenjena)
# Novo nastalo tabelo poimenujemo bolezni_melt
library(reshape)
bolezni_melt <- melt(bolezni, id.vars = c("Leto", "Drzava"))
# Preveimo novonastalo tabelo s funkcijo head()
head(bolezni_melt)
```

```
##   Leto Drzava variable value
## 1 2015 Austria HepatitisB    93
## 2 2014 Austria HepatitisB    98
## 3 2013 Austria HepatitisB    95
## 4 2012 Austria HepatitisB    92
## 5 2011 Austria HepatitisB    89
## 6 2010 Austria HepatitisB    86
```

```
#Preverimo vrednosti v stolpcu 'variable' tabele bolezni_melt
table(bolezni_melt$variable)
```

```
##
## HepatitisB      HIV      Polio
##          128      128      128
```

tidyr funkcije za preurejanje podatkov

tidyr je paket, ki vsebuje nabor funkcij za urejanje podatkov v R-u (<https://tidyr.tidyverse.org/reference/index.html>). Teži k "urejenosti" tabel in omogoča tudi podajanje podatkov med funkcijami z operatorjem %>%. Paket tidyr operira s podatkovni strukturo tibble (in ne data frame). Ta je preprostejša in omogoča manj operacij, s čimer pa omogoča hitrejšo odpravo napak in poenostavljenje kode. Paket naložimo z ukazom library(tidyr).

Funkcija pivot_longer() opravlja enako nalogo kot funkcija melt(), ki smo jo uporabili zgoraj. Tabelo preuredi tako, da ima več vrstic in manj stolpcev oz. podaljša tabelo.

```
# Ponovimo vajo
# Sfunkcijo pivot_longer() iz paketa tidyr preuredimo tabelo bolezni (ustvarjena zgoraj),
# tako da zberemo vse vredosti za stolpce HepatitisB, HIV in Polio v enem stolpcu
# Novo nastalo tabelo poimenujemo bolezni_longer
library(tidyr)
```

```
##
## Attaching package: 'tidyr'

## The following objects are masked from 'package:reshape':
##
##   expand, smiths

bolezni_longer <- pivot_longer(data = bolezni, cols = c(HepatitisB, HIV, Polio), names_to = "Bolezen")
# Preverimo novonastalo tabelo
head(bolezni_longer)
```

```
## # A tibble: 6 x 4
##   Drzava Leto Bolezen   value
##   <chr>   <fct> <chr>     <dbl>
## 1 Austria 2015 HepatitisB  93
## 2 Austria 2015 HIV           0.1
## 3 Austria 2015 Polio        93
## 4 Austria 2014 HepatitisB  98
## 5 Austria 2014 HIV           0.1
## 6 Austria 2014 Polio        98
```

Obratna funkcija je `pivot_wider()`, ki ustvari tablo z več stolpci in manj vrsticami oz. razširi tabelo.

```
# S funkcijo pivot_wider() razširimo zgornjo tabelo bolezni_longer,
# da bodo podatki za vsako državo v svojem stolpcu
bolezni_wider <- pivot_wider(data = bolezni_longer, id_cols = c(Leto, Bolezen),
                             names_from=Drzava, values_from = value)
# Preverimo novonastalo tabelo
head(bolezni_wider)
```

```
## # A tibble: 6 x 10
##   Leto Bolezen   Austria Brazil Chile Egypt France Lithuania Peru Sweden
##   <fct> <chr>     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 2015 HepatitisB  93    96    97    93    86    94    9    67
## 2 2015 HIV          0.1    0.1    0.1    0.1    0.1    0.1    0.1    0.1
## 3 2015 Polio        93    98    96    93    98    93    88    98
## 4 2014 HepatitisB  98    96    95    94    83    94    88    67
## 5 2014 HIV          0.1    0.1    0.1    0.1    0.1    0.1    0.1    0.1
## 6 2014 Polio        98    96    95    94    98    93    78    98
```

S funkcijo `separate()` združimo vrednosti dveh stolpcev. Enako lahko naredimo tudi s funkcijo `paste0()`. Nasprotje funkcije `unite()` je funkcija `separate()`.

```
# Združimo stolpca Drzava in Leto tabele bolezni z ločilom "_" v nov stolpec DrzavaLeto
head(unite(bolezni, DrzavaLeto, Drzava, Leto, sep="_"))
```

```
##   DrzavaLeto HepatitisB HIV Polio
## 1 Austria_2015      93 0.1  93
## 2 Austria_2014      98 0.1  98
## 3 Austria_2013      95 0.1  95
## 4 Austria_2012      92 0.1  92
## 5 Austria_2011      89 0.1  89
## 6 Austria_2010      86 0.1  86
```

Funkcija `expand()` prikaže vse kombinacije določenih spremenljivk v podatkih.

```
# S funkcijo expand() preverimo vse kombinacije spremenljivk Drzava in Leto
# v tabelo bolezni in rezultat shranimo v bolezni_kombo
bolezni_kombo <- expand(bolezni, Drzava, Leto)
# Preverimo število kombinacij
nrow(bolezni_kombo)
```

```
## [1] 128
```

```
# Prikažemo zadnjih par vrstic tabele
tail(bolezni_kombo)
```

```
## # A tibble: 6 x 2
##   Drzava Leto
##   <chr>   <fct>
```

```
## 1 Sweden 2010
## 2 Sweden 2011
## 3 Sweden 2012
## 4 Sweden 2013
## 5 Sweden 2014
## 6 Sweden 2015
```

S funkcijo chop() lahko podatke za določeno vrednost ali kombinacijo spremenljivk shranimo v en element - seznam.

```
# Uporabimo funkcijo chop(), da vse vrednosti za bolezni po letih shranimo
# v eno spremenljivko glede na Drzavo. Rezultat shranimo v tabelo bolezni_chop.
bolezni_chop <- chop(bolezni, c(Leto, HepatitisB, HIV, Polio))
# Preverimo rezultat
head(bolezni_chop, 3)
```

```
##      Drzava                                Leto
## 1 Austria 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
## 2 Brazil 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
## 3 Chile 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
##                                     HepatitisB
## 1 93, 98, 95, 92, 89, 86, 83, 83, 85, 83, 86, 83, 83, 81, 44, 33
## 2 96, 96, 96, 96, 98, 96, 99, 96, 99, 99, 98, 96, 97, 92, 91, 94
## 3 97, 95, 9, 9, 94, 92, 94, 95, 92, 95, NA, NA, NA, NA, NA, NA
##                                     HIV
## 1 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1
## 2 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1
## 3 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1
##                                     Polio
## 1 93, 98, 95, 92, 89, 86, 83, 83, 85, 83, 86, 83, 84, 82, 83, 71
## 2 98, 96, 96, 96, 98, 99, 99, 99, 99, 99, 99, 99, 99, 99, 99, 99
## 3 96, 95, 9, 9, 93, 92, 94, 95, 95, 94, 92, 94, 96, 97, 96, 91
```

```
# Preverimo dimenzije ustopne in novonastale tabele
dim(bolezni)
```

```
## [1] 128 5
dim(bolezni_chop)
```

```
## [1] 8 5
```

```
# Novonastala tabela je dimenzije
```

```
# Novonastala tabela vsebuje samo eno vrstico za vsako državo!
# Elementi tabele so seznamami vektorjev
# Preverimo stolpec leto tabele bolezni_chop
head(bolezni_chop$Leto, 3)
```

```
## <list_of<factor<49b95>>[3]>
## [[1]]
## [1] 2015 2014 2013 2012 2011 2010 2009 2008 2007 2006 2005 2004 2003 2002 2001
## [16] 2000
## 16 Levels: 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 ... 2015
##
## [[2]]
## [1] 2015 2014 2013 2012 2011 2010 2009 2008 2007 2006 2005 2004 2003 2002 2001
## [16] 2000
```

```
## 16 Levels: 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 ... 2015
##
## [[3]]
## [1] 2015 2014 2013 2012 2011 2010 2009 2008 2007 2006 2005 2004 2003 2002 2001
## [16] 2000
## 16 Levels: 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 ... 2015
```

```
# Preverimo stolpev Polio tabele bolezni_chop
head(bolezni_chop$Polio, 3)
```

```
## <list_of<integer>[3]>
## [[1]]
## [1] 93 98 95 92 89 86 83 83 85 83 86 83 84 82 83 71
##
## [[2]]
## [1] 98 96 96 96 98 99 99 99 99 99 99 99 99 99 99 99
##
## [[3]]
## [1] 96 95 9 9 93 92 94 95 95 94 92 94 96 97 96 91
```

```
# Elementi so sezname z vektorji za vsak nivo preostalih spremenljivk
```

Obratna funkcija je funkcija unchop().

```
# Uporabimo funkcijo unchop() za razširitev podatkov tabele bolezni_chop
# Chopped tabelo lahko razširimo za posamezno ali vse spremenljivke
head(unchop(bolezni_chop, c(Leto)), 3)
```

```
##      Drzava Leto                                     HepatitisB
## 1 Austria 2015 93, 98, 95, 92, 89, 86, 83, 83, 85, 83, 86, 83, 83, 81, 44, 33
## 2 Austria 2014 93, 98, 95, 92, 89, 86, 83, 83, 85, 83, 86, 83, 83, 81, 44, 33
## 3 Austria 2013 93, 98, 95, 92, 89, 86, 83, 83, 85, 83, 86, 83, 83, 81, 44, 33
##                                     HIV
## 1 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1
## 2 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1
## 3 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1
##                                     Polio
## 1 93, 98, 95, 92, 89, 86, 83, 83, 85, 83, 86, 83, 84, 82, 83, 71
## 2 93, 98, 95, 92, 89, 86, 83, 83, 85, 83, 86, 83, 84, 82, 83, 71
## 3 93, 98, 95, 92, 89, 86, 83, 83, 85, 83, 86, 83, 84, 82, 83, 71
```

```
# Razširitev glede na vse spremenljivke povrne tabelo osnovno stanje
head(unchop(bolezni_chop, c(Leto, HepatitisB, HIV, Polio)), 3)
```

```
##      Drzava Leto HepatitisB HIV Polio
## 1 Austria 2015          93 0.1    93
## 2 Austria 2014          98 0.1    98
## 3 Austria 2013          95 0.1    95
```

Funkcija pack() sesede stolpce v en stolpec-tabelo s skupnim imenom, ki ga določi uporabnik, in pod-imenom, ki je enako imenu vhodne spremenljivke.

```
# S funkcijo pack() sesedemo stolpce HepatitisB, HIV in Polio v stolpec-tabelo "Bolezni".
# Novonastalo tabelo poimenujemo bolezni_pack.
bolezni_pack <- pack(bolezni, Boleden = c(HepatitisB, HIV, Polio))
# Preverimo tabelo
head(bolezni_pack, 3)
```

```
##      Drzava Leto Bolezen.HepatitisB Bolezen.HIV Bolezen.Polio
## 1 Austria 2015                93          0.1          93
## 2 Austria 2014                98          0.1          98
## 3 Austria 2013                95          0.1          95

# Ustvarili smo stolpec "Bolezen", ki ima eno vrstico za vsako državo in leto,
# element vrstice pa je tabelo dimenzij 6x3
# Preverimo
```

6 Povzemanje podatkov

Pri prebranih podatkih nas ponavadi najprej zanimajo srednje vrednosti spremenljivk in razpršenost podatkov.

Vsoto lahko izračunamo s funkcijo `sum()`. Pri pregledovanju podatkov v tabelah lahko uporabimo tudi funkciji `rowSums()` in `colSums()`, ki vrnete vsoti vrstic ali stolpcev. Funkciji delujeta le na numeričnih podatkih, zato jih moramo naprej pretvoriti (`as.numeric()`) ali pa izbrati le numerične vrstice / stolpce.

```
# S funkcijo colSums() izračunajmo vsoto 4. do 30. stolpca tabele zivDoba
colSums(zivDoba[, 4:13])
```

```
##      ZivDoba Umrljivost      Alkohol HepatitisB      Ospice      ITM      Polio
##      9872.3      13163.0          NA          NA      63021.0      6440.5      11725.0
##           HIV          BDP      Solanje
##           15.9          NA      1880.3
```

```
# V stolpcih z manjkajočimi vrednostmi funkcija ne more izračunati vsote in vrne NA vrednost.
# Obnašanje funkcije lahko prilagodimo z nastavitvijo parametra na.rm (na remove) na TRUE
colSums(zivDoba[, 4:13], na.rm = TRUE)
```

```
##      ZivDoba Umrljivost      Alkohol HepatitisB      Ospice      ITM      Polio
##      9872.30      13163.00      942.88      8875.00      63021.00      6440.50      11725.00
##           HIV          BDP      Solanje
##           15.90 1822552.43      1880.30
```

Povprečje lahko izračunamo s funkcijo `mean()`. Enako kot pri vsoti lahko povprečja vrstic in stolpcev tabele izračunamo s funkcijama `rowMeans()` in `colMeans()`. Funkciji enako delujeta le na numeričnih podatkih.

```
# S funkcijo colMeans() izračunajmo vsoto 4. do 30. stolpca tabele zivDoba.
# Funkcija naj odstrani manjajoče vrednosti
colMeans(zivDoba[, 4:13], na.rm = TRUE)
```

```
##      ZivDoba      Umrljivost      Alkohol      HepatitisB      Ospice      ITM
## 7.712734e+01 1.028359e+02 7.857333e+00 8.217593e+01 4.923516e+02 5.031641e+01
##      Polio      HIV      BDP      Solanje
## 9.160156e+01 1.242188e-01 1.627279e+04 1.468984e+01
```

Kvantile lahko izračunamo s funkcijo `quantile()`. Privzeto R vrne vrednost za verjetnost 0, 0.25, 0.50, 0.75 in 1. S parametrom `probs` lahko določimo želene kvantile.

```
# S funkcijo quantile() izračunajmo kvantile z verjetnostjo
# 0.05 in 0.95 spremenljivke Umrljivost v tabeli zivDoba
quantile(zivDoba$Umrljivost, probs = c(0.05, 0.95))
```

```
##      5%      95%
##      9.70 178.65
```

Velikokrat želimo izračunati srednje vrednosti posameznih skupin oz. ravni neke spremenljivke. Za to lahko uporabimo funkcijo `aggregate()`, ki s poljubno funkcijo agregira podatke glede na podano formulo. Formulo

navedemo v obliki odvisnaSpremenljivka ~ neodvisnaSpremenljivka.

```
# S funkcijo aggregate izračunajmo povprečno umrljivost v posamezni državi
aggregate(zivDoba$Umrljivost ~ zivDoba$Drzava, FUN="mean")
```

```
##   zivDoba$Drzava zivDoba$Umrljivost
## 1      Austria          65.7500
## 2       Brazil          150.6875
## 3        Chile           63.6250
## 4        Egypt          170.6250
## 5        France           73.1250
## 6    Lithuania          117.2500
## 7         Peru           122.4375
## 8        Sweden           59.1875
```

```
# S funkcijo aggregate izračunajmo še standardni odklon umrljivost glede na status države in leto
povpUmr_stLeto <- aggregate(zivDoba$Umrljivost ~ zivDoba$Status + zivDoba$Leto, FUN = "sd")
head(povpUmr_stLeto)
```

```
##   zivDoba$Status zivDoba$Leto zivDoba$Umrljivost
## 1    Developed      2000          49.00000
## 2    Developing      2000          86.24500
## 3    Developed      2001          36.75595
## 4    Developing      2001          84.28108
## 5    Developed      2002          32.69557
## 6    Developing      2002          85.74789
```

Združevanje podatkov s funkcijami paketa dplyr

Funkcije paketa dplyr nam omogočajo bolj pregledno in uporabniku prijazno obdelavo podatkov. Prav tako kot tidyr omogoča cevovodno obdelavo podatkov

S funkcijo group_by() lahko ustvarimo skupine v obstoječih tabelah. Sama funkcija podatkov ne spremeni, jih pa interno pripiše specficiranim skupinam.

```
# Najprej naložimo paket dplyr
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
##
## The following object is masked from 'package:reshape':
##
##   rename
##
## The following objects are masked from 'package:stats':
##
##   filter, lag
##
## The following objects are masked from 'package:base':
##
##   intersect, setdiff, setequal, union
```

```
# S funkcijo group_by() združimo podatke tabele zivDoba glede na leto in status
# ter jih shranimo v zivDoba_group
zivDoba_group <- group_by(zivDoba, Leto, Status)
# Preverimo nastalo tabelo
head(zivDoba_group)
```



```
## # A tibble: 6 x 14
## # Groups:   Leto, Status [6]
##   Drzava Leto Status ZivDoba Umrlijivost Alkohol HepatitisB Ospice ITM Polio
##   <chr> <fct> <chr>    <dbl>      <int>    <dbl>      <int> <int> <dbl> <int>
## 1 Austr~ 2015 Devel~    81.5        65      NA          93   309  57.6   93
## 2 Austr~ 2014 Devel~    81.4        66    12.3        98   117  57.1   98
## 3 Austr~ 2013 Devel~    81.1        68    11.8        95     0  56.6   95
## 4 Austr~ 2012 Devel~     88         7    12.3        92    36  56.1   92
## 5 Austr~ 2011 Devel~     88        73    12.0        89    68  55.7   89
## 6 Austr~ 2010 Devel~     84        75    12.1        86    52  55.2   86
## # ... with 4 more variables: HIV <dbl>, BDP <dbl>, Solanje <dbl>, Celina <chr>
```

S funkcijo `summarize()` lahko nato povzamemo podatke. Če so v podatkih že določene skupine, bo funkcija povzela po le-teh.

```
# S funkcijo summarize() izračunajmo povprečje in standardni odklon spremenljivk
# Umrlijivost in Alkohol v zivDoba_group. Shranimo podatke v zivDoba_sum
zivDoba_sum <- summarize(zivDoba_group,
  povpUmr = mean(Umrlijivost),
  sdUmr = sd(Umrlijivost),
  povpAlk = mean(Alkohol),
  sdAlk = sd(Alkohol))
```

```
## `summarise()` regrouping output by 'Leto' (override with `groups` argument)
head(zivDoba_sum)
```

```
## # A tibble: 6 x 6
## # Groups:   Leto [3]
##   Leto Status    povpUmr sdUmr povpAlk sdAlk
##   <fct> <chr>      <dbl> <dbl> <dbl> <dbl>
## 1 2000 Developed     57  49     9.76  3.50
## 2 2000 Developing  107. 86.2    6.37  4.89
## 3 2001 Developed     62 36.8    9.73  2.93
## 4 2001 Developing  105. 84.3    6.30  5.02
## 5 2002 Developed     34 32.7   10.1  2.90
## 6 2002 Developing  104. 85.7    6.20  4.99
```

Funkcije paketa `dplyr` lahko med sabo povežemo v “cevodno” analizo z operatorjem `%>%`. Izhodni podatki prejšnje funkcije tako služijo kot vhodni podatki za naslednjo.

```
# V enem koraku v zivDoba ustvarimo skupine glede na status in leto
# ter izračunamo povprečje spremenljivke HepatitisB
zivDoba_hep <- zivDoba %>% group_by(Status, Leto) %>%
  summarize(povpHep = mean(HepatitisB, na.rm=T))
```

```
## `summarise()` regrouping output by 'Status' (override with `groups` argument)
head(zivDoba_hep)
```

```
## # A tibble: 6 x 3
## # Groups:   Status [1]
##   Status Leto povpHep
##   <chr>   <fct>    <dbl>
## 1 Developed 2000    65.5
## 2 Developed 2001    69.5
## 3 Developed 2002    87.5
## 4 Developed 2003    89
```

```
## 5 Developed 2004      88.5
## 6 Developed 2005      90.5
```

V cevovod lahko združimo funkcije iz različnih paketov.

```
# V enem koraku tabelo bolezni preuredimo s funkcijo pivot_longer(),
# da bo imela vrednosti za vse bolezni v enem stolpcu (in imena bolezni v drugem),
# združimo po državi in bolezni in izračunajmo povprečje
# Ponovno ustvarimo tabelo bolezni in naložimo tidyr, če jih nimamo več v delovnem okolju
bolezni_povp <- bolezni %>% pivot_longer(cols = c(HepatitisB, HIV, Polio), names_to = "Bolezen") %>%
  group_by(Drzava, Bolezen) %>%
  summarize(povpVrednost = mean(value))
```

```
## `summarise()` regrouping output by 'Drzava' (override with `.groups` argument)
```

```
head(bolezni_povp)
```

```
## # A tibble: 6 x 3
## # Groups:   Drzava [2]
##   Drzava Bolezen    povpVrednost
##   <chr>   <chr>         <dbl>
## 1 Austria HepatitisB      81.1
## 2 Austria HIV              0.1
## 3 Austria Polio           86
## 4 Brazil  HepatitisB     96.2
## 5 Brazil  HIV              0.1
## 6 Brazil  Polio          98.3
```