

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
```

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
```

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
```

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  
# Izpišimo "stevilo"  
# Preverimo tip spremenljivke
```

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"
```

```
# Preverimo tip spremenljivke
```

Eksplisitno lahko tip "integer" ustvarimo tudi z dodatkom črke "L" na koncu celega števila.

```
# Direktno ustvarimo "integer" spremenljivko "stevilo" z vrednostjo 10
```

Realna števila

```
# Ustvarimo spremenljivko "realnoStevilo" z vrednostjo 3.52
```

```
# Izpišimo "realnoStevilo"
```

```
# Preverimo tip spremenljivke
```

Znaki

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

```
# Ustvarimo spremenljivko "beseda" z vrednostjo "jagoda"
```

```
# Izpišimo spremenljivko "beseda"
```

```
# Preverimo tip spremenljivke
```

Logični vektor

Logične vrednosti v R-u so TRUE in FALSE.

```
# Ustvarimo spremenljivko "logicnaVrednost" z vrednostjo TRUE
```

```
# Izpišimo "logicnaVrednost"
```

```
# Preverimo tip spremenljivke
```

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

```
# Seštejmo vrednosti TRUE in TRUE
```

Manjkajoča vrednost

Manjkajoče vrednosti v R-u so predstavljene kot NA.

```
# Ustvarimo spremenljivko "manjkaVrednost" z manjkajočo vrednostjo
```

```
# Izpišimo "manjkaVrednost"
```

```
# Preverimo tip spremenljivke
```

Prazen element

Prazen element R predstavi kot NULL.

```
# Ustvarimo prazen element "prazenElement"

# Izpišimo "prazenElement"

# Preverimo tip spremenljivke
```

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

# Izpišimo vektor
```

Vektor številskega zaporedja lahko ustvarimo tudi z navedbo razpona števil.

```
# Ustvarimo številski vektor "meseci" s številkami mesecev 1 do 12
```

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

```
# Ustvarimo vektor znakov "letniCasi" z letnimi časi

# Izpišimo vektor
```

Elemente vektorjev izberemo z indeksom v oglatih oklepajih.

```
# Izberimo drugi element vektorja "letniCasi"
```

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")

# Izpišimo "letniCasi_f"
```

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

```
# Poskušajmo dodati novo raven 'zjutraj' v faktor kot deveti element

# 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
```

```
# Izpišimo seznam "leto"
```

```
# Preverimo dolžino seznama "leto"
```

Elemente seznama izberemo z indeksom v dvojnih oglatih oklepajih.

```
# Izberemo prvi element seznama "leto"
```

```
# Ker je prvi element vektor, izberimo še peti element znotraj tega
```

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 je 18, poleti 24, jeseni 13 in pozimi 5
```

```
# Izpišimo tabelo
```

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

```
# Izpišimo prvih par vrstic tabele "povpTemp"
```

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"
```

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

```
# Preverimo število vrstic in stolpcev tabele "povpTemp"
```

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

```
# Preverimo imena vrstic in stolpcev tabele "povpTemp"
```

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

```
# Izberimo prvo vrednost v prvem stolpcu tabele "povpTemp"
```

```
# Izberimo prvo vrstico tabele "povpTemp"
```

```
# Izberimo prvi stolpec tabele "povpTemp"
```

Stolpce tabel lahko izberemo tudi po imenu.

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

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
```

```
# Izpišimo matriko
```

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

```
# Izberimo element v drugi vrstici tretjega stolpca matrike vrednosti
```

Polje

Polje ustvarimo s funkcijo `array()`.

```
# Ustvarimo polje povpTempLeto s povprečnimi letnimi temperaturami v letnih časih  
# v dveh različnih letih.  
# Povprečna temperatura 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)
```

```
# Izpišimo polje "povpTempLeto"
```

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

```
# Preverimo dimenzije polja "povpTempLeto"
```

```
# Preverimo strukturo polja "povpTempLeto"
```

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
```

```
# Ustvarimo spremenljivko "vsota" kot vsoto a in b
```

```
# Ustvarimo spremenljivko "razlike" kot vsoto a in b
```

```
# Ustvarimo spremenljivko "zmnožek" kot vsoto a in b
```

```
# Ustvarimo spremenljivko "kvocient" kot vsoto a in b
```

Ponovimo vajo z vektorji.

```
# Ustvarimo vektorja "a" z vrednostmi c(1, 2, 3) in "b" z vrednostmi c(3.12, 5.44, 10)
```

```
# Ustvarimo spremenljivko "vsota" kot vsoto a in b
```

```
# Ustvarimo spremenljivko "razlike" kot vsoto a in b
```

```
# Ustvarimo spremenljivko "zmnožek" kot vsoto a in b
```

```
# Ustvarimo spremenljivko "kvocient" kot vsoto a in b
```

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

```
# Preverimo ostanek pri deljenju 10 / 3

# Preverimo celoštevilski kvocient deljenja 10 / 3

# Ustvarimo matriko m1 z vrednostmi 1:9 in tremi vrsticami
# ter matriko m2 z vrednostmi 9:18 in tremi vrsticami
```

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

# Preverimo, ali je 5 večje od 3

# Preverimo, ali je 5 enako 3

# Preverimo, ali 5 ni enako 3

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

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.

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

“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
```

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

```
# Združimo besedi Janez in Novak brez separatorja
```

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

# Seštejmo elemente vektorja s funkcijo sum()

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

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

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

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)
```

```
# Seštejmo vektor vrednosti
```

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()
```

```
# Ponovno sštejmo vektor vrednosti
```

```
# Ker želimo odstraniti manjkajočih vrednosti, nastavimo parameter na.rm na FALSE
```

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.

S funkcijo `getwd()` lahko preverimo nastavljen delovni imenik.

```
# Preverimo delovni imenik
```

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
```

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()
```

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

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

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).
```

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

```
# Preverimo število vrstic in število stolpcev
```

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

```
# Preimenujmo stolpec "Country" v "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
```

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

```
# Izberimo tretji stolpec tabele zivDoba z imenom (Status)
```

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()
```

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
```

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
```

```
# Preverimo varianco življenjske dobe
```

```
# Preverimo minimalno umrljivost v podatkih
```

```
# Preverimo maksimalno umrljivost v podatkih
```

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
```

```
# Izberimo vse vrstice, v katerih je vrednost stolpca "Drzava" enaka "France"
```

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
```



```
# Preverimo imena stolpcev nove tabele zivDoba_nov s funkcijo colnames()
```

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
```

```
# Preimenuj drugi stolpec iz 'Ospice' v 'ospiceFR'
```

```
# Ustvarimo vektor 'ospiceBR', v katerega zberemo podatke o letu in pojavnosti ošpic v Braziliji
```

```
# Preimenuj drugi stolpcev iz 'Ospice' v 'ospiceBR'
```

```
# Ustvarimo vektor 'ospiceAU' in 'ospiceEG'
```

```
# Vanju zberemo podatke o pojavnosti ošpic v Avstriji in Egiptu
```

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"
```

```
# Preverimo tabelo
```

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"
#
```

```
# "Brazil", "Chile" in "Peru" ležijo v Južni Ameriki
```

```
# Tem vrsticam spremenimo vrednost v stolpcu "Celina" na JuznaAmerika
```

```
# Preverimo zastopanost vrednosti v stolpcu "Celina"
```

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
```

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'

# Preverimo novonastalo tabelo
```

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'
```

```
# Preverimo novonastalo tabelo
```

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

```
# Najprej naložimo paket reshape
```

```
# 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
```

```
# Preverimo novonastalo tabelo
```

```
# 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
```

```
# Preverimo novonastalo tabelo s funkcijo head()
```

```
#Preverimo vrednosti v stolpcu 'variable' tabele bolezni_melt
```

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. Tabela preuredi tako, da ima več vrstic in manj stolpcev oz. podaljša tabelo.

```
# Najprej naložimo paket tidyr
```

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

```
# Preverimo novonastalo tabelo
```

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
```

```
# Preverimo novonastalo tabelo
```

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
```

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

```
# S funkcijo expand() preverimo vse kombinacije spremenljivk Drzava in Leto v tabeli bolezni
# Rezultat shranimo v bolezni_kombo

# Preverimo število kombinacij

# Prikažemo zadnjih par vrstic tabele
```

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.

# Preverimo rezultat

# Preverimo dimenzije vstopne in novonastale tabele
```

```
# Novonastala tabela vsebuje samo eno vrstico za vsako državo!
# Elementi tabele so sezname vektorjev
# Preverimo stolpec leto tabele bolezni_chop
```

```
# Preverimo stolpec Polio tabele bolezni_chop

# 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

# Razširitev glede na vse spremenljivke povrne tabelo osnovno stanje
```

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.

# Preverimo tabelo

# Ustvarili smo stolpec "Bolezni", ki ima eno vrstico za vsako državo in leto,
# element vrstice pa je tabela 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 13. stolpca tabele zivDoba

# vidimo, da 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
```

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 odstani manjajoče vrednosti
```

Kvatile 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 zibDoba
```

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
```

```
# S funkcijo aggregate izračunajmo še standardni odklon umrljivost glede na status države in leto
```

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
```

```
# S funkcijo group_by() združimo podatke tabele zivDoba glede na  
# leto in status ter jih shranimo v zivDoba_group
```

```
# Preverimo nastalo tabelo
```

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 Umrljivost in Alkohol v zivDoba_group.  
# Shranimo podatke v zivDoba_sum
```

Funkcije paketa `dplyr` lahko med sabo povežemo v “cevovodno” 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
```

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  
  
# Preverimo novonastalo tabelo
```