Opisna statistika v Rju

Luka Komidar

Ljubljana, 2017

Kazalo vsebine

Osnovne informacije (R in RStudio)

R je hkrati programski jezik in statistični paket.

Namestitev programa R

Domača stran projekta R (mnogo koristnih virov in povezav):
https://www.r-project.org/

Najprej je potrebno namestiti osnovni R program (veliko spletnih strani/zrcal):
https://cloud.r-project.org/

R je na voljo za operacijske sisteme Windows, Mac OS X in nekatere distribucije Linuxa (Debian, Red Hat, openSUSE, Ubuntu). V primeru operacijskega sistema Windows izberite "base", prenesite namestitveno datoteko na računalnik in namestite R. Trenutno (september 2017) zadnja stabilna različica je 3.4.1.

Namestitev programa RStudio

Na voljo je več različic razvojnih okolij oz. programske opreme, ki presegajo funkcionalnost programa R. V tem priročniku bomo uporabljali program **RStudio**. RStudio je na voljo za iste operacijske sisteme kot R.

Domača stran RStudia:
https://www.rstudio.com/

Namestitvene datoteke so tukaj (izberite RStudio Desktop Free Licence):
https://www.rstudio.com/products/rstudio/download3/

Namestitev dodatnih paketov

R je izrazito modularen program, kar pomeni, da lahko funkcionalnost, ki jo nudi osnovni program R, nadgradimo z namestitvijo dodatnih **paketov**. Za uporabo funkcij, uporabljenih v tem priročniku, so potrebni sledeči paketi:

- car
- DescTools
- ggplot2
- gmodels
- lmtest
- mosaic
- psych
- vcd
- xlsx (za pravilno delovanje potrebuje nameščeno Javo; 32- ali 64-bitno, odvisno od različice Rja)

Namestitev in aktivacija paketov

```
install.packages("ime_paketa")   # to storimo samo enkrat
```

Dodatne pakete lahko aktiviramo vsakič znova na začetku vsake seje:

```
library(ime_paketa)
```

Lahko pa v datoteko Rprofile.site (namestitveni direktorij R –> etc) dodamo funkcijo .First. Vse, kar bo vsebovala ta funkcija, se bo izvedlo ob začetku nove seje, npr.:

```
.First <- function() {
   library(psych)
   library(gmodels)
   …
}
```

Za spreminjanje datoteke Rprofile.site potrebujemo administratorske pravice, zato je potrebno urejevalnik besedila (npr. Notepad++ ipd.), v katerem bomo urejali to datoteko, zagnati z administratorskimi pravicami.

Lahko ustvarimo tudi skripto (v urejevalniku kode oz. v okno levo zgoraj; glej naslednje poglavje o RStudiu), ki jo shranimo in poženemo ob začetku vsake seje. V skripto nanizamo želene ukaze/funkcije, npr.

```
library(vcd)
library(xlsx)
…
```

Če želimo preveriti, kateri nameščeni paketi so aktivirani, uporabimo funkcijo `search()`. Poleg tega funkcija vrne tudi seznam "pripetih" objektov (glej poglavje o podatkovnih okvirjih).

Za izpis funkcij, ki jih vsebuje paket, uporabimo funkcijo:

```
ls("package:ime_paketa")
```

RStudio – pregled uporabniškega vmesnika

Štiri okna:

- levo zgoraj:
  - skriptno okno (zapis niza ukazov, ki jih lahko shranimo kot skripto)
  - pregled podatkov; funkcija `View(ime_objekta)`
- levo spodaj:
  - konzola (vpis ukazov in njihova takojšnja izvedba, izpis besedilnih rezultatov)
- desno zgoraj:
  - okolje (seznam in lastnosti objektov)
  - zgodovina ukazov
- desno spodaj:
  - raziskovalec direktorijev/datotek
  - izpis slik
  - seznam nameščenih paketov
  - izpis pomoči

Pomembne tehnične značilnosti Rja

- R razlikuje velike in male črke (case-sensitive); pomembno za vse, torej imena objektov, funkcij, datotek, direktorijev …
- Decimalno ločilo je pika (in ne vejica kot predpisuje slovenski pravopis).
- Ukaze lahko vpisujemo v skriptno okno in jih zaganjamo s `Ctrl+Enter` ali s klikom na `Run`. Če ima skripta več ukazov/funkcij, lahko vse izvedemo s klikom na `Source`.
- V konzoli se lahko s smernima tipkama (navzgor/navzdol) premikamo med že izvedenimi ukazi oz. po zgodovini ukazov.
- Samodokončanje imen funkcij in objektov; po začetku pisanja imena funkcije/objekta nam bo R v kontekstnem menu-ju ponudil možnosti, ki se začnejo na izpisan niz znakov – izbrano funkcijo/ime lahko dokončamo s pritiskom tipke `Tab`.
- Komentarje (besedilo, ki ga bo R tako v skripti kot v konzoli ignoriral) označimo z znakom #. Omogoča le enovrstične komentarje, torej moramo v primeru večvrstičnega komentarja na začetek vsake vrstice zapisati znak #.
- V imenih objektov se izogibamo uporabi pike, ker se uporablja za pripisovanje funkcije objektu in za skrite spremenljivke (se začnejo s piko).

Nastavitev delovnega direktorija

Posamezne projekte (skripte, podatkovne datoteke …) je smotrno imeti v določenem direktoriju, ki ga pri ukvarjanju s trenutnim projektom nastavimo kot delovni direktorij. V zvezi z delovnim direktorijem sta na voljo dve funkciji:

```
getwd()    # izpiše trenutno izbran delovni direktorij

setwd("pot do direktorija")
```

Pri vpisu poti do izbranega direktorija v operacijskem sistemu Windows za ločilo med direktoriji ne smemo uporabiti znaka "\" (backslash; poševnica nazaj), ampak znak "/" (forward slash; poševnica), npr.:

```
setwd("C:/Users/Janez Novak/R/moji projekti")
```

Delovni direktorij lahko nastavimo tudi prek uporabniškega vmesnika RStudia, in sicer prek menu-ja:

Session -> Set Working Directory -> Choose Directory …

Objekti v Rju

Ukazi v Rju običajno sledijo obliki:

```
objekt <- funkcija(arg1, arg2, …)
```

kar pomeni, da je objekt ustvarjen na podlagi določene funkcije. Z znakom `<-` objektu priredimo vrednost(i), ki jih vrne funkcija (objektu lahko neposredno priredimo določeno vrednost). Ukaz lahko vsebuje tudi samo ime funkcije in morebitne argumente, prav tako lahko ukaz vsebuje samo ime (že ustvarjenega) objekta, pri čemer bo izvedba takega ukaza vodila do izpisa vsebine objekta.

Seznam ustvarjenih objektov je prikazan v oknu desno zgoraj (zavihek Environment/Okolje), lahko jih izpišemo tudi s funkcijo `ls()`. Če želimo nek objekt povsem izbrisati, uporabimo funkcijo `rm(ime_objekta)`. Če želimo naenkrat izbrisati vse objekte v okolju, izvedemo ukaz `rm(list = ls())`.

Pomoč pri uporabi funkcij

Če želimo izpisati pomoč za določeno funkcijo, lahko uporabimo ukaza:

```
?ime_funkcije (ali) help(ime_funkcije)
```

Pomoč se bo izpisala v oknu desno spodaj.

Operatorji

Aritmetični operatorji (osnovne računske operacije)

```
+          # seštevanje
-          # odštevanje
*          # množenje
/          # deljenje
^ ali **   # potenciranje
%%         # modulo (vrne ostanek po deljenju)
```

Prireditveni operatorji (prirejanje vrednosti objektu)

```
<-          # priporočen operator za prirejanje!
<<-
=
```

Primerjalni operatorji (primerjamo dva operanda; vrne logično vrednost: TRUE/FALSE)

```
==          # je enako; vrne TRUE, če sta operanda enaka
!=          # ni enako; vrne TRUE, če sta operanda različna
>           # večje; vrne TRUE, če je levi operand večji od desnega
<           # manjše; vrne TRUE, če je levi operand manjši od desnega
>=          # večje ali enako
<=          # manjše ali enako
```

Logični operatorji

```
!x          # x ni resničen (NOT TRUE); negacija (TRUE->FALSE in obratno)
x | y       # ali; če je katerikoli operand resničen, vrne TRUE
xor(x,y)    # ekskluzivni ali; če je en od operandov resničen, drugi pa
            # ne, vrne TRUE (če sta oba operanda TRUE ali FALSE, xor
            # vrne FALSE)
x & y       # in; če sta oba operanda resnična, vrne TRUE
isTRUE(x)   # funkcija, ki preveri, če je x resničen; če je resničen,
            # vrne TRUE
```

Podatkovni tipi

R pozna sledeče podatkovne tipe:

- številski
  - *numeric* (realna števila; cela števila in števila s plavajočo vejico); npr. 3; 24,73; –6,8 …
  - *integer* (cela števila; pripišemo L!); npr. 5L, 0L, 42L … *ALI* ustvarimo vrednost/vektor znotraj funkcije `as.integer()`
  - *complex* (kompleksna števila); npr. 4 + 2i
- faktorji (*factor*); za kategorialne spremenljivke (nominalne in ordinalne)
- znakovni/besedilni (*character*); v drugih jezikih/programih je ta tip *string*
- logični (TRUE/FALSE)

Posebne vrednosti:

- NA (*not available*; ni na voljo); manjkajoče vrednosti
- NaN (*not a number*; ni število); nemogoče numerične vrednosti (npr. rezultat deljenja z 0)
- NULL (prazna vrednost)

Podatkovni tip objekta izpišemo s funkcijo

```
class(ime_objekta)    # ali s funkcijo …
str(ime_objekta)      # … str (structure); ta izpiše tudi vrednosti objekta
```

Podatkovne strukture

Skalar

Skalar je spremenljivka z eno vrednostjo oz. vektor z dolžino ena.

- Lahko vsebuje vrednosti kateregakoli podatkovnega tipa.
- Npr: `x <- 24; x <- TRUE; x <- "jabolko"`

Vektor

Vektor je spremenljivka z več vrednostmi.

- Lahko vsebuje vrednosti zgolj enega podatkovnega tipa.
- Vektor lahko ustvarimo:
  - s funkcijo `c()`, npr.: `v <- c(3,1,4); v <- c("jabolko","hruška")`
  - s funkcijo `scan()`; v tem primeru zaporedno vpisujemo vrednosti (končamo z `2*Enter`); s `scan()` lahko določimo tudi podatkovni tip vrednosti, ki jih vpisujemo, in sicer dodamo argument `what = podatkovni_tip()`, npr. `v <- scan(what = character())`
  - vektor z zaporedjem ustvarimo s funkcijo `c()`, pri čemer uporabimo dvopičje med začetno in končno vrednostjo, npr. `v <- c(1:10)`; dobimo vektor z vrednostmi 1 do 10; za isti rezultat lahko zapišemo tudi zgolj `v <- 1:10`

- Do posameznih vrednosti vektorja dostopamo s sklicem na njihov **indeks** (v oglatih oklepajih). Če vrednostim ne pripišemo oznak, so indeksi vrednosti zaporedne številke od 1 do števila vrednosti v vektorju; npr.:

```
v[2]              # sklicujemo se na indeks 2, R bo vrnil drugo
                  # vrednost v vektorju
v[3:7]            # lahko izberemo/prikažemo tudi določeno podmnožico
                  # vrednosti
v[c(1:3,10:15)]   # za izbor podmnožice lahko uporabimo ločene intervale
                  # vrednosti (od 1. do 3. in 10. do 15. vrednosti)
```

- Izbor **podmnožice** lahko izvedemo tudi tako, da R-u sporočimo, katerih vrednosti naj ne prikaže. To storimo prek negativnih indeksov:

```
v[-3]                  # izbrane bodo vse vrednosti razen tretje
v[c(-2,-5)]            # izločimo drugo in peto vrednost
v[c(-2:-4,-8:-10)]     # izločimo vrednosti, ki so na 2. do 4. mestu
                       # in 8. do 10. mestu
```

- Izbor podmnožice lahko izvedemo tudi s primerjalnimi/logičnimi operatorju:

```
v[v>6]           # izbor vrednosti, večjih od 6
v[v>3 & v<9]     # izbor vrednosti, večjih od 3 in manjših od 9
```

- S sklici na indekse lahko (podobno kot pri izborih) spreminjamo/dodajamo izbrane vrednosti vektorja

```
v[3] <- 12       # 3. vrednost smo zamenjali z 12
v[11] <- 42      # vektorju z 10 elementi smo dodali 11. vrednost
                 # Če bi vektorju z 10 elementi dodali vrednost
                 # na položaj 15, bi R izpuščene položaje
                 # (11:14) zapolnil z NA vrednostmi (prazna polja).
```

- Dolžino vektorja (oz. število elementov) dobimo s funkcijo `length(vektor)`. Če želimo iz štetja izločiti manjkajoče vrednosti, uporabimo `length(vektor[!is.na(vektor)])`.

Matrika

Matrika je pravokotna tabela z *m* vrsticami *n* stolpci.

- Vsi stolpci/vrstice morajo vsebovati vrednosti istega tipa in vsi stolpci/vrstice morajo imeti enako dolžino.
- Matriko lahko ustvarimo:
  - s funkcijo (vrednosti argumentov so njihove privzete vrednosti, če jih izpustimo)

    ```
    matrix(vektor/podatki = NA, nrow = 1, ncol = 1, byrow = FALSE,
    dimnames = NULL)
    ```

    nrow = število vrstic, ncol = število stolpcev, byrow (TRUE/FALSE) = ali naj zapolnjuje vrednosti po vrsticah ali po stolpcih, dimnames = seznam (glej naslednjo podatkovno strukturo), ki vsebuje imena vrstic in stolpcev

    Npr. (dobimo 3 * 3 matriko, ki ima v prvi vrstici vrednosti 1, 2, 3, v drugi 4, 5, 6 in v tretji 7, 8, 9)

    ```
    m <- matrix(c(1,2,3,4,5,6,7,8,9), nrow = 3, ncol = 3,
         byrow = TRUE, dimnames = list(c("r1","r2","r3"),
         c("c1","c2","c3")))
    ```

  - z združitvijo vektorjev (ki vsebujejo vrednosti enakega tipa!):

    ```
    v1 <- c(1,2,3)
    v2 <- c(7,8,9)

    m <- cbind(v1, v2)    # združimo vektorja v matriko tako, da z
                          # vrednostmi podanih vektorjev zapolnimo
                          # stolpce; ustvarili smo matriko m s tremi
                          # vrsticami in dvema stolpcema
    m <- rbind(v1, v2)    # združimo vektorja v matriko tako, da z
                          # vrednostmi podanih vektorjev zapolnimo
                          # vrstice; ustvarili smo matriko m z dvema
                          # vrsticama in tremi stolpci
    ```

- Do vrednosti v matriki dostopamo podobno kot pri vektorjih (prek indeksov), vendar pri matrikah podamo zaporedno številko (ali ime) vrstice in stolpca

  ```
  m[2,4]          # vrednost v 2. vrstici in 4. stolpcu
  m[5,]           # celotna 5. vrstica
  m[,3]           # celoten 3. stolpec
  m["r2","c1"]    # vrednost v vrstici r2 in stolpcu c1
  ```

- Na podoben način kot pri dostopanju do vrednosti v matriki (ali vektorju) lahko določene vrednosti tudi zamenjamo (prek indeksov).
- Dimenzije matrike (ali seznama, podatkovnega okvirja) izpišemo s funkcijo `dim(ime_objekta)`. Funkcija vrne število vrstic, število stolpcev.

Seznam

Seznam (angl. *list*) običajno ustvarimo/uporabljamo za združevanje objektov, katerih elementi/vrednosti lahko predstavljajo različne podatkovne tipe, poleg tega lahko imajo objekti tudi različno število elementov/vrednosti.

- Sezname ustvarimo s funkcijo `list(objekt1, objekt2, …)`

```
v1 <- c(2, 5, 8, 11)            # vektor z numeričnimi vrednostmi
v2 <- c("a", "b")               # vektor z znakovnimi vrednostmi
v3 <- c(TRUE, FALSE, FALSE)     # vektor z logičnimi vrednostmi
s <- 23                         # skalar

l <- list(v1, v2, v3, s)        # vsebuje kopije v1, v2, v3 in s

# elemente seznama lahko tudi poimenujemo
l <- list(e1 = v1, e2 = v2, e3 = v3, e4 = s)
```

- Do vrednosti dostopamo z uporabo dvojnih oglatih oklepajev

```
ime_seznama[[indeks/ime_objekta]]
l[[2]]          # izpiše v2
l[[4]]          # izpiše s
l[[3]][2]       # izpiše 2. element v3 (= FALSE)
l[["e3"]]       # izpiše objekt z imenom e3 (= v3)
l[["e2"]][1]    # izpiše 1. element objekta e2 (= v2[1] = "a")
```

Polje

Polje (angl. *array*) je podobna struktura kot matrika, le da je polje lahko večdimenzionalno. Dvodimenzionalno polje je matrika.

Faktor

- Za opredelitev spremenljivk kot nominalnih ali ordinalnih (urejenih kategorialnih).
- Če želimo numerično spremenljivko opredeliti kot faktor, to storimo s funkcijo `factor()`:

```
factor(x, levels = c(1,2,…), labels = c("oznaka1","oznaka2" …),
       exclude = NA, ordered = is.ordered, nmax = NA)
```

Argumenti:

```
x          # numerična spremenljivka
levels     # vektor z ravnmi faktorja (npr. 1 za ženske, 2 za moške)
labels     # besedilne oznake za ravni faktorja
exclude    # katere vrednosti naj bodo ignorirane pri oblikovanju
           # niza ravni (levels); privzeto so to manjkajoče
           # vrednosti (NA)
ordered    # ali je spremenljivka nominalna (FALSE) ali
           # ordinalna/urejena kategorialna (TRUE)
nmax       # zgornja meja za število ravni (privzeto NA)

# ustvarjanje nominalne spremenljivke/faktorja

n <- c(1,2,1,1,2)
n_fact <- factor(n, levels = c(1, 2), labels = c("ženski","moški"),
        ordered = FALSE)
```

```
# ustvarjanje ordinalne spremenljivke/faktorja

o <- c(1,2,3,2,2,1,3)
o_fact <- factor(o, levels = c(1, 2, 3), labels = c("OŠ","SŠ","UNI"),
       ordered = TRUE)
```

- Numerično spremenljivko lahko v faktor spremenimo s funkcijo `as.factor()`.
- Če imamo v podatkovnem okvirju z imenom "po" (glej naslednje poglavje) numerično spremenljivko num_var, ki je opredeljena kot faktor (in brez besedilnih oznak vrednosti), lahko to spremenljivko spremenimo v numerično s funkcijo `as.numeric()`. Brez uporabe funkcije `as.character` bo R obstoječe vrednosti v num_var zamenjal z indeksi od vrednosti spremenljivke, ne pa z vrednostmi spremenljivke!

```
po$num_var <- as.numeric(as.character(po$num_var))
```

Podatkovni okvir

Podatkovni okvir (angl. *data frame*) je podobna struktura kot matrika, le da lahko stolpci vsebujejo različne podatkovne tipe (znotraj stolpca še vedno enak tip!). Podatkovni okvir (v nadaljevanju "p.o.") je torej seznam vektorjev z enako dolžino.

- Podatkovni okvir je najpogosteje uporabljana struktura za statistične analize.
  - V splošnem bi lahko rekli, da podatkovni okvir predstavlja podatkovno bazo, ki ima običajno v vrsticah predmete merjenja (običajno udeleženci), v stolpcih pa spremenljivke.
- Podatkovni okvir ustvarimo z združevanjem enako dolgih vektorjev:

```
# ustvarimo dva vektorja (enega numeričnega, enega besedilnega)

n <- c(6,9,8)
s <- c("ž","ž","m")

# ustvarimo p.o. z dvema stolpcema in tremi vrsticami

podatki <- data.frame(n, s)

# dodamo smiselna (kratka) imena za spremenljivke

names(podatki) <- c("ocene","spol")

# … ali s funkcijo scan(); po izvedbi ukaza jih zapisujemo v ustreznem
# vrstnem redu

names(podatki) <- scan(what = character())
```

- Do posameznih vrednosti in celotnih spremenljivk lahko dostopamo na različne načine:

```
podatki[2]            # izpis druge spremenljivke (spol)
podatki["ocene"]      # izpis prve spremenljivke (ocene)
podatki$spol          # izpis spremenljivke spol
podatki$ocene[2]      # izpis druge vrednosti v spremenljivki ocene (9)
podatki[2,1]          # enako kot predhodni primer (9); izbor
                      # vrednosti v 2. vrstici in 1. stolpcu
```

- Imena spremenljivk v p.o. lahko po želji tudi dodamo v iskalno pot Rja, kar pomeni, da lahko spremenljivke neposredno naslavljamo z imenom in ne kot pripono imenu p.o., torej lahko v ukazu/funkciji zapišemo npr. `spol`, ne pa `po$spol`. To naredimo s funkcijo `attach()`:

```
attach(ime_po)
```

Pripenjanje imen spremenljivk v iskalno pot je priročno, a ni priporočljivo, saj predvideno vedenje po izvedbi ukaza `attach()` ni povsem konsistentno prek vseh možnih situacij (npr. pri ustvarjanju/dodajanju novih spremenljivk!), prav tako velikokrat delamo z več podatkovnimi okvirji, kar lahko vodi do zmede in napak. Podatkovni okvir lahko tudi odstranimo iz iskalne poti Rja, in sicer z `detach(ime_po)`.

- Vrednosti v p.o. lahko tako kot v vektorjih in matrikah spreminjamo prek sklicev na ustrezne indekse.
- Urejanje podatkov (dodajanje/spreminjanje vrednosti, celotnih vrstic ali spremenljivk) lahko izvedemo tudi z ukazom `edit()`.

```
podatki <- edit(podatki)    # ukaz bo odprl okno s tabelo, ki jo lahko
                            # urejamo; po zaprtju se bo p.o. posodobil
```

- Spremenljivk iz p.o. ne moremo izbrisati s funkcijo `rm()`, temveč jim pripišemo vrednost `NULL` ali pa s pripisom negativnega indeksa izbranemu stolpcu ali vektorju stolpcev; če p.o. podamo samo en indeks (ali vektor indeksov), se nanaša na stolpec.

```
po$var <- NULL
po <- po[-3]                # brisanje spremenljivke v tretjem stolpcu
po <- po[-c(2,5,8:11)]      # brisanje 2., 5. in 8. do 11. spremenljivke
```

- Vrstice v p.o. brišemo s pripisom negativnega indeksa izbrani vrstici (ali vektorju vrstic).

```
po <- po[-3,]               # brisanje 3. vrstice
po <- po[-c(4,7,9:14),]     # brisanje 4., 7. in 9. do 14. vrstice
```

- Podatke v p.o. običajno uvozimo iz podatkovnih baz, ustvarjenih v drugih programih (tekstovne datoteke, Excelove datoteke, SPSSove datoteke ipd.) …

Branje in shranjevanje objektov v datoteke

Uvoz podatkov iz tekstovnih datotek

Tekstovne (besedilne) datoteke so najenostavnejše datoteke za delo s podatki. To so datoteke, ki jih običajno beremo in urejamo z enostavnimi urejevalniki besedil (npr. Notepad, Notepad++, Wordpad ipd.).

Kot smo že izvedeli pri podatkovnih okvirjih, so podatkovne baze običajno sestavljene tako, da imamo v vrsticah enote merjenja, v stolpcih pa spremenljivke, pri čemer v prvo vrstico običajno zapišemo imena spremenljivk (glava datoteke oz. *header*). Spremenljivke v tekstovnih datotekah lahko ločimo z različnimi znaki, in sicer tabulatorjem (\t), podpičjem (;), vejico (,) itd.

Za branje tekstovnih datotek (večinoma ločenih s podpičjem ali tabulatorjem) uporabljamo funkcijo `read.table()` oz. njene bolj specifične izpeljanke, in sicer:

```
read.csv()      # ali   read.csv2()
read.delim()    # ali   read.delim2()
```

Razlika med funkcijama, ki ju loči le številka "2" na koncu imena, je malenkostna, saj se razlikujeta samo v nekaterih privzetih vrednostih. V slovenskem jeziku za decimalno ločilo uporabljamo vejico (v nekaterih drugih jezikovnih okoljih se uporablja pika), zato vejice ne moremo uporabiti kot ločilo za razmejevanje stolpcev (za ločilo zato običajno uporabimo podpičje). Funkciji z "2" na koncu imena imata torej privzete vrednosti za ločilo in decimalno ločilo, ki ustrezata našim pravopisnim pravilom. Funkcija `read.csv2()` ima sledeče argumente:

```
read.csv2(file, header = TRUE, sep = ";", quote = "\"", dec = ",",
          fill = TRUE, comment.char = "", ...)
```

Funkcija `read.delim2()` se od zgornje razlikuje le po tem, da za ločilo stolpcev uporablja tabulator (\t):

```
read.delim2(file, header = TRUE, sep = "\t", quote = "\"", dec = ",",
            fill = TRUE, comment.char = "", ...)
```

Prvi argument (file) je edini nujni argument teh funkcij, in sicer na njegovo mesto v narekovajih vpišemo ime datoteke (s končnico vred, npr. txt, dat, csv), če se ta datoteka nahaja v delovnem direktoriju. V nasprotnem primeru moramo vpisati celotno pot do datoteke. Večinoma bo vpis zgolj imena zadostoval, saj so privzete vrednosti drugih argumentov v večini primerov optimalne.

Podatke, ločene z vejico ali tabulatorjem lahko uvozimo tudi tako, da jih kopiramo iz tekstovne datoteke in jih nato s funkcijo `read.csv2()` oz. `read.delim2()` prilepimo v objekt oz. priredimo objektu. Namesto imena datoteke v tem primeru vpišemo "clipboard" (tj. odložišče). Če kopiramo podatke iz Excelove datoteke (glej naslednje poglavje), moramo uporabiti funkcijo `read.delim2()`.

```
po <- read.csv2("clipboard")     # ali      po <- read.delim2("clipboard")
```

POZOR! Ne glede na to, ali je v zunanji datoteki uporabljena vejica ali pika kot decimalno ločilo, bo R po ustreznem uvozu vrednosti z decimalnimi mesti kot decimalno ločilo vedno prikazoval piko! Zaradi tega po uvozu podatkov vedno te podatke prikažemo v oknu levo zgoraj z ukazom `View(ime_podatkov)`, saj bodo numerični podatki poravnani desno, znakovni (in faktorji!) pa levo. Prav tako je smotrno natančno pregledati podatkovni tip spremenljivk, pri čemer lahko za hitrejši pregled tipov vseh spremenljivk uporabimo funkcijo:

```
sapply(ime_objekta_s_podatki, class)
```

Informacije o podatkovem tipu objektov (ali spremenljivk v p.o.) so dostopne tudi v oknu "Environment/Okolje" (desno zgoraj). Namesto funkcije `View()` lahko za prikaz objekta zgolj kliknemo na ime objekta v tem oknu.

Uvoz podatkov iz Excelovih datotek

Pri uvozu Excelovih datotek (xls ali xlsx) lahko v Excelu datoteko shranimo kot tekstovno datoteko, ločeno s tabulatorjem ali s podpičjem; v tem primeru postopamo enako kot v predhodnem poglavju (z `read.csv2` ali `read.delim2`). Če želimo neposredno uvoziti datoteko v Excelovem formatu, je najboljše uporabiti enega od razpoložljivih paketov, in sicer xlsx. Funkcija `read.xlsx` ima mnogo argumentov (poglejte pomoč za to funkcijo), a bosta v večini primerov zadostovali le trije, in sicer (i) ime datoteke (ali celotna pot do datoteke), (ii) številka delovnega lista, ki ga želimo uvoziti, in (iii) vrsta kodiranja znakov (encoding = "UTF-8"), saj se bodo tako v imenih spremenljivk kot v vrednostih nekaterih spremenljivk lahko pojavljali šumniki. Ta funkcija nima argumenta, ki bi naslavljal vrsto decimalnega ločila, zato podatke predhodno v Excelu ustrezno uredimo (morebitne decimalne pike spremenimo v vejice). Npr.

```
# v objekt podatki bomo uvozili 1. delovni list iz datoteke podatki.xlsx

podatki <- read.xlsx("podatki.xlsx", 1, encoding = "UTF-8")
```

Izvoz podatkov

Na novo ustvarjene objekte/podatke ali spremenjene objekte lahko shranimo, pri čemer imamo več možnosti. Kot smo že zapisali, so objekti vse podatkovne strukture in funkcije. Objekte/podatke lahko:

- izvozimo v **podatkovno datoteko** (npr. tekstovno, Excelovo)

  ```
  # objekt (p.o.) bomo shranili v s podpičjem ločeno tekstovno datoteko
  # (lahko ji damo končnico txt, csv ali dat)

  write.csv2(podatki, "moji_podatki.txt")
  ```

```
# objekt (p.o.) bomo shranili v Excelovo datoteko

write.xlsx(podatki, "moji_podatki.xlsx")
```

- shranimo kot **posamezne objekte** (*.RData)

```
# shranimo samo podatkovni okvir "podatki"

save(podatki, file = "moji_podatki.RData")
```

- shranimo kot **celotno delovno okolje** (*.RData)

```
# shranimo celotno delovno okolje oz. vse objekte v okolju

save.image("moj_projekt.RData")
```

Posamezne shranjene objekte ali delovno okolje naložimo s funkcijo `load()`:

```
load("ime_objekta_ali_okolja.RData")
```

Shranjevanje ustvarjenih rezultatov/tabel in slik

Če so rezultati določene funkcije oblikovani kot matrika ali podatkovni okvir, lahko objekt, v katerem so ti rezultati shranjeni, shranimo na enak način kot podatkovni okvir, torej z uporabo funkcij `write.csv2` ali `write.xlsx`. Lahko pa tabele tudi enostavno kopiramo, prilepimo v Excel in jih tam naknadno uredimo (na začetku s funkcijo "Besedilo v stolpce").

Ustvarjene slike (običajno različni grafi) je najlažje shraniti prek gumba "Export", ki se nahaja v desnem spodnjem oknu (zavihek "Plots"). Slike lahko izvozimo v enega od standardnih slikovnih formatov (png, jpg …) ali v pdf.

Izbor določene skupine udeležencev iz celotnega vzorca

Velikokrat želimo določene statistične analize opraviti le na določeni skupini oz. podvzorcu udeležencev (vrstic). Na voljo je mnogo načinov, kako izberemo določen podvzorec vrstic (prek indeksov in pogojev za vrednosti določenih spremenljivk), najenostavnejši način pa je z uporabo funkcije `subset()`.

```
# v nov p.o. shranimo samo ženske udeleženke (vrednost "Ž" v spremenljivki
# spol) iz p.o. "po"

nov_po <- subset(po, spol == "Ž")

# v nov p.o. shranimo samo udeležence, ki so mlajši od 10 ali več kot 20 let

nov_po <- subset(po, starost < 10 | starost > 20)

# v nov p.o. shranimo moške udeležence, ki so stari 30 let ali več

nov_po <- subset(po, spol == "M" & starost >= 30)

# poleg vrstic, lahko z argumentom "select" ohranimo tudi samo izbrane
# spremenljivke (npr. spol, izobrazba)

nov_po <- subset(po, starost > 25, select = c(spol, izobrazba))
```

Kot že rečeno, lahko določeno podskupino izberemo tudi prek indeksov, npr.:

```
nov_po <- po[po$spol == "M" & po$starost > 25,]
```

Funkcije, ki smo jih spoznali v poglavju Osnovne informacije

```
? ali help()          # pomoč za funkcijo
as.character()        # spreminjanje spremenljivke v besedilno spremenljivko
as.factor()           # spreminjanje spremenljivke v faktor
as.numeric()          # spreminjanje spremenljivke v numerično spremenljivko
attach()              # pripenjanje imen spremenljivk p.o. v iskalno pot R-a
c()                   # funkcija combine, vpis vrednosti objekta
cbind()               # združevanje vektorjev po stolpcih
class()               # izpis podatkovnega tipa/strukture objekta
data.frame()          # ustvarjanje podatkovnega okvirja (p.o.)
detach()              # odstranitev pripetega po iz iskalne poti R-a
dim()                 # dimenzije objekta
edit()                # urejanje podatkovnih okvirjev
factor()              # ustvarjanje faktorja
getwd()               # izpis delovnega direktorija
install.packages()    # namestitev paketa
length()              # št. elementov vektorja/matrike, št. stolpcev (!) p.o.
library()             # aktivacija paketa
list()                # ustvarjanje seznama
load()                # nalaganje shranjenega objekta/delovnega okolja
ls()                  # izpis vseh objektov v trenutni seji
matrix()              # ustvarjanje matrike
names()               # izpis/ustvarjanje oznak spremenljivk v p.o.
rbind()               # združevanje vektorjev po vrsticah
read.csv2()           # branje tekstovne datoteke (podpičje ločilo stolpcev)
read.delim2()         # branje tekstovne datoteke (tabulator ločilo stolpcev)
read.xlsx()           # branje Excelove datoteke
sapply()              # izvedba izbrane funkcije nad objektom
save()                # shranjevanje izbranega objekta
save.image()          # shranjevanje vseh objektov/delovnega okolja
scan()                # zaporedno vpisovanje vrednosti objekta
search()              # izpis aktiviranih paketov v trenutni seji
setwd()               # nastavitev delovnega direktorija
str()                 # podobno kot class(), izpišejo se tudi podatki
subset()              # izbor določenih vrstic in stolpcev v p.o.
View()                # izpis objekta v oknu levo zgoraj (tabela s podatki)
write.csv2()          # shranjevanje podatkov v tekstovno datoteko
write.xlsx()          # shranjevanje podatkov v Excelovo datoteko
```

Urejanje podatkov, pregled porazdelitev/strukture spremenljivk

Merske ravni spremenljivk

V tem poglavju se bomo začeli ukvarjati s statističnimi analizami. Merska raven spremenljivk je ena od lastnosti spremenljivk, ki ima (med drugimi lastnostmi) ključno vlogo pri odločitvi, katere statistične postopke bomo izbrali. Z vidika merskih ravni lahko imajo spremenljivke tri lastnosti, ki so urejene hierarhično (najprej najosnovnejša):

- **velikost**: Lahko vrednosti spremenljivke uredimo po velikosti? Lahko za dve različni vrednosti neke spremenljivke rečemo, da ena odraža "več/manj" merjene lastnosti kot druga?
- **enaki intervali**: Ali so razdalje med merskimi enotami enake? Npr., je razlika med 5 cm in 10 cm enaka kot med 115 cm in 120 cm? Če različnim delom dneva pripišemo zaporedne vrednosti (1-jutro, 2-dopoldne, 3-popoldne, 4-večer, 5-noč), so razdalje med temi "enotami" oz. med zaporednimi pripisanimi vrednostmi enake?
- **absolutna ničla**: Ali vrednost 0 pomeni popolno odsotnost merjene lastnosti?

Na podlagi teh treh lastnosti lahko izpeljemo štiri merske ravni:

merska raven	velikost	enaki intervali	absolutna 0	Kaj lahko počnemo?
nominalna	✗	✗	✗	Samo štejemo pojavitve posameznih vrednosti (frekvence). Lahko računamo mere, ki temeljijo na frekvencah.
ordinalna	✓	✗	✗	Lahko tudi razvrščamo po velikosti. Lahko računamo mere, ki temeljijo na velikosti/vrstnem redu (npr. max, min, percentili).
intervalna	✓	✓	✗	Dodatno lahko računamo mere, ki temeljijo na seštevanju/odštevanju.
razmernostna	✓	✓	✓	Lahko računamo mere, ki temeljijo na katerikoli matematični operaciji.

Razvrščanje podatkov

Če delamo samo z vektorjem ali imamo v p.o. samo eno spremenljivko, lahko uporabimo funkcijo `sort()`. Ne glede na smer razvrščanja bodo manjkajoče vrednosti privzeto postavljene na konec vektorja/spremenljivke.

```
v <- sort(v)                      # vrednosti bodo razvrščene naraščajoče
                                  # (privzeta nastavitev)
v <- sort(v, decreasing = TRUE)   # vrednosti bodo razvrščene padajoče
```

Pri razvrščanju matrik in p.o. glede na eno ali več izbranih spremenljivk (stolpcev), se morajo ustrezno preurediti tudi vrednosti ostalih spremenljivk. V tem primeru uporabimo funkcijo `order()`. Privzeto razvrščanje je naraščajoče!

```
s1 <- c(5,1,8)   # imamo samo eno spremenljivko/vektor s1

order(s1)        # Vrne zaporedne številke vrednosti [indekse] iz originalne
                 # spremenljivke v takem vrstnem redu, ki ustreza naraščajoči
                 # ranžirni vrsti. V tem primeru funkcija vrne vrednosti
                 # 2 1 3, kar pomeni, da bi za naraščajočo razvrstitev na prvo
                 # mesto morali postaviti 2. vrednost (to je 1), nato 1.
                 # vrednost (to je 5) in nazadnje 3. vrednost (to je 8).
```

```
# za padajoče razvrščanje lahko uporabimo argument decreasing = TRUE ali pred
# spremenljivko zapišemo znak minus (-)

order(s1, decreasing = TRUE)     # ali
order(-s1)
```

Oba zgornja primera bosta vrnila 3 1 2; za padajočo razvrstitev bi morali na prvo mesto postaviti 3. vrednost (8), nato 1. vrednost (5), nato 2. vrednost (1).

V večini primerov bomo želeli razvrščati vse spremenljivke glede na eno ali več spremenljivk. Npr., delamo s podatkovnim okvirjem "po", ki ima štiri spremenljivke: spol, starost, višina, teža. V spodnjem primeru spreminjamo obstoječi p.o.!

```
po <- po[order(po$starost),]     # naraščajoče razvrščanje
                                 # p.o. po starosti
po <- po[order(-po$starost),]    # padajoče razvrščanje p.o.
                                 # po starosti
```

Podatke v p.o. lahko razvrščamo tudi po več spremenljivkah hkrati, pri čemer je tako razvrščanje hierarhično oz. ugnezdeno. Če npr. razvrščamo po dveh spremenljivkah, bo algoritem podatke najprej razvrstil po spremenljivki, ki smo jo v funkciji navedli na prvem mestu, nato pa še po drugi, in sicer samo pri enakih/vezanih vrednosti prve/nadredne spremenljivke. To pravilo lahko posplošimo na vse sledeče/podredne spremenljivke, po katerih želimo razvrščati podatkovni okvir.

```
# naraščajoče razvrščanje p.o. po starosti, nato po telesni višini

po <- po[order(po$starost, po$višina),]

# najprej naraščajoče po starosti, nato padajoče po telesni teži

po <- po[order(po$starost, -po$teža),]
```

Rekodiranje vrednosti spremenljivk

Pri izvedbi statističnih analiz je velikokrat potrebno spremeniti vrednosti določenih spremenljivk oz. je potrebno njihove vrednosti rekodirati. Pri enostavnih rekodiranjih (npr. želimo spremeniti samo eno vrednost) lahko to storimo prek indeksiranja:

```
# vse manjkajoče vrednosti v spremenljivki starost smo zamenjali z 999

po$starost[is.na(po$starost)] <- 999

# vse vrednosti na spremenljivki starost, večje od 80, smo zamenjali z NA

po$starost[po$starost > 80] <- NA
```

Pri kompleksnejših rekodiranjih pa si bomo pomagali s kontrolno strukturo `ifelse`:

```
ifelse(test, yes, no)

test      logični test
yes       vrednost, ki jo vrne ukaz, če je test resničen (TRUE)
no        vrednost, ki jo vrne ukaz, če je test neresničen (FALSE)
```

Na položaj "no" lahko ugnezdimo nov `ifelse` izraz (gnezdenje lahko nadaljujemo do [skoraj] poljubne globine).

```
# spol imamo trenutno zakodiran z 1 (Ž) in 2 (M), želimo pa imeti vrednosti
# 0 (Ž) in 1 (M)

po$spol <- ifelse(po$spol == 1, 0, 1)

# stopnjo izobrazbe (izob) imamo zakodirano z vrednostmi od 1 do 8, želimo pa
# nekatere kategorije združiti, in sicer 1 do 3 v 1, 4 v 2, 5 in 6 v 3 ter
# ostalo (7 in 8) v 4

po$izob <- ifelse(po$izob < 4, 1, ifelse(po$izob == 4, 2,
           ifelse(po$izob > 4 & po$izob < 7, 3, 4)))

# spremenljivko starost želimo dihotomizirati (rekodirati tako, da bomo
# ohranili samo dve vrednosti, in sicer "mlajši" (1) in "starejši" (2)); mejo
# med skupinama smo določili pri starosti 40 let (40-letniki so že v 2.
# skupini)

po$starost <- ifelse(po$starost < 40, 1, 2)
```

Rekodiranje vrednosti spremenljivk lahko izvedemo tudi s posebnimi funkcijami za rekodiranje, npr. s funkcijo `recode()` iz paketa car. Tej funkciji moramo nujno podati prva dva argumenta, in sicer spremenljivko, katere vrednosti želimo rekodirati, in opredelitev pravil za rekodiranje (pravila zapišemo v narekovajih). Vloga ostalih treh argumentov (as.factor.result, as.numeric.result, levels) je opisana v pomoči za to funkcijo. Če opredelimo več pravil, jih ločimo s podpičjem. Če v katerem od pravil nova vrednost predstavlja besedilni znak ali niz znakov, novo (besedilno) vrednost zapišemo v enojnih narekovajih. Pri rekodiranju lahko stare vrednosti prepišemo z novimi vrednostmi (torej neposredno spremenimo originalno spremenljivko) ali pa nove vrednosti shranimo v novo spremenljivko (priporočen način!).

```
# v p.o. "po" imamo spremenljivko izobrazba s pet ravnmi ("OŠ", "SŠ", "univ",
# "mag", "dr"), spremenljivko starost, ki vsebuje vrednosti v letih (npr. od
# 25 do 65 let) in spremenljivko spol, ki vsebuje vrednosti 1 in 2

# pri ustvarjanju spremenljivke izobrazba _rekod smo vrednosti 'univ'
# zamenjali z 'uni', 'mag' in 'dr' pa smo zamenjali z 'več kot uni'); za
# besedilne oznake originalnih in novih vrednosti smo uporabili enojne
# narekovaje!

po$izobrazba_rekod <- recode(po$izobrazba, "'univ'='uni';
                      c('mag','dr')='več kot uni'")

# pri ustvarjanju spremenljivke starost_rekod smo vrednosti do 40 let
# zamenjali z 'mlajši', manjkajoče vrednosti smo prekopirali, vrednosti nad
# 39 pa zamenjali s 'starejši'; za določitev vrednosti 'starejši' smo
# uporabili argument "else"; "else" pravilo moramo vedno postaviti na konec
# niza pravil, saj se nanaša na vse vrednosti, ki niso omenjene v predhodnih
# pravilih, tudi na sistemske manjkajoče vrednosti (NA); če želimo pri teh
# vrednostih ohraniti sistemsko oznako NA, moramo to pravilo tudi zapisati

po$starost_rekod <- recode(po$starost, "25:39='mlajši'; NA=NA;
                    else='starejši'")

# pri ustvarjanju spremenljivke spol_01 smo vse vrednosti 1 zamenjali z
# vrednostjo 0 in vse vrednosti 2 z vrednostjo 1

po$spol_rekod <- recode(po$spol, "1=0; 2=1")
```

Frekvenčne porazdelitve

Pred izvedbo zahtevnejših statističnih analiz običajno najprej opravimo osnovni pregled podatkov, in sicer s pregledom frekvenčnih porazdelitev spremenljivk (v obliki tabele in/ali slike), opisnih statistik ipd.

Frekvenčne tabele za pregled ene spremenljivke

V tabelah frekvenčne porazdelitve ene spremenljivke običajno prikažemo frekvence (*f*; število pojavitev posamezne vrednosti spremenljivke), deleže (*p*; delež pojavitev vrednosti) in kumulativne deleže ali frekvence (*cp*/*cf*), ki nam sporočajo, kolikšen delež/koliko podatkov se nahaja do (običajno) zgornje meje razreda (za diskretne ali grupirane spremenljivke) ali do vključno trenutne (zvezne) vrednosti. Npr. spodaj se nahaja enostavna frekvenčna porazdelitev za dosežke na testu znanja, na katerem je bilo možno doseči 5 točk (kumulativne frekvence in deleži so izračunani na zgornjo mejo vrednosti/razreda).

dosežek	*f*	*p*	*cf*	*cp*
0	5	0,07	5	0,07
1	3	0,04	8	0,11
2	14	0,20	22	0,31
3	25	0,35	47	0,66
4	16	0,23	63	0,89
5	8	0,11	71	1,00
skupaj	71	1,00		

Frekvenčno tabelo za eno spremenljivko (npr. "var") lahko ustvarimo na več načinov. Najosnovnejši vključuje uporabo funkcij `table()`, `prop.table()` in `cumsum()`. Predpostavimo, da imamo p.o. pripet (*attached*).

```
# izračunamo frekvence

f_var <- table(var)

# izračunamo deleže (iz frekvenc, ne iz osnovne spremenljivke!); če želimo
# zmanjšati število decimalnih mest, lahko prop.table ugnezdimo v funkcijo
# round(); v 2. primeru smo deleže zaokrožili na 3 decimalna mesta

p_var <- prop.table(f_var)
p_var <- round(prop.table(f_var), 3)

# dodamo še kumulativne frekvence in deleže s funkcijo cumsum()

cf_var <- cumsum(f_var)
cp_var <- cumsum(p_var)

# na koncu s funkcijo cbind() združimo vse mere kot stolpce in opredelimo
# objekt kot podatkovni okvir (ni nujno, a nam to omogoča dostopanje do mer
# po vzoru po$mera)

tbl_var <- as.data.frame(cbind(f_var, p_var, cf_var, cp_var))
```

Funkcija `table()` privzeto ignorira manjkajoče vrednosti (NA). Če jih želimo vključiti, dodamo argument `exclude = NULL`. Npr.:

```
f_var <- table(var, exclude = NULL)
```

Slike frekvenčne porazdelitve spremenljivke

Za slikovni prikaz porazdelitve **vsaj intervalnih spremenljivk** uporabljamo **histogram**. Histogram je vrsta stolpčnega grafa, pri čemer na absciso navedemo vrednosti (ali v primeru grupiranih podatkov sredine razredov/zgornje ali spodnje meje), na ordinato pa ali frekvence ali deleže. Histogram lahko pogojno uporabimo tudi v primeru ordinalnih spremenljivk, vendar v tem primeru ne smemo interpretirati oblike porazdelitve!

```
# objektu hist_var smo priredili histogram za spremenljivko "var" s funkcijo
# hist()

hist_var <- hist(var)
```

Funkcija `hist()` ima mnogo argumentov. V spodnjem primeru smo npr. določili barvo stolpcev (col = "grey"), določili (približno!) število stolpcev (breaks = 7) ter določili spodnjo in zgornjo mejo vrednosti na abscisi in ordinati (xlim = c(10, 40), ylim = c(0, 50)):

```
hist_var <- hist(var, col = "grey", breaks = 7, xlim = c(10, 40),
           ylim = c(0, 50))
```

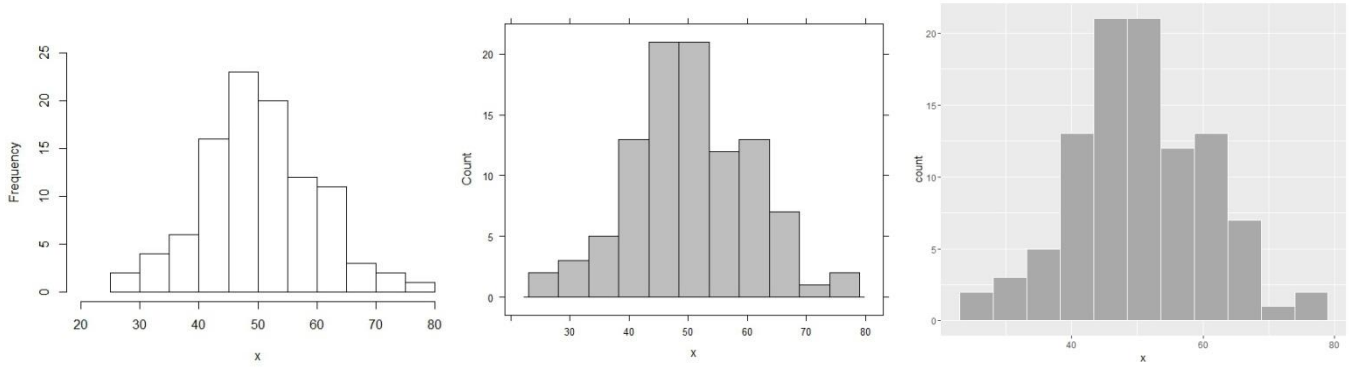
Če zgolj izpišemo vsebino objekta, ki smo mu priredili histogram, dobimo vpogled v osnovne značilnosti histograma (npr. uporabljene meje med razredi, sredine razredov, frekvence v razredih).

Histogram lahko narišemo tudi s funkcijami:

- `histogram()` iz paketa mosaic (oz. lattice). Poleg velikega števila argumentov, je potrebno izpostaviti argument type, s katerim lahko določimo, ali bodo na ordinati prikazane frekvence (count), deleži/odstotki (percent) ali pa ocenjena gostota verjetnosti (density).
- `qplot()` iz paketa ggplot2

```
# histogram s 5 razredi, modrimi stolpci, odstotki na ordinati

histogram(var, type = "percent", col = "blue", nint = 5)

# histogram (tip grafa določimo z argumentom "geom") s 7 razredi/stolpci,
# sivimi stolpci in tankimi belimi črtami med stolpci

qplot(matura, geom = "histogram", bins = 7, fill = I("grey"),
      col = I("white"))
```

Spodaj so prikazani primeri histogramov, narisanih s funkcijami `hist()`, `histogram()` in `qplot()`.



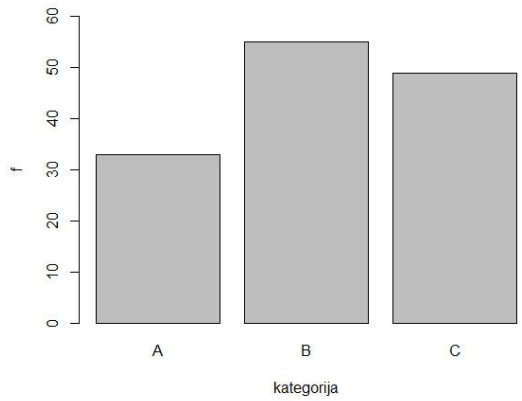
Graf z ocenjeno gostoto verjetnosti lahko narišemo s pomočjo funkcij `density()` in `plot()`.

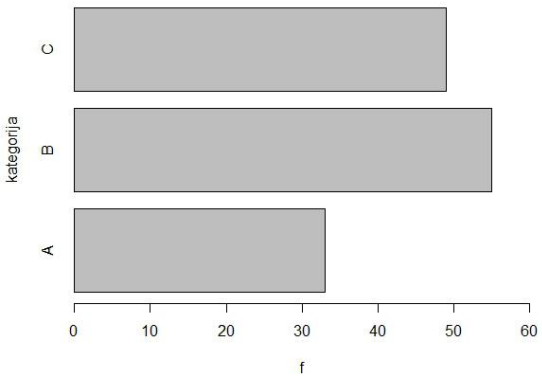
```
plot(density(var))
```

Za prikaz frekvenčnih porazdelitev **nominalnih spremenljivk** običajno uporabljamo stolpčni graf (angl. *bar graph/plot*), lahko pa tudi strukturne stolpce, pite ipd. V primerjavi s histogramom v stolpčnem grafu zaporedje kategorij na abscisi nima smisla, saj kategorij med seboj ne moremo primerjati po velikosti. Zaradi tega oblike stolpčnega grafa ni smiselno interpretirati. Histogram in stolpčni graf najhitreje ločimo tako, da se stolpci pri histogramu stikajo, pri stolpčnem grafu pa so razmaknjeni.

```
# stolpčni graf za spremenljivko barva_las; oznaka y-osi (f),
# oznaka x-osi (barva las)

barva_las_tbl <- table(barva_las)
barplot(barva_las_tbl, xlab = "barva las", ylab = "f")

# vodoravna različica prejšnjega primera

barplot(barva_las_tbl, xlab = "f", ylab = "barva las", horiz = TRUE)
```

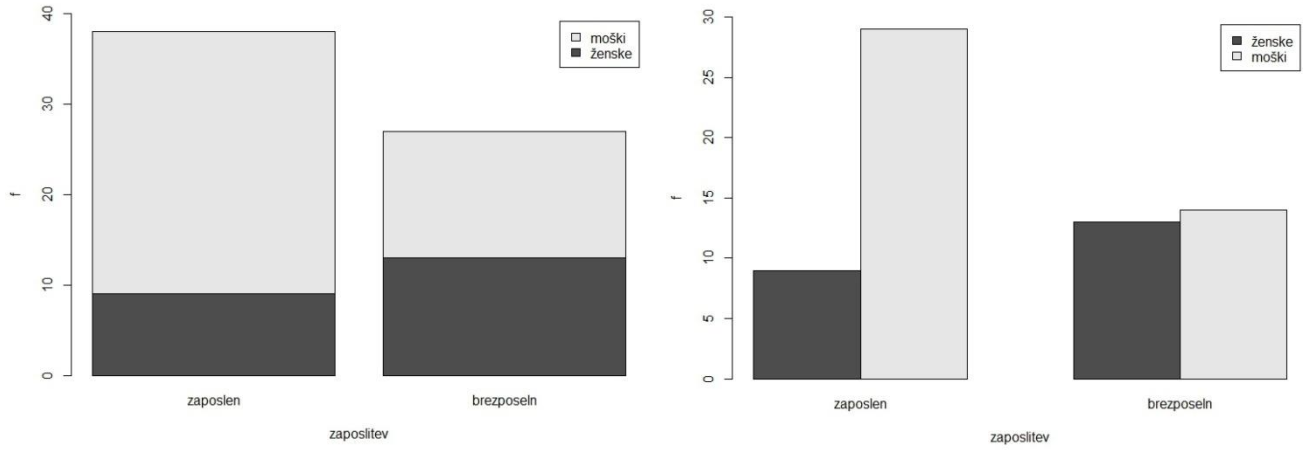
Spodaj sta prikazana primera navpičnega in vodoravnega stolpčnega grafa (`barplot()`) za tri kategorije.






```
# naložen (stacked) stolpčni graf (ali graf s strukturnimi stolpci) za
# spremenljivki spol (0-ž, 1-m) in zaposlitev (1-da, 2-ne); najprej ustvarimo
# jedro kontingenčne tabele (glej naslednje poglavje!); spremenljivka na 1.
# mestu bo predstavljala vrstice, spremenljivka na drugem mestu pa stolpce

spol_zap_tbl <- table(spol, zaposlitev)

# spremenljivka v stolpcih tabele bo predstavljala kategorije na x-osi, spr.
# v vrsticah bo "naložena" znotraj kategorije spremenljivke v stolpcih (torej
# struktura po spolu znotraj zaposlitve); zapišemo oznake kategorij na x-osi
# (names.arg) in oznake za legendo (legend)

barplot(spol_zap_tbl, xlab = "zaposlitev", ylab = "f",
        names.arg = c("zaposlen", "brezposeln"),
        legend = c("ženske", "moški"))

# namesto naloženega stolpčnega grafa, lahko z barplot() narišemo tudi
# grupiran stolpčni graf (dodamo argument beside = TRUE), pri čemer bodo
# predhodno ugnezdeni stolpci v kategorijah na x-osi v tem primeru
# postavljeni drug ob drugem; v tem primeru bomo prikazali število žensk in
# moških v ločenih stolpcih znotraj kategorij zaposlenosti

barplot(spol_zap_tbl, xlab = "zaposlitev", ylab = "f",
        names.arg = c("zaposlen", "brezposeln"),
        legend = c("ženske", "moški"), beside = TRUE)
```

Spodaj sta prikazana primera naloženega stolpčnega grafa (levi primer) in grupiranega stolpčnega grafa.




```
# strukturo nominalne spremenljivke lahko prikažemo tudi v obliki pite (npr.
# za spol: 1 = ž, 2 = m)

spol_tbl <- table(spol)
oznake <- c("ženske", "moški")
pie(spol_tbl, labels = oznake)

# oznakam lahko dodamo tudi vrednosti/odstotke

odstotki <- round(100*spol_tbl/sum(spol_tbl))
oznake <- paste(oznake, odstotki, "%")
pie(spol_tbl, labels = oznake)
```

Kontingenčne tabele (prikaz večstopenjske strukture nominalnih spremenljivk)

V primeru nominalnih spremenljivk nas v veliko primerih zanima večstopenjska struktura oz. frekvenčna porazdelitev po kombinacijah več spremenljivk hkrati (običajno za dve, lahko pa tudi za več spremenljivk). Kontingenčne tabele lahko ustvarimo na več načinov; v spodnjih primerih bomo uporabili funkcije `table()`, `prop.table()` in `margin.table()`, `ftable()`, `CrossTable` (paket gmodels) in `table2d_summary` (paket vcd).

```
# tvorba dvodimenzionalne kontingenčne tabele s funkcijami table, prop.table
# in margin.table (npr. 2 * 3 tabela; spol * barva_oci [rjava, modra,
# zelena])

# ustvarimo jedro kontingenčne tabele z dvema vrsticama in tremi stolpci

tbl <- table(spol, barva_oci)

# tabeli dodamo robne frekvence za vrstice (seštevki vrednosti v vrsticah);
# margin.table(tbl, 1); s cbind jih dodamo kot zadnji stolpec

tbl <- cbind(tbl, margin.table(tbl, 1))

# tabeli dodamo še robne frekvence za stolpce kot dodano zadnjo vrstico
# (seštevki vrednosti po stolpcih; ker imamo v tbl že dodane robne frekvence
# v vrstici, bo margin.table(tbl, 2) ustvarila tudi število vseh opazovanj

tbl <- rbind(tbl, margin.table(tbl, 2))
```

```
# namesto frekvenc lahko v jedru kontingenčne tabele prikažemo deleže glede
# na celoten numerus (torej f/N)

tbl <- table(spol, barva_oci)
tbl_p <- prop.table(tbl)

# … ali deleže znotraj vrstic; f/vsota(f) v določeni vrstici

tbl_p_r <- prop.table(tbl, 1)

# … ali deleže znotraj stolpcev; f/vsota(f) v določenem stolpcu

tbl_p_c <- prop.table(tbl, 2)
```

V primeru, če želimo ustvariti 3 ali večdimenzionalno kontingenčno tabelo, moramo eno (ali več) spremenljivk ugnezditi ali znotraj vrstic ali stolpcev. Npr., spremenljivkama spol in barva oči dodamo še zaposlitev (zaposlen, brezposeln).

```
tbl <- table(spol, barva_oci, zaposlitev)

# za lepši prikaz take tabele uporabimo funkcijo ftable(); prva spremenljivka
# bo v vrsticah, druga bo ugnezdena v vrstice, tretja pa bo v stolpcih

ftable(tbl)
```

Če želimo ustvariti zgolj dvodimenzionalne kontingenčne tabele, sta zelo uporabni tudi funkciji `CrossTable()` iz paketa gmodels in `table2d_summary()` iz paketa vcd. Če funkciji `CrossTable()` podamo le prva dva argumenta (obe nominalni spremenljivki), bo funkcija v vsaki celici jedra tabele vrnila frekvenco, prispevek k χ^2 (pride v poštev pri statističnem zaključevanju), delež znotraj vrstice, delež znotraj stolpca in delež glede na celoten numerus.

```
# s CrossTable želimo ustvariti 2 * 2 kontingenčno tabelo
# (npr. spol * ročnost), ki bo vsebovala le (jedrne in robne) frekvence

tbl_2_2 <- CrossTable(spol, ročnost, prop.r = FALSE, prop.c = FALSE,
        prop.t = FALSE, prop.chisq = FALSE)
```

Funkcija `table2d_summary()` ima zelo podobne možnosti kot predhodno obravnavana funkcija, le da vrne tabelo, ki jo lažje prenesemo v Excel in uredimo skladno z APA standardi. Tej funkciji moramo podatke posredovati v obliki tabele, ki jo vrne funkcija `table()`.

```
# ustvarimo 2 * 3 kontingenčno tabelo (spol * barva_oci); poleg frekvenc
# prikažemo še skupne odstotke in odstotke znotraj stolpcev)

tbl <- table(spol, barva_oci)

table_2_3 <- table2d_summary(tbl, percentages = TRUE,
          conditionals = "column")
```

Funkcije, ki smo jih spoznali v poglavju Urejanje podatkov, pregled porazdelitev spremenljivk

```
barplot()             # izris stolpčnega grafa
CrossTable()          # kontingenčna tabela; paket gmodels
cumsum()              # izračun kumulativnih frekvenc/deležev
density()             # izračun ocenjenih gostotnih verjetnosti frek. porazd.
ftable()              # lepši (flat) prikaz večdimenzionalnih konting. tabel
hist()                # izris histograma
histogram()           # izris histograma; paket mosaic/lattice
is.na()               # ali je vrednost manjkajoča; vrne TRUE/FALSE
margin.table()        # izračun robnih frekvenc frekvenčne tabele
order()               # razvrščanje p.o. po eni ali več spremenljivkah
pie()                 # izris pite (strukturni graf)
plot()                # splošna funkcija za risanje grafov
prop.table()          # izračun različnih deležev v frekvenčni tabeli
recode()              # rekodiranje vrednosti spremenljivke; paket car
qplot()               # izris histograma (če geom="histogram"); paket ggplot2
round()               # zaokroževanje vrednosti na želeno št. dec. mest
sort()                # razvrščanje vrednosti vektorja/ene spr. v p.o.
table()               # frekvenčna porazdelitev ene ali kombinacije spr.
table2d_summary()     # kontingenčna tabela; paket vcd
```

Položaj podatka/osebe v (katerikoli) porazdelitvi

Pri določanju položaja določenega podatka (oz. v psihologiji običajno posameznikovega podatka) v porazdelitvi si bomo pomagali z absolutnimi rangi, percentilnimi rangi in kvantili. Postopki, ki jih bomo spoznali v tem poglavju, zahtevajo vsaj ordinalno mersko raven obravnavane spremenljivke in so neodvisni od oblike porazdelitve.

Absolutni rangi

Absolutni rangi vrednosti neke spremenljivke predstavljajo tipični primer ordinalne spremenljivke, saj nam sporočajo na katerem mestu spremenljivke se nahaja določen podatek. Pri določanju položaja podatka v porazdelitvi spremenljivko (z *n* vrednostmi) običajno razvrstimo **naraščajoče**, rang 1 pripišemo minimalni vrednosti, rang *n* pa maksimalni vrednosti. Spremenljivki, urejeni po velikosti, rečemo tudi **ranžirna vrsta**.

Če so vse vrednosti spremenljivke med seboj različne (npr. 33, 37, 42), so absolutni rangi naraščajoče zaporedne številke (v tem primeru 1, 2, 3). Če se katerakoli vrednost v ranžirni vrsti ponovi, se srečamo s t. i. **vezanimi rangi**. Vezane range pri ponovljenih vrednostih lahko določamo na različne načine (npr. vsem pripišemo najnižji ali najvišji možni rang), v večini primerov pa vezane range določimo kot povprečje (aritmetično sredino) absolutnih rangov, ki bi jih imele ponovljene vrednosti, če bi bile različne. Npr. če ima spremenljivka vrednosti 12, 17, 17, 21 bi absolutni rangi teh vrednosti znašali 1, 2,5, 2,5, 4; če bi ponovljeni vrednosti (17) bili različni, bi imeli absolutna ranga 2 in 3 (aritmetična sredina 2 in 3 znaša 2,5). Za določitev absolutnih rangov spremenljivke bomo v Rju uporabili funkcijo `rank()`. Range običajno označujemo z *R*(X), kar pomeni absolutni rang vrednosti X.

Manjkajoče vrednosti bodo privzeto postavljene na konec vrste, pri čemer jim bo R tudi dodelil ustrezni rang. Če jih želimo izločiti pri določanju rangov, funkciji dodamo argument `na.last = NA`, vendar tak način ni priporočljiv, saj bo vektor vrnjenih vrednosti (rangov) vseboval manj vrednosti kot originalna spremenljivka; v tem primeru na novo ustvarjene spremenljivke z rangi ne moremo pripeti k obstoječim spremenljivkam v podatkovnem okvirju. Zaradi tega raje uporabimo argument `na.last = "keep"`, saj bo funkcija manjkajočim vrednostim priredila vrednosti NA, kar pomeni, da bosta dolžini vektorja z rangi in vektorja z originalnimi vrednostmi ostali enaki.

```
R_spr <- rank(spr, ties.method = "average", na.last = "keep")
```

Percentilni rangi

Percentilni rangi, ki jih običajno označujemo s *P*(X), so zgolj absolutni rangi v relativni obliki (so torej v podobnem odnosu kot frekvence in deleži). Pri določanju položaja podatka s pomočjo absolutnih rangov moramo vedno poročati tudi o številu podatkov (numerusu) v spremenljivki (npr. podatek/rezultat 55 točk se nahaja na 28. mestu od 145 podatkov oz. pod zgornjo mejo tega podatka se nahaja 28 podatkov). Pri percentilnem rangu poročamo o deležu (ali odstotku) podatkov, ki leži pod določeno mejo obravnavane vrednosti. Percentilne range lahko računamo na več načinov; v tem priorčniku jih bomo določili kot delež/odstotek podatkov, ki leži pod sredino obravnavane vrednosti (od absolutnega ranga odštejemo polovico intervala, ki ga zaseda):

$$P(X) = \frac{R(X) - 0{,}5}{N} \tag{1}$$

Percentilne range za izmerjene vrednosti spremenljivke (npr. dosežki na spremenljivki "dosezki") bomo izračunali po zgornji enačbi (predhodno smo že ustvarili spremenljivko z absolutnimi rangi "R_test").

```
# če spremenljivka z absolutnimi rangi nima manjkajočih vrednosti,
# je numerus enak dolžini (številu vrednosti) spremenljivke

Pdosezki <- (Rdosezki - 0.5) / length(Rdosezki)
```

```
# če spremenljivka z absolutnimi rangi vsebuje manjkajoče vrednosti,
# pri izračunu dolžine vektorja/spremenljivke izločimo manjkajoče vrednosti

Pdosezki <- (Rdosezki - 0.5) / length(Rdosezki[!is.na(Rdosezki)])
```

Določanje percentilnega ranga vrednosti, ki se pojavi v ranžirni vrsti, je enostavno, saj zgolj odčitamo *P* pri obravnavani vrednosti. Če želimo določiti percentilni rang vrednosti, ki se ne pojavi v ranžirni vrsti, si bomo pomagali z linearno interpolacijo. Po pravilu podobnih trikotnikov bo za surove vrednosti X in percentilne range *P*(X) veljalo:

$$\frac{P(X) - P_0}{P_1 - P_0} = \frac{X - X_0}{X_1 - X_0} \tag{2}$$

pri čemer je *P*(X) percentilni rang, ki ga iščemo, P_0 percentilni rang predhodne vrednosti, P_1 percentilni rang naslednje vrednosti, X vrednost, za katero iščemo *P*, X_0 predhodna vrednost, X_1 naslednja vrednost.

Če želimo z linearno interpolacijo izračunati percentilni rang določene vrednosti, iz enačbe 2 izpostavimo *P*(X).

V Rju si bomo pomagali s funkcijo za linearno interpolacijo,in sicer `approx()`. Prva dva argumenta funkcije predstavljata vrednosti dveh vektorjev/spremenljivk, med katerimi želimo izvesti linearno interpolacijo. Za iskanje percentilnih rangov moramo na prvo mesto vstaviti spremenljivko s surovimi vrednostmi (X), na drugo pa spremenljivko, ki vsebuje percentilne range teh vrednosti. Npr., na spremenljivki "dosezki" se ne pojavi vrednost 48, torej bomo z linearno interpolacijo izračunali *P*(48).

```
P_dosezki_48 <- approx(dosezki, Pdosezki, xout = 48)
```

Kvantili

Kvantili so točke, ki vrednosti spremenljivke delijo na *k* enakih delov oz. na intervale z enakih številom vrednosti. Če je npr. *k* = 100 (vzorec razdelimo na 100 enakih delov), takim kvantilom rečemo percentili. Najbolj pogosto uporabljani kvantili so:

- **percentili** (*k* = 100); npr. P_{33} (33. percentil), P_{80} (80. percentil)
- **kvartili** (*k* = 4); Q_1 (1. kvartil, P_{25}), Q_2 in Q_3
- **decili** (*k* = 10); npr. D_4 (4. decil; P_{40})
- **mediana** (*k* = 2); *Mdn* (ali Q_2, D_5, P_{50}); razdeli vzorec na dva enaka dela

Pri določanju percentilnih rangov za znano surovo vrednost (X) iščemo njen percentilni rang oz. delež podatkov, ki leži pod to vrednostjo. Pri percentilih pa je postopek ravno obraten, saj poznamo percentilni rang vrednosti (iz imena kvantila) in iščemo surovo vrednost (X) z določenim percentilnim rangom. Vrednost 22 v ranžirni vrsti 15, 22, 29 je torej mediana (ali 50. percentil, 5. decil, 2. kvartil).

Če se percentilni rang za iskan percentil pojavi v podatkih, ga zgolj odčitamo, če pa se percentilni rang iskanega percentila ne pojavi v podatkih, si ponovno pomagamo z linearno interpolacijo (tokrat izpostavimo X iz enačbe 2). V Rju bomo ponovno uporabili funkcijo `approx()`, le da bomo v tem primeru na prvo mesto postavili spremenljivko s percentilnimi rangi, na drugo mesto pa spremenljivko s surovimi vrednostmi. Npr., na spremenljivki "dosezki" se percentilni rang 0,85 ne pojavi, zato bomo 85. percentil izračunali z linearno interpolacijo.

```
X_P85 <- approx(Pdosezki, dosezki, xout = 0.85)
```

V Rju sicer obstaja funkcija `quantile()`, ki ima devet različnih načinov izračuna percentilov, vendar noben od načinov ne predstavlja navadne linearne interpolacije. Paket psych vsebuje funkcijo `interp.quantiles()`, ki prav tako izvede linearno interpolacijo nekoliko drugače in zato vodi do malenkost drugačnih rezultatov kot navadna linearna interpolacija.

Funkcije, ki smo jih spoznali v poglavju Položaj podatka/osebe v (katerikoli) porazdelitvi

```
approx()              # linearna interpolacija
interp.quantiles()    # paket psych; izračun kvantilov (drugačna linearna
                      # interpolacija!)
rank()                # določitev absolutnih rangov za vrednosti
                      # spremenljivke
quantile()            # izračun kvantilov (brez navadne linearne
                      # interpolacije!)
```

Mere centralne tendence, razpršenosti in oblike porazdelitve

Za osnovni pregled in opis spremenljivk (poleg frekvenčnih porazdelitev) izračunamo osnovne opisne statistike, ki jih lahko razvrstimo v tri kategorije:

- **Mere centralne tendence** (mere povprečja). Mera centralne tendence predstavlja vrednost, ki je tipični oz. najbolj reprezentativen predstavnik porazdelitve spremenljivke. Imenujemo jih tudi mere centralne lokacije oz. srednjega položaja (spremenljivke). Spoznali bomo modus (*Mo*), mediano (*Mdn*), (tehtano) aritmetično sredino (*M*) in geometrično sredino (*G*).

- **Mere razpršenosti**. Mere razpršenosti odražajo stopnjo različnosti podatkov oz. vrednosti spremenljivke (večja kot je razpršenost, v večji meri so podatki med seboj [v povprečju] različni). Spoznali bomo:
  - **razmike**: variacijski razmik [VR], interkvartilni razmik (Q_3–Q_1), interdecilni razmik (D_9–D_1)
  - in mere razpršenosti, ki so osnovane na **odklonih**, običajno izračunanih kot razlika med vrednostjo in določeno mero centralne tendence (najpogosteje od aritmetične sredine): interkvartilni odklon ((Q_3–Q_1) / 2), povprečni absolutni odklon od aritmetične sredine (PO_M), varianco (s^2, σ^2, SD^2), standardni odklon/deviacijo (s, σ, SD), koeficient variacije (KV).

- **Mere oblike porazdelitve**. Za opis oblike porazdelitve potrebujemo referenčno obliko oz. teoretično porazdelitev, katere oblika predstavlja referenčno obliko (običajno je to oblika normalne porazdelitve [N.P.]). Spoznali bomo meri asimetričnosti (*As*, a_3) in sploščenosti porazdelitve (*Spl*, a_4). Mera asimetričnosti kaže, ali ima porazdelitev spremenljivke v primerjavi z N.P. "podaljšan" levi rep porazdelitve (v povprečju prevladujejo negativni odkloni od aritmetične sredine) ali desni rep porazdelitve (v povprečju prevladujejo pozitivni odkloni od aritmetične sredine). Mera sploščenosti pa kaže, ali je porazdelitev spremenljivke v primerjavi z N.P. bolj koničasta ali bolj sploščena.

Izračuni različnih opisnih statistik zahtevajo različne pretvorbe ali predvsem različne računske operacije, zato moramo pred izračunom dobro premisliti, katera mera je primerna za določeno mersko raven spremenljivke.

merska raven	**Katere opisne statistike lahko računamo?**	
nominalna	Mere centralne tendence	*Mo*
	Mere razpršenosti	/
	Mere oblike porazdelitve	/
ordinalna	Mere centralne tendence	*Mo, Mdn*
	Mere razpršenosti	*VR*, Q_3–Q_1, (Q_3–Q_1)/2
	Mere oblike porazdelitve	/
intervalna	Mere centralne tendence	*Mo, Mdn, M*
	Mere razpršenosti	*VR*, Q_3–Q_1, (Q_3–Q_1)/2, PO_M, *SD*, SD^2
	Mere oblike porazdelitve	*As*, *Spl*
razmernostna	Mere centralne tendence	*Mo*, *Mdn*, *M*, *G*
	Mere razpršenosti	*VR*, Q_3–Q_1, (Q_3–Q_1)/2, PO_M, *SD*, SD^2, *KV*
	Mere oblike porazdelitve	*As*, *Spl*

Mere centralne tendence

Modus

Modus je vrednost spremenljivke z največjo frekvenco (porazdelitev ima lahko več modusov oz. jasno razvidnih "vrhov"). Čeprav lahko modus računamo za spremenljivke na vseh merskih ravneh, ga običajno računamo le na nominalnih in diskretnih spremenljivkah na višjih merskih ravneh. V Rju ni na voljo funkcije za izračun modusa, zato si bomo pri izračunu modusa pomagali s funkcijami `table()` in `which.max()`.

```
# najprej izračunamo frekvence in nato poiščemo vrednost, ki ima največjo
# frekvenco (s funkcijo which.max(), ki vrne vrednost in njen indeks v
# tabeli); če ima več vrednosti enako (največjo) frekvenco, bo which.max
# vrnil najmanjšo od teh vrednosti (za pregled si lahko pomagamo z barplot!)

f_var <- table(po$var)
Mo <- which.max(f_var)

# izračun Mo lahko združimo v en ukaz

Mo <- which.max(table(po$var))
```

Mediana

Mediana je 50. percentil (ali Q_2, D_5), torej (sredina) vrednost(i), pod/nad katero leži 50 % podatkov. Če želimo uporabiti klasično linearno interpolacijo, jo izračunamo tako kot katerikoli drugi kvantil.

```
# izračun mediane s funkcijo approx(); naprej ustvarimo percentilne range
# (Pvar), nato pa izračunamo mediano; pri izračunu percentilnih rangov pazimo
# na morebitne manjkajoče vrednosti (če so prisotne, jim pri uporabi rank()
# določimo vrednost NA (na.last = "keep"), pri length() pa jih izločimo)

Pvar <- (rank(po$var) - 0.5) / length(po$var)
Mdn <- approx(Pvar, po$var, xout = 0.5)

# za izračun mediane (in drugih percentilov) lahko (pogojno) uporabimo tudi
# funkcijo interp.quantiles (paket psych), ki uporablja nekoliko drugačno
# enačbo za interpolacijo in posledično vrne nekoliko drugačen rezultat kot
# funkcija approx

Mdn <- interp.quantiles(po$var, q = 0.5)
```

V R-u obstaja tudi funkcija `median()`, vendar ne omogoča linearne interpolacije. Mediano izračuna na najenostavnejši način, in sicer je v primeru lihega števila podatkov mediana sredinski podatek v urejeni ranžirni vrsti (npr. 6. podatek od 11 podatkov), v primeru sodega števila podatkov pa na sredini med srednjima podatkoma (npr. med 5. in 6. podatkom od 10 podatkov).

Aritmetična sredina

Aritmetično sredino izračunamo po enačbi:

$$M = \frac{\sum X}{N} \tag{3}$$

```
# za izračun aritmetične sredine bomo uporabili funkcijo mean()

M <- mean(po$var)
```

```
# izračun aritmetične sredine na podvzorcu oseb, pri čemer se pogoj za izbor
# podvzorca nanaša na spremenljivko, za katero tudi računamo aritmetično
# sredino; npr., izračunati želimo aritmetično sredino za dosežke na testu
# samo za osebe, ki so imele manj kot 50 točk

M <- mean(po$dosezki[po$dosezki < 50])

# izračun aritmetične sredine na podvzorcu oseb, pri čemer se pogoj za izbor
# podvzorca ne nanaša na spremenljivko, za katero računamo aritmetično
# sredino, ampak na eno ali več drugih spremenljivk; npr., izračunati želimo
# aritmetično sredino za dosežke na testu samo za moške udeležence, ki so
# starejši od 35 let

M <- mean(po$dosezki[po$spol == "M" & po$starost > 35])
```

Tehtana aritmetična sredina

V nekaterih primerih želimo, da imajo vrednosti, iz katerih računamo aritmetično sredino, različno "težo" oz. vpliv na končni izračun. Tehtano aritmetično sredino izračunamo po enačbi:

$$M = \frac{\sum w_k X_k}{\sum w_k} \tag{4}$$

pri čemer je w_k utež za *k*-to vrednost. Za izračun tehtane aritmetične sredine bomo uporabili funkcijo `weighted.mean()`. Funkcija zahteva vnos vsaj dveh argumentov, in sicer na prvo mesto vstavimo vektor z vrednostmi za izračun (tehtane) aritmetične sredine, na drugo mesto pa vektor s pripadajočimi utežmi (vektorja morata imeti enako dolžino!).

```
# primer izračuna tehtane aritmetične sredine v primeru, ko imamo več
# aritmetičnih sredin na neki spremenljivki (nimamo dostopa do surovih
# podatkov), pridobljenih na različno velikih vzorcih (uteži so v tem primeru
# numerusi skupin)

Mk <- c(22.5,33.2,24.1,29.8)
wk <- c(122,433,255,219)
M_teht <- weighted.mean(Mk, wk)
```

Geometrična sredina

Geometrično sredino izračunamo po enačbi:

$$G = \sqrt[n]{X_1 X_2 X_3 \dots X_n} \tag{5}$$

Za izračun geometrične sredine bomo uporabili funkcijo `geometric.mean()` iz paketa psych.

```
# primer izračuna geometrične sredine oz. povprečnega indeksa prek določenega
# časovnega obdobja (na voljo imamo tri indekse)

indeksi <- c(1.32,0.95,1.55,2.09)
G <- geometric.mean(indeksi)
```

Robustne mere centralne tendence

Pri izračunu aritmetične sredine vsaka vključena vrednost prispeva k vsoti, ki predstavlja osnovo za njen izračun, zato lahko vrednosti, ki zelo odstopajo od večine vrednosti spremenljivke (t. i. osamelci ali ekstremne vrednosti), močno spremenijo njeno vrednost (pravimo, da je aritmetična sredina "zelo občutljiva na ekstremne vrednosti"). Če se na določeni spremenljivki pojavijo ekstremne vrednosti (ali pa ima spremenljivka zelo asimetrično porazdelitev), običajna aritmetična sredina ni nujno najprimernejša mera centralne tendence. V takih primerih raje uporabimo katero od robustnih mer centralne tendence. Tovrstnih mer je zelo veliko, zato bomo spoznali le najpogosteje uporabljane:

- **mediana**; mediana je kot kvantil določena ali z vrednostjo sredinske vrednosti v ranžirni vrsti ali z vrednostima, ki ležita tik pod/nad sredino ranžirne vrste
- **prirezana aritmetična sredina**; običajno 5-odstotno prirezana *M*, kar pomeni, da na vsaki strani porazdelitve iz izračuna izločimo 5 % največjih in 5 % najmanjših vrednosti
- **Winsorizirana aritmetična sredina**; podobna mera kot prirezana *M*, le da izbranega odstotka največjih oz. najmanjših vrednosti ne izločimo iz izračuna, temveč jih zamenjamo z vrednostmi, ki ustrezajo točki, kjer bi "odrezali" največje/najmanjše vrednosti

```
# izračun 5-odstotno prirezane aritmetične sredine; funkciji mean() dodamo
# argument trim (tolikšen delež vrednosti "odrežemo" na desni in levi strani
# porazdelitve)

M_prirez <- mean(po$var, trim  = 0.05)

# izračun Winsorizirane aritmetične sredine, pri čemer bomo "zamenjali"
# zgornjih 10 % in spodnjih 10 % podatkov, torej računamo 80-odstotno
# Winsorizirano aritmetično sredino; uporabimo funkcijo winsor.mean() iz
# paketa psych

Mw <- winsor.mean(po$var, trim = 0.1)
```

Mere razpršenosti

Variacijski razmik

Za razmeroma natančno izmerjene vrednosti zveznih spremenljivk variacijski razmik (*VR*) izračunamo kot razliko med največjo in najmanjšo vrednostjo, za diskretne (in nenatančno, običajno na celo vrednost izmerjene zvezne spremenljivke) pa tej razliki prištejemo 1. V prvem primeru variacijski razmik predstavlja razdaljo med vrednostima (ali razdaljo med sredinama dveh ekstremnih diskretnih vrednosti), v drugem primeru pa število vrednosti, ki jih (lahko) vsebuje izračunan razmik/razpon. Npr., pri vrednostih 3, 4, 6 bo prvi izračun vrnil *VR* = 3 (torej razdalja med sredinama vrednosti 3 in 6), v drugem primeru pa bo VR znašal 4; toliko vrednosti se lahko pojavi od (vključno) najmanjše do (vključno) največje vrednosti. Čeprav je variacijski razmik definiran kot razlika (torej izračunamo eno vrednost), običajno ločeno poročamo o najmanjši (minimum) in največji (maksimum) vrednosti, saj lahko bralec na osnovi *min* in *max* sam izračuna *VR*. Npr., če ločeno poročamo o *min* in *max* za izmerjeno telesno višino udeležencev (npr. *min* = 148 cm, *max* = 198 cm), izvemo, da izmerjene vrednosti zasedajo razpon, ki lahko vsebuje 51 vrednosti, hkrati pa imamo vpogled v položaj *VR* na kontinuumu dolžine (v tem primeru telesne višine, položaj *VR* je torej med 148 in 198 cm).

```
# za izračun VR bomo uporabili funkcijo range(), ki vrne najmanjšo in
# največjo vrednost

VR <- range(po$var)
```

Interkvartilni razmik in odklon

Interkvartilni razmik izračunamo kot razliko med Q_3 in Q_1, interkvartilni odklon pa kot polovico interkvartilnega razmika. Podobno kot pri *VR*, tudi pri poročanju o interkvartilnem razmiku raje ločeno poročamo o Q_1 in Q_3.

Povprečni absolutni odklon od aritmetične sredine

Vse mere, ki so določen centralni moment (v brezdimenzionalni obliki), temeljijo na odklonu od aritmetične sredine, tj. X – *M*. Ker vsota odklonov od aritmetične sredine znaša 0, je potrebno uporabiti eno od pretvorb, ki vodijo do neničelne vsote odklonov od aritmetične sredine. V primeru povprečnega absolutnega odklona od aritmetične sredine (PO_M) upoštevamo le velikost, ne pa tudi predznaka odklonov (upoštevamo torej absolutne vrednosti odklonov). PO_M izračunamo po enačbi:

$$PO_M = \frac{\sum |X - M|}{N} \tag{6}$$

PO_M nam sporoča povprečno oddaljenost podatkov od aritmetične sredine. V Rju ni na voljo funkcije za PO_M, zato ga bomo izračunali s pomočjo funkcij `mean()` in `abs()`. Funkcija `abs()` vrne absolutne vrednosti za podane podatke.

```
# izračun povprečnega absolutnega odklona od aritmetične sredine

mean(abs(po$var - mean(po$var)))

# izračun povprečnega absolutnega odklona od aritmetične sredine za določen
# podvzorec (npr. samo za moške)

mean(abs(po$var[po$spol == "M"] - mean(po$var[po$spol == "M"])))
```

Standardni odklon/deviacija in varianca

Standardni odklon in varianco izračunamo po enačbah:

$$\sigma = \sqrt{\frac{\sum(X - M)^2}{N}} \quad ALI \quad \sigma' = \sqrt{\frac{\sum(X - M)^2}{N - 1}} \tag{7, 8}$$

$$\sigma^2 = \frac{\sum(X - M)^2}{N} \quad ALI \quad \sigma'^2 = \frac{\sum(X - M)^2}{N - 1} \tag{9, 10}$$

Varianca je povprečni kvadriran odklon od aritmetične sredine, standardni odklon pa kvadratni koren variance oz. povprečnega kvadriranega odklona od aritmetične sredine. Interpretacija obeh mer je nekoliko olajšana v primeru, če se porazdelitev spremenljivke dobro prilega normalni porazdelitvi (več o tem v poglavju o normalni porazdelitvi).

Varianco in standardni odklon lahko izračunamo na dva načina (primerjaj enačbi 7 in 9 ter 8 in 10). Enačbi 7 in 9 (z *N* v imenovalcu) sta meri razpršenosti za celotno populacijo. Če ju izračunamo na vzorcu iz določene populacije, predstavljata *pristranski oceni populacijske variance in standardnega odklona*. Če želimo izračunati *nepristranski oceni populacijske SD in SD*^{2}, uporabimo enačbi 8 in 10 (z *N* – 1 v imenovalcu). V Rju imamo na voljo funkciji `var()` in `sd()`, ki izračunata nepristranski populacijski oceni.

```
# izračun standardnega odklona in variance (z N - 1 v imenovalcu)

sd(po$var)
var(po$var)

# izračun standardnega odklona določen podvzorec (npr. samo za ženske in
# tiste, ki imajo na spremenljivki dosezek vrednost, manjšo od 35)

sd(po$var[po$spol == "Ž" & po$dosezek < 35])
```

Koeficient variacije

Vse mere, ki smo jih spoznali do tega trenutka, predstavljajo absolutne mere razpršenosti. Če želimo primerjati razpršenost dveh (ali več) spremenljivk, ki imajo ali drugačno enoto ali pa isto enoto, a drugačno aritmetično sredino, si z absolutnimi merami razpršenosti ne moremo pomagati. V tovrstnih primerih moramo za take spremenljivke izračunati relativno mero razpršenosti. Ena od takih mer je koeficient variacije (*KV*), ki ga izračunamo po enačbi:

$$KV = \frac{SD}{M} (* 100\ \%) \tag{11}$$

Pri koeficientu variacije torej umerimo standardni odklon neke spremenljivke na enoto aritmetične sredine te spremenljivke. *KV* nam torej pove, kolikšen delež/odstotek (ki je lahko večji od 1 oz. 100 %!) aritmetične sredine predstavlja standardni odklon neke spremenljivke. Koeficient variacije lahko računamo samo na razmernostnih spremenljivkah!

```
# izračun koeficienta variacije za telesno težo udeležencev; KV opredelimo
# kot odstotek SD glede na M (zato množimo s 100)

100 * sd(po$teza) / mean(po$teza)
```

Robustne mere razpršenosti

Podobno kot pri merah centralne tendence, poznamo tudi mnogo različnih robustnih mer razpršenosti. Spoznali bomo nekatere pogosteje uporabljane:

- **interkvartilni razmik** (ali odklon); lahko tudi interdecilni razmik ($D_9 - D_1$)
- **MAD** (angl. *median absolute deviation*); mediana absolutnih odklonov od mediane (ali aritmetične sredine)
- **Winsoriziran standardni odklon** (ali varianca)

```
# izračun MAD v osnovni obliki, torej kot enostavna mediana absolutnih
# odklonov od mediane

MAD <- mad(po$var, constant = 1)

# izračun MAD kot konsistentno oceno populacijskega standardnega odklona
# normalno porazdeljene spremenljivke; izpustimo argument "constant", s čimer
# konstanti priredimo privzeto vrednost oz. lestvični faktor za normalno
# porazdelitev (lestvični faktor izračunamo kot obratno/recipročno vrednost
# standardiziranega tretjega kvartila izbrane teoretične porazdelitve; v
# primeru normalne porazdelitve ta lestvični faktor znaša približno 1.4826)

MAD_SD <- mad(po$var)

# izračun winsoriziranega standardnega odklona (in variance), pri čemer bomo
# "zamenjali" zgornjih 5 % in spodnjih 5 % podatkov; računamo 90-odstotno
# winsoriziran standardni odklon (in varianco); uporabimo funkciji
# winsor.sd() in winsor.var() iz paketa psych

winsor.sd(po$var, trim = 0.05)
winsor.var(po$var, trim = 0.05)
```

Mere oblike porazdelitve

Asimetričnost porazdelitve

Na voljo je mnogo mer asimetričnosti, spoznali pa bomo najpogosteje uporabljano mero, ki jo izračunamo kot tretji centralni moment v brezdimenzionalni obliki:

$$As = a_3 = \frac{\sum(X - M)^3}{N \cdot SD^3} = \frac{\sum z^3}{N} \tag{12}$$

Pri izračunu tretjega centralnega momenta v brezdimenzionalni obliki se predznaki odklonov ohranijo. Zaradi tega bo a_3 za simetrično porazdelitev znašala 0, za levo (ali negativno) asimetrično manj kot 0, v primeru desne (ali pozitivne) asimetričnosti pa več kot 0. Možen razpon vrednosti koeficienta asimetričnosti znaša od –∞ do +∞.

Interpretacija asimetričnosti je zapletena, a v večini primerov za levo (negativno) asimetričnost velja, da je levi rep porazdelitve daljši (ali "debelejši") v primerjavi z desnim (vsota negativnih odklonov na tretjo potenco je večja od vsote pozitivnih odklonov na tretjo potenco). Za desno (pozitivno) asimetričnost velja ravno obratno.

Koeficient asimetričnosti, izračunan po enačbi 12, predstavlja pristransko oceno populacijske asimetričnosti spremenljivke. Nepristranske ocene populacijske asimetričnosti lahko izračunamo na več različnih načinov, zato bomo v Rju uporabljali izračun, ki vrne isto vrednost kot sorodne funkcije v drugih statističnih paketih (npr. SPSS-u ali funkcija SKEW v Excelu).

```
# izračun koeficienta asimetričnosti z uporabo funkcije skew() iz paketa
# psych; za primerljivost vrnjene vrednosti s tistimi, ki jih dobimo v Excelu
# (SKEW), bomo argumentu type priredili vrednost 2 (privzeta vrednost je 3)

As <- skew(po$var, type = 2)

# če želimo izračunati koeficient asimetričnosti na določenem podvzorcu oseb
# (npr. samo za moške), to naredimo z indeksiranjem

skew(po$var[po$spol == "M"], type = 2)
```

Sploščenost porazdelitve

Tudi za sploščenost je na voljo več koeficientov, zato bomo ponovno spoznali najpogosteje uporabljano mero, in sicer (presežni) četrti centralni moment v brezdimenzionalni obliki:

$$Spl = a_4 - 3 = \frac{\sum(X - M)^4}{N \cdot SD^4} - 3 = \frac{\sum z^4}{N} - 3 \tag{13}$$

Za normalno porazdelitev a_4 znaša 3, zato od dobljene vrednosti odštejemo 3. Na ta način dobimo mero, katere ničelna vrednost (tako kot pri asimetričnosti) ustreza vrednosti, ki velja za normalno porazdelitev (N.P.). Možen razpon vrednosti koeficienta sploščenosti znaša od –2 do +∞. Porazdelitve lahko glede vrednosti na koeficientu sploščenosti razvrstimo v tri skupine:

- *Spl* = 0 **mezokurtična** porazdelitev (velja za normalno porazdelitev)
- –2 ≤ *Spl* < 0 **platikurtična** (sploščena) porazdelitev
- *Spl* > 0 **leptokurtična** (koničasta ali "debelorepa") porazdelitev

Interpretacija sploščenosti je še nekoliko bolj zapletena kot interpretacija asimetričnosti, a v splošnem lahko rečemo, da za platikurtične (sploščene) porazdelitve velja, da imajo bolj širok in sploščen vrh ter krajše repe kot N.P., za leptokurtične (koničaste) pa, da imajo (običajno) ožji vrh in predvsem "debelejše" repe kot N.P. (zato takim porazdelitvam velikokrat rečemo "debelorepe porazdelitve"; angl. *heavy-* ali *fat-tailed distributions*).

Koeficient sploščenosti, izračunan po enačbi 13, predstavlja pristransko oceno populacijske sploščenosti. Tako kot v primeru koeficienta sploščenosti, imamo tudi pri sploščenosti na voljo več različnih izračunov nepristranske populacijske ocene, in ponovno bomo v Rju uporabili izračun, ki vrne isto vrednost kot sorodne funkcije v drugih statističnih paketih (npr. v SPSS-u ali funkcija KURT v Excelu).

```
# izračun koeficienta sploščenosti z uporabo funkcije kurtosi() iz paketa
# psych; za primerljivost vrnjene vrednosti s tisto, ki jo dobimo v Excelu
# (KURT), bomo argumentu type priredili vrednost 2 (privzeta vrednost je 3)

Spl <- kurtosi(po$var, type = 2)

# če želimo izračunati koeficient sploščenosti na določenem podvzorcu oseb
# (npr. samo za moške), to naredimo z indeksiranjem

kurtosi(po$var[po$spol == "M"], type = 2)
```

Funkcije za izračun več opisnih statistik hkrati

V Rju imamo na voljo tudi funkcije, s katerimi lahko izračunamo več opisnih statistik hkrati. Primeru tovrstnih funkcij so `summary()`, `describe()` in `describeBy()`; slednji funkciji sta iz paketa psych.

```
# izračun opisnih statistik s funkcijo summary(); summary vrne min in max,
# vse kvartile in aritmetično sredino; kvartili so izračunani na
# najenostavnejši način in ne ustrezajo vrednostim, dobljenim z linearno
# interpolacijo!

summary(po$var)
```

Funkcija `summary()` ima tudi argument digits, s katerim nadziramo natančnost izračunanih opisnih statistik. Argument se **ne** nanaša na število decimalnih mest, temveč na število vseh cifer v vrednosti. Npr., če bomo argumentu digits priredili vrednost 5, bo neka aritmetična sredina (ali katera druga statistika) izpisana npr. kot 25.327 (vrednost ima pet cifer, tj. 5 pomembnih mest).

```
# če želimo uporabiti funkcijo summary() samo na podvzorcu udeležencev, to
# storimo z indeksiranjem (npr. samo za ženske udeleženke); v tem primeru smo
# vrednostim določili tudi želeno natančnost (digits)

summary(po$var[po$spol == "Ž"], digits = 5)

# izračun opisnih statistik s funkcijo describe() iz paketa psych; funkcija
# vrne N, M, SD, Mdn, (privzeto 10-odstotno) prirezano M, MAD, min, max, VR
# (brez +1!), As, Spl in standardno napako za M; v spodnjem primeru smo
# argumentu trim spremenili privzeto vrednost oz. zapisali svojo (0.05) in za
# izračun As in Spl argumentu type priredili vrednost 2

describe(po$var, trim = 0.05, type = 2)
```

Funkcija `describe()` omogoča tudi izračun kvantilov, in sicer z uporabo argumenta quant, npr. `quant = c(0.25, 0.75)`. Funkcija ima tudi argument interp (s privzeto vrednostjo FALSE), ki se nanaša na to, ali želimo mediano izračunati z interpolacijo; ta interpolacija je identična tisti v funkciji `interp.quantiles()`, torej vrne nekoliko drugačno vrednost kot postopek s funkcijo `approx()`. Argument interp se nanaša na izračun mediane, ne pa tudi na izračun kvantilov prek argumenta quant (ti bodo vedno izračunani na najenostavnejši način)!

```
# izračun opisnih statistik na podvzorcu udeležencev lahko s funkcijo
# describe() izvedemo z indeksiranjem (kot pri summary), lahko pa uporabimo
# funkcijo describeBy(), ki ima dodaten argument za določitev spremenljivke,
# ki določa podskupine (argument group; spremenljivko vstavimo na drugo
# mesto, takoj za objektom, za katerega računamo opisne statistike); če bomo
# uporabili spol kot spremenljivko za grupiranje, bo describeBy izračunala
# opisne statistike za vse skupine hkrati

describeBy(po$var, po$spol, type = 2, digits = 6)

# podvzorce udeležencev lahko določimo tudi kot kombinacije več
# spremenljivk, pri čemer na položaj argumenta group vstavimo seznam
# spremenljivk za grupiranje; v spodnjem primeru smo skupine določili kot
# kombinacije spremenljivk spol in starost2 (vsebuje samo dve vrednosti za
# dve starostni skupini, npr. mlajši, starejši); funkcija describeBy bo
# vrnila opisne statistike za vse štiri kombinacije: Ž/mlajši, Ž/starejši,
# M/mlajši in M/starejši

describeBy(po$var, list(po$spol,po$starost2), type = 2)
```

Funkcije, ki smo jih spoznali v poglavju Mere centralne tendence, razpršenosti, oblike porazdelitve

```
abs()                # vrne absolutne vrednosti
describe()           # izračun najpogosteje uporabljanih opisnih statistik;
                     # paket psych
describeBy()         # hkratni izračun najpogosteje uporabljanih opisnih
                     # statistik za različne skupine udeležencev; paket
                     # psych
geometric.mean()     # izračun geometrične sredine
kurtosi()            # izračun koeficienta sploščenosti; paket psych
mad()                # mediana absolutnih odklonov od mediane
mean()               # izračun aritmetične sredine
range()              # izračun variacijskega razmika
sd()                 # izračun nepristranske ocene populacijske SD
skew()               # izračun koeficienta asimetričnosti; paket psych
summary()            # izračun nekaterih opisnih statistik
var()                # izračun nepristranske ocene populacijske variance
weighted.mean()      # izračun tehtane aritmetične sredine
which.max()          # vrne indeks največje vrednosti podanega objekta; če
                     # funkciji podamo tabelo (rezultat funkcije table),
                     # vrne vrednost z največjo frekvenco in indeks te
                     # vrednosti; obratno velja za which.min()
winsor.mean()        # izračun winsorizirane aritmetične sredine
winsor.sd()          # winsorizirana SD; paket psych
winsor.var()         # winsorizirana varianca; paket psych
```

Normalna porazdelitev

Normalna porazdelitev (N.P.) je zvezna verjetnostna porazdelitev, ki jo v psihologiji in drugih znanostih pogosto uporabljamo kot matematični model za porazdelitev velikega števila spremenljivk. Spremenljivka se bo porazdeljevala po N.P. v primeru, če na vrednosti te spremenljivke vpliva veliko število neodvisnih dejavnikov (enostavna ponazoritev tega principa je Galtonova naprava *bean machine* (ali *quincunx*) , npr. https://en.wikipedia.org/wiki/Bean_machine). Normalna porazdelitev je limita binomske porazdelitve, je unimodalna, simetrična in mezokurtična.

Funkcija **gostote verjetnosti** (angl. *probability density function*; PDF) za normalno porazdelitev je:

$$f(x|\mu,\sigma^2) = \frac{1}{\sqrt{\sigma^2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{14}$$

pri čemer je *x* vrednost zvezne naključne spremenljivke, μ aritmetična sredina (oz. pričakovana vrednost) in σ^2 varianca te spremenljivke.

Funkcija gostote verjetnosti za standardizirano normalno porazdelitev je:

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2} \tag{15}$$

Za standardizirano normalno porazdelitev velja $\mu = 0$ in $\sigma^2 = 1$ ali $\sigma = 1$.

Funkcija gostote verjetnosti za *x* (ali *z*) pri določeni aritmetični sredini in varianci vrne relativno verjetnost, da bo imela zvezna naključna spremenljivka točno določeno vrednost iz množice možnih vrednosti (tj. *x* ali *z*). V primeru diskretnih naključnih spremenljivk (oz. diskretnih verjetnostih porazdelitev) bomo v ta namen uporabili verjetnostno masno funkcijo (angl. *probability mass function*; PMF), včasih imenovano zgolj funkcija verjetnosti.

Integral funkcije gostote verjetnosti prek določenega intervala vrednosti imenujemo **(kumulativna) porazdelitvena funkcija** (angl. *cumulative distribution function*; CDF). S porazdelitveno funkcijo bomo ocenjevali verjetnost, da vrednosti naključne spremenljivke ležijo v določenem intervalu. To funkcijo bomo torej uporabljali za **določitev percentilnih rangov** za vrednosti normalno porazdeljenih spremenljivk; npr. kolikšen delež podatkov leži pod/nad/med določenimi vrednostmi.

Inverzno porazdelitveno funkcijo [$\Phi^{-1}(p)$] **imenujemo kvantilna funkcija** (v primeru N.P. jo imenujemo tudi *probit funkcija*). S kvantilno funkcijo izračunamo določen kvantil normalne porazdelitve, torej vrednost, ki leži nad (ali pod) določeno površino pod normalno porazdelitvijo. To funkcijo bomo torej uporabljali za **določitev kvantilov** v normalno porazdeljenih spremenljivkah; npr. koliko znaša 35. percentil v normalno porazdeljeni spremenljivki (pri določeni aritmetični sredini in standardnem odklonu)?

Preverjanje prileganja porazdelitve spremenljivke normalni porazdelitvi

Odločitev o tem, ali se določena empirična porazdelitev prilega normalni porazdelitvi, sprejmemo z upoštevanjem rezultatov več različnih postopkov.

Preverjanje prileganja normalni porazdelitvi na podlagi opisnih statistik

- Primerjava *Mo*, *Mdn*, *M*; za levo asimetrično porazdelitev bo običajno (ne nujno!) veljalo $M < Mdn < Mo$, za desno asimetrično pa $M > Mdn > Mo$.
- Pregled vrednosti koeficientov asimetričnosti in sploščenosti.

Preverjanje prileganja normalni porazdelitvi z grafičnimi metodami

- Narišemo **histogram ali histogram z vrisano normalno krivuljo**. Histogram z vrisano normalno krivuljo bomo v Rju najhitreje narisali tako, da bomo najprej narisali histogram, pri čemer bomo na y-osi prikazali gostote verjetnosti namesto frekvenc:

```
hist(po$var, freq = FALSE)
```

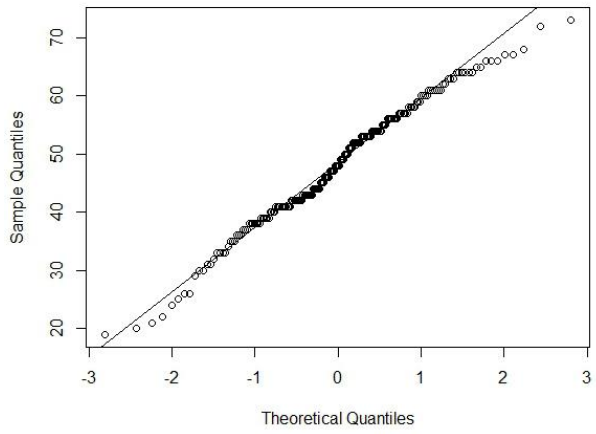
nato pa temu histogramu dodamo normalno krivuljo, za izris katere bomo potrebovali funkcije `lines()`, `seq()` in `dnorm()`; za slednjo funkcijo glej naslednje poglavje o določanju položaja podatka v N.P. S funkcijo `seq()` ustvarimo zaporedje vrednosti, in sicer tako, da ji podamo argumente from (začetna vrednost), to (končna vrednost) in length.out (število vrednosti med from in to).

```
# najprej ustvarimo večje število vrednosti za dano x-os (npr. 100 ali
# 1000), da bo normalna krivulja zadostno zglajena, in sicer v razponu
# vrednosti spremenljivke, za katero smo narisali histogram

x <- seq(min(po$var), max(po$var), length.out = 1000)

# v naslednjem koraku ustvarimo vrednosti za y-os (enako število kot za
# x!), pri čemer uporabimo funkcijo dnorm(), s katero za vse vrednosti
# v vektorju x izračunamo verjetnost, da bo imela uporabljena (zvezna)
# spremenljivka točno določeno vrednost iz množice možnih vrednosti;
# funkciji dnorm() moramo podati vektor x ter aritmetično sredino in
# standardni odklon obravnavane spremenljivke
# v zadnjem koraku s funkcijo lines() dodamo krivuljo na predhodno
# narisan histogram

y <- dnorm(x, mean(po$var), sd(po$var))
lines(x, y)
```

- Narišemo ***Q-Q* graf** (angl. *Q-Q plot*) za primerjavo empiričnih in teoretičnih kvantilov. Na *Q-Q* grafu prikažemo točke, katerih koordinate so določene z dejanskimi (empiričnimi) kvantili in teoretičnimi kvantili, ki izhajajo iz normalne porazdelitve z *M* in *SD* obravnavane spremenljivke. Če se spremenljivka popolnoma prilega normalni porazdelitvi, točke ležijo na premici y = x. V Rju uporabimo funkcijo `qqnorm()`.

```
# najprej narišemo Q-Q graf s funkcijo qqnorm(), nato pa na ta
# (nazadnje narisan graf!) dodamo premico y = x (funkcija qqline)

qqnorm(po$var)
qqline(po$var)
```

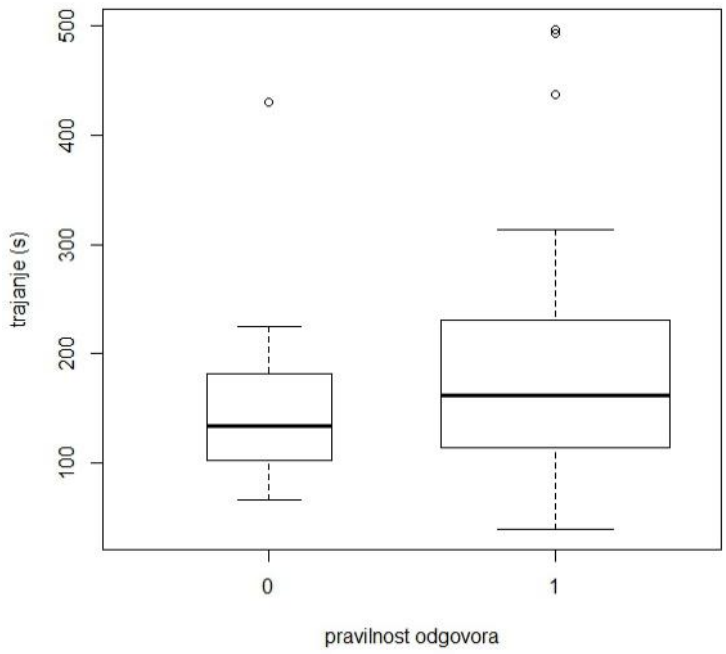
Spodaj je prikazan primer Q-Q grafa.



- Narišemo **boxplot** (oz. "prikaz zaboja z ročaji"). Odebeljena črta v zaboju predstavlja mediano, zgornji rob zaboja tretji kvartil, spodnji rob zaboja pa prvi kvartil (zaboj je torej Q_3–Q_1). Ročaja lahko predstavljata različne vrednosti, in sicer:
    - Če za največjo vrednost spremenljivke (max) velja max < Q_3 + 1,5 * (Q_3–Q_1), potem se zgornji ročaj konča pri največji vrednosti. Podobno velja za končno točko spodnjega ročaja; če je min > Q_1 – 1,5 * (Q_3–Q_1), se spodnji ročaj konča pri najmanjši vrednosti.
    - Če je max > Q_3 + 1,5 * (Q_3–Q_1), se bo zgornji ročaj končal pri največji vrednosti, za katero še velja, da je manjša od Q_3 + 1,5 * (Q_3–Q_1). Vse večje vrednosti bodo prikazane kot krogci (osamelci/vplivne točke, ekstremne vrednosti …). Če je min < Q_1 – 1,5 * (Q_3–Q_1), se bo spodnji ročaj končal pri najmanjši vrednosti, za katero še velja, da je večja od Q_1 – 1,5 * (Q_3–Q_1). Vse manjše vrednosti bodo prikazane kot krogci.

```
# risanje boxplota s funkcijo boxplot()

boxplot(po$var, ylab = "dosežek na var")

# na istem grafu prikažemo več zabojev z ročaji za različne podvzorce
# udeležencev (npr. ločeno po spolu), hkrati pa želimo s širino zabojev
# sporočiti velikost podvzorcev oz. približno prikazati natančnost ocen
# kvantilov v določeni skupini (z argumentom varwidth = TRUE; širina
# zaboja je izračunana kot kvadratni koren iz numerusa skupine)

boxplot(po$var~po$spol, varwidth = TRUE, ylab = "dosežek na var",
        xlab = "spol")
```

Spodaj je prikazan primer boxplota za dve skupini udeležencev (širina zabojev sporoča velikost skupin oz. natančnost ocen kvantilov v določeni skupini).



Preverjanje prileganja normalni porazdelitvi s statističnimi testi

- Obstaja precejšnje število različnih statističnih testov za preverjanje prileganja normalni porazdelitvi, zato bomo spoznali enega od pogosteje uporabljanih, in sicer **Shapiro-Wilkov test normalnosti**.

```
# testiranje prileganja normalni porazdelitvi s Shapiro-Wilkovim testom

shapiro.test(po$var)
```

Določanje položaja podatka/osebe v normalni porazdelitvi

V tem poglavju bomo spoznali postopke za določanje položaja podatka/posameznika v normalno porazdeljenih spremenljivkah. Za izračun percentilnih rangov in kvantilov bomo v tem primeru potrebovali le podatek o *M* in *SD* obravnavane normalno porazdeljene spremenljivke.

Izračun standardiziranega dosežka (*z*-vrednost)

Položaj podatka lahko določamo ali v nestandardizirani ali standardizirani porazdelitvi. Standardizirano porazdelitev sestavljajo standardizirane vrednosti oz. *z*-vrednosti, ki jih izračunamo po enačbi:

$$z = \frac{X - M}{SD} \tag{16}$$

Standardizirani dosežek izračunamo kot odklon tega dosežka/podatka od aritmetične sredine v enotah *SD* spremenljivke in nam torej pove, za koliko standardnih odklonov je določen podatek manjši ali večji od aritmetične sredine [*M*(*z*) = 0].

```
# izračun z-vrednosti v Rju (po enačbi z uporabo funkcij mean in sd)

z <- (po$var - mean(po$var)) / sd(po$var)
```

Standardizirane dosežke lahko izračunamo tudi s funkcijo `scale()`. Funkcija vključuje tri argumente, in sicer ime spremenljivke ali objekta z več spremenljivkami, argument center (ali TRUE/FALSE ali vektor vrednosti, ki jih odštejemo od vrednosti podanih spremenljivk) ter argument scale (ali TRUE/FALSE ali vektor vrednosti, s katerimi delimo vrednosti podanih spremenljivk). Če argumentu center priredimo vrednost TRUE, bo funkcija od vseh vrednosti podane spremenljivke odštela aritmetično sredino te spremenljivke (t. i. centriranje spremenljivke). Če argumentu scale priredimo vrednost TRUE, bo funkcija vse vrednosti spremenljivke delila s standardnim odklonom te spremenljivke. Torej, če argumentoma center in scale priredimo logični vrednosti TRUE, bo funkcija vrnila *z*-vrednosti za podane spremenljivke.

```
# izračun z-vrednosti s funkcijo scale(); argumenta center in scale imata
# privzeti vrednosti TRUE, zato ju lahko izpustimo

z <- scale(po$var)
```

Percentilni rangi

Pri določanju percentilnih rangov (kolikšen delež podatkov leži pod/nad določeno vrednostjo) bomo uporabili kumulativno porazdelitveno funkcijo za normalno porazdelitev (funkcija `pnorm()` v Rju, privzeti vrednosti argumentov mean in sd sta 0 in 1).

```
# izračun percentilnega ranga (delež pod vrednostjo) na primeru
# standardizirane (za z = 1,2) in nestandardizirane N.P. (za X = 142,8 pri
# M = 150, SD = 20)

pnorm(1.2)
pnorm(142.8, mean = 150, sd = 20)

# izračunamo lahko tudi delež med dvema surovima ali standardiziranima
# vrednostma, npr. med z-vrednostma 0,5 in 1,5

pnorm(1.5) - pnorm(0.5)
```

```
# če želimo izračunati delež podatkov nad določeno vrednostjo, lahko
# spremenimo (privzeto) vrednost argumenta lower.tail v FALSE (ali pa od 1
# odštejemo rezultat pnorm s privzeto vrednostjo lower.tail argumenta); npr.
# delež podatkov nad z-vrednostjo −0,85

pnorm(-0.85, lower.tail = FALSE)

# ali (privzeta vrednost argumenta lower.tail je TRUE)

1 - pnorm(-0.85)
```

Do sedaj prikazani postopki so primerni le za (razmeroma natančno izmerjene) zvezne spremenljivke. Za določanje percentilnih rangov nezveznih spremenljivk (pa tudi nenatančno izmerjenih zveznih, kot npr. točke na maturi ali testu znanja) bi morali uporabiti binomsko porazdelitev, a lahko pogojno uporabimo tudi (zvezno) normalno porazdelitev. Pri zveznih spremenljivkah lahko v kontekstu izračunavanja površine pod N.P. zastavljamo vprašanje o deležu pod ali nad neko vrednostjo, pri nezveznih (ali zveznih, ki jih obravnavamo kot nezvezne), pa lahko izvemo, kolikšen delež podatkov je:

- manjših od neke vrednosti,
- manjših ali enakih kot neka vrednost,
- večjih od neke vrednosti,
- večjih ali enakih kot neka vrednost ter
- enakih kot neka vrednost.

Pri vseh zgornjih vprašanjih moramo upoštevati, da so posamezne vrednosti intervali vrednosti s spodnjo in zgornjo mejo (npr. dosežek na testu znaša 25, pri čemer njegova spodnja meja znaša 24,5, zgornja meja pa 25,5).

```
# izračun površine pod/nad normalno porazdelitvijo za nezvezno porazdelitev
# (primer porazdelitve z M = 50 in SD = 10)
# 1. primer: kolikšen delež posameznikov ima dosežek nižji od 36?
# 2. primer: kolikšen delež posameznikov ima dosežek enak ali višji od 55?
# 3. primer: kolikšen delež posameznikov ima dosežek enak 63?

pnorm(35.5, mean = 50, sd = 10)
pnorm(54.5, mean = 50, sd = 10, lower.tail = FALSE)
pnorm(63.5, mean = 50, sd = 10) - pnorm(62.5, mean = 50, sd = 10)
```

Kvantili

Pri določanju kvantilov bomo uporabili kvantilno funkcijo oz. inverzno porazdelitveno funkcijo; funkcija `qnorm()` v Rju; privzeti vrednosti argumentov mean in sd sta 0 in 1.

```
# izračun sedmega decila standardizirane normalne porazdelitve (funkcija vrne
# z-vrednost)

qnorm(0.7)

# izračun 45. percentila v normalni porazdelitvi z M = 50 in SD = 10

qnorm(0.45, mean = 50, sd = 10)
```

Funkcije, ki smo jih spoznali v poglavju Normalna porazdelitev

```
boxplot()        # risanje boxplota (prikaz zabojev z ročaji)
dnorm()          # funkcija gostote verjetnosti za N.P.
lines()          # dodajanje črtnega grafa na predhodno ustvarjen graf
pnorm()          # kumulativna porazdelitvena funkcija za N.P.
qnorm()          # kvantilna funkcija za N.P.
qqline()         # dodajanje premice y = x na Q-Q plot
qqnorm()         # risanje Q-Q plota
scale()          # centriranje ali standardiziranje vrednosti spremenljivk(e)
seq()            # generiranje aritmetičnih zaporedij
shapiro.test()   # Shapiro-Wilkov test normalnosti porazdelitve
```

Korelacija (Pearsonov *r*) in bivariatna linearna regresija

Doslej smo se ukvarjali s statistikami, ki se nanašajo na eno spremenljivko (t. i. univariatne statistike), v tem poglavju pa bomo spoznali nekatere statistike in postopke, ki se nanašajo na odnos med dvema spremenljivkama (t. i. bivariatne statistike oz. postopki). Ukvarjali se bomo z različnimi koeficienti korelacije in bivariatno linearno regresijo.

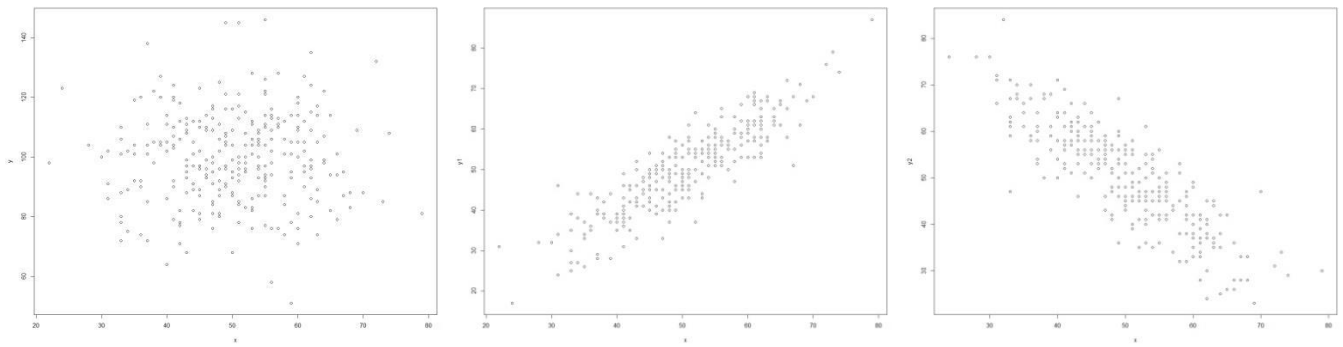
Najpogosteje uporabljan koeficient korelacije je Pearsonov *r*. Linearna regresija in Pearsonov *r* sta neločljivo povezana:

- **Bivariatna linearna regresija** je namenjena opisu (linearnega) odnosa med dvema spremenljivkama oz. napovedovanju vrednosti ene spremenljivke na podlagi vrednosti druge spremenljivke.
- **Pearsonov koeficient korelacije** uporabljamo za opis velikosti povezanosti dveh spremenljivk, v kontekstu linearne regresije pa višina Pearsonovega *r* odraža natančnost napovedovanja.

Za ustrezno uporabo Pearsonovega koeficienta korelacije moramo biti pozorni predvsem na sledeče:

- Obe spremenljivki morata biti vsaj na **intervalni ravni** (računamo ga lahko tudi v nekaterih drugih primerih, npr., če je ena od spremenljivk nominalna oz. dihotomna spremenljivka; več o tem v poglavju o drugih korelacijskih koeficientih).
- Spremenljivki morata biti v **linearnem odnosu**.
- Pozorni moramo biti na morebitne **osamelce**, saj močno vplivajo na vrednost Pearsonovega *r*.

Korelacija odraža stopnjo (in smer) sovariiranja vrednosti dveh spremenljivk. Vrednost Pearsonovega *r* je lahko **ničelna** (spremenljivki nista povezani oz. sta med seboj povsem neodvisni), **pozitivna** (z višanjem vrednosti ene spremenljivke se [v povprečju] višajo vrednosti druge spremenljivke) in **negativna** (z višanjem vrednosti ene spremenljivke se [v povprečju] nižajo vrednosti druge spremenljivke). Na spodnji sliki so prikazani primeri grafičnih prikazov (od leve proti desni) skoraj ničelne korelacije, visoke pozitivne in visoke negativne korelacije med dvema spremenljivkama. Prikazani razsevni diagrami so narisani s funkcijo `plot()` (glej naslednje poglavje o razsevnih diagramih).



Povezanost oz. sovariiranje vrednosti dveh spremenljivk je lahko posledica vzročno-posledičnega odnosa med tema dvema spremenljivkama, lahko pa povezanost dveh spremenljivk izhaja tudi iz posrednega vpliva neke druge (ali več) spremenljivk(e) na ti dve povezani spremenljivki. Torej, korelacija med dvema spremenljivkama je lahko posledica vzročno-posledičnega odnosa med spremenljivkama, a sam koeficient korelacije ne omogoča sklepanja o vzročno-posledičnih odnosih – **korelacija ne implicira vzročnosti!** Sklepanje o vzročno-posledičnih odnosih omogočajo le meritve, ki sledijo eksperimentalnemu načrtu v strogo nadzorovanih pogojih.

Razsevni diagram

Korelacijo med dvema (vsaj intervalnima) spremenljivkama si je najlažje predstavljati v grafični obliki oz. na grafu, v katerem so prikazane točke, katerih položaj/koordinate so določene s pari vrednosti na x- in y-osi. Tak graf imenujemo razsevni diagram (angl. *scatterplot*).

```
# risanje enostavnega razsevnega diagrama s funkcijo plot(); funkciji lahko
# podamo običajne argumente kot so npr. main, xlab, ylab ipd.

plot(os$var1, os$var2)
```

Včasih želimo hkrati pregledati (in primerjati) več razsevnih diagramov za vse možne kombinacije obravnavanih spremenljivk. V takih primerih narišemo matriko razsevnih diagramov. V Rju lahko v ta namen uporabimo funkcijo `pairs()`. Spremenljivke za matriko lahko funkciji podamo po privzeti metodi ali metodi s formulo. Pri privzeti metodi lahko funkciji podamo cel p.o. (prvi spodnji primer) ali pa le izbrane spremenljivke prek njihovih številskih indeksov (drugi in tretji primer). Pri metodi s formulo lahko funkciji podamo izbrane spremenljivke (njihova imena) po vzoru `~po$var1+po$var2+po$var8+po$var11` (četrti primer). Funkcija vsebuje številne dodatne argumente, s katerimi lahko spreminjamo vrsto prikazov pod/v/nad diagonalo, spreminjamo oblikovne lastnosti ipd.

```
# različni načini risanja matrike razsevnih diagramov s funkcijo pairs()

pairs(po)
pairs(po[4:8])
pairs(po[c(2,6,7,11,22)]
pairs(~po$var2+po$var5+po$var10+po$var17)
```

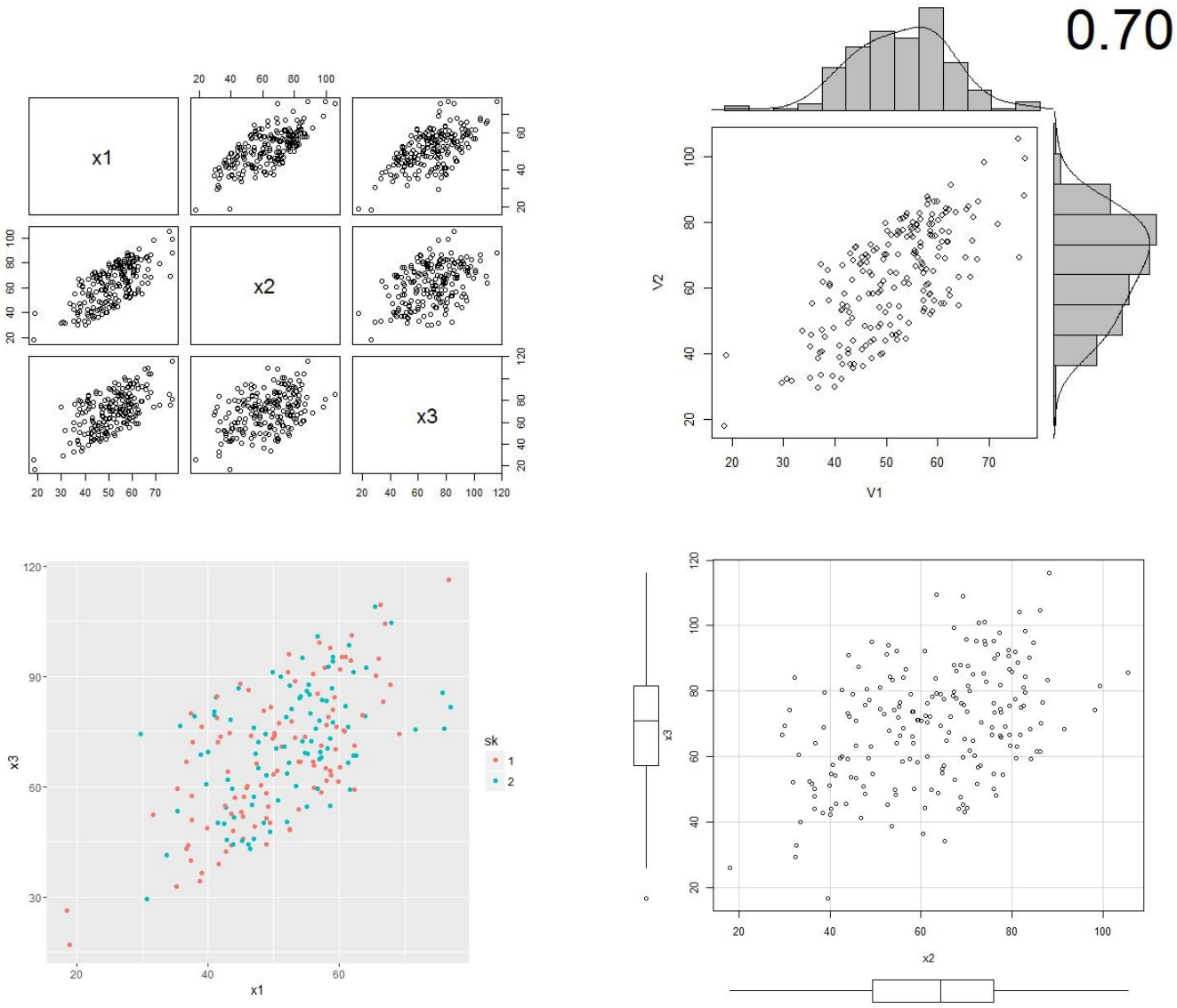
Nekateri paketi vsebujejo funkcije za risanje razsevnih diagramov, ki jih lahko obogatimo z dodatnimi elementi ali informacijami (npr. dorišemo regresijsko premico, boxplote ali histograme za obravnavani spremenljivki, dopišemo vrednosti korelacij na graf, točke udeležencev iz določenih podskupin pobarvamo z različnimi barvami ipd.). Ena od takih funkcij je `scatter.hist()` iz paketa psych.

```
# risanje razsevnega diagrama z dodanima histogramoma (in krivuljama
# ocenjene gostote verjetnosti) za obe spremenljivki; funkcija privzeto
# omogoča dodajanje še nekaterih drugih elementov, ki smo jih v spodnjem
# primeru onemogočili (vsi argumenti, ki smo jim pripisali vrednost FALSE)

scatter.hist(po$var1, po$var2, smooth = FALSE, correl = FALSE,
             ellipse = FALSE, title = NULL)

# s funkcijo qplot() iz paketa ggplot2 lahko točke  udeležencev iz različnih
# podvzorcev pobarvamo z različnimi barvami, in sicer tako, da argumentu
# colour podamo spremenljivko (ki mora biti faktor!), ki predstavlja osnovo
# za tvorbo podvzorcev; v spodnjem primeru smo npr. različno pobarvali točke
# za moške in ženske udeležence

qplot(po$var1, po$var2, geom = "point", colour = po$spol)
```

Nekatere funkcije imajo še dodatne možnosti za risanje enostavnih in matričnih prikazov razsevnih diagramov, npr. funkcija `cor.plot()` iz osnovnega paketa, `corrplot()` iz paketa corrplot, `corrgram()` iz paketa corrgram, `scatterplot()` in `scatterplot.matrix()` iz paketa car.

Spodaj so prikazani primeri:

- matričnega prikaza razsevnih diagramov (funkcija `pairs()`; zgoraj levo);
- razsevnega diagrama z vrisanima histogramoma porazdelitev obeh spremenljivk (funkcija `scatter.hist()` ; zgoraj desno);
- razsevnega diagrama z različno pobarvanimi točkami udeležencev iz dveh podvzorcev (funkcija `qplot()`; levo spodaj);
- razsevnega diagrama z vrisanima boxplot prikazoma za porazdelitvi obeh spremenljivk (funkcija `scatterplot()`; desno spodaj).



Kovarianca in Pearsonov koeficient korelacije

V primeru vsaj intervalnih spremenljivk je kovarianca najosnovnejša mera sovariiranja oz. povezanosti dveh spremenljivk. Izračunamo jo po enačbi:

$$cov_{xy} = \frac{\sum(X - M_x)(Y - M_y)}{N} \quad ALI \quad cov_{xy} = \frac{\sum(X - M_x)(Y - M_y)}{N - 1} \qquad (17, 18)$$

Kovarianco lahko izračunamo na dva načina (primerjaj enačbi 17 in 18). Enačba 17 (z *N* v imenovalcu) je kovarianca za celotno populacijo. Če ju izračunamo na vzorcu iz določene populacije, predstavlja *pristransko oceno populacijske kovariance*. Če želimo izračunati *nepristransko oceno populacijske kovariance*, uporabimo enačbo 18 (z *N* – 1 v imenovalcu). Iz enačb 17 in 18 je razvidno, da je kovarianca povprečni produkt odklonov (od aritmetične sredine) obravnavanih spremenljivk. Če sta odklona za določen par podatkov pozitivna ali negativna,

bo produkt pozitiven (tak produkt bo impliciral pozitivno povezanost), če pa je en od odklonov pozitiven, drug pa negativen, bo njun produkt negativen (tak produkt bo impliciral negativno povezanost). Pozitivna vsota vseh odklonov kaže na pozitivno povezanost, negativna vsota pa na negativno povezanost dveh spremenljivk.

V Rju imamo na voljo funkcijo `cov()`, ki izračuna nepristransko populacijsko oceno kovariance (enačba 18). Podatke lahko funkciji `cov()` podamo v obliki celotnega p.o. (ali z izbranimi spremenljivkami) ali matrike (v obeh primerih podatki predstavljajo samo en argument), lahko pa funkciji podamo dve spremenljivki (v tem primeru so podatki podani v dveh zaporednih argumentih). Pri tej (in sorodnih bivariatnih) funkcijah imamo več možnosti glede ravnanja z manjkajočimi podatki, če računamo kovariance (ali korelacije) med pari več spremenljivk hkrati. En od načinov je *casewise* ali *listwise* izločanje podatkov (argument use = "complete.obs"), drug pa *pairwise* izločanje podatkov (argument use = "pairwise.complete.obs"). Pri prvem načinu uporabimo samo podatke tistih udeležencev, ki nimajo manjkajočih vrednosti na nobeni od obravnavanih spremenljivk (kovariance za vse pare spremenljivk bodo izračunane na istem številu udeležencev), pri drugem pa bodo v izračun za določen par spremenljivk vključeni vsi udeleženci, ki nimajo manjkajočih podatkov na teh dveh spremenljivkah (kovariance za vse pare spremenljivk bodo lahko izračunane na različnem številu udeležencev).

```
# izračun nepristranske ocene populacijske kovariance s funkcijo cov();
# 1. primer: kovariančna matrika za cel p.o., pri čemer smo uporabili
# listwise/casewise izločanje udeležencev;
# 2. primer: kovariančna matrika za manjši izbor spremenljivk, uporabili smo
# casewise izločanje udeležencev;
# 3. primer: kovarianca samo za en par spremenljivk iz p.o.

cov(po, use = "complete.obs")
cov(po[c(2:5, 10:14, 23)], use = "pairwise.complete.obs")
cov(po$var1, po$var2)
```

Kovarianci lahko rečemo tudi "surova" ali "nestandardizirana" mera povezanosti, saj je njena vrednost odvisna od uporabljene enote na določenem kontinuumu (najvišja možna absolutna vrednost kovariance je enaka produktu standardnih odklonov obeh spremenljivk) . Iz vrednosti kovariance zaradi tega izvemo le smer/predznak povezanosti, ne pa tudi velikosti povezanosti. Zaradi te pomanjkljivosti kovariance ne moremo uporabljati za opis povezanosti v kontekstu opisnih statistik.

Pearsonov *r* je standardizirana ali brezdimenzionalna mera povezanosti (*r* je standardizirana kovarianca; glej enačbo 19) in **lahko zavzema vrednosti od –1 (popolna negativna korelacija) do 1 (popolna pozitivna korelacija)**. Izračunamo ga lahko na mnogo načinov, spodaj pa sta predstavljeni enačbi, ki temeljita na kovarianci in *z*-vrednostih obeh obravnavanih spremenljivk.

$$r_{xy} = \frac{cov_{xy}}{\sigma_x \sigma_y} \quad ALI \quad r_{xy} = \frac{\sum z_x z_y}{N-1} \qquad (19, 20)$$

Pri uporabi enačbe 19 moramo paziti, da so vse mere (kovarianca in oba standardna odklona) izračunane na enak način, torej morajo biti vse ali pristranske ali nepristranske ocene populacijskih parametrov. Enako velja za enačbo 20, saj moramo v primeru enačbe z $N-1$ v imenovalcu *z*-vrednosti izračunati z uporabo nepristranske ocene populacijske *SD*, v primeru enačbe z N v imenovalcu pa moramo pri izračunu *z*-vrednosti uporabiti pristranske ocene populacijske *SD*. Če "surove" vrednosti obeh spremenljivk standardiziramo (oz. pretvorimo v *z*-vrednosti), bo kovarianca med standardiziranima spremenljivkama enaka korelaciji med tema dvema spremenljivkama. V Rju Pearsonov *r* izračunamo s funkcijo `cor()`, ki jo uporabljamo na povsem enak način kot funkcijo `cov()`.

```
# izračun Pearsonovega koeficienta korelacije s funkcijo cor(); spodnji trije
# primeri so primerljivi s predhodno predstavljenimi izračuni kovariance (za
# razlago posameznih primerov zato glej komentar nad predhodnimi primeri
# izračunov kovariance in kovariančnih matrik)

cor(po, use = "complete.obs")
cor(po[c(2:5, 10:14, 23)], use = "pairwise.complete.obs")
cor(po$var1, po$var2)
```

Če imamo v določenem objektu (npr. v objektu "kov") shranjeno kovariančno matriko, lahko te kovariance v matriki spremenimo v korelacije s funkcijo `cov2cor()`.

```
# pretvorba kovariančne matrike v korelacijsko matriko (objekt "kor" vsebuje
# kovariančno matriko)

cov2kor(kor)
```

Osnovna opredelitev bivariatne linearne regresije

Z bivariatno linearno regresijo napovedujemo vrednosti ene spremenljivke na osnovi vrednosti druge spremenljivke. Spremenljivko, na osnovi katere napovedujemo, imenujemo **napovednik** ali **prediktor** (ali prediktorska spremenljivka), spremenljivko, katere vrednosti napovedujemo, pa **kriterijska spremenljivka** (ali zgolj **kriterij**). V enačbah za linearno regresijo kriterij običajno označimo s črko Y, prediktor pa s črko X.

Regresijski modeli so lahko kompleksnejši kot bivariatni model, v katerega vključimo samo en kriterij in en prediktor. Če napovedujemo vrednosti kriterija na osnovi več prediktorjev, potem govorimo o **multipli linearni regresiji**, če pa v model vključimo tudi več kriterijev, pa govorimo o **multivariatni (multipli) regresiji**.

Za opis odnosa med dvema spremenljivkama z bivariatno linearno regresijo uporabljamo linearno funkcijo kot model za opis tega odnosa:

$$\mathrm{Y}' = a_{\mathrm{YX}} + b_{\mathrm{YX}}\mathrm{X} \tag{21}$$

pri čemer je Y' napovedana vrednost kriterija, a_{YX} je **presečišče z ordinato**, b_{YX} je (nestandarizirani) regresijski koeficient/nagib, X pa vrednost prediktorja. Če presečišču in nagibu v indeks dodamo imena obeh spremenljivk, na prvo mesto postavimo kriterij, na drugo pa prediktor (torej $a_{YX} \neq a_{XY}$ in $b_{YX} \neq b_{XY}$).

Regresijsko premico lahko točkam v korelacijskem/regresijskem oblaku prilagodimo na različne načine oziroma z uporabo različnih metod. Najosnovnejša in še vedno najpogosteje uporabljana je **metoda najmanjših kvadratov**. S to metodo regresijsko premico prilagodimo podatkom tako, da vsota odklonov dejanskih vrednosti kriterija (Y) od napovedanih vrednosti kriterija (Y') znaša 0 (odklon Y–Y' imenujemo **rezidual**) oziroma tako, da ima vsota kvadriranih rezidualov najmanjšo možno vrednost ($\Sigma(Y–Y')^2$ = min).

Presečišče z ordinato nam pove, koliko znaša napovedana vrednost kriterija, če je prediktor enak 0. **Regresijski nagib** pa nam pove, za koliko se spremeni (poveča ali zmanjša) napovedana vrednost kriterija, če se vrednost prediktorja poveča za eno enoto. Presečišče z ordinato in regresijski nagib izračunamo po sledečih enačbah:

$$a_{\mathrm{YX}} = M_{\mathrm{Y}} - b_{\mathrm{YX}}M_{\mathrm{X}} \tag{22}$$

$$b_{\mathrm{YX}} = \frac{cov_{\mathrm{XY}}}{\sigma_{\mathrm{X}}^2} \quad ALI \quad b_{\mathrm{YX}} = r_{\mathrm{XY}}\frac{\sigma_{\mathrm{Y}}}{\sigma_{\mathrm{X}}} \tag{23, 24}$$

Parametra regresijske enačbe (*a* in *b*) v Rju izračunamo s funkcijo `lm()`, ki je sicer splošna funkcija za uporabo linearnega modela (lm je okrajšava za *linear model*). Funkcija ima veliko število argumentov, a jih lahko za osnovno uporabo večino izpustimo oz. uporabimo njihove privzete vrednosti. Podatke in opredelitev regresijskega modela funkciji podamo v obliki formule (kriterij ~ prediktor).

```
# izračun parametrov (bivariatnega) regresijskega modela s funkcijo lm();
# vrednosti spremenljivke "var_k" napovedujemo na osnovi spremenljivke
# "var_p"; funkcija lm ima tudi argument subset, s katerim lahko določimo
# podvzorec udeležencev, na katerih izvajamo linearno regresijo (npr., samo
# za tiste, ki so starejši od 25 let (2. primer); samo za ženske udeleženke
# (3. primer)

reg <- lm(po$var_k ~ po$var_p)
reg <- lm(po$var_k ~ po$var_p, subset = po$starost > 25)
reg <- lm(po$var_k ~ po$var_p, subset = po$spol == "Ž")
```

Izpis vsebine objekta, ki mu priredimo rezultat funkcije `lm()`, vsebuje le informacije o opredelitvi modela in obeh parametrih; *a* = (Intercept), *b* = vrednost pod imenom prediktorja. Objekt z rezultati, ki ga vrne funkcija `lm()`, je seznam različnih spremenljivk, do katerih lahko dostopamo na enak način kot do spremenljivk v p.o. (po vzoru po$var). Npr., spremenljivka **coefficients** vsebuje vse parametre regresijskega modela (presečišče in nagib/e), spremenljivka **residuals** vsebuje (nestandardizirane) reziduale (Y–Y'), spremenljivka **fitted.values** pa napovedane vrednosti kriterija (Y').

Če objekt, ki ga vrne funkcija `lm()`, podamo funkciji `summary()`, bo izpis vseboval tudi opisne statistike za reziduale, standardne napake za regresijske parametre (in informacije o statistični značilnosti parametrov), nepristransko oceno populacijske standardne napake napovedi in koeficient determinacije (nekatere od teh mer bomo spoznali v poglavju o razstavljanju variance kriterija). Objekt, ki ga v tem primeru vrne funkcija `summary()`, je (podobno kot objekt, ki ga vrne `lm`) seznam spremenljivk. Npr., spremenljivka **residuals** vsebuje (obtežene) reziduale, **coefficients** vsebuje presečišče in nagibe, **sigma** vsebuje nepristransko oceno populacijske standardne napake napovedi, **r.squared** vsebuje koeficient determinacije, **adj.r.squared** pa nepristransko oceno populacijskega koeficienta determinacije (tj. popravljen koeficient determinacije). Noben od pravkar obravnavnih objektov z rezultati linearne regresije ne vsebuje standardiziranih rezidualov. Če jih potrebujemo, jih lahko izračunamo s funkcijo `rstandard()`, kateri podamo objekt, ki ga vrne funkcija `lm()`.

Standardizirana regresijska enačba

Regresijski model lahko opredelimo tudi v standardizirani obliki, pri čemer lahko ali (i) standardiziramo prediktor in kriterij ter določimo parametre regresijske enačbe ali (ii) standardiziramo regresijske parametre, določene na surovih vrednostih prediktorja in kriterija. Enačba za standardizirano bivariatno linearno regresijo je:

$$z(\mathrm{Y})' = \beta_{z(\mathrm{Y})z(\mathrm{X})}\, z(\mathrm{X}) \tag{25}$$

pri čemer je β **standardiziran regresijski nagib**. V standardizirani obliki presečišče z ordinato znaša 0, saj aritmetična sredina katerokoli standardizirane spremenljivke znaša 0 (glej enačbo 22). Regresijska premica v (standardiziranem) razsevnem diagramu torej vedno poteka skozi izhodišče koordinatnega sistema (x-os v tem primeru predstavlja aritmetično sredino kriterija, y-os pa aritmetično sredino prediktorja).

```
# določitev standardizirane regresijske enačbe s standardiziranjem kriterija
# in prediktorja

reg <- lm(scale(po$var_k) ~ scale(po$var_p))

# vrednosti parametra regresijske premice za z-vrednosti sta privzeto
# oblikovana na znanstven način (tj. X * 10^n); če želimo vrednosti parametrov
# izpisati na običajen način, lahko zaokrožimo oba parametra (npr. na 3
# decimalna mesta natančno)

round(reg$coefficients, 3)
```

Beta koeficient lahko izračunamo tako, da standardiziramo nestandardizirani regresijski nagib:

$$\beta_{z(Y)z(X)} = b_{YX}\frac{\sigma_X}{\sigma_Y} \tag{26}$$

```
# standardiziranje nestandardiziranega regresijskega nagiba; v objektu reg so
# shranjeni (nestandardizirani) rezultati funkcije lm(); po$var_p je
# prediktor, po$var_k pa kriterij

beta <- reg$coefficients["po$var_p"] * sd(po$var_p) / sd(po$var_k)
```

Standardiziran regresijski nagib nam pove, za koliko standardnih odklonov se spremeni napovedana vrednost (standardiziranega) kriterija, če se (standardiziran) prediktor poveča za en standardni odklon. V primeru bivariatne linearne regresije je beta koeficient enak korelaciji med prediktorjem in kriterijem. V primeru multiple linearne regresije za določeno β velja ta enakost le v primeru, če je obravnavan prediktor popolnoma neodvisen od ostalih prediktorjev v modelu!

Dodajanje regresijske premice na razsevni diagram

Na razsevni diagram, ki smo ga narisali s funkcijo `plot()`, regresijsko premico dodamo s funkcijo `abline()`. Za izris regresijske premice funkciji podamo objekt, ki ga vrne funkcija `lm()`.

```
# dodajanje regresijske premice na razsevni diagram s funkcijo abline(); v
# objektu reg so shranjeni rezultati, ki jih vrne funkcija lm();
# 1.primer: narišemo razsevni diagram s funkcijo plot
# 2. primer: na graf dodamo (modro) regresijsko premico s funkcijo abline(),
# pri čemer ji podamo objekt reg

plot(po$var_p, po$var_k)
abline(reg, col = "blue")
```

Če za risanje razsevnega diagrama uporabimo funkcijo `qplot()` iz paketa ggplot2, lahko regresijsko premico dodamo s funkcijo `geom_abline()`, pri čemer funkciji podamo vrednosti presečišča in nagiba (iz objekta reg, ki ga je vrnila funkcija `lm`).

```
# dodajanje regresijske premice na razsevni diagram, narisan s funkcijo
# qplot(); funkciji geom_abline podamo presečišče in regresijski nagib

qplot(po$var_p, po$var_k) + geom_abline(intercept =
      reg$coefficients["(Intercept)"], slope = reg$coefficients["po$var_p"])
```

Na razsevni diagram, narisan s funkcijo `scatter.hist()` iz paketa psych, regresijsko premico dodamo tako, da argumentu ab priredimo vrednost TRUE.

```
# dodajanje regresijske premice na razsevni diagram, narisan s funkcijo
# scatter.hist(); nekatere privzeto omogočene možnosti smo onemogočili

scatter.hist(po$var_p, po$var_k, smooth = FALSE, ellipse = FALSE,
             title = NULL, correl = FALSE, ab = TRUE)
```

Razsevni diagram, ki ga lahko obogatimo z mnogimi koristnimi informacijami, lahko narišemo tudi s funkcijo `scatterplot()` iz paketa car. Ta funkcija je glede dodajanja različnih elementov na razsevni diagram zelo podobna funkciji `scatter.hist()`, le da za prikaz porazdelitev obeh obravnavanih spremenljivk na graf doda boxplot prikaza namesto histogramov.

```
# dodajanje regresijske premice na razsevni diagram, narisan s funkcijo
# scatterplot(); prikaz regresijske premice nadziramo prek argumenta
# reg.line, ki je privzeto omogočen, zato ga lahko izpustimo! (za jasnejši
# prikaz onemogočimo vris zglajene regresijske krivulje)

scatterplot(po$var_p, po$var_k, smooth = FALSE)
```

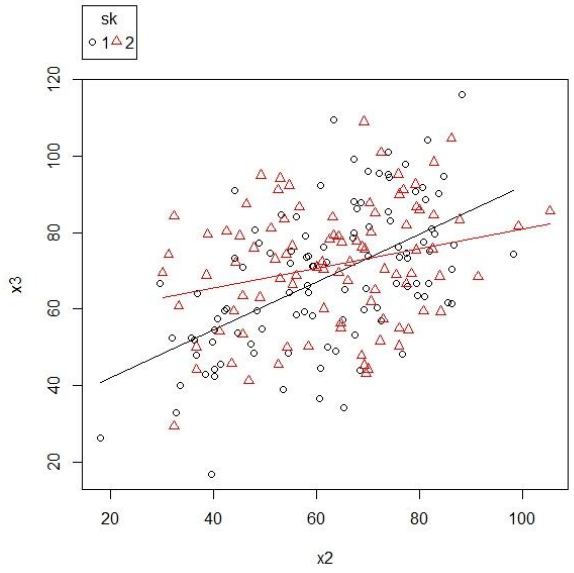
Funkcija `scatterplot()` ima tudi argument groups, s katerim lahko točke udeležencev, ki prihajajo iz različnih skupin oz. podvzorcev, prikažemo v različni barvi in z različnimi znaki (argumentu podamo spremenljivko/faktor, ki definira podvzorce; npr. spol ipd.). Če želimo imeti dodano regresijsko premico za celoten vzorec, argumentu by.groups priredimo vrednost FALSE, če pa želimo dodati regresijske premice za vsak podvzorec posebej, mu priredimo vrednost TRUE (privzeta vrednost).

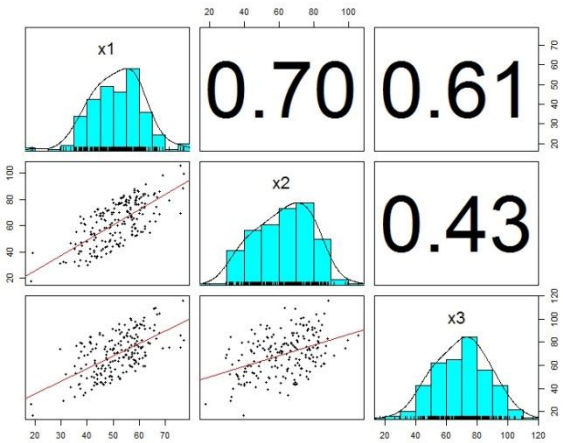
```
# risanje razsevnega diagrama s funkcijo scatterplot(), pri čemer uporabimo
# različno oblikovanje točk za moške in ženske; v prvem primeru dodamo
# regresijsko premico za celoten vzorec, v drugem primeru pa ločeni
# regresijski premici za moške in ženske (argument by.groups lahko v tem
# primeru izpustimo, ker je privzeto omogočen); v 2. primeru smo tudi
# onemogočili izris pomožnih črt na grafu (grid = FALSE)

scatterplot(po$var_p, po$var_k, smooth = F, groups = po$spol,
            by.groups = FALSE)
scatterplot(po$var_p, po$var_k, smooth = F, groups = po$spol,
            by.groups = TRUE, grid = FALSE)
```

Zelo uporabna je tudi funkcija `pairs.panels()` iz paketa psych, s katero lahko ustvarimo matrični prikaz (za dve ali več spremenljivk), ki pod diagonalo vsebuje razsevne diagrame, v diagonali histograme z vrisano ocenjeno gostoto verjetnosti, nad diagonalo pa korelacijske koeficiente.

```
# risanje matričnega prikaza za odnos med štirimi spremenljivkami (ki se v
# p.o. nahajajo v stolpcih z indeksi 8 do 11) s funkcijo pairs.panels(); za
# jasnejši prikaz smo onemogočili argument ellipses in naročili vris
# regresijske premice (lm = TRUE)

pairs.panels(po[8:11], ellipses = FALSE, lm = TRUE)
```

Spodaj je prikazan (i) razsevni diagram z različno označenimi točkami udeležencev iz dveh podzvorcev in vrisanima regresijskima premicama za ta dva podvzorca (funkcija `scatterplot()`; levi prikaz) in (ii) matrični prikaz razsevnih diagramov s funkcijo `pairs.panels()`.





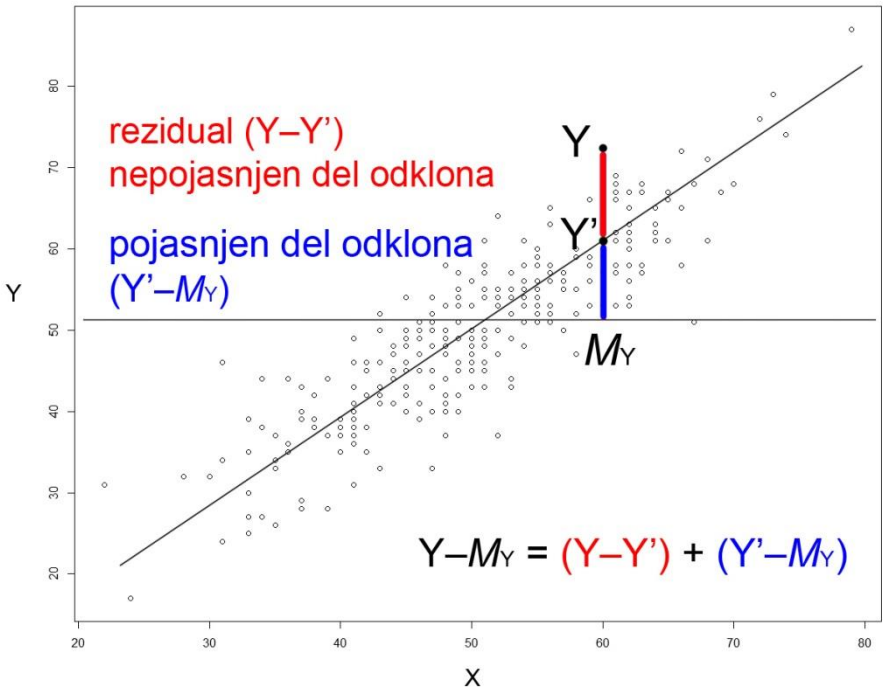
Razstavljanje variance kriterija in mere prileganja regresijskega modela podatkom

Napovedni model lahko oblikujemo tudi v primeru, če imamo na voljo samo eno spremenljivko (npr. Y). V tem primeru je aritmetična sredina te spremenljivke najboljša napoved za katerokoli osebo iz izbrane populacije (regresijsko enačbo zapišemo kot Y' = M_Y). Tovrstnemu regresijskemu modelu rečemo **prazni model** (angl. *null model*) ali ga opišemo kot **"napovedovanje na osnovi aritmetične sredine (kriterija)"**. Prazni model predstavlja osnovo za ocenjevanje učinkovitosti napovedovanja s katerimkoli drugim (ugnezdenim) kompleksnejšim regresijskim modelom. Če poznamo vrednosti neke druge spremenljivke (npr. X), pri čemer za ti dve spremenljivki velja $r_{XY} \neq 0$, lahko oblikujemo regresijski model, ki omogoča učinkovitejše oz. natančnejše napovedovanje vrednosti kriterija v primerjavi s praznim modelom[1].

Varianco kriterija (oz. odklonov vrednosti kriterija od aritmetične sredine kriterija; tj. Y–M_Y) lahko razstavimo na varianco, ki jo lahko pojasnimo z regresijskim modelom (varianca Y'–M_Y = **modelna ali pojasnjena varianca**), in varianco, ki je ne moremo pojasniti z regresijskim modelom (varianca Y–Y' = **nepojasnjena varianca**; glej tudi spodnjo sliko, ki prikazuje princip razstavljanja variance kriterija na primeru ene osebe/točke v razsevnem diagramu):

$$\frac{\sum(\mathrm{Y}-M_\mathrm{Y})^2}{N} = \frac{\sum(\mathrm{Y}'-M_\mathrm{Y})^2}{N} + \frac{\sum(\mathrm{Y}-\mathrm{Y}')^2}{N} \quad ALI \quad \sigma_\mathrm{Y}^2 = \sigma_{\mathrm{Y}'}^2 + \sigma_e^2 \tag{27}$$

Odklon dejanske vrednosti na kriteriju od aritmetične sredine kriterija (Y–M_Y) lahko razstavimo na pojasnjen del (Y'– M_Y) in nepojasnjen del oziroma rezidual (Y–Y').



Modelna oz. pojasnjena varianca je torej varianca napovedanih vrednosti, nepojasnjena varianca pa je varianca rezidualov. Standardnemu odklonu rezidualov rečemo tudi **standardna napaka napovedi** in jo običajno označimo kot $SE_{Y.X}$ ali $\sigma_{Y.X}$. Standardno napako napovedi za populacijo izračunamo tako, da korenimo nepojasnjeno varianco (glej enačbo 27) ali standardni odklon kriterija pomnožimo s $\sqrt{1-r^2}$, nepristransko oceno populacijske standardne napake napovedi pa po enačbah 28 ali 29 (v enačbi 29 je σ_Y izračunana z *N* v imenovalcu):

$$\sigma_{\mathrm{Y.X}} = \sqrt{\frac{\sum(\mathrm{Y}-\mathrm{Y}')^2}{N-k-1}} \quad ALI \quad \sigma_{\mathrm{Y.X}} = \sigma_\mathrm{Y}\sqrt{1-r^2}\sqrt{\frac{N}{N-k-1}} \tag{28, 29}$$

[1] Če za prediktor X in kriterij Y velja $r = 0$ (=> $b = 0$; glej enačbo 24), imamo ponovno opravka s praznim modelom: $a_{YX} = M_Y - 0*M_X = M_Y$ => Y' = a_{YX} + 0*X = M_Y

pri čemer je *k* število prediktorjev, torej v primeru bivariatne linearne regresije vsoto kvadriranih rezidualov delimo z *N* – 2 (enačba 28; enako velja za imenovalec zadnjega člena enačbe 29). Standardna napaka napovedi je ena od mer prileganja regresijskega modela podatkom, in sicer visoke vrednosti pomenijo slabše, nižje pa boljše prileganje. V primeru popolne korelacije med prediktorjem in kriterijem $\sigma_{Y.X}$ znaša 0, v primeru popolne neodvisnosti kriterija in prediktorja pa je enaka standardnemu odklonu kriterija (v tem primeru je celotna varianca kriterija nepojasnjena). Na tak način izračunano standardno napako napovedi bomo uporabili tudi za izračun napovednega intervala za populacijo oz. na zelo velikih vzorcih, ki omogočajo izračun zelo natančnih ocen populacijskih parametrov regresijskega modela (glej naslednje poglavje o napovednem intervalu).

Razmerje med pojasnjeno varianco (tj. varianco napovedanih vrednosti $\sigma_{Y'}^2$; glej enačbo 27) in varianco kriterija (σ_Y^2) imenujemo **koeficient determinacije** (r^2).

$$r^2 = \frac{\sigma_{Y'}^2}{\sigma_Y^2} = 1 - \frac{\sigma_e^2}{\sigma_Y^2} \tag{30}$$

Koeficient determinacije lahko izračunamo tudi kot produkt obeh regresijskih nagibov ($b_{YX}b_{XY}$), najpogosteje pa ga izračunamo kot **kvadrat Pearsonovega *r***. Ta koeficient nam pove, kolikšen delež variance kriterija lahko pojasnimo s prediktorjem (ali obratno; ker vrednost koeficienta determinacije ni odvisna od tega, katero spremenljivko napovedujemo, nam r^2 v bistvu pove, koliko variance si spremenljivki delita).

Do vrednosti (nepristranske ocene) standardne napake napovedi in koeficienta determinacije v Rju dostopamo tako, da objekt z rezultati linearne regresije (npr. z imenom reg) podamo funkciji `summary()`. V izpisu, ki ga v tem primeru vrne ta funkcija, je standardna napaka napovedi poimenovana "Residual standard error", koeficient determinacije pa "Multiple R-squared". Če rezultat funkcije `summary(reg)` priredimo objektu (npr. reg_rezultati), bo vrednost standardne napake napovedi shranjena v spremenljivki `reg_rezultati$sigma`, koeficient determinacije pa v spremenljivki `reg_rezultati$r.squared`.

Kot mero prileganja regresijskega modela podatkom lahko uporabimo tudi **indeks učinkovitosti napovedi (*E*)**, ki nam pove, za koliko odstotkov se zmanjša standardna napaka napovedi, če napovedujemo na osnovi regresijske enačbe namesto na osnovi aritmetične sredine kriterija (oz. s praznim modelom). Običajno ga izračunamo po enačbi 31 in ga izrazimo v odstotkih (zavzema lahko vrednosti od 0 do 1 oz. od 0 do 100 %). Kvadratni koren deleža nepojasnjene variance (tj. $\sqrt{1-r^2}$) imenujemo **koeficient alienacije**.

$$E = 100\,\% * \left(1 - \sqrt{1 - r^2}\right) \tag{31}$$

```
# indeks učinkovitosti napovedi v Rju izračunamo sami z uporabo koeficienta
# determinacije (npr. uporabimo lahko spremenljivko r.squared iz objekta, ki
# ga vrne funkcija summary() (1. primer), ali pa uporabimo kvadrat
# Pearsonovega r, ki ga izračunamo s funkcijo cor() (2. primer)

E <- 100 * (1 - sqrt(1 - reg_rezultati$r.squared))
E <- 100 * (1 - sqrt(1 - cor(po$var_k, po$var_p)^2)
```

Napovedni interval

Poleg točkovne napovedi najverjetnejšega dosežka neke osebe na kriteriju, nas velikokrat zanima tudi intervalna ocena za prihodnji dejanski dosežek te osebe na kriteriju. To intervalno oceno imenujemo **napovedni interval**, tj. interval vrednosti (območje med zgornjo in spodnjo mejo), v katerem se z določeno verjetnostjo nahaja prihodnji dejanski dosežek osebe na kriteriju. Za veljaven izračun napovednega intervala morajo biti zadovoljene predpostavke linearne regresije (glej naslednje poglavje o preverjanju predpostavk).

Če imamo opravka s populacijo ali zelo velikim vzorcem (torej, ko imamo na voljo povsem ali zelo natančne ocene regresijskih parametrov), napovedni interval za prihodnji dejanski dosežek na kriteriju izračunamo po spodnjem obrazcu:

$$Y' \pm z * \sigma_{Y.X} \tag{32}$$

pri čemer je Y' napovedana vrednost na kriteriju, $\sigma_{Y.X}$ standardna napaka napovedi (glej enačbo 28 ali 29), *z* pa je vrednost iz standardizirane normalne porazdelitve, ki določa želeno širino intervala (npr. 1,96 za 95-odstotni napovedni interval). V Rju ni na voljo funkcije za tovrstni izračun napovednih intervalov, lahko jih pa sami izračunamo po obrazcu 32.

```
# izračun 95-odstotnih napovednih intervalov (njihovih zgornjih in spodnjih
# mej), pri čemer imamo opravka s populacijo ali zelo velikim vzorcem
# udeležencev; rezultate regresijske analize imamo shranjene v objektu reg:
# reg <- lm(po$var_k ~ po$var_p); povzetek rezultatov regresijske analize pa
# v objektu reg_rezultati: reg_rezultati <- summary(reg)

zg_meje <- reg$fitted.values + qnorm(0.975) * reg_rezultati$sigma
sp_meje <- reg$fitted.values - qnorm(0.975) * reg_rezultati$sigma
```

Pri majhnih vzorcih natančnost napovedi ni odvisna zgolj od višine korelacije med prediktorjem in kriterijem, temveč tudi od napake ocene regresijskih parametrov. V takih primerih bomo pri izračunu napovednih intervalov v enačbi 32 *z*-vrednost zamenjali s *t*-vrednostjo (ki izhaja iz *t* porazdelitve, ki je odvisna od stopenj prostosti; *df* = *N* – 2), standardno napako napovedi pa bomo izračunali po enačbi 33 (za vsakega udeleženca oz. za vse vrednosti na prediktorju!):

$$\sigma_{Y.X} = \sqrt{\frac{N}{N-2}}\sigma_Y\sqrt{1-r^2}\sqrt{1+\frac{1}{N}+\frac{z_i^2}{N}} \tag{33}$$

pri čemer je σ_Y standardni odklon kriterija, izračunan z *N* v imenovalcu, z_i pa standardizirana vrednost osebe *i* na prediktorju (to *z*-vrednost izračunamo na osnovi standardnega odklona prediktorja z *N* v imenovalcu). Standardna napaka napovedi se torej veča z oddaljenostjo dosežka na prediktorju od aritmetične sredine prediktorja.

Napovedne intervale bomo v takih primerih v Rju izračunali s pomočjo funkcije `predict.lm()`, pri čemer moramo funkciji nujno podati objekt z rezultati funkcije `lm()` in argumentu interval prirediti vrednost "prediction". Z argumentom level določimo širino napovednega intervala (privzeta vrednost znaša 0.95, kar pomeni, da z ignoriranjem tega argumenta tvorimo 95-odstotne napovedne intervale).

```
# izračun 90-odstotnih napovednih intervalov za vse udeležence; rezultate
# regresijske analize (rezultat funkcije lm) imamo shranjene v objektu reg

nap_int <- predict.lm(reg, interval = "prediction", level = 0.9)
```

Funkcija `predict.lm()` vrne matriko z *N* vrsticami in tremi stolpci/spremenljivkami, in sicer fit (napovedane vrednosti kriterija), lwr (spodnje meje napovednih intervalov) in upr (zgornje meje napovednih intervalov). Če želimo do teh treh spremenljivk dostopati z znakom $ (po vzorcu po$var), matriko spremenimo v podatkovni okvir s funkcijo `as.data.frame()`.

Zgornje in spodnje meje napovednih intervalov lahko prikažemo tudi v grafični obliki na razsevnem diagramu. Če razsevni diagram narišemo s funkcijo `plot()`, lahko krivulji, ki prikazujeta meje napovednih intervalov, dodamo s funkcijo `lines()`.

```
# dodajanje zgornjih in spodnjih mej napovednega intervala na razsevni
# diagram; po$var_k napovedujemo na osnovi po$var_p; rezultate regresijske
# analize imamo shranjene v objektu reg, zgornje in spodnje meje napovednih
# intervalov pa v objektu nap_int, ki smo ga iz matrike spremenili v
# podatkovni okvir s funkcijo as.data.frame()
# 1. primer: ustvarimo razsevni diagram s funkcijo plot()
# 2. primer: na razsevni diagram dodamo regresijsko premico
# 3. in 4. primer: s funkcijo lines() na razsevni diagram dodamo modro
# krivuljo, ki povezuje zgornje/spodnje meje napovednih intervalov
# POZOR! Vrednosti na prediktorju in meje napovednih intervalov je potrebno
# naraščajoče razvrstiti po prediktorski spremenljivki!

plot(po$var_p, po$var_k)
abline(reg)
lines(po$var_p[order(po$var_p)], nap_int$upr[order(po$var_p)], col = "blue")
lines(po$var_p[order(po$var_p)], nap_int$lwr[order(po$var_p)], col = "blue")
```

Preverjanje predpostavk linearne regresije

Rezultati linearne regresije so veljavni le v primeru, če držijo predpostavke, na katerih temelji regresijska analiza. Preverjanju predpostavk linearne regresije rečemo tudi **regresijska diagnostika**. Pri preverjanju veljavnosti teh predpostavk si pomagamo z grafičnimi in statističnimi postopki. Predpostavke za ustrezno izvedbo regresijske analize se nanašajo na:

- neodvisnost opazovanj oz. naključno vzorčenje,
- mersko raven spremenljivk v regresijskem modelu,
- linearnost odnosa med spremenljivkama,
- normalnost porazdelitve rezidualov,
- homoscedastičnost.

Neodvisnost opazovanj / naključno vzorčenje

Predpostavka neodvisnosti opazovanj pomeni, da mora biti vsak udeleženec vzorčen neodvisno od drugih, pri čemer kršitev te predpostavke povzroči neveljavnost testov statistične značilnosti. Spodaj je navedenih nekaj običajnih primerov merskih situacij, ki vodijo do kršitve te predpostavke:

- V neki raziskavi smo vse udeležence večkrat izmerili na obravnavanih spremenljivkah in vse meritve obravnavali kot neodvisne oz. kot zbrane na povsem različnih osebah. Npr. zanimala nas je korelacija med enostavnimi ($RČ_E$) in izbirnimi ($RČ_I$) reakcijskimi časi. Meritve smo izvedli na 50 udeležencih, vendar smo pri vseh zbrali 10 meritev $RČ_E$ in $RČ_I$ in te ponovljene meritve obravnavali, kot da bi jih pridobili na neodvisnih/drugih osebah (v podatkovni bazi bomo v tem primeru imeli 50 * 10 = 500 vrstic oz. "udeležencev" namesto 50). Podatki znotraj osebe med seboj niso neodvisni. Ustrezen postopek bi vključeval izračun povprečnih $RČ_E$ in $RČ_I$ znotraj oseb in nato vnos teh povprečnih vrednosti v podatkovno bazo (vsak udeleženec bi v bazo prispeval po eno vrednost na spremenljivko).
- Med merjenjem znanja dovolimo/spregledamo "sodelovanje" med nekaterimi učenci. Čeprav smo na vsaki osebi izvedli le eno meritev, dosežki teh učencev med seboj ne bodo neodvisni.
- Preizkušance vzorčimo iz različnih enot, kot so npr. šole, podjetja, študijski programi ipd. (takšnemu vzorčenju rečemo gnezdeno vzorčenje). Npr., v neki raziskavi smo zbrali po 30 študentov iz treh študijskih programov. Tako zbranih podatkov ne moremo obravnavati kot neodvisnih, saj so si študenti znotraj določenega programa po mnogih spremenljivkah med seboj bolj podobni, kot so si podobni s študenti iz drugih programov. Za razrešitev tega problema uporabimo bodisi hierarhično linearno modeliranje ali pa podatke znotraj posameznih enot povprečimo in to povprečno vrednost znotraj enote obravnavamo kot podatek "ene osebe".

Z izjemo gnezdenega vzorčenja je kršitev neodvisnosti opazovanj težko zaznati, zaradi česar skušamo veljavnost te predpostavke zagotoviti pri pripravi in izvedbi meritev. Če imamo opravka s časovnimi vrstami (to so podatki, zbrani v zaporednih, običajno med seboj enako oddaljenih časovnih točkah), se lahko odvisnost opazovanj kaže v neničelni (avto)korelaciji (oz. serialni korelaciji) med zaporednimi (ali z večjim korakom ločenimi) reziduali.

Prisotnost avtokorelacije v tovrstnih podatkih lahko v preverimo z Durbin-Watsonovim testom, ki vrne testno statistiko (*D*) z razponom od 0 do 4 (*D* = 2 kaže na odsotnost avtokorelacije, *D* > 2 kaže na negativno avtokorelacijo, *D* < 2 pa na pozitivno avtokorelacijo). Po približnem kriteriju kot problematične obravnavamo vrednosti, ki so manjše od 1 ali večje od 3. V Rju Durbin-Watsonov test izvedemo s funkcijama `durbinWatsonTest()` ali `dwt()` iz paketa car ali s funkcijo `dwtest()` iz paketa lmtest. Slednja funkcija poleg *D* statistike vrne tudi vrednost avtokorelacije.

```
# Durbin-Watsonov test serialne korelacije za linearne regresijske modele;
# funkciji moramo nujno podati objekt z rezultati regresije (tj. reg), ki ga
# vrne funkcija lm(); v 1. primeru smo uporabili funkcijo dwt() iz paketa
# car, v 2. primeru pa funkcijo dwtest() iz paketa lmtest (pri slednji smo
# spremenili privzeto vrednost argumenta alternative v "two.sided")

dwt(reg)
dwtest(reg, alternative = "two.sided")
```

Merska raven spremenljivk

Prediktor mora biti vsaj intervalen ali dihotomen (tj. nominalna spremenljivka s samo dvema kategorijama), kriterij pa vsaj intervalen. Na tem mestu lahko tudi omenimo, da kriterij ne sme imeti omejenega variiranja. Večina podatkov, ki jih zbiramo s psihološkimi testi, imajo omejen razpon možnih vrednosti (npr. najmanjši ali največji možni dosežek), vendar to ne predstavlja resnega problema, če je delež podatkov v bližini zgornje ali spodnje meje možnega razpona zelo majhen.

Linearnost odnosa med spremenljivkama

Za linearnost odnosa med prediktorjem in kriterijem bi lahko rekli, da je najosnovnejša predpostavka, saj kot model za odnos uporabljamo linearno funkcijo. Če sta spremenljivki v nelinearnem odnosu, bomo kot model za odnos uporabili primernejšo (nelinearno) funkcijo. Predpostavko linearnosti lahko v primeru bivariatne regresije preverimo ali s pregledom običajnega razsevnega diagrama ali **rezidualnega grafa** (na x-os nanesemo napovedane vrednosti, na y-os pa reziduale). V Rju bomo za izris rezidualnega grafa uporabili funkcijo `plot()`.

```
# risanje rezidualnega grafa za regresijski model; rezultate regresije
# imamo shranjene v objektu reg; za lažji pregled smo pri rezidual = 0
# narisali referenčno črto s funkcijo abline(presečišče, nagib)

plot(reg$fitted.values, reg$residuals)
abline(0, 0)
```

Rezidualni graf lahko narišemo tudi s funkcijo `plot()`, pri čemer ji podamo objekt s shranjenimi rezultati regresije, npr. `plot(reg)`. Prvi od štirih grafov, ki jih vrne na ta način uporabljena funkcija, je rezidualni graf.

Normalnost porazdelitve rezidualov

Kršitev predpostavke o normalnosti porazdelitve rezidualov ima negativne posledice za veljavnost testov statistične značilnosti in natančnost napovednih intervalov. To predpostavko preverjamo s postopki, ki so predstavljeni v poglavju o normalni porazdelitvi (torej lahko uporabimo grafične metode, pregledamo relevantne opisne statistike in izvedemo enega ali več statističnih testov normalnosti rezidualov).

Homoscedastičnost

Homoscedastičnost drži v primeru, če so variance napak napovedi (rezidualov) enake pri vseh napovedanih vrednostih. Če poenostavimo – razpršenost točk v razsevnem ali rezidualnem grafu mora biti enaka vzdolž celotne regresijske premice. Kršitev te predpostavke ima negativne posledice za veljavnost testov statistične značilnosti in natančnost napovednih intervalov (kršitvi homoscedastičnosti pravimo heteroscedastičnost). Predpostavko homoscedastičnosti običajno preverjamo s pregledom rezidualnega grafa, lahko pa uporabimo tudi Breusch-

Paganov test heteroscedastičnosti. B-P test preverja veljavnost ničelne hipoteze, ki pravi, da homoscedastičnost drži (če testiramo hipotezo na ravni 5-odstotnega tveganja in je *p*-vrednost manjša od 0,05, zaključimo, da je bolj verjetna alternativna hipoteza, ki pravi, da drži heteroscedastičnost). V Rju lahko B-P test izvedemo s funkcijo `ncvTest()` iz paketa car ali s funkcijo `bptest()` iz paketa lmtest. Funkciji sta ekvivalentni, če pri `bptest()` privzeto vrednost argumenta studentize (TRUE) zamenjamo z logično vrednostjo FALSE.

```
# preverjanje predpostavke o homoscedastičnosti s funkcijama ncvTest() iz
# paketa car in bptest() iz paketa lmtest; rezultate regresijske analize
# imamo shranjene v objektu reg

ncvTest(reg)
bptest(reg, studentize = FALSE)
```

Analiza vplivnih točk

Točke, ki so na razsevnem ali rezidualnem grafu zelo oddaljene od regresijskega oblaka, lahko pomembno vplivajo na oceno regresijskih parametrov (in/ali na višino korelacije). Takim točkam pravimo **vplivne točke**. Nekateri avtorji ločijo pojma osamelci in vplivne točke, pri čemer osamelce identificiramo izključno na osnovi velikosti njihovih rezidualov, pri vplivnih točkah pa dodatno upoštevamo tudi oddaljenost točke od aritmetične sredine prediktorja.

Pri analizi rezidualov zaradi lažje interpretacije običajno pregledamo standardizirane reziduale (ZRESID), studentizirane reziduale (SRESID) ali izbrisane studentizirane reziduale (SDRESID). **Standardizirane reziduale** v Rju izračunamo s funkcijo `rstandard()`, ki ji podamo objekt z rezultati regresije. Običajno smo pozorni na standardizirane reziduale, ki so večji od |2|. V primerjavi s standardiziranimi reziduali pri **studentiziranih rezidualih** upoštevamo (v praksi pogosto veljavno) dejstvo, da imajo reziduali pri različnih vrednostih prediktorja različne variance. V Rju lahko izračunamo zgolj **izbrisane studentizirane reziduale** (angl. *deleted studentized residuals*), in sicer s funkcijo `rstudent()`, ki ji podamo objekt z rezultati regresije. Pri tvorbi izbrisanih studentiziranih rezidualov pri izračunu regresijskih parametrov za neko osebo izločimo podatke te osebe. Vse navedene reziduale običajno pregledamo na rezidualnem grafu, pri čemer na x-os nanesemo napovedane vrednosti, na y-os pa ZRESID, SRESID ali SDRESID.

Vplivnost posamezne točke/osebe je poleg njenega reziduala določena tudi z "ročico" (angl. *leverage*), ki je odvisna od oddaljenosti točke od aritmetične sredine prediktorja in števila vseh podatkov (glej enačbo 34). Točke, ki imajo velik rezidual in veliko ročico, bodo imele (v primerjavi z drugimi manj vplivnimi točkami) nesorazmerno velik vpliv na regresijski nagib.

Ročico (h_i) izračunamo po enačbi:

$$h_i = \frac{1}{N} + \frac{(\mathrm{X} - M_\mathrm{X})^2}{\sum(\mathrm{X} - M_\mathrm{X})^2} \tag{34}$$

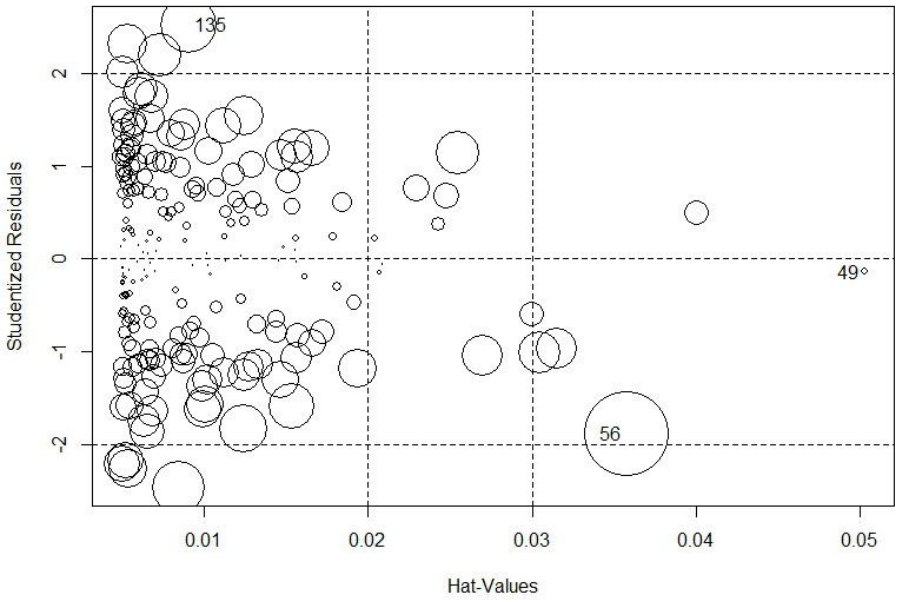
pri čemer je X vrednost osebe *i* na prediktorju. Ročica lahko zavzema vrednosti v intervalu od $1/N$ (ko je $\mathrm{X} = M_\mathrm{X}$) do 1. Praktično priporočilo pravi, da moramo biti pozorni na vrednosti ročice, ki so večje od $2(k + 1)/N$, pri čemer je k število prediktorjev, vendar je smotrno biti pozoren na vse ročice, ki (ne glede na svojo vrednost) opazno odstopajo od večine ročic. Ročice v Rju izračunamo s funkcijo `hatvalues()`, ki ji podamo objekt z rezultati regresije.

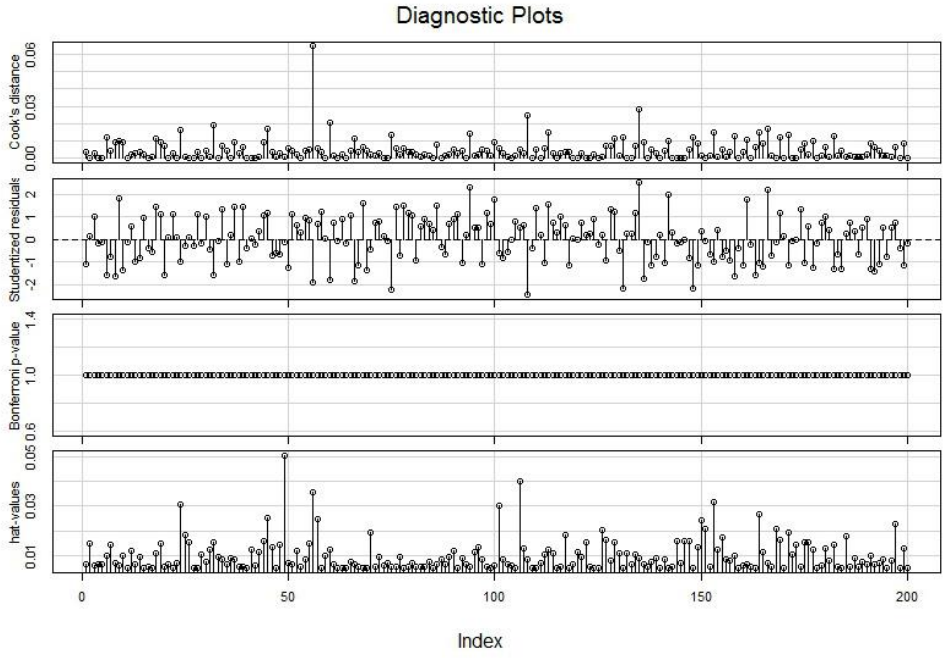
Pogosto uporabljana mera vplivnosti je **Cookova razdalja** (Cookov *D*), ki pri izračunu upošteva rezidual in ročico:

$$D_i = \frac{SRESID_i^2}{k + 1} * \frac{h_i}{1 - h_i} \tag{35}$$

V literaturi bomo našli priporočili, da moramo biti pozorni na osebe z $D > 1$ ali $D > 4/N$ (N je število oseb), vendar je ponovno boljše biti pozoren na vse Cookove razdalje, ki opazno odstopajo od drugih (vrednost Cookove razdalje je navzgor neomejena). Kot je razvidno iz enačbe 35, bo imela določena točka veliko Cookovo razdaljo, če ima velik rezidual, veliko ročico ali oboje. Cookove razdalje v Rju izračunamo s funkcijo `cooks.distance()`, ki ji podamo objekt z rezultati regresije.

Za analizo vplivnih točk imamo v Rju na voljo dve funkciji, ki nudita odličen pregled različnih mer vplivnosti. S funkcijo `influencePlot()` iz paketa car narišemo graf, na katerem so na x-osi vrednosti ročic, na y-osi izbrisani studentizirani reziduali, površine krogov, ki predstavljajo posamezne točke, pa so sorazmerne z velikostjo Cookove razdalje. S funkcijo `influenceIndexPlot()` iz paketa car narišemo (maksimalno štiri) navpično naložene grafe, ki prikazujejo Cookove razdalje, izbrisane studentizirane reziduale in ročice. Obema funkcijama moramo nujno podati le objekt z rezultati regresije. Spodaj sta prikazana primera slik, ki ju vrneta funkciji `influencePlot()` (zgoraj) in `influenceIndexPlot()`.





Na voljo so še druge mere vplivnosti posameznih točk; nekatere bomo na tem mestu le omenili:

– DFBETA$_i$ nam pove, za koliko se spremenita presečišče in regresijski nagib, če iz izračuna regresijskih parametrov izločimo osebo i. V Rju jih izračunamo s funkcijo `dfbeta()`, grafično pa jih prikažemo s funkcijo `dfbetaPlots()` iz paketa car.

- DFBETAS$_i$ je standardizirana DFBETA$_i$. V Rju jih izračunamo s funkcijo `dfbetas()`, grafično pa jih prikažemo s funkcijo `dfbetasPlots()` iz paketa car.
- DFFITS$_i$ nam pove, za koliko se spremeni napovedana vrednost osebe *i*, če iz izračuna regresijskih parametrov izločimo podatke te osebe. V Rju jih izračunamo s funkcijo `dffits()`.

Funkcije, ki smo jih spoznali v poglavju Korelacija (Pearsonov *r*) in bivariatna linearna regresija

```
abline()              # izris regresijske premice na predhodno narisan
                      # razsevni diagram
bptest()              # Breusch-Paganov test heteroscedastičnosti; paket
                      # lmtest
cooks.distance()      # izračun Cookovih razdalj
cor()                 # izračun korelacije/korelacijske matrike
cov()                 # izračun kovariance/kovariančne matrike
cov2cor()             # pretvorba kovariančne matrike v korelacijsko matriko
dfbeta()              # izračun regresijskih parametrov za osebo i, pri čemer
                      # iz izračuna izločimo podatke osebe i
dfbetaPlots()         # grafični prikaz regresijskih parametrov, ki jih vrne
                      # funkcija dfbeta(); paket car
dfbetas()             # izračun standardiziranih vrednosti, ki jih vrne
                      # funkcija dfbeta()
dfbetasPlots()        # grafični prikaz regresijskih parametrov, ki jih vrne
                      # funkcija dfbetas(); paket car
dffits()              # izračun spremembe v napovedani vrednosti osebe i, če
                      # iz izračuna regresijskih parametrov izločimo podatke
                      # osebe i
durbinWatsonTest()    # Durbin-Watsonov test; paket car
dwt()                 # dwt() = durbinWatsonTest(); paket car
dwtest()              # Durbin-Watsonov test; paket lmtest
geom_abline()         # izris regresijske premice na predhodno narisan
                      # razsevni diagram s funkcijo qplot() iz paketa ggplot2
hatvalues()           # izračun ročic za vse osebe (mera vplivnosti)
influenceIndexPlot()  # grafični prikaz različnih mer vplivnosti točk pri
                      # uporabi linearne regresije; paket car
influencePlot()       # grafični prikaz različnih mer vplivnosti točk pri
                      # uporabi linearne regresije; paket car
lm()                  # splošna funkcija za linearne modele
ncvTest()             # Breusch-Paganov test heteroscedastičnosti; paket
                      # car
pairs()               # risanje matrike razsevnih diagramov
pairs.panels()        # risanje matrike z različnimi grafičnimi
                      # prikazi/informacijami, npr. z razsevnimi diagrami,
                      # Pearsonovimi rji in histogrami
plot()                # generična funkcija za grafični prikaz objektov; lahko
                      # uporabimo za risanje razsevnega diagrama, rezidualnih
                      # grafov itd.
predict.lm()          # izračun napovedanih vrednosti in različnih
                      # intervalnih ocen na osnovi danega regresijskega
                      # modela
rstandard()           # izračun standardiziranih rezidualov
rstudent()            # izračun izbrisanih studentiziranih rezidualov
scatter.hist()        # risanje razsevnega diagrama z vrisanima histogramoma
                      # za porazdelitvi obeh spremenljivk; paket psych
scatterplot()         # risanje razsevnega diagrama z vrisanima boxplot
                      # prikazoma za porazdelitvi obeh spremenljivk; paket
                      # car
```

Drugi korelacijski koeficienti

Do te točke smo od korelacijskih koeficientov spoznali le Pearsonov *r*, ki ga lahko izračunamo le takrat, ko imamo opravka z dvema vsaj intervalnima spremenljivkama, ki sta v linearnem odnosu. V spodnji tabeli so prikazani primeri koeficientov korelacije, ki jih računamo takrat, ko vsaj ena od spremenljivk ni na (vsaj) intervalni merski ravni.

merska raven	nominalna	ordinalna
nominalna	• ϕ koeficient • r_{tet}; tetrahorični koeficient korelacije • C; koeficient kontingence • Cramerjev V (ϕ_C)	
ordinalna	• Kendallov τ Če je nominalna spremenljivka dihotomna!	• Spearmanov ρ • Kendallov τ • Kendallov W (koeficient konkordance oz. skladnosti med ocenjevalci)
intervalna ali razmernostna	• r_{pb}; točkovno-biserialni koeficient korelacije • r_b; biserialni koeficient korelacije • η (eta)	• Spearmanov ρ • Kendallov τ

Korelacije za kombinacijo intervalne in nominalne spremenljivke

Točkovno biserialni koeficient korelacije (r_{pb})

Za izračun korelacije med (vsaj) intervalno in dihotomno spremenljivko (ima samo dve kategoriji), uporabimo točkovno biserialni koeficient korelacije (r_{pb}). Točkovno biserialni koeficient je le posebna oblika Pearsonovega *r*, zato ga lahko izračunamo na enak način kot *r* ali po enačbah 36 ali 37 (če nimamo na voljo podatkov, ampak le opisne statistike):

$$r_{pb} = \frac{M_p - M_q}{\sigma_X}\sqrt{pq} \quad ALI \quad r_{pb} = \frac{M_p - M_X}{\sigma_X}\sqrt{\frac{p}{q}} \qquad (36, 37)$$

pri čemer sta M_p in M_q aritmetični sredini intervalne spremenljivke (X) pri skupinah, ki ju določata vrednosti dihotomne spremenljivke, σ_X je standardni odklon intervalne spremenljivke (z *N* v imenovalcu), M_X aritmetična sredina intervalne spremenljivke, *p* in *q* pa sta deleža obeh možnih odgovorov na dihotomni spremenljivki. Oznaki *p* in *q* lahko kategorijama pripišemo poljubno, a če je možno, *p* dodelimo deležu "pozitivnega izida/uspeha" (npr. pravilno rešena naloga), *q* pa deležu "negativnega izida/neuspeha" (npr. nepravilno rešena naloga).

Kot je razvidno iz enačbe 36, je višina točkovno biserialnega koeficienta premo sorazmerna z razliko med aritmetičnima sredinama intervalne spremenljivke obeh podskupin, njen predznak pa je določen z dodelitvijo *p* in *q* kategorijama dihotomne spremenljivke. Točkovno biserialni koeficient lahko zavzema vrednosti od –1 do 1, a njegova višina ni odvisna le od stopnje povezanosti spremenljivk, temveč tudi od razmerja med *p* in *q*, in sicer se z odstopanjem od razmerja 1:1 (pri katerem sta *p* = *q* = 0,50) maksimalna možna vrednost niža.

V Rju točkovno biserialni koeficient izračunamo na enak način kot Pearsonov *r*, torej z uporabo funkcije `cor()`. Če je dihotomna spremenljivka opredeljena kot faktor, jo moramo predhodno spremeniti v numerično spremenljivko.

```
# izračun točkovno biserialne korelacije s funkcijo cor(); po$var_dih je
# opredeljena kot faktor, zato smo jo v izračunu spremenili v numerično

cor(po$var_int, as.numeric(po$var_dih))
```

Biserialni koeficient korelacije (r_b)

Nekatere dihotomne spremenljivke so **naravno dihotomne**, npr. spol, nekatere pa t. i. **umetno dihotomne** (ali dihotomizirane, če neko intervalno/razmernostno spremenljivko naknadno dihotomiziramo), npr. dihotomna postavka na osebnostnem vprašalniku ali dihotomiziran dosežek na testu znanja (opravil/ni opravil). Če želimo zgolj opisati povezanost intervalne in dihotomne spremenljivke, vedno izračunamo točkovno biserialni koeficient korelacije, ne glede na to, ali imamo opravka z naravno ali umetno dihotomno spremenljivko.

Če obravnavamo korelacije med umetno dihotomno in intervalno spremenljivko, pri čemer je ta dihotomna spremenljivka v resnici zvezna in normalno porazdeljena, lahko uporabimo tudi biserialni koeficient korelacije (r_b). V takem primeru z biserialno korelacijo naslovimo (hipotetično) vprašanje, kolikšna bi bila **korelacija med to (trenutno) umetno dihotomno in vsaj intervalno** (zvezno) **spremenljivko, če bi bila ta dihotomna spremenljivka zvezna** (npr. če bi postavko na vprašalniku izmerili na zvezni, ne pa na dihotomni lestvici) **in normalno porazdeljena**. Tudi biserialna korelacija je odvisna od razmerja med *p* in *q*, in sicer se izračun tega koeficienta odsvetuje, če *p* ali *q* močno odstopa od 0,50 (npr. višji od 0,90 ali nižji od 0,10), saj lahko v tovrstnih primerih dobimo zelo izkrivljene vrednosti korelacije.

Biserialni koeficient korelacije lahko izračunamo po enačbah 38 ali 39:

$$r_b = \frac{M_p - M_q}{\sigma_X} * \frac{pq}{y} \quad ALI \quad r_b = r_{pb}\frac{\sqrt{pq}}{y} \qquad (38, 39)$$

pri čemer imajo oznake M_p, M_q, σ_X (z *N* v imenovalcu), *p* in *q* enak pomen kot v enačbah 36 in 37 za izračun točkovno-biserialne korelacije (glej opis enačb 36 in 37), *y* pa je ordinata normalne krivulje (oz. vrednost funkcije gostote verjetnosti) pri *z*-vrednosti pod/nad katero leži delež *p* ali *q* (*z*-vrednost lahko poiščemo pri *p* ali *q*, saj je normalna porazdelitev simetrična). Npr., vrednost ordinate normalne porazdelitve pri *p* = 0,6 (in *q* = 0,4) v Rju izračunamo kot `dnorm(qnorm(0.6))` ali `dnorm(qnorm(0.4))`.

V Rju za izračun biserialne korelacije (na osnovi surovih podatkov) izračunamo s funkcijo `biserial()` iz paketa psych. Funkciji podamo obe spremenljivki, pri čemer intervalno/razmernostno zvezno spremenljivko podamo kot prvi, dihotomno pa kot drugi argument.

```
# izračun biserialne korelacije s funkcijo biserial() med spremenljivkama
# po$var in po$umetna_dih

biserial(po$var, po$umetna_dih)
```

Eta koeficient korelacije (η)

Eta koeficient korelacije (η; ali tudi korelacijsko razmerje) uporabimo v primerih, ko želimo izračunati korelacijo med (vsaj) intervalno spremenljivko in nominalno spremenljivko z več kot dvema kategorijama (lahko je tudi dihotomna; v tem primeru velja $\eta = r_{pb}$). Običajno oz. najhitreje ga izračunamo tako, da najprej izvedemo enosmerno analizo variance za neponovljene meritve, tj. statistični test za preverjanje statistične značilnosti razlik med aritmetičnimi sredinami dveh ali več skupin udeležencev. Koeficient eta nato izračunamo kot razmerje med vsoto kvadratov za učinek ($SS_{učinek}$) in skupno vsoto kvadratov (SS_{skupaj}). Eta lahko zaseda vrednosti od 0 do 1. Razlaga analize variance (in različnih vsot kvadratov) presega namen tega priročnika, zato naj zadostuje sledeča razlaga. Višina koeficienta eta je odvisna od razlik med aritmetičnimi sredinami skupin (variance med skupinami) in od razpršenosti podatkov znotraj skupin (varianc podatkov znotraj skupin). Eta koeficient bo visok, če je varianca med skupinami velika, varianca znotraj skupin pa majhna, nizek pa v primeru, če je varianca med skupinami majhna, znotraj skupin pa velika.

V primerjavi s Pearsonovim *r* eta ne temelji na predpostavki o linearnosti odnosa med spremenljivkama, zato ga pogosto uporabljamo kot test linearnosti; če je pri računanju korelacije med dvema spremenljivkama eta koeficient višji od absolutne vrednosti Pearsonovega *r*, lahko sklepamo, da odnos med spremenljivkama ni linearen.

V Rju lahko eto najhitreje izračunamo s pomočjo funkcije `EtaSq()` iz paketa DescTools. Tej funkciji moramo podati le objekt, ki ga vrne funkcija `lm()`, pri čemer intervalno spremenljivko v model vključimo kot kriterij, nominalno pa kot prediktor. `EtaSq()` izračuna kvadriran eta koeficient, zato jo ugnezdimo v funkcijo `sqrt()`; v izpisu bo v tem primeru še vedno pisalo eta.sq (tj. *eta squared*), vendar bo vrednost pravilno prikazovala kvadratni koren ete na kvadrat, torej eto.

```
# izračun koeficienta eta s funkcijama lm() in EtaSq(); računamo korelacijo
# med spremenljivko po$dosezek (intervalna spremenljivka, npr. rezultati
# nekega testa) in po$stud_prog (nominalna spremenljivka z npr. 5
# kategorijami/študijskimi programi)

eta <- sqrt(EtaSq(lm(po$dosezek ~ po$stud_prog)))
```

Za pomoč pri interpretaciji eta koeficienta si lahko narišemo graf z aritmetičnimi sredinami skupin in intervale zaupanja za te aritmetične sredine (privzeto so 95-odstotni; intervali zaupanja med drugim odražajo tudi razpršenost znotraj skupin), in sicer s funkcijo `error.bars.by()` iz paketa psych.

```
# risanje črtnega grafa z aritmetičnimi sredinami različnih skupin; omogočili
# smo argument by.var, ker želimo imeti aritmetične sredine prikazane
# zaporedno v vodoravni, ne pa navpični smeri, poleg tega pa smo onemogočili
# argument eyes, ker želimo imeti okoli aritmetičnih sredin izrisane običajne
# ročaje za intervale zaupanja

error.bars.by(po$dosezek, group = po$stud_prog, by.var = TRUE, eyes = FALSE)
```

Korelacije za kombinacijo dveh (ali več) ordinalnih spremenljivk

Če imamo opravka z dvema ordinalnima spremenljivkama (ali eno ordinalno in eno vsaj intervalno), uporaba Personovega *r* ni več primerna. V takih primerih običajno izberemo ali Spearmanov ro (ρ) ali Kendallov tau (τ). Koeficiente za ordinalne spremenljivke uporabljamo tudi v primerih, ko sta obe spremenljivki na intervalni ali razmernostni ravni, a kršimo eno od pomembnejših predpostavk za izračun Pearsonovega *r*, npr., če odnos med spremenljivkama ni linearen (zato poleg koeficienta eta na podoben način uporabljamo tudi Spearmanov ρ za preverjanje linearnosti odnosa), če imamo v podatkih izrazite osamelce, ki močno vplivajo na višino korelacije ipd.

Spearmanov ρ

Spearmanov koeficient korelacije (ρ ali r_s) je neparametrična korelacija med rangi oz. rangiranimi vrednostmi dveh spremenljivk in ga uporabljamo za **oceno konsistentnosti smeri povezave oz. monotonosti povezave** med dvema spremenljivkama. Spremenljivki sta lahko v (pozitivnem ali negativnem) monotonem odnosu, ne glede na to, ali je ta odnos linearen ali nelinearen.

Spearmanov ρ lahko izračunamo kot Pearsonov *r* med absolutnimi rangi dveh spremenljivk in lahko zaseda vrednosti med –1 in 1. Interpretacija tega koeficienta je zato enaka kot pri Pearsonovem *r*, le da se nanaša na range, ne pa na surove vrednosti spremenljivk. Izračunamo ga lahko tudi po enačbi 40 (ta enačba velja za primere, ko se v podatkih ne pojavijo vezani rangi; v primeru vezanih rangov postane enačba precej kompleksnejša (zainteresirani bralci si jo lahko ogledajo v različnih statističnih učbenikih in spletnih straneh):

$$\rho = 1 - \frac{6\sum d^2}{N(N^2-1)} \tag{40}$$

pri čemer je *d* razlika med rangoma obeh spremenljivk pri določeni osebi. V Rju Spearmanov ρ izračunamo s funkcijo `cor()`, pri čemer privzeto vrednost argumenta method (tj. "pearson") zamenjamo z vrednostjo "spearman". Uporabimo lahko tudi funkcijo `SpearmanRho()` iz paketa DescTools.

```
# izračun Spearmanovega koeficienta korelacije med ordinalnima
# spremenljivkama po$var1 in po$var2 s funkcijo cor() in s funkcijo
# SpearmanRho() iz paketa DescTools

cor(po$var1, po$var2, method = "spearman")
SpearmanRho(po$var1, po$var2)
```

Kendallov τ

Kendallov τ je prav tako neparametrična mera korelacije za spremenljivke na ordinalni merski ravni (τ lahko uporabimo tudi v primeru kombinacije ordinalne in dihotomne spremenljivke!). Medtem ko lahko Spearmanov ρ obravnavamo (in interpretiramo) kot Pearsonov *r* (le da je ρ izračunan iz rangiranih vrednosti), Kendallov τ predstavlja verjetnost, in sicer **razliko med verjetnostjo, da so vrednosti obeh spremenljivk urejene v istem vrstnem redu, in verjetnostjo, da so vrednosti obeh spremenljivk urejene v povsem obratnem vrstnem redu**. V splošnem ga torej izračunamo kot razmerje med razliko med skladnimi in neskladnimi pari in vsemi možnimi (neponovljenimi) pari (tj. $N(N-1)/2$; glej enačbo 41). Npr., obravnavamo spremenljivki X in Y; če za par opazovanj oseb *i* in *j* (x_i, y_i) in (x_j, y_j) velja `(xi > xj & yi > yj) || (xi < xj & yi < yj)`, je ta par **skladen** (ujemanje), če pa za nek par velja `(xi > xj & yi < yj) || (xi < xj & yi > yj)`, je ta par **neskladen** (neujemanje). Paru opazovanj, za katerega velja `xi == xj || yi == yj`, rečemo **vezan par**. Koeficient znaša 1, če so vsi pari skladni (oz. je število skladnih parov enako številu vseh možnih neponovljenih parov), in –1, če so vsi pari neskladni.

Kendallov τ lahko izračunamo po enačbi 41, pri čemer ta enačba velja za primere, ko se v podatkih ne pojavijo vezani pari (tej obliki koeficienta rečemo **tau-a** oz. τ_a):

$$\tau = \frac{S}{S_{max}} = \frac{S}{\frac{N(N-1)}{2}} \tag{41}$$

pri čemer je *S* razlika med številom skladnih in neskladnih parov opazovanj. Če se v podatkih pojavijo vezani pari, postane enačba precej kompleksnejša (zainteresirani bralci si jo lahko ogledajo v različnih statističnih učbenikih in spletnih straneh). Koeficient τ, ki v izračunu upošteva vezane pare, pravimo **tau-b** (τ_b). Če v podatkih ni vezanih parov, velja $\tau_a = \tau_b$.

V Rju Kendallov τ (a ali b) izračunamo s funkcijo `cor()`, pri čemer privzeto vrednost argumenta method zamenjamo z vrednostjo "kendall". Uporabimo lahko tudi funkciji `KendallTauA()` in `KendallTauB()` iz paketa DescTools.

```
# izračun Kendallovega tau-b med dvema ordinalnima spremenljivkama s funkcijo
# cor() in KendallTauB() iz paketa DescTools

cor(po$var1, po$var2, method = "kendall")
KendallTauB(po$var1, po$var2)
```

Kendallov W (koeficient konkordance)

Kendallov W oz. koeficient konkordance je mera korelacije med več ordinalnimi spremenljivkami oz. mera skladnosti rangov več spremenljivk, zaradi česar ga pogosto poimenujemo tudi **koeficient skladnosti med ocenjevalci**; npr. več ocenjevalcev za več objektov (npr. učence, prehrambne izdelke) poda ocene/range na izbrani lastnosti (npr. ocena šolskega izdelka, všečnost). Kendallov W lahko zavzema vrednosti od 0 (popolna neskladnost med ocenjevalci) do 1 (popolna skladnost med ocenjevalci).

Tudi v primeru Kendallovega W imamo na voljo dve obliki enačb, pri čemer je ena namenjena primerom, v katerih se ne pojavijo vezane ocene/rangi (običajno ocenjevalcem ne dovolimo uporabe vezanih ocen/rangov), druga pa pri izračunu koeficienta W upošteva vezane range. Kendallov W za podatke brez vezanih rangov izračunamo po enačbi 42:

$$W = \frac{S}{S_{max}} = \frac{S}{\frac{m^2(N^3 - N)}{12}} \quad kjer\ je\ S = \sum_{j=1}^{N}\left(R_j - \frac{\sum R_j}{N}\right)^2 \tag{42}$$

pri čemer je R_j vsota rangov objekta *j*, *m* število ocenjevalcev, *N* pa število objektov. V Rju Kendallov W izračunamo s funkcijo `KendallW()` iz paketa DescTools. Funkciji moramo nujno podati matriko ali p.o., v katerih vrstice predstavljajo objekte, stolpci pa ocenjevalce. Če želimo, da izračun W po tej funkciji upošteva vezane range znotraj ocenjevalcev (oz. želimo uporabiti popravek za vezane range), moramo privzeto vrednost argumenta correct (tj. FALSE) zamenjati z vrednostjo TRUE.

```
# izračun Kendallovega W s popravkom za vezane range; ocene/rangi objektov
# (npr. 30 ocenjevalcev je podalo range všečnosti za 7 vrst sladoledov, pri
# čemer smo dovolili uporabo vezanih ocen/rangov) se nahajajo v podatkovnem
# okvirju podatki_W

KendallW(podatki_W, correct = TRUE)
```

Korelacije za kombinacijo dveh nominalnih spremenljivk

Fi koeficient (ϕ)

Koeficient ϕ je **mera povezanosti oz. odvisnosti dveh dihotomnih spremenljivk** (torej ga računamo na osnovi 2*2 kontingenčne tabele). Računamo ga lahko tudi na dveh nominalnih spremenljivkah z več kategorijami, a v teh primerih (če imata obe spremenljivki več kot 2 kategoriji) maksimalna možna vrednost koeficienta presega 1, zato raje uporabimo enega od drugih koeficientov korelacije, ki te težave nimajo (npr. Cramerjev V).

Tudi koeficient ϕ je posebna oblika Peasonovega *r*. Če imamo na voljo surove podatke, ga lahko v Rju izračunamo kot Pearsonov *r* s funkcijo `cor()`. Če smo dihotomni spremenljivki opredelili kot faktor, ju moramo za izračun korelacije spremeniti v numerični spremenljivki s funkcijo `as.numeric()` (enako kot za dihotomno spremenljivko pri izračunu točkovno-biserialnega koeficienta).

Če računamo ϕ koeficient na osnovi 2*2 kontingenčne tabele (npr. objekt, ki ga vrne funkcija `table()`, če ji podamo dve dihotomni spremenljivki), lahko uporabimo enačbi 43 ali 44.

$$\Phi = \frac{AD - BC}{\sqrt{(A+B)(C+D)(A+C)(B+D)}} \quad ALI \quad \Phi = \frac{p_{xy} - p_x p_y}{\sqrt{p_x q_x p_y q_y}} \tag{43, 44}$$

pri čemer so oznake (velike tiskane črke in različni deleži) nanašajo na vrednosti celic v spodnjih tabelah s frekvencami in deleži (odgovora 0 in 1 lahko npr. pomenita napačni in pravilni odgovor, odgovora NE in DA ipd.).

frekvence

x \ y	0	1
0	A	B
1	C	D

deleži

x \ y	0	1	skupaj
0			q_x
1		p_{xy}	p_x
skupaj	q_y	p_y	

Celici A in D predstavljata ujemanje odgovorov na obeh dihotomnih spremenljivkah (x = 0 in y = 0 ter x = 1 in y = 1), celici B in D pa neujemanje. Pri uporabi teh oznak moramo biti pozorni na to, da se položaj celic z oznakami A do D lahko z različno razvrstitvijo kategorij pri eni in/ali drugi spremenljivki spremeni. Pri izračunu koeficienta ϕ

na ta način moramo torej vedno preveriti, kateri celici predstavljata ujemanje oz. neujemanje. O (ne)ujemanju je smiselno govoriti le takrat, ko imata obe spremenljivki isti odgovorni alternativi (kot je prikazano v zgornjih kontingenčnih tabelah). Če imata spremenljivki različna možna odgovora (npr. zanima nas korelacija med spolom [Ž, M] in pravilnostjo določene naloge [0, 1]), oznake celicam kontingenčne tabele dodelimo arbitrarno, a pri tem pazimo, da sta oznaki A in D uvrščeni v celici ene diagonale, oznaki B in C pa v celici druge diagonale; v tovrstnih situacijah naj označevanje vedno sledi primeru v zgornji kontingenčni tabeli s frekvencami! Predznak koeficienta ϕ je v takih primerih določen z izbiro položaja teh oznak. Pri računanju korelacije med dvema dihotomnima spremenljivkama, ki sta resnično nominalni (npr. spol, zakonski stan ipd.), lahko predznak ignoriramo oz. upoštevamo le absolutno vrednost koeficienta, medtem ko ima v primeru dveh ordinalnih dihotomnih spremenljivk (npr. pravilnost naloge), predznak enak pomen kot npr. pri Pearsonovem *r*.

Koeficient ϕ lahko izračunamo tudi na osnovi χ^2 statistike (izračuna jo npr. funkcija `table2d_summary()` iz paketa vcd, če jo uporabimo za tvorbo kontingenčne tabele), in sicer po enačbi 45:

$$\Phi = \sqrt{\frac{\chi^2}{N}} \qquad \Phi_{max} = \sqrt{\frac{p_i q_j}{p_j q_i}} \quad kjer\ je\ p_j > p_i \qquad (45, 46)$$

pri čemer so p_i, q_i, p_j in q_j robni deleži spremenljivk *i* in *j* (p_i in q_i, sta robna deleža spremenljivke z manjšim *p*). Koeficient ϕ je lahko enak |1| le takrat, ko so robni deleži obeh spremenljivk enaki, torej, ko velja $p_i = p_j$ in (posledično) $q_i = q_j$. Z enačbo 46 izračunamo najvišji možni ϕ med dvema dihotomnima spremenljivkama in ustrezno prilagodimo interpretacijo koeficienta ϕ glede na vrednost ϕ_{max}. V Rju robne deleže najhitreje izračunamo s funkcijo `CrossTable()` iz paketa gmodels ali s funkcijo `table2d_summary()` iz paketa vcd (argumentu percentages moramo prirediti vrednost TRUE).

Če izračun koeficienta ϕ temelji na kontingenčni tabeli oz. objektu z jedrom kontingenčne tabele, ki jo vrne funkcija `table()`, lahko ϕ izračunamo tudi s funkcijo `phi()` iz paketa psych, funkcijo `assocstats()` iz paketa vcd ali s funkcijo `Assocs()` iz paketa DescTools. Slednji dve funkciji izračunata tudi nekatere druge korelacije. Koeficient ϕ, ki ga izračunata funkciji `assocstats()` in `Assocs()`, je vedno pozitiven (ϕ izračunata po enačbi 45), zato v primeru ordinalnih dihotomnih spremenljivk raje uporabimo funkcijo `phi()`.

```
# izračun koeficienta fi (na osnovi jedra 2*2 kontingenčne tabele) s
# funkcijami phi(), assocstats() in Assocs(); jedro kontingenčne tabele je
# shranjeno v objektu tabela

phi(tabela, digits = 3)          # spremenili smo privzeto natančnost
                                 # rezultata (tj. dve decimalni mesti) na tri
                                 # decimalna mesta
assocstats(tabela)
Assocs(tabela, verbose = 1)      # z verbose = 1 (privzeta vrednost je 2) smo
                                 # naročili najbolj jedrnat izpis rezultatov
```

Tetrahorični koeficient korelacije (r_{tet})

Za poročanje o povezanosti dveh dihotomnih spremenljivk vedno izračunamo koeficient ϕ (ali kateri drug primeren koeficient), ne glede na to, ali imamo opravka z dvema naravno ali umetno dihotomnima spremenljivkama. Lahko pa v primeru odnosa med dvema umetno dihotomnima spremenljivkama (ki sta v resnici zvezni in normalno porazdeljeni) naslovimo podobno (hipotetično) vprašanje kot z biserialno korelacijo. Tetrahorični koeficient korelacije (r_{tet}) nam torej pove, **koliko bi znašala korelacija med dvema (trenutno) dihotomnima spremenljivkama, če bi bili ti dve dihotomni spremenljivki zvezni in normalno porazdeljeni**. Izračun tetrahorične korelacije je brez uporabe statistične programske opreme zelo zapleten, zato ga običajno izračunamo po t. i. cos π približku (enačba 47).

$$r_{tet} \approx \cos\left(\frac{\pi}{1+\sqrt{\frac{AD}{BC}}}\right) \quad (47)$$

pri čemer se oznake od A do D nanašajo na celice (frekvence) v jedru 2*2 kontingenčne tabele (glej kontingenčno tabelo in opis te tabele v poglavju o ϕ koeficientu). Če funkcija, ki jo v določenem programu uporabimo za izračun kosinusa kota, uporablja stopinje namesto radianov, moramo namesto vrednosti π vstaviti 180 (π radianov = 180°; vse kotne funkcije v Rju uporabljajo radiane).

V Rju tetrahorično korelacijo izračunamo s funkcijo `tetrachoric()` iz paketa psych, pri čemer ji podamo jedro 2*2 kontingenčne tabele. V izpisu rezultata te funkcije je r_{tet} vedno prikazan na dve decimalni mesti natančno. Če želimo dostopati do natančnejše vrednosti, rezultat funkcije priredimo objektu in prikličemo natančnejšo vrednost prek spremenljivke $rho.

```
# izračun tetrahorične korelacije med dvema umetno dihotomnima
# spremenljivkama; jedro kontingenčne tabele je shranjeno v objektu tabela; v
# 2. drugem primeru smo s klicem spremenljivke $rho iz objekta, ki ga vrne
# funkcija tetrachoric(), izpisali natančnejšo vrednost (z več kot dvema
# decimalkama)

r_tet <- tetrachoric(tabela)
r_tet$rho
```

Opozorilo: ker je izračun tetrahorične korelacije po enačbi 47 približek prave vrednosti, se rezultata, ki ga vrneta enačba 47 in funkcija `tetrahoric()`, nekoliko razlikujeta.

Koeficient kontingence (C)

Koeficient kontingence (C) je mera korelacije med dvema nominalnima spremenljivkama, ki lahko imata več kot dve kategoriji (in tudi različno število kategorij). Tako kot ϕ, tudi C temelji na χ^2 statistiki. Vrednost koeficienta kontingence je lahko le pozitivna, nikoli pa ne more doseči 1, saj je *N* vedno večji od 1 (glej enačbo 48; možen razpon za C torej znaša $0 \leq C < 1$). Koeficient kontingence je navzgor omejen tudi z velikostjo kontingenčne tabele, in sicer se njegova maksimalna vrednost veča z velikostjo kontingenčne tabele. Zaradi te pomanjkljivosti nekateri avtorji priporočajo, da se C uporabi le pri večjih kontingenčnih tabelah (npr. vsaj 5*5), saj je pri manjših tabelah velikost korelacije podcenjena. Koeficient kontingence izračunamo po enačbi 48:

$$C = \sqrt{\frac{\chi^2}{N + \chi^2}} \quad (48)$$

V Rju lahko koeficient kontingence izračunamo s funkcijami `assocstats()` iz paketa vcd ter `Assocs()` in `ContCoef()` iz paketa DescTools.

Zaradi navedenih težav koeficienta kontingence se njegova uporaba odsvetuje (namesto C raje uporabimo Cramerjev V).

Cramerjev V

Cramerjev V, včasih poimenovan tudi Cramerjev ϕ (ϕ_C), je mera povezanosti dveh nominalnih spremenljivk, ki lahko imata več kot dve kategoriji (in tudi različno število kategorij). Tudi Cramerjev V temelji na χ^2 statistiki. Vrednost Cramerjevega V je lahko le pozitivna, ni odvisna od velikosti kontingenčne tabele in lahko doseže 1, če so robni deleži obeh spremenljivk enaki (enako kot velja za ϕ; možen razpon za Cramerjev V torej znaša $0 \leq V \leq 1$). Cramerjev V izračunamo po enačbi 49:

$$\text{Cramerjev V} = \sqrt{\frac{\chi^2}{N * \min(m-1, n-1)}} \tag{49}$$

pri čemer je *m* število vrstic, *n* pa število stolpcev (npr. pri kontingenčni tabeli velikosti 5*3 vrednost drugega člena v imenovalcu znaša *min*(5 – 1, 3 – 1) = 2). Kot je razvidno iz enačbe 49, pri 2*2 kontingenčnih tabelah velja Cramerjev V = ϕ.

V Rju lahko Cramerjev V izračunamo s funkcijami `assocstats()` iz paketa vcd ter `Assocs()` in `CramerV()` iz paketa DescTools.

Če bi od vseh predstavljenih mer povezanosti nominalnih spremenljivk, ki temeljijo na χ^2 statistiki, želeli uporabljati le enega, priporočam uporabo Cramerjevega V, saj je primeren za vse velikosti in oblike kontingenčnih tabel, hkrati pa ima manj pomanjkljivosti kot koeficient kontingence.

Funkcije, ki smo jih spoznali v poglavju Drugi korelacijski koeficienti

```
Assocs()            # izračun različnih mer povezanosti; paket DescTools
assocstats()        # izračun različnih mer povezanosti nominalnih
                    # spremenljivk; paket vcd
biserial()          # izračun biserialne korelacije; paket psych
ContCoef()          # izračun koeficienta kontingence; paket DescTools
CramerV()           # izračun Cramerjevega V; paket DescTools
error.bars.by()     # risanje grafa z aritmetičnimi sredinami in intervali
                    # zaupanja za prikazane arit. sredine; paket psych
EtaSq()             # izračun ete kvadrat; paket DescTools
KendallTauA()       # izračun Kendallovega tau-a; paket DescTools
KendallTauB()       # izračun Kendallovega tau-b; paket DescTools
KendallW()          # izračun Kendallovega W (koeficienta konkordance);
                    # paket DescTools
phi()               # izračun koeficienta fi; paket psych
SpearmanRho()       # izračun Spearmanovega ro; paket DescTools
tetrachoric()       # izračun tetrahoričnega koeficienta korelacije; paket
                    # psych
```