

# ITAP - Osnove uporabe R

A. Blejec

[andrej.blejec@nib.si](mailto:andrej.blejec@nib.si)

February 8, 2009

## Abstract

Osnove uporabe R .

### Povezave

Domača stran R projekta <http://www.r-project.org>

Bližnji repozitorij **CRAN** (Dunaj) <http://cran.at.r-project.org/>

Najnovejša verzija R na **CRAN** (Dunaj) <http://cran.at.r-project.org/bin/windows/base/release.htm>

Tinn-R GUI/Editor <http://www.sciviews.org/Tinn-R/>

WinEdt - ASCII urejevalnik (shareware) <http://www.winedt.com/>

Materiali o R na moji spletni strani <http://ablejec.nib.si/R> in <http://ablejec.nib.si/ITAP>

Moj e-mail naslov [andrej.blejec@nib.si](mailto:andrej.blejec@nib.si)

# 1 Osnove

## 1.1

### Elementarna aritmetika

Operatorji: + - / \* ^

```
> 2+3 # izračuna 2+3, vse za znakom # je komentar
[1] 5
> 2*3 # aritmetika v R: + - * /
[1] 6
> 2^3 # in potenca
[1] 8
> sqrt(4) # funkcije imajo v oklepajih argumente
[1] 2
> # sqrt(-4) # kar tako ne bo šlo :
> sqrt(as.complex(-4)) # R pozna kompleksna števila
[1] 0+2i
> sqrt(-4+0i) # isto kot zgoraj
[1] 0+2i
```

### Zaokroževanje

```
> x <- 7/6
> round(x)
[1] 1
> round(x, 3)
[1] 1.167
> floor(x)
[1] 1
> ceiling(x)
[1] 2
> format(1/2, nsmall=3)
[1] "0.500"
```

### Nekaj konstant

pi  $\pi$

NA manjkajoča vrednost

NaN "not a number"

Inf neskončno (1/0)

NULL prazna vrednost

letters male črke angleške abecede

LETTERS velike črke angleške abecede

Funkcije vrste `is.nekaj` in `as.nekaj`

```
> is.na(111)
[1] FALSE
> is.numeric(111)
[1] TRUE
> as.character(111)
[1] "111"
> (x <- c(1,2,NA,NULL,5,"bla")) # prisiljeni znaki!!
[1] "1"  "2"  NA    "5"  "bla"
> is.null(x)
[1] FALSE
> is.na(x)
[1] FALSE FALSE  TRUE FALSE FALSE
```

Rezultate lahko shranimo v imenovane objekte

```
> x <- 2+2
> y <- 2^3
> x
[1] 4
> y
[1] 8
> ime <- x # v imenih razlikuje velike in male črke
> IME <- y
> ls() # seznam objektov
[1] "ime" "IME" "x"  "y"
```

Logični vrednosti in operatorji

```
> x<y      # "logical"
[1] TRUE
> x>=20    # primerjalni operatorji: < <= > >= == !=
[1] FALSE
> x!=y     # x ni enak y
[1] TRUE
> z <- T   # logični konstanti T in F
> z
[1] TRUE
```

Priporočljivo je uporabljati polni besedi **TRUE** in **FALSE**

Tipi vrednosti: numeric, logical, character, factor, ordered

```

> is.character("Besedilo")
[1] TRUE
> is.numeric(x<y)
[1] FALSE
> as.numeric(x<y)
[1] 1
> as.character(x<y)
[1] "TRUE"
> library(chron)      # knjižnica datumskih funkcij
> apropos("date")    # ko že govorimo o datumih :)
[1] "-.Date"           ".__C__Date"
[3] "[.Date"           "[[.Date"
[5] "[<-.Date"         "+.Date"
[7] "as.character.Date" "as.data.frame.Date"
[9] "as.Date"           "as.Date.character"
[11] "as.Date.date"      "as.Date.dates"
[13] "as.Date.default"  "as.Date.factor"
[15] "as.Date.numeric"   "as.Date.POSIXct"
[17] "as.Date.POSIXlt"   "as.POSIXct.date"
[19] "as.POSIXct.Date"   "as.POSIXct.dates"
[21] "as.POSIXlt.date"   "as.POSIXlt.Date"
[23] "as.POSIXlt.dates"  "axis.Date"
[25] "c.Date"            "cut.Date"
[27] "date"              "dates"
[29] "diff.Date"         "format.Date"
[31] "is.numeric.Date"   "ISOdate"
[33] "ISOdatetime"       "julian.Date"
[35] "Math.Date"         "mean.Date"
[37] "months.Date"       "Ops.Date"
[39] "print.Date"        "quarters.Date"
[41] "rep.Date"          "round.Date"
[43] "seq.Date"          "seq.dates"
[45] "split.Date"        "summary.Date"
[47] "Summary.Date"     "Sys.Date"
[49] "trunc.Date"        "update"
[51] "update.default"    "update.formula"
[53] "update.packages"   "update.packageStatus"
[55] "weekdays.Date"    "xtfrm.Date"

```

## 2 Podatkovne strukture

### 2.1 vector

vector

Funkcija *combine* `c(...)` zlepi vrednosti v vektor:

```

> c(1,2,3,10,20,4)      # posamezne vrednosti lahko zlepim v vektor
[1] 1 2 3 10 20 4
> x <- c(1,2,3,10,20,4)  # vektor shranim v objekt x
> x                     # izpiše objekt po vrednostih
[1] 1 2 3 10 20 4
> print(x)              # isto kot zgoraj,
[1] 1 2 3 10 20 4
> length(x)             # vektorji imajo dolžino
[1] 6

```

## Vektorska aritmetika

R uporablja aritmetiko na komponentah vektorjev

```

> y <- c(3, 4, 5, 11, 12, 13)  # še en vektor
> x
[1] 1 2 3 10 20 4
> y
[1] 3 4 5 11 12 13
> x + y                       # vsota
[1] 4 6 8 21 32 17
> x * y                       # in produkt po komponentah
[1] 3 8 15 110 240 52
> round(sqrt(x),2)           # tudi funkcije delujejo na komponente
[1] 1.00 1.41 1.73 3.16 4.47 2.00
> x < y                       # pa primerjave tudi
[1] TRUE TRUE TRUE TRUE FALSE TRUE

```

## Dopolnjevanje krajših operandov

Če operanda nista enako dolga, se krajši podaljša na potrebno dolžino.

```

> x * 2                       # množenje s konstanto
[1] 2 4 6 20 40 8
> x * c(2, -2)                # če zmanjka vrednosti za račun, se ponovijo
[1] 2 -4 6 -20 40 -8
> rep(c(2, -2), 3)           # v tem primeru takole
[1] 2 -2 2 -2 2 -2

```

Če daljši ni monogokratnik krajšega, se izpiše opozorilo (**warning**)

Zaporedja števil: `seq`

```

> args(seq.default)      # argumenti funkcije
function (from = 1, to = 1, by = ((to - from)/(length.out - 1)),
         length.out = NULL, along.with = NULL, ...)
NULL
> 1:6                    # zaporedje števil (by = 1)
[1] 1 2 3 4 5 6
> 6:1                    # padajoče (by = -1)
[1] 6 5 4 3 2 1
> seq(1, 6, 2)           # by = 2
[1] 1 3 5
> seq(6, 1, -2)         # by = -2
[1] 6 4 2
> seq(1, 6, length=3)   # znana dolžina
[1] 1.0 3.5 6.0
> seq(1, 18, along=x)   # vzdolž vektorja x
[1] 1.0 4.4 7.8 11.2 14.6 18.0

```

Ponavljanje vrednosti: rep

```

> rep(2, 6)
[1] 2 2 2 2 2 2
> rep(c(1, 2), times=3)
[1] 1 2 1 2 1 2
> rep(c(1, 2), each=3)
[1] 1 1 1 2 2 2
> rep(c(1, 2), length=6)
[1] 1 2 1 2 1 2
> expand.grid(1:2, 1:3)
  Var1 Var2
1     1     1
2     2     1
3     1     2
4     2     2
5     1     3
6     2     3

```

Izbiranje komponent vektorjev

```

> (x <- c(1,2,3,1,2,4))
[1] 1 2 3 1 2 4
> x[4]          # četrty element
[1] 1
> x[3:5]       # 3., 4. in 5. element
[1] 3 1 2
> x[c(1,4,2)] # pa še malo mešamo :-)
[1] 1 1 2
> x[-1]        # prvega spustimo
[1] 2 3 1 2 4

```

### Spreminjanje vrednosti izbranim komponentam vektorjev

```

> x
[1] 1 2 3 1 2 4
> x[4] <- 40          # posameznemu elementu
> x
[1] 1 2 3 40 2 4
> x[3:5] <- c(33,44,55); x # 3. do 5. element; izpis
[1] 1 2 33 44 55 4
> x[c(1,4,2)] <- 11   # vsem enako vrednost
> x
[1] 11 11 33 11 55 4
> x[-1] <- 22        # vsem razen prvemu
> x
[1] 11 22 22 22 22 22

```

### Logične operacije in izbira vrednosti

```

> x
[1] 11 22 22 22 22 22
> y
[1] 3 4 5 11 12 13
> x>5
[1] TRUE TRUE TRUE TRUE TRUE TRUE
> which(x>5)
[1] 1 2 3 4 5 6
> y[x>5]          # elementi y, ki ustrezajo legi x>5
[1] 3 4 5 11 12 13
> (x>3)|(y<2) # logične operacije z vektorji & (and) | (or) in ! (not)
[1] TRUE TRUE TRUE TRUE TRUE TRUE

```



## Vrednosti lahko vpišemo s tipkovnico

Vnos zaključimo s tipko `Enter` v prazni vrstici

```
z <- scan()
11
22
33
44
55
66
<Enter>

> (z <- c(11,22,33,44,55,66)) # priredi in izpiši
[1] 11 22 33 44 55 66
> y
[1] 3 4 5 11 12 13
> x
[1] 11 22 22 22 22 22
```

## 2.2 matrix

### Lepljenje vektorjev

```
> cbind(x,y,z) # vektorje lahko zlepimo v tabelo, kot stolpce (c)
      x y z
[1,] 11 3 11
[2,] 22 4 22
[3,] 22 5 33
[4,] 22 11 44
[5,] 22 12 55
[6,] 22 13 66

> rbind(x,y,z) # ali pa kot vrstice (r)
      [,1] [,2] [,3] [,4] [,5] [,6]
x      11  22  22  22  22  22
y       3   4   5  11  12  13
z      11  22  33  44  55  66
```

### matrix in transpozicija

`t(...)` transponira matriko (zamenja indeksa vrstic in stolpcev)

```

> u <- cbind(x,y,z)
> dim(u)          # u ima dve dimenziji (6 vrstic, 3 stolpce)
[1] 6 3
> is.matrix(u)   # zato je matrika
[1] TRUE
> t(u)
      [,1] [,2] [,3] [,4] [,5] [,6]
x      11  22  22  22  22  22
y       3   4   5  11  12  13
z      11  22  33  44  55  66

```

### Preoblikovanje vektorja v matriko

```

> c(x,y,z)       # tole pa zlepi vektorje v en sam dolg vektor
[1] 11 22 22 22 22 22  3  4  5 11 12 13 11 22 33 44 55
[18] 66
> matrix(c(x,y,z),ncol=3) # ki ga lahko preoblikujemo v matriko,
      [,1] [,2] [,3]
[1,]    11     3    11
[2,]    22     4    22
[3,]    22     5    33
[4,]    22    11    44
[5,]    22    12    55
[6,]    22    13    66

```

### Preoblikovanje vektorja v matriko

```

> matrix(c(x,y,z),nrow=6) # ce mu predpišem št. vrstic ali stolpcev
      [,1] [,2] [,3]
[1,]    11     3    11
[2,]    22     4    22
[3,]    22     5    33
[4,]    22    11    44
[5,]    22    12    55
[6,]    22    13    66
> matrix(c(x,y,z),nrow=6,byrow=T) # matriko lahko napolnim po vrsticah
      [,1] [,2] [,3]
[1,]    11    22    22
[2,]    22    22    22
[3,]     3     4     5
[4,]    11    12    13
[5,]    11    22    33
[6,]    44    55    66

```

## 2.3 data.frame

data.frame - podatkovni okvir

```
> data.frame(x,y,z) # podobno cbind
  x y z
1 11 3 11
2 22 4 22
3 22 5 33
4 22 11 44
5 22 12 55
6 22 13 66

> cbind(x,y,z)
      x y z
[1,] 11 3 11
[2,] 22 4 22
[3,] 22 5 33
[4,] 22 11 44
[5,] 22 12 55
[6,] 22 13 66

> names(cbind(x,y,z)) # cbind ne naredi imen
NULL
```

Imena spremenljivk

```
> v <- data.frame(x=x,drugi=y,z=z) # data frame pa
> dimnames(v) # imena vseh dimenzij
[[1]]
[1] "1" "2" "3" "4" "5" "6"

[[2]]
[1] "x"      "drugi" "z"

> names(v) # imena stolpcev (navadno spremenljivk)
[1] "x"      "drugi" "z"

> dimnames(v)[[2]]
[1] "x"      "drugi" "z"
```

read.table prebere podatke v data.frame

```
> dbmi <- read.table("../data/bmi.txt",header=T)
> head(dbmi)
  id spol lroj starost visina teza
1  1   Z 1941      19  165.0 48.8
2  2   Z 1941      18  155.7 48.9
3  3   Z 1941      19  153.6 55.5
4  4   Z 1941      18  163.4 69.7
5  5   Z 1941      19  164.7 60.8
6  6   Z 1941      19  163.6 58.6
```

## 2.4 list

Najbogatejša struktura: list

```
> w <- list("Mešane komponente",1:3,dva=x,mat=u) # kratek vektor, dolg vektor
> w

[[1]]
[1] "Mešane komponente"

[[2]]
[1] 1 2 3

$dva
[1] 11 22 22 22 22 22

$mat
      x y z
[1,] 11 3 11
[2,] 22 4 22
[3,] 22 5 33
[4,] 22 11 44
[5,] 22 12 55
[6,] 22 13 66
```

**list:** Izbiranje komponent

Komponente lahko izberem z navedbo lege `[[...]]`

```
> w[[3]]
[1] 11 22 22 22 22 22
```

ali pa imena (`$`)

```
> w$dva
[1] 11 22 22 22 22 22
```

**list:** Izbiranje komponent

Šlo bi tudi z `[...]`

Enojni oklepaji vrnejo komponento kot **list**:

```
> w[3]
$dva
[1] 11 22 22 22 22 22
> mode(w[3]);mean(w[3])
[1] "list"
[1] NA
```

```

> w[[3]]
[1] 11 22 22 22 22 22
> mode(w[[3]]);mean(w[[3]])
[1] "numeric"
[1] 20.16667

```

### 3 Vzorec podatkov

#### Izbira vzorca

```

> N <- dim(dbmi)[1]
> n <- 20
> (select <- sample(1:N,n))
 [1] 166 277 1005 49 673 417 855 1115 576 1418
[11] 225 483 115 211 578 465 863 610 947 265
> data <- dbmi[select,]
> dimnames(data)

[[1]]
 [1] "166" "277" "1005" "49" "673" "417" "855"
 [8] "1115" "576" "1418" "225" "483" "115" "211"
[15] "578" "465" "863" "610" "947" "265"

[[2]]
 [1] "id" "spol" "lroj" "starost" "visina"
 [6] "teza"
> dimnames(data)[[1]] <- 1:n

```

#### Nova spremenljivka

```

> attach(data)
> bmi <- teza/(visina/100)^2
> data <- cbind(data,bmi) # podatkom dodamo novo spremenljivko

```

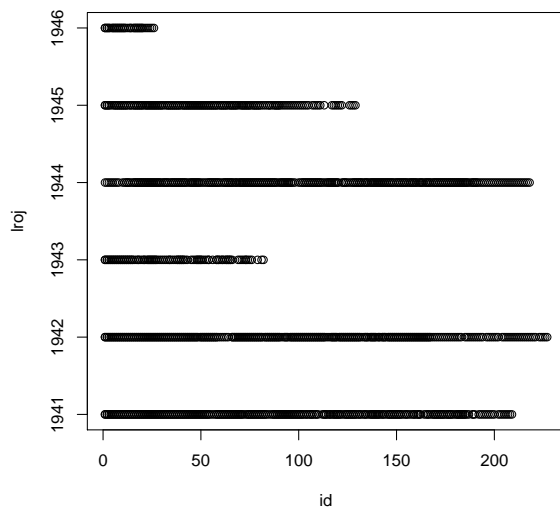
#### Vzorec

	id	spol	lroj	starost	visina	teza	bmi
1	183	Z	1941	19	157.0	55.6	22.55670
2	111	Z	1942	19	162.2	55.2	20.98154
3	137	M	1942	19	171.4	59.3	20.18520
4	49	Z	1941	20	170.1	65.4	22.60316
5	113	Z	1945	20	155.0	50.7	21.10302
6	41	Z	1944	20	166.3	61.1	22.09310
7	175	M	1941	19	181.5	66.2	20.09577
8	27	M	1943	20	169.5	73.0	25.40876
9	10	Z	1945	20	161.5	56.5	21.66224
10	20	M	1946	18	177.4	72.0	22.87838
11	55	Z	1942	19	166.3	53.2	19.23655
12	120	Z	1944	19	173.4	59.0	19.62248
13	126	Z	1941	19	149.9	60.5	26.92478
14	40	Z	1942	18	163.4	55.8	20.89922
15	12	Z	1945	19	167.0	53.8	19.29076
16	101	Z	1944	20	169.5	61.5	21.40601
17	185	M	1941	19	184.1	79.7	23.51531
18	46	Z	1945	20	152.8	46.7	20.00185
19	72	M	1942	19	169.8	63.5	22.02411
20	99	Z	1942	19	155.5	58.1	24.02787

Nenavaden id?

id se ponavlja znotraj leta

```
> plot(dbmi[,c("id", "lroj")])
```



Lahki

```

> which(teza<60)
[1] 1 2 3 5 9 11 12 14 15 18 20
> data[teza<60,]
  id spol  lroj starost visina teza      bmi
1  183   Z 1941      19  157.0  55.6 22.55670
2  111   Z 1942      19  162.2  55.2 20.98154
3  137   M 1942      19  171.4  59.3 20.18520
5  113   Z 1945      20  155.0  50.7 21.10302
9   10   Z 1945      20  161.5  56.5 21.66224
11  55   Z 1942      19  166.3  53.2 19.23655
12 120   Z 1944      19  173.4  59.0 19.62248
14  40   Z 1942      18  163.4  55.8 20.89922
15  12   Z 1945      19  167.0  53.8 19.29076
18  46   Z 1945      20  152.8  46.7 20.00185
20  99   Z 1942      19  155.5  58.1 24.02787

```

## 4 Tipi rezultatov

Vrste vrnjenih rezultatov

- Posamezne vrednosti
- `vector`
- `matrix`
- `list`

### 4.1 Vrednosti

Posamezne vrednosti

Rezultat enostavnih funkcij je posamezna vrednost

```

> x <- visina
> xBar <- mean(x)
> xBar
[1] 166.18
> str(xBar)
num 166

```

Rezultate lahko uporabimo v analizi

```

> x-xBar
[1] -9.18 -3.98  5.22  3.92 -11.18  0.12 15.32
[8]  3.32 -4.68 11.22  0.12  7.22 -16.28 -2.78
[15]  0.82  3.32 17.92 -13.38  3.62 -10.68
> sum(x-xBar)      # skoraj nič :)
[1] -8.526513e-14

```

## 4.2 vector

vector

```
> xq <- quantile(x, seq(0, 1, .25))
> xq
      0%      25%      50%      75%     100%
149.900 160.375 166.650 170.425 184.100
> str(xq)
Named num [1:5] 150 160 167 170 184
- attr(*, "names")= chr [1:5] "0%" "25%" "50%" "75%" ...
> length(xq)
[1] 5
```

### Izbor posameznih elementov

Posamezne elemente izberemo z navedbo indeksov v oglatih oklepajih [...]

```
> xq[2] # drugi element
      25%
160.375
> xq[2:4] # kvartili
      25%      50%      75%
160.375 166.650 170.425
> xq[-1] # brez minimuma
      25%      50%      75%     100%
160.375 166.650 170.425 184.100
```

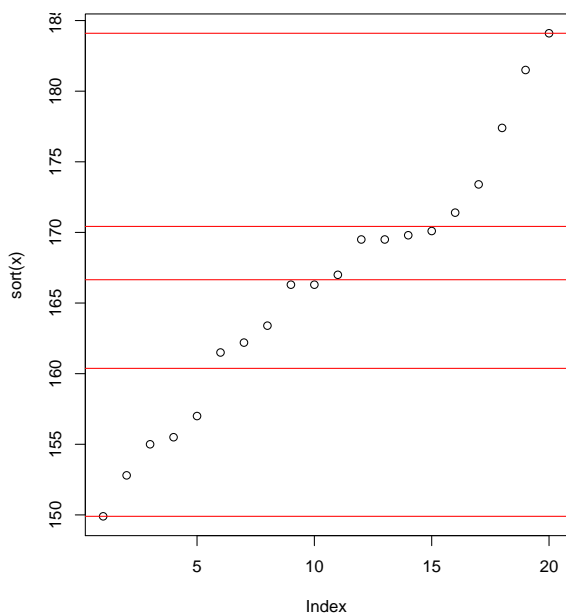
### Razrez podatkov



```

> par(mar=c(4, 4, 0, 2))
> plot(sort(x))
> abline(h=xq, col="red")

```



## Uporaba vektorja

Razvrstitev v razrede z mejami **xq**

```

> xClass <- cut(x, xq)
> xClass

 [1] (150,160] (160,167] (170,184] (167,170] (150,160]
 [6] (160,167] (170,184] (167,170] (160,167] (170,184]
[11] (160,167] (170,184] <NA>          (160,167] (167,170]
[16] (167,170] (170,184] (150,160] (167,170] (150,160]
Levels: (150,160] (160,167] (167,170] (170,184]

> str(xClass)
Factor w/ 4 levels "(150,160]","(160,167]","(167,170]","(170,184]": 1 2 4 3 1 2 4 3 2 4 2 4 NA 2 3 3 4

> as.numeric(xClass)

 [1]  1  2  4  3  1  2  4  3  2  4  2  4 NA  2  3  3  4
[18]  1  3  1

```

## 4.3 matrix

Frekvenčna tabela

```

> tabela <- table(xClass, spol)
> dim(tabela)
[1] 4 2
> tabela
      spol
xClass  M Z
(150,160] 0 4
(160,167] 0 5
(167,170] 2 3
(170,184] 4 1

```

## Pravokotne strukture

- **matrix** – 2 dimenziji
- **array** – več dimenzij
- **data.frame**

V matrikah so lahko le podatki enakega tipa (števila ali pa faktorji), v **data.frame** pa so lahko stolpci različnih tipov.

**data.frame** je klasična struktura statističnih podatkov: enote v vrsticah, spremenljivke v stolpcih (enake dolžine)

## Izbira delov pravokotnih struktur

Iz pravokotnih struktur dobimo vrednosti z navedbo številke vrstice in stolpca, ...

```

> tabela[3:4, 1]
(167,170] (170,184]
      2      4
> tabela[ -4, ] # brez četrte vrstice
      spol
xClass  M Z
(150,160] 0 4
(160,167] 0 5
(167,170] 2 3
> tabela[ , 2] # samo drugi stolpec
(150,160] (160,167] (167,170] (170,184]
      4      5      3      1

```

array

```
> array(letters,c(2,3,2))
```

```
, , 1
```

```
      [,1] [,2] [,3]
[1,] "a"  "c"  "e"
[2,] "b"  "d"  "f"
```

```
, , 2
```

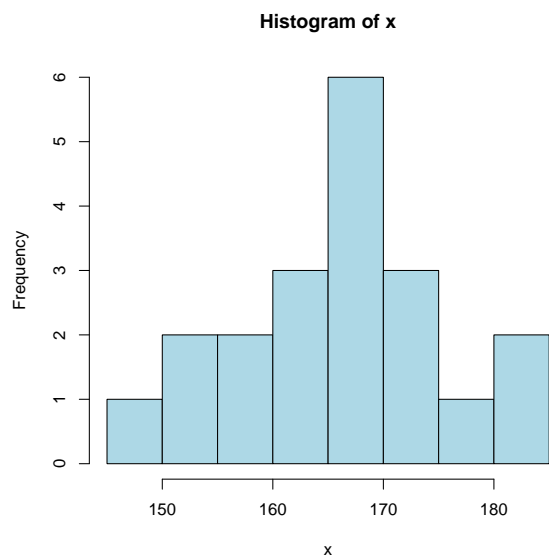
```
      [,1] [,2] [,3]
[1,] "g"  "i"  "k"
[2,] "h"  "j"  "l"
```

## 4.4 list

### list

Funkcija `hist` vrne uporabne informacije o histogramu

```
> opis <- hist(x,col="lightblue")
```



### Struktura list

```

> names(opis)
[1] "breaks"      "counts"      "intensities"
[4] "density"     "mids"        "xname"
[7] "equidist"

> str(opis)
List of 7
 $ breaks      : num [1:9] 145 150 155 160 165 170 175 180 185
 $ counts      : int [1:8] 1 2 2 3 6 3 1 2
 $ intensities: num [1:8] 0.01 0.02 0.02 0.03 0.06 ...
 $ density     : num [1:8] 0.01 0.02 0.02 0.03 0.06 ...
 $ mids        : num [1:8] 148 152 158 162 168 ...
 $ xname       : chr "x"
 $ equidist    : logi TRUE
- attr(*, "class")= chr "histogram"

```

### Posamezni deli list

```

> names(opis)
[1] "breaks"      "counts"      "intensities"
[4] "density"     "mids"        "xname"
[7] "equidist"

```

Izbor komponente: z navedbo imena

```

> opis$counts
[1] 1 2 2 3 6 3 1 2

```

Izbor komponente: z navedbo številke komponente

```

> opis[[2]]
[1] 1 2 2 3 6 3 1 2

```

### Dopolnjen histogram

Objekt `opis` vsebuje vse, kar rabimo za risanje (`class:histogram`)

```
> plot (opis, col="lightblue", main="Dopolnjen histogram")  
> points (opis$mids, opis$counts, pch=16, col="red", cex=2)  
> lines (opis$mids, opis$counts, col="blue", lwd=3, lty=2)
```

