

Elements of R usage

A. Blejec

andrej.blejec@nib.si

July 31, 2010

Abstract

Elements of R usage.

Links

R project home page <http://www.r-project.org>

Nearest repository **CRAN** (Austria) <http://cran.at.r-project.org/>

Newest R version on **CRAN** (Austria) <http://cran.at.r-project.org/bin/windows/base/release.htm>

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

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

R materials on my web page <http://ablejec.nib.si/R> in <http://ablejec.nib.si/ITAP>

My e-mail address andrej.blejec@nib.si

1 Basics

1.1

Elementary arithmetic

Operators: + - / * ^

```
> 2+3 # calculate 2+3, everything after # is a comment
[1] 5
> 2*3 # arithmetics in R: + - * /
[1] 6
> 2^3 # power
[1] 8
> sqrt(4) # Functions have arguments in brackets ()
[1] 2
> # sqrt(-4) # kar tako ne bo šlo :
> sqrt(as.complex(-4)) # R knows complex numbers
[1] 0+2i
> sqrt(-4+0i) # same as above
[1] 0+2i
```

Rounding

```
> 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"
```

Some constants

pi π

NA Missing value (Not Available)

NaN "not a number"

Inf infinite (1/0)

NULL empty, null value

letters lower case characters

LETTERS upper case characters

Functions: `is.something` in `as.something`

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

Results can be saved as named objects

```
> x <- 2+2
> y <- 2^3
> x
[1] 4
> y
[1] 8
> ime <- x # R is case sensitive
> IME <- y
> ls() # list of objects
[1] "ime" "IME" "x"  "y"
```

Logical values and operators

```
> x<y      # "logical"
[1] TRUE
> x>=20    # comparison operators: < <= > >= == !=
[1] FALSE
> x!=y     # x not equal y
[1] TRUE
> z <- T   # logical constants T in F
> z
[1] TRUE
```

It is advisable to use words **TRUE** in **FALSE**

Types: numeric, logical, character, factor, ordered

```

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

```

2 Data structures

2.1 vector

vector

Function *combine* `c(...)` combines values into a vector:

```

> c(1,2,3,10,20,4)      # values combined
[1] 1 2 3 10 20 4
> x <- c(1,2,3,10,20,4)  # vector saved to x
> x                      # type values in x
[1] 1 2 3 10 20 4
> print(x)              # same as above,
[1] 1 2 3 10 20 4
> length(x)             # vectors have length: n of elements
[1] 6

```

Vector arithmetics

R uses arithmetic component by component

```

> 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                        # sum
[1] 4 6 8 21 32 17
> x * y                        # and product by components
[1] 3 8 15 110 240 52
> round(sqrt(x),2)            # functions work on components
[1] 1.00 1.41 1.73 3.16 4.47 2.00
> x < y                        # so do comparisons
[1] TRUE TRUE TRUE TRUE FALSE TRUE

```

Reuse of short operands

If operands have different lengths, shorter one is reused.

```

> x
[1] 1 2 3 10 20 4
> x * 2                        # multiply by constant
[1] 2 4 6 20 40 8
> x * c(2, -2)                 # if there is not enough values, shorter are reused
[1] 2 -4 6 -20 40 -8
> rep(c(2, -2), 3) # like this
[1] 2 -2 2 -2 2 -2

```

If length of longer is not a multiple of the shorter one, we get a (**warning**)

Sequence of numbers: seq

```
> args(seq.default)      # function arguments
function (from = 1, to = 1, by = ((to - from)/(length.out - 1)),
         length.out = NULL, along.with = NULL, ...)
NULL
> 1:6                    # sequence of numbers (by = 1)
[1] 1 2 3 4 5 6
> 6:1                    # decreasing (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)   # known length
[1] 1.0 3.5 6.0
> seq(1, 18, along=x)   # make it as long as length of x
[1] 1.0 4.4 7.8 11.2 14.6 18.0
```

Repeating values: 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
```

Element selection

```

> (x <- c(1,2,3,1,2,4))
[1] 1 2 3 1 2 4
> x[4]          # fourth element
[1] 1
> x[3:5]       # 3., 4. in 5. element
[1] 3 1 2
> x[c(1,4,2)] # mixed positions :-
[1] 1 1 2
> x[-1]       # don't use the first one
[1] 2 3 1 2 4

```

Changing values to selected elements

```

> x
[1] 1 2 3 1 2 4
> x[4] <- 40          # only one element
> 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   # replace selected values with 11
> x
[1] 11 11 33 11 55 4
> x[-1] <- 22        # change all but the first element
> x
[1] 11 22 22 22 22 22

```

Logical operations and selection

```

> x <- c(2, 6, 3, 10, 1, 2)
> y <- c(3, 4, 5, 11, 12, 13)
> x>5
[1] FALSE TRUE FALSE TRUE FALSE FALSE
> which(x>5)
[1] 2 4
> y[x>5]          # elements of y, corresponding to values of x>5
[1] 4 11

```

Logical operators: & (and) | (or) in ! (not)

```

> (x>3)|(y<2) # logical operations with vectors
[1] FALSE TRUE FALSE TRUE FALSE FALSE

```


Manual data typing

Finnish the entry of data with `Enter` in an empty line

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

> (z <- c(11,22,33,44,55,66)) # print assigned value note outer brackets
[1] 11 22 33 44 55 66
> y
[1] 3 4 5 11 12 13
> x
[1] 2 6 3 10 1 2
```

2.2 matrix

Vector binding

```
> cbind(x,y,z) # vectors bind as columns (c)
      x y z
[1,]  2 3 11
[2,]  6 4 22
[3,]  3 5 33
[4,] 10 11 44
[5,]  1 12 55
[6,]  2 13 66

> rbind(x,y,z) # or as rows (r)
      [,1] [,2] [,3] [,4] [,5] [,6]
x         2    6    3   10    1    2
y         3    4    5   11   12   13
z        11   22   33   44   55   66
```

matrix and transpose

`t(...)` matrix transpose (exchange row and column indices)

```

> u <- cbind(x,y,z)
> dim(u)          # u has two dimesions (6 rows, 3 columns)
[1] 6 3
> is.matrix(u)   # u is a matrix
[1] TRUE
> t(u)
      [,1] [,2] [,3] [,4] [,5] [,6]
x         2    6    3   10    1    2
y         3    4    5   11   12   13
z        11   22   33   44   55   66

```

Reshaping vector into matrix

```

> c(x,y,z)       # one long vector
[1]  2  6  3 10  1  2  3  4  5 11 12 13 11 22 33 44 55
[18] 66
> matrix(c(x,y,z),ncol=3) # is now a matrix with 3 columns
      [,1] [,2] [,3]
[1,]     2     3    11
[2,]     6     4    22
[3,]     3     5    33
[4,]    10    11    44
[5,]     1    12    55
[6,]     2    13    66

```

Reshaping vector into matrix

```

> matrix(c(x,y,z),nrow=6) # 6 rows
      [,1] [,2] [,3]
[1,]     2     3    11
[2,]     6     4    22
[3,]     3     5    33
[4,]    10    11    44
[5,]     1    12    55
[6,]     2    13    66
> matrix(c(x,y,z),nrow=6,byrow=T) # Fill matrix by rows
      [,1] [,2] [,3]
[1,]     2     6     3
[2,]    10     1     2
[3,]     3     4     5
[4,]    11    12    13
[5,]    11    22    33
[6,]    44    55    66

```

2.3 data.frame

data.frame - main stat data structure

```
> data.frame(x,y,z) # similar as cbind
  x y z
1 2 3 11
2 6 4 22
3 3 5 33
4 10 11 44
5 1 12 55
6 2 13 66

> cbind(x,y,z)
      x y z
[1,]  2 3 11
[2,]  6 4 22
[3,]  3 5 33
[4,] 10 11 44
[5,]  1 12 55
[6,]  2 13 66

> names(cbind(x,y,z)) # cbind drops the names
NULL
```

Variable names (column names)

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

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

> names(v) # column names (variable names)
[1] "x"      "drugi" "z"

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

read.table: read data as data.frame

```
> dbmi <- read.table("../data/bmi.txt", header=T)
> head(dbmi) # show first few lines
  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

The richest structure: list

```
> w <- list("Mixed components",1:3,dva=x,mat=u) # short vector, long vector
> w
[[1]]
[1] "Mixed components"

[[2]]
[1] 1 2 3

$dva
[1] 2 6 3 10 1 2

$mat
      x y z
[1,]  2 3 11
[2,]  6 4 22
[3,]  3 5 33
[4,] 10 11 44
[5,]  1 12 55
[6,]  2 13 66
```

list: component selection

Selection by position `[[...]]`

```
> w[[3]]
[1] 2 6 3 10 1 2
```

or name (`$`)

```
> w$dva
[1] 2 6 3 10 1 2
```

list: component selection

Also possible: `[...]`

Single brackets return component as **list**:

```
> w[3]
$dva
[1] 2 6 3 10 1 2
> mode(w[3]);mean(w[3])
[1] "list"
[1] NA
```

```

> w[[3]]
[1] 2 6 3 10 1 2
> mode(w[[3]]);mean(w[[3]])
[1] "numeric"
[1] 4

```

3 Data sample

Sample selection

```

> 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

```

New variable

```

> attach(data)
> bmi <- teza/(visina/100)^2 # teza = weight, visina = height
> data <- cbind(data,bmi) # new variable bmi added to the data frame

```

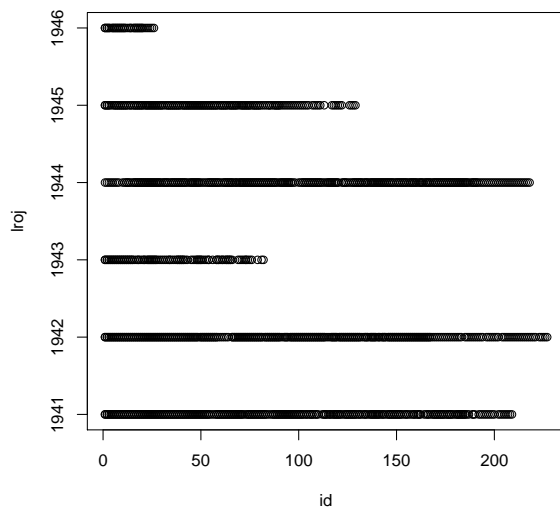
Sample

	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

Strange id?

id is repeated within a year

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



Light (teza)

```

> 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 Type of results

Types of returned values

- single values
- vector
- matrix
- list

4.1 Single values

Posamezne vrednosti

Results of elementary functions are single values

```

> x <- visina <- rnorm(20,170,8)
> xBar <- mean(x)
> xBar
[1] 171.2762
> str(xBar)
num 171

```

Intermediate results can be used in the analysis

```

> x-xBar
 [1] 15.16655001  6.00565072 -4.34805674
 [4]  8.65181935  0.30278832 -8.96655844
 [7] 13.29657158 -13.45650231  0.06145109
[10] -11.77971669  4.63303367 -0.86156690
[13] -5.26933163  5.28075710 -5.62170120
[16] -1.28649033  0.98040813  0.74958449
[19] -4.44928166  0.91059145

> sum(x-xBar)      # almost nothing :)
[1] -1.989520e-13

```

4.2 vector

vector

```

> xq <- quantile(x, seq(0, 1, .25))
> xq
      0%      25%      50%      75%     100%
157.8197 166.6219 171.4584 176.0712 186.4428

> str(xq)
Named num [1:5] 158 167 171 176 186
- attr(*, "names")= chr [1:5] "0%" "25%" "50%" "75%" ...

> length(xq)
[1] 5

```

Selection of elements

Individual elements can be selected by vector of indices insquare brackets [...]

```

> xq[2]      # second element
      25%
166.6219

> xq[2:4]    # elements 2, 3, 4 (quartiles)
      25%      50%      75%
166.6219 171.4584 176.0712

> xq[-1]     # no minimum
      25%      50%      75%     100%
166.6219 171.4584 176.0712 186.4428

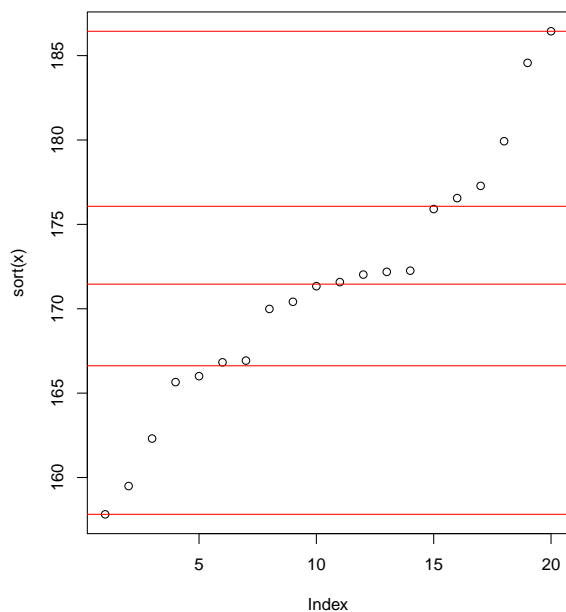
```

Cutting numerical variables


```

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

```



Vector use

Distribute values into classes `xq`

```

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

 [1] (176,186] (176,186] (167,171] (176,186] (171,176]
 [6] (158,167] (176,186] <NA>         (167,171] (158,167]
[11] (171,176] (167,171] (158,167] (176,186] (158,167]
[16] (167,171] (171,176] (171,176] (167,171] (171,176]
Levels: (158,167] (167,171] (171,176] (176,186]

> str(xClass)
Factor w/ 4 levels "(158,167]", "(167,171]", ...: 4 4 2 4 3 1 4 NA 2 1 3 2 1 4 1 2 3

> as.numeric(xClass)

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

```

4.3 matrix

Frequency table

```

> tabela <- table(xClass, spol)
> dim(tabela)
[1] 4 2
> tabela
      spol
xClass  M Z
(158,167] 1 3
(167,171] 2 3
(171,176] 1 4
(176,186] 1 4

```

Rectangular structures

- **matrix** – 2 dimensions
- **array** – 3 or more than dimensions
- **data.frame**

All elements of the same type (numbers, characters, logical), in **data.frame** columns can be of different type.

data.frame classical structure of stat data: cases (units) in rows, variables in columns (same length!)

Selection from rectangular structures

Values are extracted referencing rows and columns

```

> tabela[3:4, 1]
(171,176] (176,186]
      1      1
> tabela[-4, ] # skip row no 4
      spol
xClass  M Z
(158,167] 1 3
(167,171] 2 3
(171,176] 1 4
> tabela[, 2] # just column 2
(158,167] (167,171] (171,176] (176,186]
      3      3      4      4

```

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"

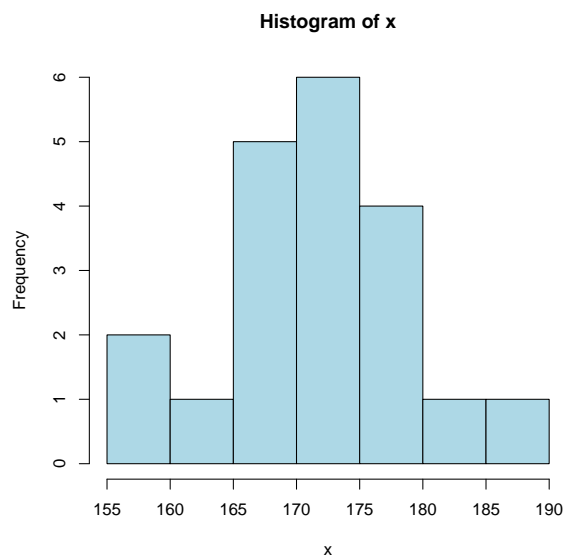
```

4.4 list

list

Function `hist` returns useful information about the histogram.

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



Structure list

Returned values have a list structure. Why?

```

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

> str(desc)
List of 7
 $ breaks      : num [1:8] 155 160 165 170 175 180 185 190
 $ counts      : int [1:7] 2 1 5 6 4 1 1
 $ intensities: num [1:7] 0.02 0.01 0.05 0.06 0.04 ...
 $ density     : num [1:7] 0.02 0.01 0.05 0.06 0.04 ...
 $ mids        : num [1:7] 158 162 168 172 178 ...
 $ xname       : chr "x"
 $ equidist    : logi TRUE
- attr(*, "class")= chr "histogram"

```

Elements of list

```

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

```

Select list component: referenced by name

```

> desc$counts
[1] 2 1 5 6 4 1 1

```

Select list component: referenced by component number

```

> desc[[2]]
[1] 2 1 5 6 4 1 1

```

Histogram with added polygon

Object `desc` has details needed for plotting (`class:histogram`)

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

