Osnove

solutions by Alen Kahteran

Contents

Programska orodja za statistiko

- splošni programski jeziki, low-level (nizko-nivojski), hitrejši (Fortran, C/C++, Java, Python,...)
- bolj matematično statistični, uporabniku prijaznejši (Mathematica, Matlab/Octave, R, Julia)

R

- uporablja vektorje, matrike in delo z njimi
- open-source
- najbolj uporabljan jezik v statistiki
- uporabniki dograjujejo njegovo funkcionalnost s programskimi paketi
- instalacija z osnovnim grafičnim vmesnikom
  - RStudio za dodatne funkcionalnosti
  - IDE (integrated development environment)
- dokumentacija dosegljiva prek RStudia, na spletu, forum *stackexchange* (omejite se z R)

RStudio, Rmarkdown

cheatsheet **rstudio-IDE**

- ukazna vrstica, R Markdown izpis
- help, knjižnice, pregled grafov, pregled datotek v mapi
- gobalno okolje (Global Environment), zgodovina ukazov
- datoteke (bližnjice <Ctrl> + <Enter>, <Ctrl> + <Shift> + <Enter>)

cheatsheet: **rmarkdown**

- .Rmd datoteka
- reproducible research (ponovljive raziskave)
- koda v R med tremi enojnimi narekovaji in crko `r`: “‘{r } <code> “‘
- R koda znotraj besedila med enima enojnima narekovajema in crko `r`: ‘{r <code>}‘
- enačbe kot v LaTeX-u (cheatsheet **LaTeX**)
- za kreiranje PDF dokumentov: `install.packages('tinytex'); tinytex::install_tinytex()`

Naloge

- Naredite nov .Rmd dokument kot je tukajle.

Osnovni gradniki v R

- znak pred komentarjem

? - znak za izpis pomoči (ali pa uporaba funkcije `help()`)

cheatsheet: **base-r**

Vrste stavkov

Izrazi

```
2+4
```

Prirejanja

```
a1 = 1+6
a2 <- "vaje"
a3 = a1 == a2
```

`a1` je objekt, ki je shranjen v globalnem okolju.

Logične primerjave: `==, !=, <, >, is.na(), is.null()`

Operacije: `+, -, /, *, %%, %/%` (zadnja dva: deljenje po modulu in celoštevilsko deljenje)

(Osnovne) vrste objektov

Skalar

```
as1 = 1 # numeric
as2 = "Uporabna statistika" # character
as3 = TRUE # logical
str(as3)
class(as3)
is.numeric(a1)
as.numeric(a3)
ls() # list of objects
```

Datume bomo obravnavali kasneje.

Naloge

- Kakšna je numerična vrednost `as3`?
- Kako izbrišemo vse, kar je trenutno v globalnem okolju? Poglejte v help funkcije `rm()`.

```r
# setting global seed for reproducability
set.seed(8)

# solutions

# as3 = TRUE
as.numeric(as3)
```

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

```r
# equal to
as.numeric(TRUE)
```

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

```r
# a way to remove all global variables (found in ?rm)
rm(list=ls())
```

Posebne vrednosti:

- `NA` - not available
- `pi` - π
- `NaN` - not a number
- `Inf` - infinite value
- `NULL` - brez vrednosti, prazno
- `TRUE` in `FALSE` (okrajšavi `T` in `F`)

Vektor

```r
av1 = c(1,2,3,4,5)
av2 = vector(mode="character",length=4)
av2
class(av2)
av2[1] = "\u017Div\u00E9" # Žive
av2[2] = "naj"
av2[4] = "narodi"
av2
av2[-1]
av3 = 1:10
length(av3)
```

Naloge

- Naredite vektor `av4`, v katerem so števila in znaki. Izpišite ga na zaslon. Kakšnega tipa je?
- Vektor `av3` skrajšajte:
  - na prve tri znake,
  - na zadnja dva znaka,
  - izberite le vsak drugi znak,
  - vektor naj vsebuje samo lihe številke
- Vektor `av2` podaljšate za naslednjo vrstico Zdravljice.
- Kaj se zgodi, če seštejemo `av1` in `av3[1:2]`?

Seštevamo lahko tudi vektorje neenakih dolžin, vendar moramo biti pri tem **zelo pazljivi**! (npr. prištevamo skalar)

```r
# solutions
```

```r
# av4 creation
av4 = c(8, 22, 9, "Alen", "Kahteran")
# printing av4
print(av4)
```

```
## [1] "8"        "22"       "9"        "Alen"     "Kahteran"
```

```r
# type of av4
typeof(av4)
```

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

```r
# shortening av3 to first 3 elements
av3[1:3]
```

```
## [1] 1 2 3
```

```r
# shortening av3 to last 2 elements
av3[(length(av3) - 1):length(av3)]
```

```
## [1]  9 10
```

```r
# only selecting every second element in av3
av3[seq(2, 10, 2)]
```

```
## [1]  2  4  6  8 10
```

```r
# only selecting odd numbers in av3
av3[av3 %% 2 == 1]
```

```
## [1] 1 3 5 7 9
```

```r
# imputing 3rd element of av2 as it's missing
av2[3] <- "vsi"

# adding the next row of Zdravljica
next_row <- c("ki", "hrepené", "dočakat", "dan")

for (i in 5:8) {
    av2[i] <- next_row[i-4]
}

print(av2)
```

```
## [1] "Živé"     "naj"      "vsi"      "narodi"   "ki"       "hrepené" "dočakat"
## [8] "dan"
```

```r
# what happens when summing av1 and av3[1:2]
print(sum(av1, av3[1:2])) # we get the sum of all elements
```

```
## [1] 18
```

```r
# as I assume the above wasn't the anwser that was expected
av1 + av3[1:2]
```

```
## [1] 2 4 4 6 6
```

```r
# this above returns a warning message that we can't sum objects with different
# lengths which are not a multiple of eachother.
# what it does it expands the second vector to match the length and then sums.
# the warning is raised as it has to slice the second vector at the end.
```

```r
# this would work (without warning) as only 1 element of av3 was added to each element
# of av1 (objects match by multiple length)
av1 + av3[1]
```

```
## [1] 2 3 4 5 6
```

```r
# this could work as well
av1 + av3[1:5]
```

```
## [1]  2  4  6  8 10
```

```r
# this sums av1 and av3[1:5] element-wise (first with first, second with second, etc.)

# in cases where some object lengths are a multiple of eachother R automatically
# multiplies the shorter one by the multiplier so the lenghts match.
```

Matrika

```r
am1 = matrix(c(1:6),nrow=2)
am1
am1[2,3] = 10
am1
am1[2,] # second row
is.matrix(am1)
dim(am1)
# dimensions can be added to the vector
dim(av3) = c(1,10)
# matrix as binded vectors
am2 = cbind(av1,av1)
class(am2)
str(am2)
colnames(am2)
am3 = rbind(av1,av1,rev(av1))
am1[1,]
am1[1,,drop=FALSE] # preserve dimensions
```

Naloge

- Seštejte dve matriki `am1`.
- Zmnožite `am1`z 2.
- Kaj dobite z naslednjim izrazom: `am1*c(1,2)` ? Zakaj?
- Kaj dobite z naslednjim izrazom: `am1*c(1,2,3)` ? Zakaj?
- Kaj dobite, če po vrsticah skupaj združite `av1` in vektor `1:8`? Zakaj?
- Kaj dobite, če želite po stolpcih združiti `av1` in `av2`? Zakaj?
- Kaj omogoča parameter `byrow` v funkciji `matrix`? Zapišite primer.

Vektorsko množenje in množenje matrik (`%*%`):

```r
# solutions

# sums element-wise
am1 + am1
```

```
##      [,1] [,2] [,3]
## [1,]    2    6   10
## [2,]    4    8   20
```

```r
# multiplies 2 with every element
am1 * 2
```

```
##      [,1] [,2] [,3]
## [1,]    2    6   10
## [2,]    4    8   20
```

```r
# multiplies 1st row with 1 and 2nd row with 2
am1*c(1, 2)
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    4    8   20
```

```r
# multiplies 1st column with 1, 2nd column with 2, 3rd column with 3
am1*c(1, 2, 3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    9   10
## [2,]    4    4   30
```

```r
# why? I assume R checks the lengths, as you can't multiply column-wise
# in this particular example if the vector we're multiplying with is shorter
# than the matrix size.

# returns a matrix of size 2x8, as av1 vector is length 5, it expands it
# so the lenghts match (adds first three elements to the end).
# why? it's somewhat senseless to bind if the lenghts don't match.
# also, if the lenghts are a multiple of eachother, it does the same but dosen't
# raise a warning.
rbind(av1, 1)
```

```
##     [,1] [,2] [,3] [,4] [,5]
## av1    1    2    3    4    5
##        1    1    1    1    1
```

```r
# this does similarly as above, but in this case as the types don't match
# it converts the numeric type to character type as the data in matrix can only
# be of one type. (also expands the first vector as it is shorter)
cbind(av1, av2)
```

```
##      av1 av2
## [1,] "1" "Živé"
## [2,] "2" "naj"
## [3,] "3" "vsi"
## [4,] "4" "narodi"
## [5,] "5" "ki"
## [6,] "1" "hrepené"
## [7,] "2" "dočakat"
## [8,] "3" "dan"
```

```r
# returns a matrix of size 2x3 where the selected method of filling is column-wise
# which means first fills the values sequentially in first column, then second, etc.
matrix(c(1, 1, 1, 2, 2, 2), nrow = 2)
```

```
##      [,1] [,2] [,3]
## [1,]    1    1    2
## [2,]    1    2    2
```

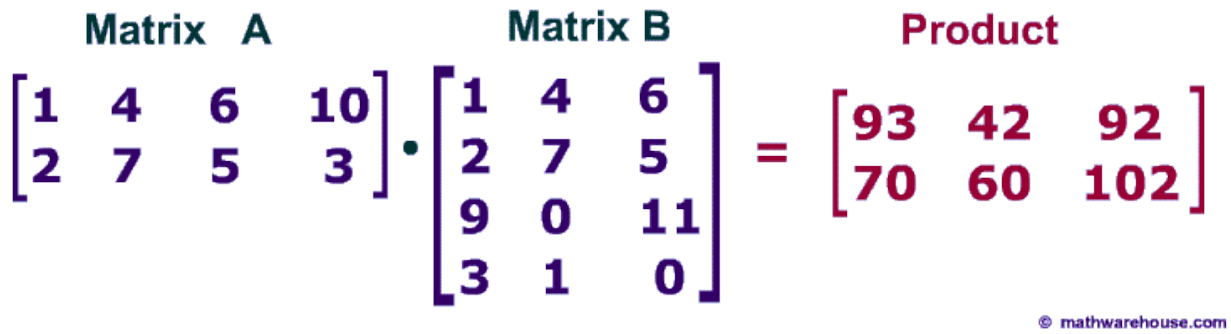
```r
# returns a matrix of size 2x3 (byrow=FALSE by default)
# similarly as above, byrow setting just changes the filling method to rows.
matrix(c(1, 1, 1, 2, 2, 2), nrow = 2, byrow=TRUE)
```

```
##      [,1] [,2] [,3]
## [1,]    1    1    1
## [2,]    2    2    2
```

```r
am1 %*% t(am1) # transpose
av1 %*% t(av1)
as.matrix(av1)
av1 %*% av1
t(av1) %*% av1
```

Naloge

- Izračunajte naslednji produkt (preverite, ali je rezultat na desni pravilen)



+ Preverite v help-u, kaj naredi funkcija `solve`. (`?solve`) + Uporabite funkcijo `solve`, da boste dobili kvadratno matriko X dimenzije 4x4, za katero velja, da `Y %*% X = Y`. Matrika Y naj bo katerakoli matrika dimenzije 4x4. Zakaj dobite tak rezultat?

```r
# solutions

# defining matrix A and matrix B
matr_a <- matrix(c(1, 4, 6, 10,
                   2, 7, 5, 3),
                 nrow=2,
                 byrow=TRUE)

print(matr_a)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    6   10
## [2,]    2    7    5    3
```

```r
matr_b <- matrix(c(1, 4, 6,
                   2, 7, 5,
                   9, 0, 11,
                   3, 1, 0),
                 nrow=4,
                 byrow=TRUE)

print(matr_b)
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    6
## [2,]    2    7    5
## [3,]    9    0   11
## [4,]    3    1    0
```

```
# product
prod_matr <- matr_a %*% matr_b

print(prod_matr)
```

```
##      [,1] [,2] [,3]
## [1,]   93   42   92
## [2,]   70   60  102
```

```
# result from image
result <- matrix(c(93, 42, 92,
                   70, 60, 102),
                 nrow=2,
                 byrow=TRUE)

# do all values match?
all(prod_matr == result)
```

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

```
# solve() solves the equation a %*% x = b for x, where b can be either a vector or a matrix.
# (from ?solve)
matr_y <- matrix(c(1,  2,  3,  4,
                   5,  6,  7,  8,
                   9,  10, 11, 12,
                   13, 14, 15, 16),
                 nrow=4,
                 byrow=TRUE)

print(matr_y)
```

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

```
# this case can't be solved as the matrix is not invertable
# solve(matr_y, matr_y)
# commented due to otherwise .rmd not compiling.

# as this inverts one of the matrixes to solve the problem the matrix must be invertable
# one of the cases when a matrix is not invertable is when the determinant is equal to 0.
det(matr_y)
```

```
## [1] 4.733165e-30
```

```
# 4.733165e-30 - which is basically 0, within machine precision (floating point error).

# example of a matrix with non-zero determinant
matr_y2 <- matrix(c( 1,  0, 2, 0,
```

```
                    -4, -3, 1, 5,
                    -6,  2, 4, 1,
                     0,  1, 3, 0),
                  nrow=4,
                  byrow=TRUE)

print(matr_y2)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    2    0
## [2,]   -4   -3    1    5
## [3,]   -6    2    4    1
## [4,]    0    1    3    0
```

```
# determinant
det(matr_y2)
```

```
## [1] -32
```

```
# usage of solve again.
ret_matrix <- solve(matr_y2, matr_y2)

# returns an identity matrix, as this is the only possible solution anyway.
# This is the only matrix which if any matrix is multiplied by the identity matrix
# it returns the "original" matrix

print(ret_matrix)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    0
## [2,]    0    1    0    0
## [3,]    0    0    1    0
## [4,]    0    0    0    1
```

Seznam

```
al1 = list(ime=c("Anton","Janez"),priimek=c("Novak","Trilar"),starost=c(67,58,34))
al1[1]
al1$priimek
al1$starost[3]
al1[[3]]
al1[[3]][1]
names(al1)
```

Seznami so uprabni za združevanje več (različnih tipov) objektov v skupni objekt. Rezultati funkcij so ponavadi seznami različnih objektov.

Podatkovni okvir (osnovna statistična tabela)

```
#df1 = data.frame(ime=c("Anton","Janez"),priimek=c("Novak","Trilar"),starost=c(67,58,34))
df1 = data.frame(ime=c("Anton","Janez"),priimek=c("Novak","Trilar"),starost=c(67,58))
df1
df1$starost
df1[,"priimek"]
df1[[2]]
```

```r
dim(df1)
names(df1)
rownames(df1) = c("oseba1","oseba2")
df1
str(df1)
class(df1)
class(df1$ime)
```

Faktor je poseben tip podatka. Gre za opis kategorialnih podatkov, ki jim lahko priredimo opis posameznih kategorij in povemo, ali so urejeni.

```r
af1 = c(0,0,0,0,1,1,1,1) # 4 men, 4 women, numeric
af2 = as.factor(af1)
af2
as.numeric(af2) # factor  starts with 1, levels sorted
af3 = factor(af1,levels = c(0,1),labels=c("M","F")) # ordered=TRUE for ordered factors
af3
```

Naloge

- Iz `af3` izbrišite vse moške in shranite rezultat v `af4`. Kakšen je izpis? Katere vrednosti ima lahko faktor?
- Ali lahko faktor dodamo novo kategorijo? V `af3` poskusite dodati kategorijo `O` - otrok. Kaj se zgodi? Kako boste dodali otroka, da bo to tudi nova kategorija?
- Poženite ukaz `data()`.
- Za podatkovje `USArrests` ugotovite, kaj so statistične enote in preverite, kakšne spremenljivke imamo na voljo (katerega tipa, kaj pomenijo).
- Prikažite le imena držav.
- Izberite iz podatkov vse države, ki imajo vsaj 70% populacije urbane. Koliko jih je?
- Iz podatkov izbrišite spremenljivko `Rape`.
- Kaj se zgodi, če `USArrests` spremenimo v matriko? Zakaj?

```r
# solutions

# remove all "M" values.
af4 <- af3[af3 != "M"]

print(af4)
```

```
## [1] F F F F
## Levels: M F
```

```r
# output returns only "F" values. Factor can only hold specific values which are usually
# predetermined. In this case af4 can hold "M" and "F".

# can we add a new factor category? yes
# labels and levels needed to be changed as they now have different values.
factor(af3,
       levels=c("M", "F", "O"),
       labels=c("M", "F", "O")
       )
```

```
## [1] M M M M F F F F
## Levels: M F O
```

```r
# if we wanted to do the same from af1, and number 2 represent a child ("O")
factor(af1,
```

```
       levels=c(0, 1, 2),
       labels=c("M", "F", "O")
       )
```

```
## [1] M M M M F F F F
## Levels: M F O
```

```
# data() returns all data sets within R.

### USArrests

# the statistical units are US states.

# Murder represents Murder arrests per 100.000 people.
# Assault represents Assault arrests per 100.000 people.
# UrbanPop represents percent of urban population.
# Rape represents Rape arrests per 100.000 people.

# all of the values are numeric (double or integer)
typeof(USArrests$Murder)
```

```
## [1] "double"
```

```
typeof(USArrests$Assault)
```

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

```
typeof(USArrests$UrbanPop)
```

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

```
typeof(USArrests$Rape)
```

```
## [1] "double"
```

```
# displaying US state names.
rownames(USArrests)
```

```
##  [1] "Alabama"        "Alaska"         "Arizona"        "Arkansas"
##  [5] "California"     "Colorado"       "Connecticut"    "Delaware"
##  [9] "Florida"        "Georgia"        "Hawaii"         "Idaho"
## [13] "Illinois"       "Indiana"        "Iowa"           "Kansas"
## [17] "Kentucky"       "Louisiana"      "Maine"          "Maryland"
## [21] "Massachusetts"  "Michigan"       "Minnesota"      "Mississippi"
## [25] "Missouri"       "Montana"        "Nebraska"       "Nevada"
## [29] "New Hampshire"  "New Jersey"     "New Mexico"     "New York"
## [33] "North Carolina" "North Dakota"   "Ohio"           "Oklahoma"
## [37] "Oregon"         "Pennsylvania"   "Rhode Island"   "South Carolina"
## [41] "South Dakota"   "Tennessee"      "Texas"          "Utah"
## [45] "Vermont"        "Virginia"       "Washington"     "West Virginia"
## [49] "Wisconsin"      "Wyoming"
```

```
# selecting US states with at least 70% urban population
USArrests[USArrests$UrbanPop >= 70, ]
```

```
##               Murder Assault UrbanPop Rape
## Arizona          8.1     294       80 31.0
## California       9.0     276       91 40.6
```

```
## Colorado         7.9     204       78 38.7
## Connecticut      3.3     110       77 11.1
## Delaware         5.9     238       72 15.8
## Florida         15.4     335       80 31.9
## Hawaii           5.3      46       83 20.2
## Illinois        10.4     249       83 24.0
## Massachusetts    4.4     149       85 16.3
## Michigan        12.1     255       74 35.1
## Missouri         9.0     178       70 28.2
## Nevada          12.2     252       81 46.0
## New Jersey       7.4     159       89 18.8
## New Mexico      11.4     285       70 32.1
## New York        11.1     254       86 26.1
## Ohio             7.3     120       75 21.4
## Pennsylvania     6.3     106       72 14.9
## Rhode Island     3.4     174       87  8.3
## Texas           12.7     201       80 25.5
## Utah             3.2     120       80 22.9
## Washington       4.0     145       73 26.2
```

```r
# how many?
nrow(USArrests[USArrests$UrbanPop >= 70, ])
```

```
## [1] 21
```

```r
# remove Rape from data.frame
USArrests$Rape <- NULL

print(USArrests)
```

```
##                Murder Assault UrbanPop
## Alabama          13.2     236       58
## Alaska           10.0     263       48
## Arizona           8.1     294       80
## Arkansas          8.8     190       50
## California        9.0     276       91
## Colorado          7.9     204       78
## Connecticut       3.3     110       77
## Delaware          5.9     238       72
## Florida          15.4     335       80
## Georgia          17.4     211       60
## Hawaii            5.3      46       83
## Idaho             2.6     120       54
## Illinois         10.4     249       83
## Indiana           7.2     113       65
## Iowa              2.2      56       57
## Kansas            6.0     115       66
## Kentucky          9.7     109       52
## Louisiana        15.4     249       66
## Maine             2.1      83       51
## Maryland         11.3     300       67
## Massachusetts     4.4     149       85
## Michigan         12.1     255       74
## Minnesota         2.7      72       66
## Mississippi      16.1     259       44
## Missouri          9.0     178       70
```

```
## Montana           6.0     109       53
## Nebraska          4.3     102       62
## Nevada           12.2     252       81
## New Hampshire     2.1      57       56
## New Jersey        7.4     159       89
## New Mexico       11.4     285       70
## New York         11.1     254       86
## North Carolina   13.0     337       45
## North Dakota      0.8      45       44
## Ohio              7.3     120       75
## Oklahoma          6.6     151       68
## Oregon            4.9     159       67
## Pennsylvania      6.3     106       72
## Rhode Island      3.4     174       87
## South Carolina   14.4     279       48
## South Dakota      3.8      86       45
## Tennessee        13.2     188       59
## Texas            12.7     201       80
## Utah              3.2     120       80
## Vermont           2.2      48       32
## Virginia          8.5     156       63
## Washington        4.0     145       73
## West Virginia     5.7      81       39
## Wisconsin         2.6      53       66
## Wyoming           6.8     161       60
```

```r
# what happens when converting `USArrests` into a matrix, why?
matrix(USArrests)
```

```
##      [,1]
## [1,] Numeric,50
## [2,] Integer,50
## [3,] Integer,50
```

```r
# it uses each column as an object which is then placed sequentially
# Why? as data.frame is basically a list of lists (or list of vectors)
# in the background. So basically matrix() function is trying to convert a list
# of 3 objects into a matrix which it succeeds. It is usually not what we want
```

Array (večdimenzionalna tabela)

```r
aa = array(dim=c(3,5,2))
aa[1,,2] = "M"
aa[,,1] = 3
dimnames(aa) = list(c("oseba1","oseba2","oseba3"),c("cas1","cas2","cas3","cas4","cas5"),c("treatment","
```