Az R nyelv alapjai

Baran Ágnes

Az R objektumai

- vektorok
- tömbök
- faktorok
- listák
- adatkeretek
- függvények

Vektorok

Minden elemük azonos típusú: numeric (integer vagy double), complex, logical, character, raw

`mode(a)` vagy `typeof(a)` az *a* vektor típusa

Vektorok

Az $x = (-1.2, 3.1, 4.7, 1.9)$ vektor megadása

- az "$<-$" operátorral:
  `x <- c(-1.2, 3.1, 4.7, 1.9)`
- vagy:
  `c(-1.2, 3.1, 4.7, 1.9)->x`
- az `assign()` függvénnyel:
  `assign(‘‘x’’, c(-1.2, 3.1, 4.7, 1.9))`

Vektorok kiegészítése, összefűzése:

```
y <- c(1.4, x, 6.2)
z <- c(2.2, x, 0.3, y)
```

`a<-numeric(0)` egy üres numerikus vektor

Vektorok, mint szabályos sorozatok

Az $x = (1, 2, 3, 4, 5)$ vektor:
`x <- 1:5`
(**a kettőspont operátor magas prioritású!!!**)
Az `y <- 5:1` eredménye $y = (5, 4, 3, 2, 1)$

A `rep()` függvény:

- `y <- rep(x, times=3)`
  Az x vektort 3-szor egymásután másolja.
- `y <- rep(x, each=3)`
  Az x vektor minden egyes koordinátáját 3-szor megismétli.

Vektorok, mint szabályos sorozatok

A `seq()` függvény:

```
x <- seq(1,5)
x <- seq(from=1, to=5)
x <- seq(to=5, from=1)
```

Mindhárom utasítás eredménye az $x = (1, 2, 3, 4, 5)$ vektor.

A `seq()` függvény további lehetséges argumentumai:

- `by` (lépésköz)
  `y <- seq(from=1, to=3, by=0.5)`
  eredménye: $y = (1, 1.5, 2, 2.5, 3)$
- `length` (koordináták száma)
  `z <- seq(from=1, by=0.1, length=11)`
  eredménye: $z = (1, 1.1, 1.2, ..., 1.9, 2)$
- `along` (egyedül szerepel argumentumként)
  `v <- seq(along=y)`
  előállítja az $(1, 2, 3, ..., \text{length}(y))$ vektort.

Műveletek vektorokkal

- $+, -, *, /$ a négy alapművelet, ^ a hatványozás, mindegyik elemenként.
  A vektoroknak nem kell ugyanannyi elemből állni!!
  Pl. ha:

  ```
  x <- 1:5
  y <- c(0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20)
  z <- x+3*y
  ```

  akkor z a $(1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1)$ (azaz x 2.2-szer egymás után másolva) és a $3y$ vektorok összege.

- `sin`, `cos`, `tan`, `exp`, `log`, `sqrt`, `abs` stb. mind elemenként hajtódik végre.

Műveletek vektorokkal

- `min(x)` és `max(x)` az x vektor legkisebb és legnagyobb eleme
- `sort(x)` az x elemeit növekvő sorrendbe rendezi
- `rev(x)` az x elemeit fordított sorrendben sorolja fel
- `length(x)` az x vektor elemeinek a száma
- `sum(x)` az x vektor elemeinek összege
- `prod(x)` az x vektor elemeinek szorzata
- `mean(x)` az x vektor elemeinek átlaga
- `x[3]` az x vektor harmadik eleme
- `x[1:3]` az x vektor első három eleme
- `x[-3]` az x vektor minden eleme, kivéve a harmadikat
- `x[-c(3,5)]` az x vektor minden eleme, kivéve a 3. és 5.
- `sample(x)` az x vektor koordinátáit véletlenszerűen permutálja

Hiányzó értékek

- `NA` : Not Available (hiányzó érték)
- `NaN` : Not a Number (pl. 0/0, `Inf/Inf`)

Pl. `a <- 1:3; a[5] <- 1` esetén a=(1, 2, 3, NA, 1)

Ha `a <- c(-2, 0, NA, 1, 0/0)` akkor

- `is.na(a)` eredménye: `FALSE FALSE TRUE FALSE TRUE`
- `is.nan(a)` eredménye: `FALSE FALSE FALSE FALSE TRUE`

Fontos! `NA==NA` és `NaN==NaN` értéke egyaránt `NA`

NA, NaN értéke általában nem változik műveletvégzéssel

Logikai vektorok

Logikai operárorok

- $<$, $<=$, $>$, $>=$, $==$, $!=$
- & (AND), | (OR), ! (NOT), `xor(,)`

Logikai vektorok

Elemei a TRUE, a FALSE és az NA lehetnek

Pl. az

```
a <- c(-1, 3, 2, 0)
b <- a>1
```

utasítások eredménye a `b=(FALSE TRUE TRUE FALSE)` vektor, míg

```
a <- c(1,-1,NA,0)
b <- a<0.5
```

eredménye: `FALSE TRUE NA TRUE`

Logikai függvények

- **which** : egy logikai vektor TRUE értékeinek indexeit adja
- **any** : értéke TRUE, ha a logikai vektor valamely komponense TRUE.

  Hívása: `any(x,na.rm=FALSE)`
  ahol x logikai vektor, `na.rm` opcionális (alapértelmezés FALSE), TRUE érték esetén a vizsgálatból kihagyja az x vektor NA komponenseit.
- **all** : értéke TRUE, ha a logikai vektor minden komponense TRUE.

  Hívása: `all(x,na.rm=FALSE)`
  ahol x logikai vektor, `na.rm` opcionális

Logikai függvények

- **identical()** Két objektum teljes egyezését vizsgálja
- **all.equal()** Két objektum egyezését vizsgálja valamilyen toleranciaszint mellett.

Példák.

```
> a <- c(1,-1,NA,0)
> b <- c(1,-1,NA,0)
> c <- c(1,-1,NA,0.0001)
> identical(a,b)
[1] TRUE
> all.equal(a,c)
[1] "Mean absolute difference:  1e-04"
```

Ha megadjuk a toleranciaszintet:

```
> all.equal(a,c,0.01)
[1] TRUE
```

Objektumok típusa, konverziók

- Típus lekérdezése:
  `mode()`, `typeof()`
- Logikai függvények annak eldöntésére, hogy egy objektum adott típusú-e:
  `is.integer()`, `is.double()`, `is.complex()`, `is.caracter()`, `is.logical()`, `is.numeric()`, `is.list()`,...
- Konverziós függvények:
  `as.integer()`, `as.double()`, `as.complex()`, `as.caracter()`, `as.logical()`, `as.numeric()`, `as.list()`,...

Feladatok

- Az elemek egyenkénti begépelése nélkül állítsa elő az alábbi vektorokat!
  (1) $a = (0, 1, \cdots, 30)$
  (2) $b = (2, 4, 6, \ldots, 100)$,
  (3) $c = (2, 1.9, 1.8, \cdots, 0.1, 0)$
  (4) $d = (0, 3, 6, \ldots, 27, 30, -100, 30, 27, \cdots, 6, 3, 0)$
  (5) $e = \left(\frac{1}{2}, \frac{1}{3}, \cdots, \frac{1}{20}\right)$
  (6) $f = \left(\frac{1}{2}, \frac{2}{3}, \cdots, \frac{19}{20}\right)$
- Legyen x egy adott 100 elemű sorvektor. Az x vektorból állítsa elő azt az y vektort, melynek elemei
  (1) az x vektor elemei fordított sorrendben felsorolva,
  (2) az x vektor első 5 eleme,
  (3) az x vektor elemei ugyanolyan sorrendben, kivéve az x 4. elemét
  (4) az x vektor elemei ugyanolyan sorrendben, kivéve az x 5., 72. és 93. elemét
  (4) az x vektor páratlan sorszámú elemei
  (5) az x vektor 2., 5., 17. és 81. eleme.

Feladatok

- Legyen x egy 100 elemű véletlen vektor. Állítsa elő azt az y vektort, amely az x vektor 3 legnagyobb elemét tartalmazza.
- Legyen x egy 100 elemű véletlen vektor. Állítsa elő azt az y vektort, amely az x vektor minden 10. elemét tartalmazza.
- Legyen x egy adott vektor. Állítsa elő azt az y vektort, amely az x vektor 3-nál nagyobb elemeit tartalmazza.
- Legyen x egy adott vektor. Állítsa elő azt az y vektort, amely az x vektor $(0, 1)$ intervallumon kívül eső elemeit tartalmazza.
- Legyen x egy adott vektor. Állítsa elő azt az y vektort, amely az x vektor elemeit tartalmazza, kivéve az NA és NaN elemeket.
- Legyen x egy adott vektor. Állítsa elő azt az y vektort, amely az x vektor NA-tól és NaN-tól különböző elemeinek kétszereseit tartalmazza.

Mátrixok

Adott a vektor és n, m értékek esetén

`A <- matrix(a, nrow = n, ncol = m, byrow = FALSE)`

egy $n \times m$-es A mátrixot ad (az a vektorból oszlopfolytonosan)

- Ha length(a) $< nm$, akkor ciklikusan ismétli az a vektort
- `nrow` és `ncol` közül elegendő az egyik
- a `byrow` opcionális, alapértelmezés a FALSE

`A[i,j]` az A mátrix (i, j)-edik eleme
`A[1:3,2:4]` az A mátrix 1.,2.,3. sorainak és 2.,3.,4. oszlopainak metszete

Mátrixok

- `A[i,]` : a mátrix i-edik sora
- `A[,j]` : a mátrix j-edik oszlopa
- `A[,c(2,4)]` : a mátrix 2. és 4. oszlopa
- `B <- A[-2,]` : elhagyja a mátrix 2. sorát
- `B <- A[,-c(1,3)]` : elhagyja a mátrix 1. és 3. oszlopát
- `C <- cbind(A,B)` : horizontálisan összefűzi az A és B mátrixokat (A-nak és B-nek ugyanannyi sora legyen!)
- `C <- rbind(A,B)` : vertikálisan összefűzi az A és B mátrixokat (A-nak és B-nek ugyanannyi oszlopa legyen!)

Mátrixműveletek

- $+, -, *, /, \hat{}$ elemenkénti műveletek (azonos méretű mátrixok!)
- $\%*\%$ mátrixszorzás

Mátrix-vektor műveletek

Ha x egy vektor, A egy mátrix, akkor `x*A` és `A*x` eredménye ugyanaz: az elemenkénti szorzata az A mátrixnak és annak a mátrixnak, mely ugyanolyan méretű mint A és az x vektor elemeinek oszlopfolytonos ciklikus ismételgetésével kapunk.

Az `A%*%x` és `x%*%A` eredménye (ahol x elemeinek száma első esetben A oszlopainak, második esetben A sorainak számával egyezik meg) az Ax és xA szorzat.

Külső szorzat

a%o%b : az a tömb minden elemét szorozzuk a b tömb minden elemével.

A diag() függvény

Használata:

`diag(x , nrow, ncol)`
`diag(x) <- value`
ahol

- x : egy szám, egy vektor, egy mátrix vagy hiányzik
- nrow, ncol : opcionális, de csak akkor szerepelhet, ha x nem mátrix
- value : egy szám, vagy az x diagonálisával megegyező méretű vektor

Példák:

- `diag(5)` : az 5×5-ös egységmátrix
- `diag(5,3,4)` : egy 3×4-es diagonális mátrix, átlójában csupa 5-össel
- `diag(1:3,5,4)` : egy 5×4-es diagonális mátrix, átlójában ciklikusan az (1,2,3) vektor

A diag() függvény

Példák:

Ha A egy adott mátrix

- `diag(A)` : az a vektor, mely az A mátrix diagonálisát tartalmazza
- `diag(A) <- 3` : az A diagonálisának minden elemét 3-ra cseréli
- `diag(A) <- a` : (ahol a ugyanakkora vektor, mint az A diagonálisa) az A diagonálisát kicseréli az a vektorral

Néhány hasznos függvény

- `t(A)` : az A transzponáltja
- `dim(A)` : az A mérete
- `nrow(A)` : az A sorainak száma
- `ncol(A)` : az A oszlopainak száma
- `det(A)` : az A determinánsa
- `solve(A)` : az A inverze
- `solve(A,b)` : az $Ax = b$ egyenletrendszer megoldása
- `colSums(A)` : az A oszlopaiban álló elemek összegei
- `rowSums(A)` : az A soraiban álló elemek összegei
- `sum(A)` : az A elemeinek összege

Tömbök

Az `a <- 1:12; b <- a; dim(b) <- c(3,4)` utasítások eredménye az

$$b = \begin{pmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{pmatrix}$$

mátrix.

Az `a <- 1:12; b <- array(a,dim=c(3,4))` utasítások is a fenti b mátrixot adják.

Tömbök

DE az

```
a <- 1:12; b <- a; dim(b) <- c(3,5)
```

utasításra hibaüzenetet kapunk:

```
Error in dim(b) <- c(3, 5) :
dims [product 15] do not match the length of object
[12]
```

míg az

```
a <- 1:12; b <- array(a,dim=c(3,5))
```

utasítás eredménye a

$$b = \begin{pmatrix} 1 & 4 & 7 & 10 & 1 \\ 2 & 5 & 8 & 11 & 2 \\ 3 & 6 & 9 & 12 & 3 \end{pmatrix}$$

mátrix.

Tömbök

Többdimenziós tömböket is definiálhatunk:

```
a <- array(v,dim=d,dimname=L)
```

ahol

- v : egy vektor (a tömb elemeinek vektora)
- d : a dimenziók vektora
- L : opcionális, egy lista (ld később), a dimenziók nevei

Példa

```
a <- array(1:12,dim=c(3,2,2))
```

Az objektum szerkezetéről a `str()` függvény ad információt.

Az apply() függvény

Végrehajt egy megadott függvényt egy mátrix, vagy tömb megadott dimenziói mentén.

`apply(A,v,fv)`

ahol

- *A* egy tömb
- *v* egy szám, vagy egy vektor
- *fv* a végrehajtandó függvény

Példa

```
> A <- matrix(1:12,3)
> s1 <- apply(A,1,sum)     (sorösszegek)
> s2 <- apply(A,2,sum)     (oszlopösszegek)
> A <- array(1:12,c(3,2,2))
> S <- apply(A,1:2,sum)
```

Feladatok

- Állítsa elő azt a 3×4-es A mátrixot, mely sorfolytonosan tartalmazza az $1, \ldots, 12$ elemek kétszereseit!
- Állítsa elő a csupa 1-esekből álló 3×5-ös mátrixot.
- Állítsa elő azt a 4×5-ös mátrixot, melynek átlójában az 1,2,3,4 elemek állnak, minden más eleme 0.
- Legyen A egy adott 4×5-ös mátrix. Cserélje ki az A diagonálisának minden elemét (-1)-re.
- Legyen A egy adott 3×4-es mátrix. Mi lesz a `cbind(A,1)`, `cbind(1,A)`, `rbind(A,1)`, `rbind(1,A)` utasítások eredménye?
- Keresse meg egy 6×8-as véletlen mátrix minden sorában a maximális elemet.
- Egy adott A mátrix esetén konstruálja meg azt a B mátrixot, melyet úgy kapunk, hogy az A sorainak végére odaírjuk az adott sorban álló elemek átlagát.

Feladatok

- Legyen `A <- matrix(1:12,3,byrow=TRUE)`. Konstruálja meg (az elemek felsorolása nélkül) azt a B mátrixot, melyet úgy kapunk, hogy
  (1) elhagyjuk az A mátrix első sorát,
  (2) elhagyjuk az A mátrix 2. és 4. oszlopát,
  (3) elhagyjuk az A mátrix utolsó sorát és oszlopát
  (4) kétszer egymás mellé írjuk az A mátrixot,
  (5) transzponáljuk az A mátrixot,
  (6) felcseréljük az A mátrix 2. és 4. oszlopát
  (7) négyzetre emeljük az A elemeit
  (8) az A minden elemét megnöveljük 3-mal
  (9) A minden elemének vesszük a négyzetgyökét
  (10) az A első sorának második elemét kicseréljük -2-re
  (11) az A 2. sorát kicseréljük a $(-1 \quad 0 \quad -2 \quad 3)$ vektorra

Feladatok

- Vizsgálja meg a beépített `Titanic` tömböt. Itt a tömb 4 dimenziója az utasosztály (1., 2., 3., legénység), a nem, a kor (gyerek, felnőtt), és az, hogy túlélte-e a katasztrófát. Az `apply()` függvény segítségével állapítsa meg, hogy a túlélők között a nők, vagy a férfiak voltak-e többen. Milyen osztályon utazott a legtöbb túlélő? A legénység hány százaléka élte túl a katasztrófát?
- Határozza meg a beépített `women` tömbben szereplő nők átlagos magasságát és súlyát. Melyik nő esetén lesz maximális a súly/magasság hányados?

Faktorok

Egy olyan objektum, amely más vektorok elemeinek osztályozására szolgál.

Példa

```
>szak <- c("GI","GI","PTI","PTI",
"MI","PTI","MI","GI","GI","MI","MI","GI","PTI","PTI")
>szakF <- factor(szak)
```

Ekkor a `szakF` faktor:

```
[1] GI GI PTI PTI MI PTI MI GI GI MI MI GI PTI PTI
Levels:  GI MI PTI
```

A szintek lekérdezhetőek a `levels()` függvénnyel:

```
>levels(szakF)
[1] "GI" "MI" "PTI"
```

Faktorok

Ha adottak a tanulmányi eredmények:

```
>eredm <-
c(4.2,3.1,4.8,3.7,5,4.5,4.2,3.9,4,3.8,4.4,3.6,4.8,4.4)
```

akkor kiszámíthatjuk a szakonkénti átlagot:

```
> atlag <- tapply(eredm,szakF,mean)
```

Ennek eredménye:

```
   GI MI PTI
3.76 4.35 4.44
```

A **tapply()** függvény:

```
tapply(a,b,fv)
```

ahol *a* egy vektor, *b* egy faktor (ugyanannyi elemű mint *a*), *fv* egy függvény.
Hatása: az *a* elemeit csoportosítja *b*-nek megfelelően és a csoportokra alkalmazza az *fv* függvényt.

Faktorok

A **cut()** függvény:
numerikus vektort faktorrá konvertál.

PÉLDA: a beépített `women` tömb `weight` oszlopával:

```
f <- cut(women$weight,3)
f<-cut(women$weight,3,labels=c('Low','Medium','High'))
```

3 egyforma hosszú intervallumba sorolja az értékeket, a 2. esetben a szinteknek nevet is adunk.

```
f <- cut(women$weight,breaks=c(114,135,164))
```

a megadott határok szerinti intervallumokba sorolja az értékeket

A **table()** függvény elkészíti a gyakoriságtáblát:

```
table(f)
```

Faktorok

A **cut()** függvény

PÉLDA: Megadott kvantilisek szerinti faktorizáció:

```
> x <- sample(1:15)
> cut(x,breaks=quantile(x,c(0,0.25,0.5,0.75,1)),
+ labels=c("I","II","III","IV"))
```

Ha nem szeretnénk a legkisebb elemet sem kihagyni a vizsgálatból:

```
> cut(x,breaks=quantile(x,c(0,0.25,0.5,0.75,1)),
+ labels=c("I","II","III","IV"),include.lowest=T)
```

Feladatok

- Legyenek adottak a faktoroknál leírt `szak` és `eredm` vektorok. Vizsgálja meg az alábbi 2 utasítás után a `szakF` és az `atlag` objektumok értékét.

```
>szakF <- factor(szak,levels=c("GI","MI"))
>atlag <- tapply(eredm,szakF,mean)
```

- Egy cég tovább szeretné képezni a dolgozóit, ezért 3 napon át egy felmérést végzett körükben. Minden nap 6 feladatot kellett megoldani, az erre kapott pontszámokat a `teszt.dat` táblázat `e1`–`e6` oszlopai tartalmazzák. Az elért pontszámok és a betöltött munkakör alapján számított összpontszámokat az `eredm` oszlop tartalmazza, a `valasz` oszlpoban az szerepel, hogy az adott dolgozó hajlandó-e résztvenni a programban. Határozza meg az egyes napokra behívottak számát, az egyes napokon meg nem jelentek számát és az összpontszámok átlagát. Számítsa ki az egyes napokon azon dolgozók összpontszámának átlagát, akik hajlandóak résztvenni a programban.

Listák

Nem feltétlenül azonos típusú objektumok vektora.

Példa:

```
> a <- c("Adam","Bela","Cecil","Denes","Elemer")
> b <- c(12,2,3)
> c <- matrix(1:12,3)
> L <- list(a,b,c)
```

Hivatkozás az *L* lista *i*-edik komponensére: `L[[i]]` , az *i*-edik komponens *j*-edik elemére: `L[[i]][j]`

Példa:

```
> L[[1]][2] <- "Bandi"
```

Kicseréli a lista első komponensében `"Bela"`-t `"Bandi"`-ra (de az eredeti *a* vektort nem írja felül)

Listák

Példa: Tömb dimenzió-neveinek megadása listával.

```
> a <- c(8,23,11,5,24,14,4,28,16,10,21,4)
> termet <- c("kicsi","kozepes","nagy")
> nem <- c("ffi","no")
> kor <- c("gyerek","felnott")
> L <- list(termet, nem, kor)
> A<-array(a,c(3,2,2),L)
```

lapply() függvény:
`lapply(lista,fuggv)`

az `lista` minden elemére alkalmazza a `fuggv` függvényt.

Példa:

`lapply(L,length)`

Adatkeretek

Különböző típusú adatok táblázatos formában tárolására.

Példa:

```
> n <-c("Antal A.","Balogh B.","Cseh C.","David D")
> l <- c("Debrecen", "Szeged", "Bp", "Bp")
> s <- c(1992,1987,1971,1985)
> v <- c(TRUE, TRUE, FALSE, TRUE)
> df <- data.frame(n,l,s,v)
```

	n	l	s	v
1	Antal A.	Debrecen	1992	TRUE
2	Balogh B.	Szeged	1987	TRUE
3	Cseh C.	Bp	1971	FALSE
4	David D	Bp	1985	TRUE

Az oszlopok neveit megváltoztathatjuk:

```
names(df) <- c("nev","lakhely","szul","valaszolt")
```

Adatkeretek

Fontos! A függvényeknek átadott karakter vektorok automatikusan faktorrá konvertálódnak. Pl a `df` adatkeret szerkezete:

```
> str(df)
'data.frame':  4 obs.  of 4 variables:
 $ nev :  Factor w/ 4 levels "Antal A.","Balogh
B.",..:  1 2 3 4
 $ lakhely :  Factor w/ 3 levels "Bp","Debrecen",..:
2 3 1 1
 $ szul :  num 1992 1987 1971 1985
 $ valaszolt:  logi TRUE TRUE FALSE TRUE
```

Ezt úgy lehet megakadályozni, hogy a vektort az `I(x)` függvénnyel levédve adjuk át a `data.frame()` függvénynek.

```
> df <- data.frame(I(n),I(l),s,v)
```

Adatkeretek

- Hivatkozás elemekre:

```
> df[2,3]
[1] 1987

> df[2,"szul"]
[1] 1987
```

- Hivatkozás sorokra

```
> df[1,]
 nev lakhely szul valaszolt
1 Antal A. Debrecen 1992 TRUE
```

- Hivatkozás oszlopokra

```
> df[,1]
[1] Antal A. Balogh B. Cseh C. David D
Levels:  Antal A. Balogh B. Cseh C. David D
```

Adatkeretek

Hivatkozás oszlopra névvel:

```
> df$valaszolt
[1] TRUE TRUE FALSE TRUE
```

Az attach() függvény:

```
attach(df)
```

Ezután a `lakhely, nev, szul, valaszolt` változók közvetlenül hívhatók:

```
> lakhely
[1] "Debrecen" "Szeged" "Bp" "Bp"
```

A `detach()` függvénnyel kivehetjük a közvetlen elérési útból a változókat.

read.table()

Táblázat beolvasása fájlból.

PÉLDA: ha a tableexample.txt fájl a köv.:

nev	P1	P2	P3	P4
Albert	2	5	8	0
Bela	4	0	7	3
Cecil	7	9	10	6
Denes	3	4	4	0
Edit	5	1	2	1
Flora	4	8	5	4

```
eredm <- read.table("tableexample.txt",header=T)
```

Ekkor az `eredm` objektum egy adatkeret.

Ha a fájlban hiányzik a fejrész, akkor

```
eredm <- read.table("tableexample.txt")
```

Adatkeretek

Hivatkozás adatkeretek részeire:

```
> df[3:4,1:2]
     nev lakhely
3 Cseh C.      Bp
4 David D      Bp
```

Ha azok az 1990 előtt születettek érdekelnek, akik válaszoltak:

```
> subset(df,szul<1990 & valaszolt==T)
      nev lakhely szul valaszolt
2 Balogh B. Szeged 1987      TRUE
4 David D      Bp 1985      TRUE
```

Adatkeretek

Oszlop hozzáadása:

```
> df$nem <- c("ffi","ffi","no","ffi")
```

	nev	lakhely	szul	valaszolt	nem
1	Antal A.	Debrecen	1992	TRUE	ffi
2	Balogh B.	Szeged	1987	TRUE	ffi
3	Cseh C.	Bp	1971	FALSE	no
4	David D	Bp	1985	TRUE	ffi

Sor hozzáadása (az új 5 oszlopos táblához): (**Csak akkor megy, ha figyeltünk arra, hogy a karakter vektorból ne legyen faktor!**)

```
> ujsor <- c("Elek E", "Pecs", 1977, FALSE, "no")
> dfuj <- rbind(df,ujsor)
```

merge()

Mátrixok, adattáblák összefésülése. Ha

```
> d1
```

	a	b
1	Adam	12
2	Bela	31
3	Csaba	2
4	Denes	54
5	Elek	9

```
> d2
```

	a	c
1	Balazs	11
2	Adam	21
3	Gabor	8
4	Csaba	26
5	Denes	14
6	Imre	30

akkor

```
> merge(d1,d2,by="a")
```

	a	b	c
1	Adam	12	21
2	Csaba	2	26
3	Denes	54	14

merge()

Ha az összefésült táblázatban minden adatot szerepeltetni akarunk:

```
> merge(d1,d2,by="a",all=T)
```

	a	b	c
1	Adam	12	21
2	Bela	31	NA
3	Csaba	2	26
4	Denes	54	14
5	Elek	9	NA
6	Balazs	NA	11
7	Gabor	NA	8
8	Imre	NA	30

Elágazások, ciklusok

if elágazás

```
if(logikai kifejezes) { utasitas1} else { utasitas2}
```

ifelse függvény (az if-else vektorizált alakja)

```
ifelse(a,b,c)
```

ahol *a* logikai vektor, *b* és *c* vektorok. Vektorral tér vissza, ha `a[i]` igaz, akkor `b[i]`-vel, egyébként `c[i]`-vel.

Példa:

```
x <- c(-3, 1, 0, -2, 2)
ifelse(x>0,x,-x)
```

eredménye az abs(x) vektor

Elágazások, ciklusok

for-ciklus

```
for(valtozo in vektor) utasitas
```

Példa:

```
s <- 0
for(i in 1:10) s <- s+i
```

FONTOS! Ha lehet kerüljük a for-ciklus használatát, használjunk helyette egész vektorra alkalmazható függvényeket.

while-ciklus

```
while(feltetel) utasitas
```

Példa:

```
x <- -2
while(x<8) { x <- x+2; print(x)}
```

Elágazások, ciklusok

repeat

`repeat { utasitas}`

Példa:

```
x <- -2
repeat { x <- x+2; print(x); if(x>8) break}
```

- **break** : kiugrik a ciklusból
- **next** : a ciklusváltozó következő értékére ugrik

Függvények

Szintaktika:

`fvneve <- function(arg1,arg2,...)  {utasitasok }`

Hívása:

`fvneve(arg1,arg2,...)`

Példa:

```
sajatfv <- function(x){
x ^ 2 − 3 ∗ x + 1
}
```

Ezután `sajatfv(4)` az 5 értéket adja vissza.

Feladatok.

- Írjon egy függvényt, mely tetszőleges a numerikus vektor és x valós szám esetén megadja az a vektor x-hez legközelebbi elemét.
- Írjon egy függvényt, mely kiszámolja egy tetszőleges numerikus vektor elemeinek átlagát úgy, hogy a legnagyobb és legkisebb elemet nem veszi figyelembe.
- Írjon egy függvényt, mely tetszőleges numerikus A mátrix sorait úgy cserélgeti meg, hogy az első oszlop elemei növekvő sorrendben álljanak.
- Írjon egy függvényt, mely egy vektor minden koordinátájához hozzáad 1-et `for`-ciklussal, majd mérje meg a futási időt akkor, ha a függvényt az `x <- runif(10000000)` vektorral hívja meg. Hasonlítsa össze a futási időt `x+1` utasítás végrehajtásának idejével.

Grafika

Magas szintű függvények. Egy utasítással ábra készíthető, pl.:

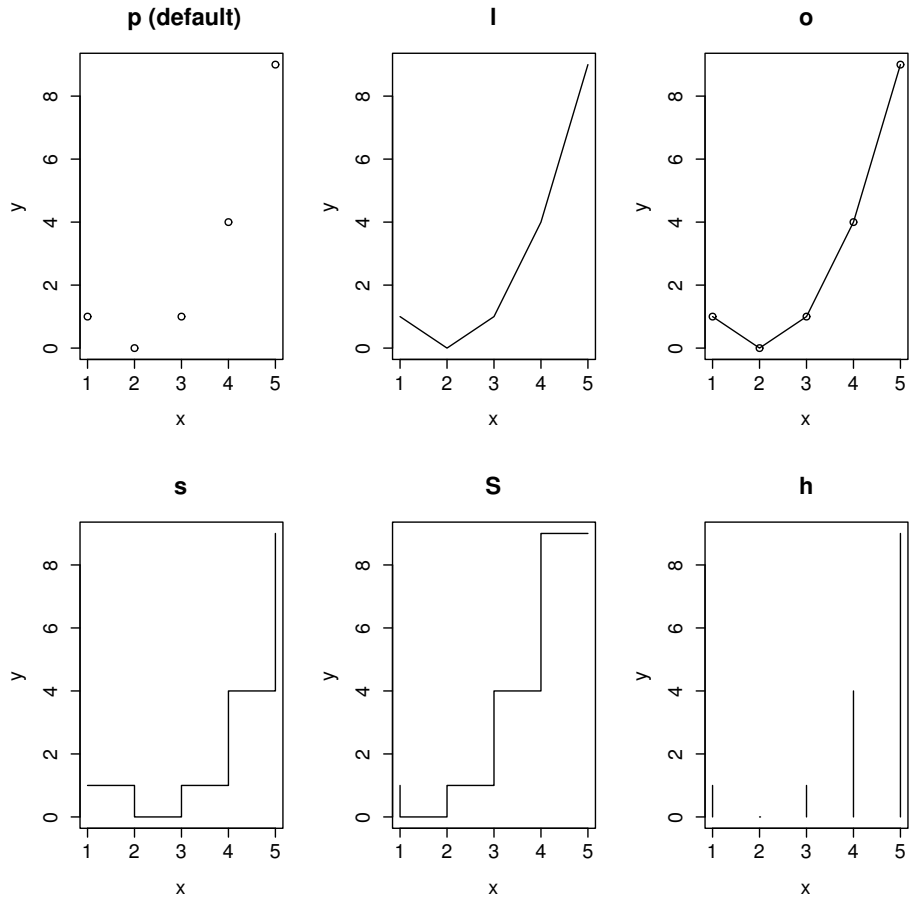
- `plot()`
- `hist()`
- `boxplot()`
- `barplot()`

Alacsony szintű függvények. Meglévő ábrák módosítása, pl:

- `lines()`
- `points()`
- `axis()`
- `legend()`
- `text()`
- `title()`

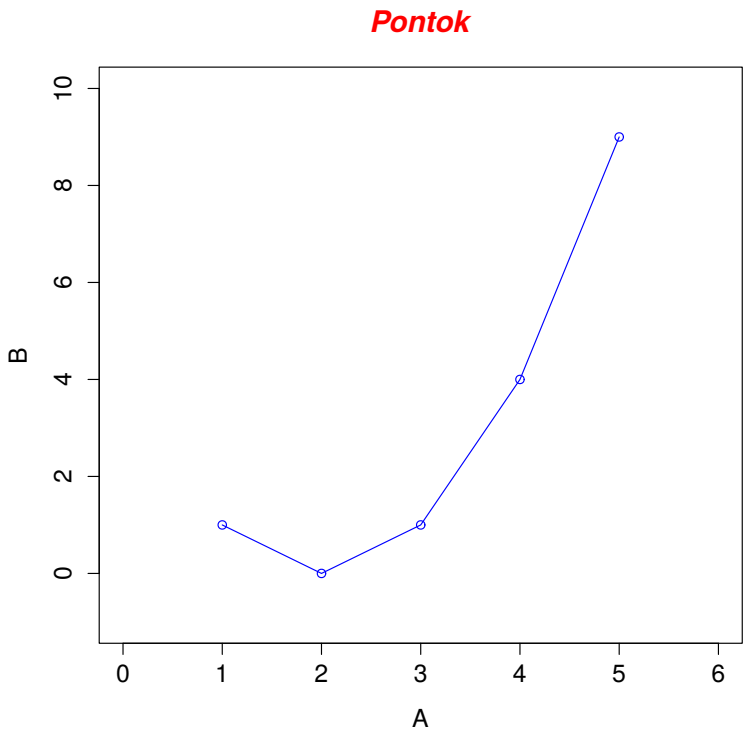
plot()

`plot(x,y,''l'')`
ahol x és y vektorok, idézőjelek között a vonal típusa.
Néhány vonaltípus:



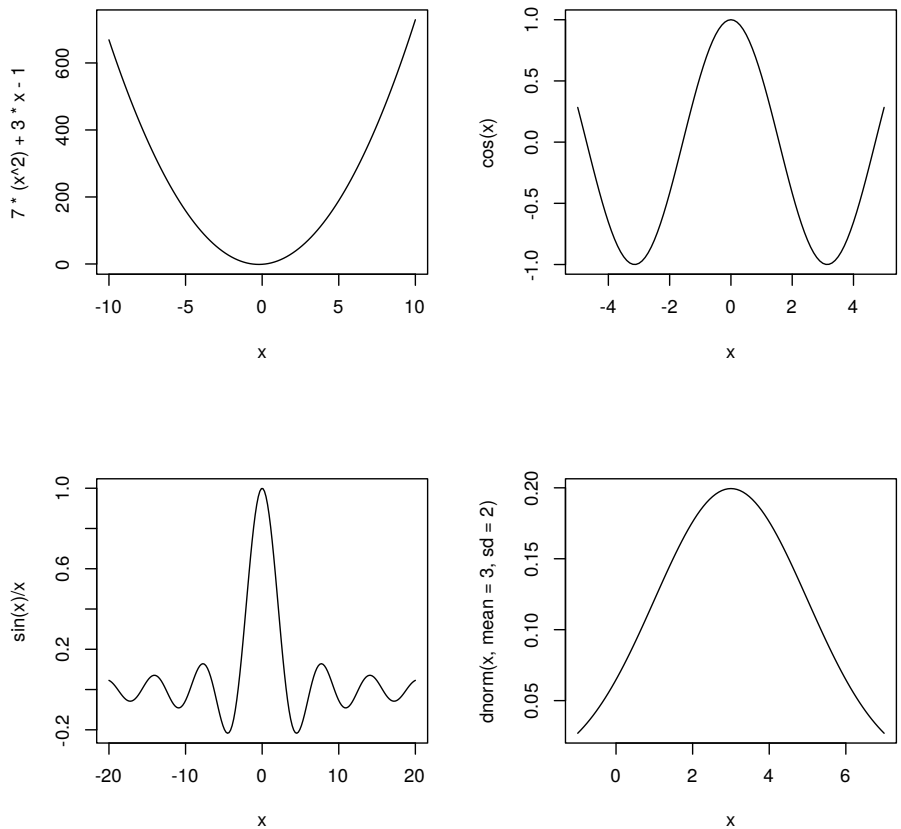
plot()

```
> x <- 1:5
> y <- c(1, 0, 1, 4, 9)
> plot(x,y,type="o",col="blue",xlim=c(0,6),
+ ylim=c(-1,10),ann=F)
> title(main="Pontok",col.main="red",font.main=4,
+ xlab="A",ylab="B")
```



curve()

```
par(mfrow=c(2,2))
curve(7 * (x^2) + 3 * x - 1, from=-10, to=10)
curve(cos, from=-5, to=5)
curve(sin(x)/x, from=-20, to=20, n=200)
curve(dnorm(x,mean=3,sd=2), from=-1, to=7)
```



```
> a <- c(2,4,-1,1,0)
> b <- c(1,3,2,-2,1)
> c <- c(3,0,2)
> l <- min(c(a,b,c))
> u <- max(c(a,b,c))
> plot(1:5,a,type="o",col="blue",ylim=c(l,u),ann=F)
> lines(1:5,b,type="o",pch=20,lty=2,col="red")
> points(c(1,3,4),c,type="p",pch=8,col="green")
```