Bevezetés az R programozási nyelv meteorológiai használatába: I. Alapismeretek

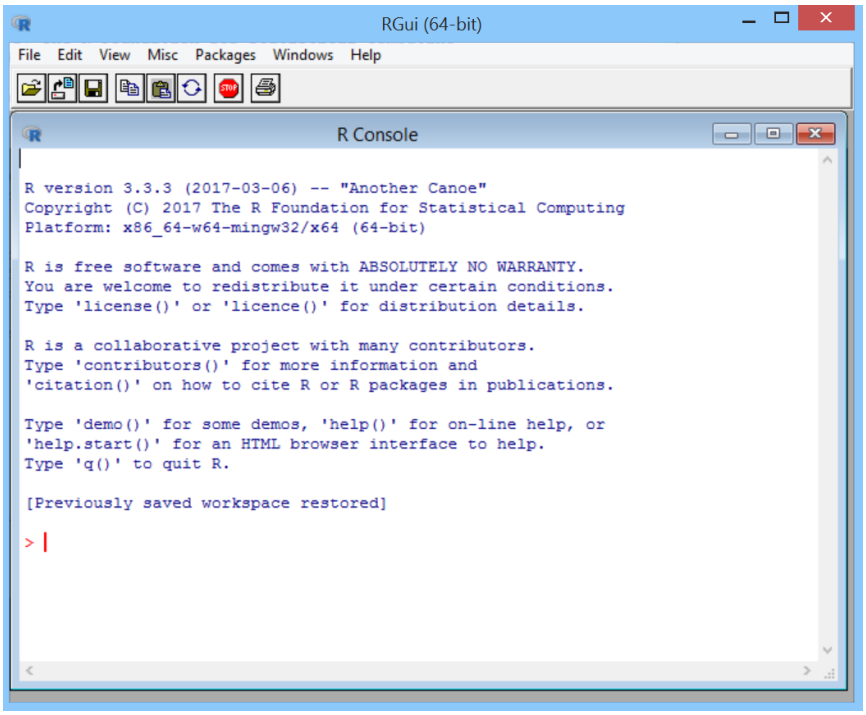
TARTALOM:

1. BEVEZETÉS:

Az R programozási nyelvet 1993-ban hozták létre statisztikai számítások elvégzésére, illetve eredményeinek grafikus megjelenítésére. A legfrissebb stabil kiadás verziószáma: 3.4.1, amely letölthető az alábbi linkről: https://cran.r-project.org/bin/windows/base/ (2017. június 30-án jelent meg). A programnyelv nyílt forráskódú, jól dokumentált.

Az R-ben számos statisztikai eljárás alkalmazható. Használható többek közt idősorok és mezők elemzésére, tartalmaz klaszterezési és simítási eljárásokat, használható hipotézisvizsgálatra. Az R rendelkezik grafikus megjelenítő eszközzel, így ábrázolhatjuk adatainkat például térképsík felett.

Az R-ben a kódok kétféleképpen értelmezhetők: parancssori és szkript módban. Előbbi esetben a terminál ablakba írjuk be a parancsokat. (A sorok a > prompt karakterrel kezdődnek.) Utóbbi esetben a szkript fájlban kerül tárolásra, amely a terminál ablakban nyitható meg, amelynek Windows alatti 3.3.3 verziója az alábbi képen látható.



Az R nyelv szabályai szerint megírt utasításokat terminálon keresztül juttatjuk el a fordítóhoz. Az ENTER-rel jóváhagyott kódok egyesével kerülnek értelmezésre és végrehajtásra az R parancsértelmezője által.

Az R-ben találhatók beépített függvények a számítások, eljárások végrehajtására, azonban számos kiegészítő csomag (package) is letölthető, amelyek különböző függvényeket, megjelenítési lehetőségeket (plotokat) tartalmaznak. A függvények szintaxisa: `függvénynév(paraméter).`

Az R kis- és nagybetű érzékeny. A megnyitandó állományok nevében, elérési útjában található ékezetek, szóközök általában nem okoznak problémát, de ezek használata inkább kerülendő.

Az R programozási nyelv használatára való hivatkozás módja a `citation()` utasítással kérhető le.

Az alábbiakban található példák az R Windows alá írt változatában kerülnek bemutatásra.

<u>Munkafolyamat létrehozása és mentése, alapvető utasítások</u>:

Az R-munkafolyamat (session) a program indításakor kezdődik és a leállításával végződik. A létrehozott objektumok együttese a workspace (munkaterület), amelyhez working directory (munkakönyvtár) tartozik. A továbbiakban a munkavégzés mappájaként hivatkozunk erre. A munkafolyamat során létrehozott objektumok törlődnek, ha a munkaterületet nem mentjük. A mentett objektumokat tartalmazó bináris állomány kiterjesztése: .Rdata. Emellett menthető a (felfelé és lefelé mutató nyilakkal) visszahívható parancsokat tartalmazó szöveges állomány, amelynek kiterjesztése: .Rhistory. A visszahívott parancsok szerkeszthetők is, a sorokban a balra és jobbra mutató nyilakkal pozícionálhatjuk a kurzort. A teljes parancs végrehajtása ENTER-rel történhet, ehhez nem szükséges a sor végére navigálni.

A létrehozott objektumok mentésére és betöltésére a terminál ablakban a Save workspace / Load workspace ikonra kattintva vagy a `save.image("név.Rdata")` és a `load("név.Rdata")` utasítások kiadásával kerülhet sor.

A létrehozott parancsok mentése és betöltése a `savehistory("név.Rhistory")` és a `loadhistory("név.Rhistory")` parancsokkal történhet.

Ha a két állományt `.Rdata` és `.Rhistory` néven az R telepítése után beállított alapértelmezett mappába mentjük (pl. C:\\Users\\Felhasználónév\\Documents mappa), akkor az R újabb indításakor ezek az állományok automatikusan betöltődnek.

A munkavégzés mappájának neve, elérési útja megjeleníthető a `getwd()` utasítással.

A munkavégzés mappájában található állományok listázása a `list.files()` utasítással történhet. Ha nincs állomány a mappában, akkor a `character(0)` eredményt kapjuk.

A munkavégzés mappájának megváltozatási módja:
`setwd("meghajtó azonosítója:\\főmappanév\\almappanév")`

A \ jel nem elég a könyvtárnevek előtt. A \\ jelek helyett / is szerepelhet.

Az "…" idézőjelek helyett '…' aposztróf is használható.

1. példa: Hozzuk létre az üres `proba` mappát a Windows Intézőben!
- Jelenítsük meg a munkavégzés alapértelmezett mappájának nevét és helyét a `getwd()` parancs kiadásával!
- A munkavégzés mappájaként állítsuk be a `proba` mappát! Ehhez a `setwd` függvényt használjuk.
- Listázzuk ki a mappában található állományokat a `list.files()` utasítással!

Később is felhasználandó utasítássorozatok létrehozatalakor célszerű szkriptet írni a parancsok terminál ablakba való beírása helyett. Erre alkalmas például az R szkript szerkesztője, vagy bármilyen szövegszerkesztő, amelyből a vágólapra másolhatók az utasítások és beilleszthetők az R terminálba. Amennyiben a szkriptet az R szrikptszerkesztőjével készítjük, akkor azt a File/New script menüpontban érhetjük el. A mentett szkript kiterjesztése .R.

A szkript a `source("szkriptnév.R")` paranccsal futtatható, ha a szkript a munkavégzés mappájában található. (A szkript a .R fájl ikonját a terminál ablakra „ejtve” is futtatható).

A munkafolyamat befejezése:
A munkafolyamatot a `quit()` vagy `q()` parancsokkal fejezhetjük be. (A terminál ablak jobb felső sarkában a bezárás ikonra kattintva a terminál ablakban megjelenik a `q()` parancs.)

Jelölések:
A megjegyzések (kommentek) a # jellel kezdődnek.

Be nem fejezett utasításokra sortörést alkalmazva az új sor + jellel kezdődik > helyett.

Az utasítás bevitele az Esc gombbal szakítható meg.

Az utasítások sortöréssel végződnek.

Az R-ben a tizedesjel alapértelmezetten pont, nem vessző. A megjelenítésben ez megváltoztatható.

A képernyő a `Ctrl+l` utasítással törölhető.

Az R súgórendszere:
A súgórendszer a `help.start()` utasítás kiadásával indítható el. Egy adott függvényre vonatkozó súgórészlet a függvények neve elé `?` írásával, vagy a `help()` argumentumába a kérdéses függvény nevének beillesztésével jeleníthető meg, html kiterjesztésű állományban. Ez tartalmazza többek között az adott függvény leírását, a szintaxisát, az alkalmazására példát és a hivatkozásait.

2. példa: A `?sum` vagy `help(sum)` parancs begépelésével kérjünk információt az összegfüggvény alkalmazásáról!

A telepített R program mappájában a *`Doc/Manual`* mappában érhető el az R részletes dokumentációja.

2. ÉRTÉKADÁS:

Az adatok lehetnek többek között skalár- vagy vektorértékűek (utóbbi numerikus, szöveg vagy logikai), mátrixok, tömbök, adatstrukturák. Utóbbiak a data frame elnevezésű objektumok, amelyek abban különböznek a mátrixoktól és a tömböktől, hogy oszlopaikban különböző típusú adatok (pl. számok, dátumok és szövegek) is szerepelhetnek.

Értékadás nélkül az R megjeleníti egy művelet eredményét, de nem tárolja el azokat.

3. példa: Adjunk össze két számot!

```
5+8
```

Az eredmény:

```
[1] 13
```

A parancs utáni sorban jelenik meg az eredmény. A szögletes zárójelekben található `[1]` arra utal, hogy az eredmény sorvektor (ebben az esetben egy komponensű vektor, azaz skalár).

Az értékadás az alábbi módokon lehetséges:

```
objektumnév <- kifejezés
kifejezés -> objektumnév
objektumnév = kifejezés
```

Az **objektum** az a kezelt egység, amellyel dolgozunk, pl. változó, konstans, függvény, tömb, adatstruktúra (data frame).

Egymás után több azonos nevű objektumot létrehozva, az utóbbi felülírja az előbbit. Mivel az R kis- és nagybetű érzékeny, ezért ugyanaz a név kis- és nagybetűvel írva különböző objektumokat fog jelölni.

4. példa: Tároljuk el a 10 és a 20 számot objektumként, majd jelenítésük meg a képernyőn!

```
objektum <- 10
OBJEKTUM <- 20
objektum
[1] 10
OBJEKTUM
[1] 20
```

Az objektumok, változók nevében szerepelhet pont, alulvonás és különleges karakter is, utóbbiak használata azonban kerülendő. Ha a név ponttal kezdődik, akkor a második karakter nem lehet szám. Névként nem használhatók az alábbi „foglalt” kifejezések, noha egy objektumnév részeként alkalmazhatók: *TRUE FALSE NULL Inf NaN NA NA_integer_ NA_real_ NA_complex_ NA_character_*. (A kis- és nagybetű érzékenység miatt kizárólag a teljes egyezés kerülendő az előbb felsorolt szavakkal.)

Beépített állandók: az R-ben kevés beépített változó érhető el. Ezek az alábbiak:

- `pi`: értéke: 3,141593
- `letters`: az angol abc kisbetűi
- `LETTERS`: az angol abc nagybetűi
- `month.name`: a hónapok nevei angolul
- `month.abb`: a hónapok hárombetűs rövidítése angolul

Egyéb konstansok függvényként adhatók meg.

5. példa: Adjuk meg az Euler-féle számot függvénnyel (az exponenciális függvény 1 helyen felvett értéke), tároljuk objektumként és írassuk ki a képernyőre!

```
e <- exp(1)
e
[1] 2.718282
```

Alapműveletek:

Két szám összege: `x+y`

Két szám különbsége: `x-y`

Két szám szorzata: `x*y`

Két szám hányadosa: `x/y`

Hatványozás: Ha `x` a hatvány alapja és `y` a kitevője, akkor: `x**y`

A műveletek `()` zárójelekkel csoportosíthatók, így a műveleti sorrend megváltoztatható.

6. példa: Tároljuk `x` és `y` objektumként egy 1+2 és 2*5 eredményét, majd mentsük ezek hányadosát `z` objektumként, illetve jelenítsük meg a képernyőn!

```
x <- 1+2
y <- 2*5
z <- x/y
z
[1] 0.3
```

Előbbi esetben több utasítást adtunk meg sortöréssel, amelyeket az R egymás után értelmezett. Az utasítások szintaktikailag teljesek voltak. Ellenkező esetben hibajelzést kapunk.

Több utasítás megadása esetén az utasítások **blokk**okba csoportosíthatók, kapcsos zárójelek használatával.

```
{x <- 1+2
y <- 2*5
x/y
z}
[1] 0.3
```

A `{` után a `>` prompt jel + jelre változik. Az utasítások blokkja addig nem teljes, amíg `}` be nem fejezi azt. Az utasítások tehát a `}` jel után hajtódnak végre.

<u>Objektumokkal kapcsolatos alapvető utasítások</u>:

A létrehozott objektumok listázása az `ls()` vagy az `objects()` utasítással lehetséges.
Ha nincs tárolt objektum, akkor a képernyőn `character(0)` szerepel.

Objektum törlésére az `rm(`*`objektum`*`)` utasítás alkalmazható.
Az összes létrehozott objektum az `rm(list=ls())` paranccsal törölhető.

7. példa:

- Listázzuk a meglévő objektumainkat az `ls()` paranccsal!
- Töröljük az 5. példában létrehozott `e` objektumot az `rm(e)` utasítással!

Több érték vagy objektum egymás utáni felsorolására, „összefűzésére” alkalmazható például a `c` függvény, amelynek szintaxisa:

```
c(érték1, érték2, érték3...)
```

8. példa:
Töröljük egyszerre az `x` és az `y` objektumot, a `c` függvény alkalmazásával:

```
rm(list=c("x","y"))
```

Az objektumok típusa, dimenziója:
Az objektumok típusa és dimenziója például a `class` és `dim` függvényekkel jeleníthető meg. Szintaxisuk:

```
class(objektumnév)
dim(objektumnév)
```

Vektorok:

Numerikus vektor:
9. példa: Hozzunk létre két, egész számokból álló vektort és tároljuk ezeket objektumokként!

```
a <- c(1,2,3,4,5)
b <- c(5,4,3,2,1)
a
[1]  1  2  3  4  5
b
[1] 5  4  3  2  1
```

Előbbi objektumok létrehozhatók a következő módon is, hiszen a számsorozatok szabályosak, tagjaik egyenlő távolságra vannak egymástól.

```
a2 <- 1:5
b2 <- 5:1
a2
[1] 1 2 3 4 5
b2
[1] 5 4 3 2 1
```

Az elemek tehát kettőspont alkalmazásával sorolhatók fel.

Fűzzük össze az `a2` és a `b2` vektort a `c` függvény segítségével! (Ne tároljuk objektumként.)

```
c(a2,b2)
[1] 1 2 3 4 5 5 4 3 2 1
```

A vektor eltolható, nyújtható, zsugorítható:
10. példa: Hozzunk létre a [0,9] intervallumon szabályos számsorozatot az [1,10] intervallumon létrehozott számsorozat eltolásával és tároljuk objektumként! (Fontos: A `c` függvény mellett létrehozhatunk `c` nevű objektumot is. Egyidejű létezésük nem okoz problémát.)

```
c <- 1:10-1
c
[1] 0 1 2 3 4 5 6 7 8 9
```

Nyújtsuk a kétszeresére az `a` vektort és zsugorítsuk a felére a `b` vektort! Tároljuk ezeket is objektumokként!

```
d <- a*2
d
[1]  2  4  6  8 10
d2 <- b/2
d2
[1] 2.5 2.0 1.5 1.0 0.5
```

Alapműveletek a numerikus vektorokkal is végezhetők:

11. példa: Végezzünk alapműveleteket a vektorokkal!

Két vektor komponensenkénti összege, pl.

```
d+d2
[1] 4.5  6.0  7.5  9.0 10.5
```

Két vektor komponensenkénti különbsége, pl.

```
a-b
[1] -4 -2  0  2  4
```

Két vektor komponensenkénti szorzata, pl.

```
d*d2
[1] 5 8 9 8 5
```

Megjegyzés: két, különböző hosszúságú vektor esetén a rövidebb vektor komponenseit újból felsorolja az R és ezekre hajtja végre a további műveleteket, pl.

```
a+c
[1]  1  3  5  7  9  6  8 10 12 14
```

Vektor szorzása skalárral, pl.

```
a*9
[1]  9 18 27 36 45
```

Skalárszorzás, pl.

```
skalarszorz <- a%*%b
skalarszorz
      [,1]
[1,]   35
```

Ellenőrizzük le a `skalarszorz` objektum típusát a következőképpen:

```
class(a%*%b)
[1] "matrix"
```

Az eredményül kapott egyelemű mátrix a `drop` függvénnyel alakítható skalárrá. Alakítsuk skalárrá és jelenítsük meg az objektum típusát a `class` függvénnyel!

```
drop(skalarszorz)
[1] 35
class(drop(skalarszorz))
[1] "numeric"
```

Egy objektum, így vektor adott komponense vagy komponensei a [] zárójelek használatával jelölhető ki.

```
objektum[sorszám]
objektum[c(sorszám_1,sorszám_2)]
objektum[sorszám_1:sorszám_n]
```

Vektor adott komponense vagy komponensei ki is hagyhatók a felsorolásból:

```
objektum[-sorszám]
objektum[c(-sorszám_1,-sorszám_2)]
```

(Ezek természetesen új objektumokként is tárolhatók.)

12. példa: Jelenítsük meg az a vektor 5. komponensét, majd a b vektor 1. és 4. komponensét. végül a c vektor 2. és 8. komponense közötti értékeket!

```
a[5]
[1] 5

b[c(1,4)]
[1] 5 2

c[2:8]
[1] 1 2 3 4 5 6 7
```

Üres vektor is előállítható a c függvény alkalmazásával:

```
ures <- c()
ures
NULL
```

Ugyanezt az eredményt kapjuk, ha a c függvény argumentumába nullát írunk:

```
ures <- c(0)
```

Stringek:

13. példa: Hozzunk létre karakterláncból álló vektort a c függvény segítségével!

```
gyumolcs <- c("alma","mandarin","szilva","narancs")
gyumolcs
[1] "alma" "mandarin" "szilva" "narancs"
```

Ennek a vektornak is megjeleníthető adott sorszámú komponense, illetve lekérhető a típusa:

```
class(gyumolcs)
[1] "character"
```

Logikai értékű vektor: TRUE és FALSE értékekből álló vektorok

14. példa: Hozzuk létre a „hamis”, „igaz”, „igaz” logikai értékű vektort a c függvénnyel és tároljuk objektumként!

```
logikai <- c(FALSE,TRUE,TRUE)
logikai
[1] FALSE  TRUE  TRUE
```

Faktorok:

Olyan speciális objektumok, amelyek azonos hosszúságú vektorok komponenseit osztályozzák. A faktor tehát numerikus kódolású vektort jelent szövegértékű szintekkel.

15. példa: Egy adatsor `honapok` oszlopában (amelyet most a `honapok <- rep(1:12,3)` függvénnyel hoztunk létre beolvasott állomány hiányában) numerikusan szerepelnek a hónapok, három évre vonatkozóan. (A `rep` függvényt lásd az Adatgeneralas fejezetben.)

```
honapok
[1]  1  2  3  4  5  6  7  8  9 10 11 12  1  2  3  4  5  6  7
8  9 10 11 12  1  2  3  4  5  6  7  8  9 10 11 12
```

Rendeljük ezekhez a hónapok nevét! Az osztályozásra hozzuk létre a `honap_faktor` nevű faktort!

```
honap_faktor <- factor(honapok, levels=c(1:12),
labels=c("január", "február", "március", "április", "május",
"június", "július", "augusztus", "szeptember", "október",
"november", "december"))
```

A `honap_faktor` objektum létrehozatalával az osztályozásra sor került.

```
honap_faktor
[1] január     február    március    április    május
június     július     augusztus  szeptember október
november   december   január

[14] február    március    április    május      június
július     augusztus  szeptember október    november
december   január     február

[27] március    április    május      június     július
augusztus  szeptember október    november   december

Levels: január február március április május június július
augusztus szeptember október november december
```

Ellenőrizzük, hogy az objektum típusa valóban faktor!

```
class(honap_faktor)
[1] "factor"
```

Mátrixok, tömbök:

Egy mátrix vagy tömb lehet többek közt numerikus, string, logikai, de egy tömbön belül többféle adattípus nem alkalmazható.

Létrehozásuk módja:

```
matrix(objektum vagy érték, nrow=sorok száma, ncol=oszlopok
száma, byrow = FALSE vagy TRUE)
```

vagy:

```
array(objektum vagy érték, dim=c(dim_1, dim_2, ..., dim_n))
```

Kétdimenziós tömbök:

16. példa: Állítsuk elő az alábbi két mátrixot!

Az `A` mátrix legyen 10x10-es, és töltsük fel az 1, 2, …, 100 számokkal!

```
A <- array(1:100, dim=c(10,10))
A
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1   11   21   31   41   51   61   71   81    91
 [2,]    2   12   22   32   42   52   62   72   82    92
 [3,]    3   13   23   33   43   53   63   73   83    93
 [4,]    4   14   24   34   44   54   64   74   84    94
 [5,]    5   15   25   35   45   55   65   75   85    95
 [6,]    6   16   26   36   46   56   66   76   86    96
 [7,]    7   17   27   37   47   57   67   77   87    97
 [8,]    8   18   28   38   48   58   68   78   88    98
 [9,]    9   19   29   39   49   59   69   79   89    99
[10,]   10   20   30   40   50   60   70   80   90   100
```

A `B` mátrix legyen 10x10-es, és töltsük fel a 100,99, …, 1 elemekkel!

```
B <- matrix(100:1, nrow=10, ncol=10)
B
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]  100   90   80   70   60   50   40   30   20    10
 [2,]   99   89   79   69   59   49   39   29   19     9
 [3,]   98   88   78   68   58   48   38   28   18     8
 [4,]   97   87   77   67   57   47   37   27   17     7
 [5,]   96   86   76   66   56   46   36   26   16     6
 [6,]   95   85   75   65   55   45   35   25   15     5
 [7,]   94   84   74   64   54   44   34   24   14     4
 [8,]   93   83   73   63   53   43   33   23   13     3
 [9,]   92   82   72   62   52   42   32   22   12     2
[10,]   91   81   71   61   51   41   31   21   11     1
```

A mátrix alapértelmezetten oszloponként kerül feltöltésre az adatokkal. A `matrix` függvény `byrow = TRUE` paraméterével állítható be soronkénti feltöltés.

A két, különböző függvénnyel létrehozott tömb mátrix típusú:

```
class(A)
[1] "matrix"
class(B)
[1] "matrix"
```

Ha a sorok és az oszlopok számának szorzata nem egyenlő a mátrix elemeinek számával, akkor a szükségesnél kevesebb elemet fog tartalmazni a mátrix.

17. példa:

```
C <- matrix(1:100, nrow=2, ncol=5)
C
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
```

Megjegyzés: Idősorok és mezők elemzése során a számítások eredményeit tömbben tárolhatjuk. A tömb szerkezetének pontos ismerete hiányában az adatok tévesen kerülhetnek ábrázolásra, téves következtetéseket vonhatunk le.

Alapműveletek: mátrix komponensei között is végezhetők műveletek.

18. példa:

Szorozzuk meg az A mátrix komponenseit kettővel:

```
D <- matrix(2*1:100, nrow=5, ncol=10) # vagy
D <- matrix(2*A, nrow=5, ncol=10)
D
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    2   12   22   32   42   52   62   72   82    92
[2,]    4   14   24   34   44   54   64   74   84    94
[3,]    6   16   26   36   46   56   66   76   86    96
[4,]    8   18   28   38   48   58   68   78   88    98
[5,]   10   20   30   40   50   60   70   80   90   100
```

Mátrixműveletek:

19. példa:

Szorozzuk be az A mátrixot egy skalárral!

```
3 * A
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    3   33   63   93  123  153  183  213  243   273
 [2,]    6   36   66   96  126  156  186  216  246   276
 [3,]    9   39   69   99  129  159  189  219  249   279
 [4,]   12   42   72  102  132  162  192  222  252   282
 [5,]   15   45   75  105  135  165  195  225  255   285
 [6,]   18   48   78  108  138  168  198  228  258   288
 [7,]   21   51   81  111  141  171  201  231  261   291
 [8,]   24   54   84  114  144  174  204  234  264   294
 [9,]   27   57   87  117  147  177  207  237  267   297
[10,]   30   60   90  120  150  180  210  240  270   300
```

Transzponáljuk az A mátrixot!

```
t(A)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1    2    3    4    5    6    7    8    9    10
 [2,]   11   12   13   14   15   16   17   18   19    20
 [3,]   21   22   23   24   25   26   27   28   29    30
 [4,]   31   32   33   34   35   36   37   38   39    40
 [5,]   41   42   43   44   45   46   47   48   49    50
 [6,]   51   52   53   54   55   56   57   58   59    60
 [7,]   61   62   63   64   65   66   67   68   69    70
 [8,]   71   72   73   74   75   76   77   78   79    80
 [9,]   81   82   83   84   85   86   87   88   89    90
[10,]   91   92   93   94   95   96   97   98   99   100
```

Adjuk meg A és a B mátrixszorzatát!

```
A %*% B
      [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8] [,9] [,10]
 [1,] 43105 38505 33905 29305 24705 20105 15505 10905 6305  1705
 [2,] 44060 39360 34660 29960 25260 20560 15860 11160 6460  1760
 [3,] 45015 40215 35415 30615 25815 21015 16215 11415 6615  1815
 [4,] 45970 41070 36170 31270 26370 21470 16570 11670 6770  1870
 [5,] 46925 41925 36925 31925 26925 21925 16925 11925 6925  1925
 [6,] 47880 42780 37680 32580 27480 22380 17280 12180 7080  1980
 [7,] 48835 43635 38435 33235 28035 22835 17635 12435 7235  2035
 [8,] 49790 44490 39190 33890 28590 23290 17990 12690 7390  2090
 [9,] 50745 45345 39945 34545 29145 23745 18345 12945 7545  2145
[10,] 51700 46200 40700 35200 29700 24200 18700 13200 7700  2200
c(1:2) %*% t(a)
```

Invertáljuk a solve függvénnyel az alábbi két soros, két oszlopos mátrixot, amelynek elemei:

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

```
solve(matrix(4:1,2,2))
     [,1] [,2]
[1,] -0.5    1
[2,]  1.5   -2
```

Mátrix adott sora, oszlopa vagy eleme a [] zárójelekkel jeleníthető meg:

```
objektum[sorszám,]
objektum[,oszlopszám]
objektum[sorszám,oszlopszám]
```

Mátrix adott sora vagy oszlop a következőképpen hagyható ki a megjelenítésből:

```
objektum[-sorszám,]
objektum[,-oszlopszám]
```

Megjegyzés: számítások elvégzése vagy adatábrázolás során szükség lehet a tömb adott részének kijelölésére.

20. példa: Jelenítsük meg a D mátrix első oszlopát, első sorát, az első elemét, végül pedig a második, harmadik sorának és második, harmadik oszlopának közös elemeit!

```
D[,1]
[1]  0 20 40 60 80
```

```
D[1,]
[1]  2 12 22 32 42 52 62 72 82 92
```

```
D[1,1]
2
```

```
D[2:3,2:3]
     [,1] [,2]
[1,]   14   24
[2,]   16   26
```

Mátrixok összekapcsolása:
Az első objektumhoz a második objektum kijelölt oszlopának kapcsolása:
`objektum <- cbind(objektum1, objektum2[,oszlopszám])`

Az első objektumhoz a második objektum kijelölt sorának kapcsolása:
`objektum <- rbind(objektum1, objektum2[sorszám,])`

21. példa: Készítsünk új mátrixot az `A` mátrix első és a `B` mátrix második sorából!

```
E <- rbind(A[1,], B[2,])
E
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1   11   21   31   41   51   61   71   81    91
[2,]   99   89   79   69   59   49   39   29   19     9
```

Többdimenziós tömbök:
Az `array` függvénnyel kettőnél több dimenziós tömböket is létrehozhatunk.

22. példa: Hozzuk létre a `tomb` nevű háromdimenziós, 10x10x10-es tömböt és töltsük fel 1-től 1000-ig a természetes számokkal!

```
tomb <- array(1:1000,dim=(c(10,10,10)))
```

Ellenőrizzük a típusát és dimenzióját!

```
class(tomb)
[1] "array"

dim(tomb)
[1] 10 10 10
```

A tömb 10 darab kétdimenziós táblázatból áll. Jelenítsük meg az 1. és a 10. táblázatot a következőképpen!

```
tomb[,,1]
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1   11   21   31   41   51   61   71   81    91
 [2,]    2   12   22   32   42   52   62   72   82    92
...
[10,]   10   20   30   40   50   60   70   80   90   100

tomb[,,10]
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]  901  911  921  931  941  951  961  971  981   991
 [2,]  902  912  922  932  942  952  962  972  982   992
 ...
[10,]  910  920  930  940  950  960  970  980  990  1000
```

Esetenként szükségünk lehet adott dimenziójú, üres tömb létrehozatalára. Ekkor az `array` függvény argumentumából a tömbbe töltendő adatokat elhagyjuk, azaz, pl. 10x10x10-es üres tömb létrehozatalára a következő utasítás adandó ki:

```
tomb <- array(dim=(c(10,10,10)))
```

vagy:

```
tomb <- array(0, dim=(c(10,10,10)))
```

3. ADATGENERÁLÁS:

A `seq` függvénnyel szabályos sorozatok hozhatók létre:

```
seq(from=sorozat kezdete, to=sorozat vége,
by=növekedés/csökkenés mértéke)
```

23. példa: Hozzunk létre az [1,20] intervallumon kettesével növekvő számsorozatot!

Megjegyzés: A `from=`, `to=`, `by=` paraméterek kiírása nem szükséges, elegendő megadni a kezdeti és a végső értéket, valamint a növekedés vagy a csökkenés mértékét.

```
sor1 <- (seq(1,20,2))
sor1
[1]  1  3  5  7  9 11 13 15 17 19
```

Hozzunk létre a [20,1] intervallumon kettesével csökkenő számsorozatot!

```
sor2 <- (seq(20,1,-2))
sor2
 [1] 20 18 16 14 12 10  8  6  4  2
```

A sorozat tagjainak sorrendje a `rev` függvénnyel is megfordítható:

24. példa: A `sor3` legyen a [100,0] intervallumon tízesével csökkenő számok sorozata!

```
sor3 <- rev(seq(0,100,10))
sor3
[1]  100  90  80  70  60  50  40  30  20  10   0
```

A `rep` függvény egy sorozat valahányszor történő ismétlésére alkalmazható:

```
rep(ismétlendő sorozat, times vagy each=ismétlődés száma)
```

25. példa: Hozzunk létre az [1,5] intervallumon egyesével növekvő sorozatot, ismételjük meg kétszer a teljes sorozatot egymás után és írassuk ki a képernyőre!

```
sor4 <- 1:5
sor4
[1] 1 2 3 4 5
sor5 <- rep(sor4, times=2)
sor5
[1] 1 2 3 4 5 1 2 3 4 5
```

Előbbi egy lépésben is elvégezhető:

```
sor4 <- rep(1:5, times=2)
sor4
[1] 1 2 3 4 5 1 2 3 4 5
```

Az `each` paraméterrel minden egyes elem egymás után is megismételhető. Például ismételjük meg a sor4 objektum minden elemét egymást követően kétszer!

```
rep(sor4, each=2)
[1] 1 1 2 2 3 3 4 4 5 5 1 1 2 2 3 3 4 4 5 5
```

A létrehozott objektumok hossza a `length` függvénnyel határozható meg.

26. példa: Ellenőrizzük a tíz tagból álló `sor4` sorozat hosszát!

```
length(sor4)
[1] 10
```

Az R-ben létrehozhatók adott valószínűségi eloszlású véletlenszám-sorozatok is:

Az R-be épített eloszlások és szintaxisuk az alábbi címen érhetők el, általánosan a generálandó adatsor elemszáma és az eloszlás paraméterei adandók meg.
https://stat.ethz.ch/R-manual/R-devel/library/stats/html/Distributions.html

27. példa: Hozzuk létre az alábbi eloszlású számsorozatokat!

Geometriai eloszlású, 100 tagú számsorozat képzése, p= 0,5 valószínűséggel:

```
rgeom(100, p=1/2)
```

Poisson-eloszlású, 100 tagú számsorozat képzése, $\lambda = 5$ paraméterrel:

```
rpois(100, lambda=5)
```

Normális eloszlású, 200 tagú számsorozat képzése, 0 átlaggal és 1 szórással:

```
rnorm(200, mean=0, sd=1)
```

Egyenletes eloszlású, 200 tagú számsorozat képzése, a 0-1 zárt intervallumon:

```
runif(200, min=0, max=1)
```

4. ADATOK FÁJLBA ÍRATÁSA:

Adatok fájlba való kiíratására alkalmas a `cat` függvény .

28. példa: Írassuk ki fájlba, hogy „Hello, világ!”, két sorban!

```
cat("Hello", file="proba.txt", sep="\n")
cat(" világ!", file="proba.txt", append=TRUE)
```

Sortörés a `sep="\n"` paraméterrel, szóköz a `sep="\ "` paraméterrel adható meg:

Az `append=FALSE` paraméter nem kapcsolja össze a sorokat, csak az utolsó sor tartalma, a példa esetében a `"világ!"` jelenne meg az állományban. Ez az alapértelmezett beállítás.

A `write.table` függvénnyel is kiírathatók az adatsorok fájlba.

```
write.table(objektumnév, "fájlnév.txt", sep="", na="NA",
dec=",", col.names=TRUE vagy FALSE, row.names=TRUE vagy FALSE)
```

`objektumnév`: Az objektum, amit ki szeretnénk íratni.
`"fájlnév.txt"`: A létrehozandó állomány neve.

Adott sor elemeit tagoló szimbólumot a `sep` paraméterrel lehet szabályozni.
`sep=""` – tagolás tabulátorral vagy szóközzel (alapértelmezett)
`sep=","` – tagolás vesszővel

`sep=";"` – tagolás pontosvesszővel

A tizedesjel a `dec` paraméterrel állítható be, alapértelmezetten pont.

Hibás vagy hiányzó értéket helyettesítő szöveg vagy szám az `na` paraméterrel állítható be. (A fenti esetben a létrehozott állományban `NA` jelölné ezeket az értékeket.)

`col.names`: Az oszlopok azonosítóit tartalmazza-e az állomány. Alapértelmezetten igen.
`row.names`: A sorok azonosítóit tartalmazza-e az állomány. Alapértelmezetten igen.

A fájl nemcsak txt kiterjesztésű lehet, hanem például prn vagy csv is. Utóbbi kiterjesztésű állomány létrehozatalára a `write.csv` függvény is alkalmazható, a `write.table` függvényhez hasonló szintaktikával.

29. példa: Írassuk ki fájlba az `A` mátrixot, sorazonosítók nélkül, tabulátorral tagolva, oszlopazonosítókkal!

```
write.table(A, "adat.txt", sep="\t", row.names=FALSE)
```

(A `row.names=FALSE` nélkül több oszlopunk volna, mint oszlopazonosítónk.)
Az adatok vágólapra is másolhatók és táblázatkezelőbe illeszthetők.
Az állomány a munkavégzés mappájába mentődik.

5. R CSOMAGOK (R PACKAGES):

Az R-ben telepíthetők kiegészítő csomagok, azaz package-ek, amelyek az R-ben alapértelmezetten el nem érhető algoritmusokat, függvényeket tartalmaznak.

Telepítés:

A telepítés legegyszerűbb módja az R-ben a Load/Install package(s)... menüpont alatt a legördülő listából kiválasztani a szükséges csomagot, a HTTPS CRAN mirror kijelölése után. (Általában a földrajzilag legközelebbi hely kiválasztását javasolják.)

A csomagok a https://cran.r-project.org/web/packages/ oldalról is letölthetők. Ez akkor lehet célszerű, ha nem a legújabb R verzióval dolgozunk és régebbi verziószámú csomagot kell telepítenünk. Ezek a https://cran.r-project.org/src/contrib/Archive/ oldalon találhatók meg.

Rendszergazdai jogosultsággal a következőképpen is telepíthetünk csomagokat: az egyes csomagok honlapján letölthetők Windows alatt az r-release zip fájlok. A kitömörített mappa a telepített R programot tartalmazó könyvtár *Library* mappájába másolandó, majd az alábbi utasítással indítható a telepítés:

```
install.packages("csomagnév")
```

A csomagot tetszőleges mappába is telepíthetjük. Ekkor a telepítés és a betöltés módja:

```
install.packages("csomagnév", lib="/elérési út/")
library(csomagnév, lib.loc="/elérési út/")
```

Ha korábban már letöltöttük a csomagokat, akkor internetkapcsolat nélkül is telepíthetők:

```
install.packages("csomagnév.kiterjesztés", repos="NULL",
type=...)
```

`type="source"`: ha tar.gz kiterjesztésű állomány áll rendelkezésünkre,

`type="win.binary"`: ha zip állományt töltöttünk le.

Internetkapcsolat nélküli telepítéskor problémát jelenthet, hogy számos csomag adott verziószámú egyéb csomagok meglététől függ, amelyek beszerzéséről vagy frissítéséről szintén gondoskodnunk kellett korábban. Pl. a `fields` csomag nem működik a `maps` csomag nélkül. Ha van internetkapcsolatunk, akkor a függőségben lévő csomagok is telepítésre vagy frissítésre kerülnek, manuális beavatkozás nélkül.

A telepített csomagok a következő paranccsal nyithatók meg és csukhatók be:

```
library(csomagnév)
detach_package(csomagnév)
```

Az R-ben alapértelmezetten megtalálható, illetve a telepített csomagok az `installed.packages()` utasítással listázhatók ki. Az elérhető csomagok a `library()` utasítással is megtekinthetők.

Megjegyzés: A bevezetés során az általunk használt csomagok és dokumentációjuk:
NetCDF állományok kezeléséhez:
ncdf4: https://cran.r-project.org/web/packages/ncdf4/index.html

Térképes ábrázoláshoz:
maps: https://cran.r-project.org/web/packages/maps/index.html
maptools: https://cran.r-project.org/web/packages/maptools/index.html (szükséges csomag: sp)
mapdata: https://cran.r-project.org/web/packages/mapdata/index.html (szükséges: maps)
rgeos: https://cran.r-project.org/web/packages/rgeos/index.html

Image.plot függvényt tartalmazó csomag:
fields: https://cran.r-project.org/web/packages/fields/index.html (szükséges: spam, maps)

Színskála készítéséhez:
RColorBrewer: https://cran.r-project.org/web/packages/RColorBrewer/index.html

FELHASZNÁLT IRODALOM:

Solymosi Norbert: Bevezetés az R-nyelv és környezet használatába, jegyzet
https://cran.r-project.org/doc/contrib/Solymosi-Rjegyzet.pdf

Abari Kálmán: Bevezetés az R-be, jegyzet
http://psycho.unideb.hu/munkatarsak/abari_kalman/szamitastechnika_II/bevezetes_az_R_be_2008_04.pdf