Introduction to R

Zoltan Kekecs

September 7, 2021

Contents

1 Bevezetés az R programnyelvbe

1.1 Absztrakt

Ez a gyakorlat egy bevezető az R program alkalmazásába. A gyakorlat bemutatja az R legalapvetőbb funkcióit, a konzol és a script használatát. Bemutatja a különböző adatfajtákat, mint pl. a vektor, a mátrix és az adattábla, és megismertet néhány alapvető adatkezelési művelettel.

2 Bevezetés az R programozásba

Az R egy programozási nyelv. Ha az R programnyelv kifejezéseit kell használnunk ahhoz, hogy használjuk a programot.

Például az R ismeri a matematika alapvető kifejezéseit.

```
21 + 16
```

```
## [1] 37
```

```
3 * 5
```

```
## [1] 15
```

```
10 * 2 / 5 + 2
```

```
## [1] 6
```

3 Konzol

Az R programozás során mondatokat használunk, hasonlóképpen ahhoz mintha emberekkel kommunikálnánk írásban. Az R-nek szánt parancsokat (mondatokat) beírhatjuk egyenesen a konzolba (ez általában a bal alsó panel szokott lenni az RStúdióban, amin az a felirat szerepel hogy "Console").

Próbáld ki most ezt, írj be egy matematikai műveletet közvetlenül a konzolba a fenti kódhoz hasonlóan, és nyomd meg az ENTER-t (pl. 12/3, egyenlőség jelet nem kell tenni).

Az R lefuttatja a kért műveletet, és a konzolban megjeleníti az eredményt közvetlenül a lefuttatott parancs sora után. Vedd észre hogy a lefuttatott parancsot tartalmazó sor elején mindig egy ">" jel van. Az eredménysor eleján pedig a következővel kezdődik: "[1]" Ezeket a jeleket nem kell értelmeznünk.

4 Script

Bár megtehetnénk, hogy mindent közvetlenül a konzolba írunk, így nagyon nehezen lehetne reprodukálni amit csináltunk, mert a konzolban a kód egy idő után kitörlődik, és különben is nehezen átlátható a konzol néhány parancs után. Ehelyett a kódunkat általában egy szöveges fájlba mentjük, ezt gyakran "script"-nek hívják, mert olyan mint egy forgatókönyv. Ezt a script-et előre megírjuk, és ez alapján a forgatókönyv alapján fogja teljesíteni az R a parancsokat, amikor szeretnénk azokat lefuttatni.

A script általában a bal felső panel az RStudio-ban. Néha először csak a konzolt látjuk. Ilyenkor a File menüben a New file > R Script kiválasztásával tudunk egy script panelt nyitni. Most tedd ezt meg, és írj a script panelba néhány matematikai műveletet enterrel elválasztva. Pl.:

2 * 4

10 / 5

Láthatod, hogy amikor itt enter-t nyomsz, a kód nem fut le, nincs eredmény. Ez jó így, itt előre megírhatjuk a kódot, és addig nem fut le, amíg mi erre utasítást nem adunk.

Most adjunk utasítást a kód lefuttatására. Ezt több módon is megtehetjük. A legegyszerűbb módja, ha az egérrel kijelöljük a kódot amit szeretnénk lefuttatni, és megnyomjuk a Ctrl+ENTER billentyűket egyszerre. Ekkor a kijelölt kódrész lefut a konzolban, mintha egyenesen oda írtuk volna be.

Ennél méggyorsabb megoldás, ha a lefuttatni kívánt kód sorába kattintunk az egérrel, és megnyomjuk a Ctrl+ENTER-t. Ekkor az a sor ahol a kurzor volt lefut, és a kurzor átugrik a következő sor elejére.

Próbáld most ki a következőt: Írj be 3 matematikai műveletet egymás alá új sorokba a scriptedbe (az ENTER segítségével tudsz új sort kezdeni). Majd vidd a korzort a legelső sorba (nem kell semmit kijelölni, csak bizonyosodj meg róla hogy a kurzor valahol a megfelelő sorban van), és nyomd meg a Ctrl+ENTER billenyűket többször egymás után. Azt kell tapasztalnod, hogy a parancsok soronként lefutnak az R-ben.

5 Befejezett és befejezetlen parancsok

Most próbáld meg lefuttatni a következő parancs-részletet:

```
9 / 3 + (6 *
```

Vedd észre hogy ez egy **parancs-töredék**, nincs befejezve, mert az R vár még valamit a "mondat" végére (tudnia kell, mivel kell megszorozni a 6-ot). Amikor a konzolban megpróbáljuk ezt lefuttatni, vedd érszre hogy a szokásos ">" vagy "[1]" helyett egy "+" jel szerepel a sor elején! Ez azt jelenti, hogy az R vár még valamit a parancs befejezéséhez, hogy parancs-töredéket futtattunk le. Az R tanulása közben előfordul, azt vesszük érszre, hogy az R "nem teljesíti a parancsokat", nem ad ki eredményt. Ez általában amiatt van hogy a kódsoroknak véletlenül csak töredékét futtatjuk le, és emiatt valamit befejezetlenül hagytunk, és az R várja, hogy "befejezzük a mondatot". Ezt a legkönnyebben úgy vehetjük észre, hogy az R konzolban "+" jelet látunk a sor elején.

A fenti példában tudjuk hogy mi a kódtöredék, és tudjuk, mivel lehet befejezni a kódot, egy számmal és egy bezáró zárójellel. Szóval írj egy számot a konzolba és nyomd meg az ENTER-t. Ezzel befejeződik a kód és a konzolban láthatjuk a művelet eredményét. Azonban gyakran amikor a konzolban "+" jelet látunk, nehéz visszafejteni, hogy mi az a mondatbefejezés amire az R vár, mert nem tudjuk, hol rontottuk el a kódot. Ilyenkor a legegyszerűbb, ha egy vagy néha több bezáró zárójelet) -et írunk a konzolba és megnyomjuk az ENTER-t. Ekkor általában egy hibaüzenetet kapunk, vagyis a kódunk nem fut le, de legalább újra az ismerős ">" jel fogad minket, ami azt mutatja, hogy az R készen áll új parancsra. Szóval ha valamikor megakadtok és úgy tűnik az R nem válaszol, ellenőrizzétek hogy ">" jel van-e a konzolban. Ha ">" jel helyett "+"-t találtok, akkor zárjátok le a mondatot, ha kell a) használatával. Ha nem vagytok benne biztosak, meddig futott le a kódotok, sokszor érdemes a script-et újra a legelejétől lefuttatni. Sőt, néha a programot is érdemes újraindítani ez előtt, vagy legalább a "munkaterületet" (Workspace) kitisztítani a korábbról fennmaradó kódoktól, objektumoktól.

6 Funkciók

Az R-ben a különböző műveleteket legtöbbször funciók segítségével végezzük el. Ezek az R programnyelv "igéi".

Például egy szám természetes alapú logaritmusát a log() funcióval tudhatjuk meg.

```
log(2)
```

```
## [1] 0.6931472
```

Rengeteg funkcióval fogunk megismerkedni a kurzus során.

Magunk is csinálhatunk funkciót (erre számos tananyagot találhatsz az interneten), de a legtöbb funkció ami ehhez a bevezetés kurzushoz kell már megtalálható az R-ben.

```
# This function adds 1 to the entered number

pluszegy <- function(szam){
  eredmeny = szam + 1
  return(eredmeny)
}

# This is how you use it

pluszegy(szam = 5)
```

```
## [1] 6
```

7 Objektumok

Ha a funckiók az R programnyelv igéi, akkor az objektumok a főnevek. Az objektumok általában olyan adatok amiknek valami nevet adtunk, hogy később könnyebben elérhessük, hivatkozhassunk rá.

pl. az alábbi kóddal hozzárendelhetjük a "Tesla model S" szöveget a dreamcar objektumhoz.

```
dreamcar <- "Tesla model S"
```

Innentől kezdve mindig amikor előhívjuk ezt a dreamcar objektumot, akkor az R tudja hogy a "Tesla model S"-re hivatkozunk.

Az objektum lefuttatásával az R a konzolban **kilistázza** a benne foglalt elemeket. (A "kilistázás" annyit jelent, hogy kiírja a konzolba az objektum tartalmát)

```
dreamcar
```

```
## [1] "Tesla model S"
```

A "<-" és a "=" felcserélhető. Ugyan azt csinálja.

```
dreamcar = "Tesla model S"
```

Az objektumokkal ugyan úgy tudunk műveleteket végezni, mint azok tartalmával. Pl.:

```
number1 <- 5
number2 <- 2

number1 + number2
```

```
## [1] 7
```

```
sum_total = number1 + number2

sum_total
```

```
## [1] 7
```

_____*Gyakorlás*_____

1. Rendeld hozzá a 4-et egy új *number* nevű objektumhoz.
2. vond ki a number1 és a number2 összegéből ezt az objektumot, és az eredményt rendeld hozzá egy *result* nevű objektumhoz.
3. listázd ki a *result* objektumot hogy leellenőrizd, helyes eredményt kaptál-e

_____

8 Munkaterület (workspace, environment)

Az RStúdióban jobb felül található általában a munkakörnyezet (Environment), ami tartalmazza az éppen aktuálisan használt adatokat, objektumokat, egyéb nevesített értékeket. Az itt található kis seprű ikon megnyomásával kitisztíthatjuk a "munkaterületet" (Workspace) a korábbról fennmaradó objektumoktól. Ezt érdemes megtenni ha újra akarjuk futtatni a kódunkat.

9 Csomagok (Packages)

Az R egyik előnye hogy a közösség folyamatosan fejleszti. Ezt úgy érik el, hogy az R rendelkezik néhány nagyon stabil és standardizált alapfunkcióval, amire mások ráépíthetik saját kódjaikat. Amikor ezek a saját kódok és funkciók egy hasznos egységgé állnak össze, ezeket úgynevezett funkció "csomag" (package)-ként adják ki a programozók. Ezek a package-ek nagyon megnövelik az R funkcionalitását és felhasználóbarátságát.

Számunkra a legalapvetőbb csomag amit használni fogunk a "tidyverse". Ez valójában egy csomag-család, ami tiszta, olvasható programozást tesz lehetővé.

A jobb alsó panel az RStúdióban általában tartalmaz egy **Packages tab-ot**. Erre kattintva megláthatjuk hogy mik a telepített package-ek, és hogy ezek közül melyik van betöltve (amik mellett pipa van).

9.1 Package installálása

A következő parancs arra szolgál, hogy egy google spreadsheet-ből olvassunk be adatot, viszont jelenleg hibaüzenetet ad ki.

```
data = gsheet2tbl("https://docs.google.com/spreadsheets/d/1fxgUb2SOZ_gvXGPIuaTJTkRqt5rvjht0PWeBa_vXvLA/
```

Ez azért van mert nincs benne az R alapcsomagjában (amit R base-nek is szoktak nevezni), hanem egy package, a googlesheets, installálása és betöltése kell hozzá.

Package-eket installálhatunk a konzolból, vagy script-ből a következő kód lefuttatásával: install.packages(). A zárójelbe a package nevét idézőjelben kell betenni.

Pl.:

```
install.packages("gsheet")
```

9.2 Package betöltése

Fontos, hogy nem elég egy package-et installálni, ha a tartalmát használni akarjuk. A package-et be is kell töltenünk. Ezt a library() funkcióval tudjuk megtenni (itt már nincs szükség az idézőjelre a package neve körül).

```
library(gsheet)
```

```
## Warning: package 'gsheet' was built under R version 4.1.3
```

Most már le fog futni a parancs amit az előbb futtatni akartunk, és meg is nézhetjük az adatokat a View() funcióval.

```
data = gsheet2tbl("https://docs.google.com/spreadsheets/d/1fxgUb2SOZ_gvXGPIuaTJTkRqt5rvjht0PWeBa_vXvLA/
```

```
View(data)
```

A jobb alsó panelben a Packages tab-on a package neve mellett lévő üres checkbox-ra, vagy a pipára kattintva is tudjuk befolyásolni, hogy az adott package be legyen-e töltve, vagy ne legyen betöltve.

10 Objektum típusok

10.1 Adatosztályok

Az R többféle adatosztályt és típust különböztet meg.

Vektorok:

A vektor adat-elemek gyűjteménye melyek egy sorban helyezkednek el egymás után. A vektor minden eleme ugyan olyan adatosztályba tartozik.

- karakter vektor (character): "Ez egy karakter érték"
- faktor (factor): Egy olyan karater vektor ami csak rögzített értékeket vehet fel
- szám vektor (numeric): 2 vagy 13.5. A szám vektornak két típusa van: egész szám vektor (integer): 2L (Az L mondja meg az R-nek, hogy az előtte lévő számot, mint egész számot kezelje), vagy racionális szám (double): pl.: 13.5
- logikai vektor (logical): csak TRUE vagy FALSE értékeket vehet fel (nagybetű is számít)
- complex szám vektor (complex): 1+5i

Az adat osztályát a class() függvénnyel tudjuk ellenőrizni, az adott osztályon belüli típusát pedig a typeof() függvénnyel.

```
x_character <- "I love R"

class(x_character)
```

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

```
x_numeric <- 2.34
x_integer <- 2L
x_logical <- TRUE
x_complex <- 1+4i
```

Egy objektumban több adat is szerepelhet, vagyis az adatok adatstruktúrákat alkotnak. Amikor egy objektumban csak egy elem szerepel, azt "atomic vector"-nak, elemi vektornak nevezzük. (Mint a fenti példában x).

A c() fügvénnyel viszont összefűzhetünk több (ugyan olyan típusú) adatot egy vektorrá.

```
numbers <- c(5, 2, 3, 24, 6, 5, 9, 10)

numbers
```

```
## [1]  5  2  3 24  6  5  9 10
```

Komplexebb adatstruktúrák:

- mátrix (matrix): egy olyan vektor, aminek az elemei táblázatba vannak rendezve, fix számú sorban és oszlopban. Csakúgy mint a vektor esetén, a mátrixban minden adat-elem típusa csak ugyan az lehet.
- adattábla (data.frame): egy olyan mátrix, aminben megengedett hogy az oszlopokban egymáshoz képest más adat típus szerepeljen
- lista (list): egy olyan vektor, aminek az elemei lehetnek más adatstruktúrák.

10.2 Objektumok osztályozása (object class)

A *matrix*, amiben az adat-elemek egy táblázatot alkotnak. A mátrixban minden vektor/adat típusa csak ugyan az lehet.

```
my_matrix <- matrix(numbers, nrow = 2)

my_matrix
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    5    3    6    9
## [2,]    2   24    5   10
```

Az **adattábla** (data frame), ami egy olyan mátrix, aminben oszloponként megengedett hogy különbözzön az adat típusa.

```
my_dataframe <- data.frame(my_matrix)

my_dataframe
```

```
##   X1 X2 X3 X4
## 1  5  3  6  9
## 2  2 24  5 10
```

És a **lista**, ami gyakorlatilag egy olyan vektor, aminek az elemei lehetnek más objektumok, vagy akár más listák.

```
my_list <- list(my_dataframe, numbers, x_character)

my_list
```

```
## [[1]]
##   X1 X2 X3 X4
## 1  5  3  6  9
## 2  2 24  5 10
##
## [[2]]
## [1]  5  2  3 24  6  5  9 10
##
## [[3]]
## [1] "I love R"
```

A következő funckiókkal különböző információkat tudhatunk meg az objektumokról: class() - milyen osztályba tartozik az adat (magasabb szintű)? typeof() - milyen al-osztályba tartozik az adat (alacsonyabb szintű)? length() - milyen hosszú a vektor (hány eleme van)? attributes() - objektunm tulajdonságait adja meg, pl. hogy hány sort és oszlopot tartalmaz?

```
class(my_dataframe)
```

```
## [1] "data.frame"
```

```
typeof(my_dataframe)
```

```
## [1] "list"
```

```
length(numbers)
```

```
## [1] 8
```

```
attributes(my_matrix)
```

```
## $dim
## [1] 2 4
```

11 Néhány hasznos rövidítés:

Ha két szám közé kettőspontot teszünk, akkor az R azt egy számsorozatként értelmezi, aminek az első eleme a kettőspont előtti szám, egyesével növekszik, és a kettőspont utáni szám az utolsó eleme. Vagyis ha monoton egyesével növekvő számsort akarunk beírni, akkor ezt könnyedén lerövidíthetjük al alábbi módon

```
1:5
```

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

Az R-ben szintén megtalálható az angol ABC betűi. Az angol ABC kisbetűi a letters objektumban vannak benne, a nagybetűk pedig a LETTERS objektumban. A fenti trükkel tehát elérhetjük.

```
letters
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
## [20] "t" "u" "v" "w" "x" "y" "z"
```

Ha egy vektornak csak bizonyos elemeit szeretnénk használni, akkor a [] és egy szám segítségével megmondhatjuk, melyik részét szeretnénk használni a vektornak. Vagyis a fenti két trükk kombinációjával nagyon rövid kóddal kilkistáztathatjuk az angol ABC első 22 betűjét:

```
letters[1:22]
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
## [20] "t" "u" "v"
```