Az R nyelv

Jeszenszky Péter
Debreceni Egyetem, Informatikai Kar
jeszenszky.peter@inf.unideb.hu

2019. június 12.

Általános tudnivalók

- Az R kisbetű-nagybetű érzékeny.

Objektumok

- Az R által kezelt különböző memóriában tárolt adatszerkezeteket objektumoknak nevezzük.
  - Fajtái közé tartoznak például a vektorok, listák, kifejezés objektumok, környezetek, …
- Az `str(`*x*`)` függvényhívás az argumentumként adott objektum belső felépítését jeleníti meg tömör formában.

Objektumok típusa, konverziók

- A `typeof(`*x*`)` függvény visszaadja az argumentumként adott objektum típusát.
  - A visszatérési érték egy karakterlánc, mint például `"character"`.
- A típusok között elég szabad átjárás van.
  - Kifejezések kiértékelése során automatikus típuskonverziók történnek.
  - Típuskonverziók kényszerítésére számos konverziós függvény áll rendelkezésre.
    - `as.logical(`*x*`)`, `as.integer(`*x*`)`, `as.character(`*x*`)`, ...

NULL objektum

- Egyetlen NULL objektum létezik, melyet a `NULL` speciális konstans reprezentál.
- Általa jelezhető egy objektum hiánya (például az, hogy egy kifejezés vagy függvény értéke nem definiált).
- Nincs típusa.
- Az `is.null(`*x*`)` függvény visszaadja, hogy az argumentuma a NULL objektum-e.
  - Kizárólag ezt használjuk a NULL objektummal való összehasonlításhoz vagy az `identical(`*x*`,NULL)` kifejezést.

Az objektumok attribútumai (1)

- A `NULL` objektum kivételével minden objektumnak lehetnek attribútumai.
- Az attribútumok az objektumokhoz tartozó név-érték párok.
- Egy objektum attribútumainak halmazként kezelése.
  - Egy objektum legfeljebb egy adott nevű attribútummal rendelkezhet.
- Az attribútumok neve tetszőleges karakterlánc, értéke tetszőleges objektum lehet.
  - Azonban vannak olyan attribútumok, melyeknek speciális jelentése van, ezek értékeire előírások vonatkozhatnak.

Az objektumok attribútumai (2)

- Az `attributes(`*x*`)` függvénnyel lehet lekérdezni az argumentumként adott objektum attribútumait.
  - A visszatérési érték az attribútumok listája.
- Az `attributes(`*x*`) <-` *érték* kifejezés beállítja az argumentumként adott objektum attribútumait.
  - *érték* az attribútumokat felsoroló lista, melynek minden elemét névvel kötelező megadni.
  - Az `attributes(`*x*`) <- NULL` kifejezés az adott objektum összes attribútumát törli.

Az objektumok attribútumai (3)

- Az `attr(`*x*`,`*név*`)` függvénnyel lehet lekérdezni egy adott nevű attribútum értékét.
  - A második argumentum az attribútum nevét megadó karakterlánc.
  - Az attribútum nevével teljes vagy egyértelmű prefix egyezés szükséges.
    - Teljes vagy egyértelmű prefix egyezés hiányában `NULL` a visszaadott érték.
- Az `attr(`*x*`,`*név*`) <-` *érték* kifejezés állítja be az adott nevű attribútum értékét.

Az objektumok attribútumai (4)

- Vannak speciális attribútumok értékének lekérdezésére és beállítására szolgáló függvények.
  - Speciális attribútumok például a `names`, `dim`, `dimnames`, `class`.

A names attribútum (1)

- Lehetővé teszi vektorok, listák elemeinek elnevezését.
  - Objektumok kiírásánál fejlécként megjelenik az elemek neve is.
  - Indexelésnél lehet a neveket indexként használni (indexelés karakterláncokkal).
- Értéke egy megfelelő elemszámú karakter vektor.

A names attribútum (2)

- A `names(`*x*`)` függvény használható az attribútum értékének lekérdezésére.
- A `names(`*x*`) <-` *érték* kifejezés beállítja az attribútum értékét.
  - *érték* karakter vektorrá konvertálása.
  - Ha a karakter vektorrá konvertált *érték* rövidebb *x*-nél, kiegészítése `NA` értékekkel (hosszabb azonban nem lehet).
  - Üres karakterlánc (`""`) a vektorban azt jelzi, hogy a megfelelő elemnek nincs neve.
    - Ezt indexként megadva azonban egyetlen elem sem lesz kiválasztva.
  - *érték* lehet speciálisan `NULL`, mely az attribútum törlését jelenti.

Objektumok hossza

- A `length(`*x*`)` függvény szolgáltatja vektorok, listák, faktorok és egyéb objektumok hosszát.
  - Vektorok, listák, faktorok esetében ez az elemek számát jelenti.
- A `length(`*x*`) <-` *érték* kifejezés, ahol *érték* egyelemű egész vektor, beállítja az adott vektor vagy objektum hosszát .
  - Vektor vagy lista hosszának csökkentése csonkolást eredményez.
  - Vektor vagy lista hosszának növelése NA értékekkel kiegészítést eredményez az adott hosszra.
- Karakterláncok karaktereinek számának meghatározására szolgál az `nchar(x)` függvény.

Vektorok

- A legalapvetőbb fajta objektumok a vektorok.
- A tárban folytonosan elhelyezkedő elemekből állnak, melyekhez indexeléssel lehet hozzáférni.
- Nincsenek a vektoroknál egyszerűbb objektumok, a konstansok egyelemű vektorokat ábrázolnak.

A konstansok öt fajtája

- **Logikai konstansok**: `TRUE`, `FALSE`
- **Numerikus konstansok**: `1`, `.5`, `3.141593`, `10e-7`, ...
  - Előjel nélküliek és 64-bites valós számokat ábrázolnak.
- **Egész konstansok**: `1L`, `1234L`, `1e4L`, `0xffffL`, ...
  - Előjel nélküliek 32-bites egész számokat ábrázolnak.
- **Komplex konstansok**: `0i`, `1.5i`, `1e-7i`, ...
  - Nincs valós rész, csak képzetes, valós és képzetes részből álló komplex számok létrehozása operátorokkal történhet.
- **Karakterlánc konstansok**: `"Hello, world!\n"`, `'Hello, world!\n'`

Speciális konstansok: NULL

- A NULL objektum jelzésére szolgáló speciális konstans.

Speciális konstansok: NA

- Hiányzó értéket jelölő speciális konstans.
- Alapértelmezésben logikai típusú konstans (egyelemű logikai vektor), melyet azonban tetszőleges típusúvá lehet konvertálni, így tetszőleges típusú vektorban jelölhet hiányzó elemet.
  - Kivételt képeznek a bináris vektorok, melyekhez nincs `NA` érték.
- `NA` értékeken végzett műveletek eredménye általában `NA`.
  - Kivéve akkor, ha a művelet eredménye az `NA` értékektől függetlenül meghatározható.
- Az `is.na(`*x*`)` függvény szolgál NA érték tesztelésre.
  - Kizárólag ezt használjuk erre a célra, mivel például az `NA==NA` kifejezés értéke is `NA`!

Speciális konstansok: Inf

- A végtelent jelölő speciális numerikus konstans.
- `Inf` és `-Inf` minden matematikai műveletnél és függvénynél megjelenhet operandusként, paraméterként és eredményként is.
  - Például `Inf` az értéke az `1/0` aritmetikai kifejezésnek, a `-1/0` kifejezésnek pedig `-Inf`.

Speciális konstansok: NaN

- Speciális numerikus konstans, mely azt jelenti, hogy egy művelet nem értelmezett.
  - Ez az értéke például a `0/0` vagy az `Inf-Inf` aritmetikai kifejezésnek.
- Minden matematikai operátornál, függvénynél megjelenhet operandusként, paraméterként és eredményként is.
- Az `is.nan(`*x*`)` függvény szolgál `NaN` érték tesztelésre.
  - Kizárólag ezt használjuk erre a célra vagy az `identical(`*x*`,NaN)` kifejezést, mivel például a `NaN==NaN` kifejezés értéke `NA`!

Atomi vektorok (1)

- Atomi vektoroknak nevezzük az azonos típusú elemekből álló vektorokat, melyeknek az alábbi 6 fajtájuk van:
  - Logikai (`"logical"`)
  - Egész (`"integer"`)
  - Valós (`"double"`)
  - Komplex (`"complex"`)
  - Karakter (`"character"`)
  - Bináris (`"raw"`)
- Zárójelben a `typeof()` függvény által adott típus.
- Az egész és valós vektorokat közös néven numerikus vektoroknak hívjuk.

Atomi vektorok (2)

- Atomi vektorokat a `c()` függvénnyel lehet létrehozni.
  - Közös típusra konvertálja és egyetlen vektorrá fűzi össze az argumentumokat.
  - Példa:

```
c("X", "Y", "Z")                # "X" "Y" "Z"
c(c(1, 2), 3, c(4, 5))          # 1 2 3 4 5
c(0i, TRUE, 2)                  # 0+0i 1+0i 2+0i
c(6, "Days", 7, "Nights")       # "6" "Days" "7" "Nights"
```

Vektor-aritmetika (1)

- Vektorok bárhol előfordulhatnak aritmetikai kifejezésekben, ekkor a műveletek elemenként lesznek végrehajtva.
  - Van néhány speciális művelet, mely kivételt jelent, például %*% vektorok esetén a belső szorzatot adja (az eredmény egy egy sorból és oszlopból álló mátrix).
- Az aritmetikai függvények vektorokra alkalmazása is elemenkénti alkalmazást jelent.
- Nagyon komplex kifejezéseket nagyon tömören le lehet írni ilyen módon.

Vektor-aritmetika (2)

- Kifejezésekben szereplő vektorok normalizálása közös hosszra.
  - Ez a rövidebb vektor elemeinek ismétlését jelenti: addig ismétlődnek az elemek, amíg el nem érjük a hosszabb vektor hosszát.
  - Ha a rövidebb vektor elemeinek száma nem osztja a hosszabb vektor elemeinek a számát, akkor is elvégzésre kerül a művelet, de figyelmeztetést kapunk.
  - Egy speciális eset az, ha a rövidebb vektor üres vektor, ekkor az eredmény is üres vektor.

Vektor-aritmetika (3)

- Példa:

```
c(1, 2, 3, 4) * c(2, 3, 4, 5)   # 2 6 12 20
c(5, 6, 7, 8) * 2               # 10 12 14 16
2^(0:7)                 # 1 2 4 8 16 32 64 128

cos(c(-pi, 0, pi))              # -1 1 -1
sin(seq(from = 0, to = pi / 2, length = 100))
exp(-1) / factorial(0:5)

c(1, 2, 3) %*% c(1, 2, 3)       # 14
```

Vektorok indexelése [ ] operátorral (1)

- Az *x* vektor indexelése történhet x[*i*] módon, ahol az *i* index egész, valós, logikai vagy karakter típusú lehet.
  - Az első elem kiválasztásához az 1 indexet kell megadni.
  - Valós indexek egésszé konvertálása.
- Az indexelés nem csak az elemek értékének kinyerésére, hanem az elemek helyettesítésére is használható.

Vektorok indexelése [ ] operátorral (2)

- Az index tetszőleges elemszámú vektor lehet, így akár több elemet is ki lehet választani.
- Az indexet akár el is lehet hagyni, ami az összes elem kiválasztását jelenti.
- Az indexhatár-túllépés megengedett, megfelelően kezelt.

Vektorok indexelése [ ] operátorral (3)

- Ha az index pozitív egészekből álló vektor, akkor a megfelelő elemek kiválasztása történik az adott sorrendben.
  - A vektor hosszánál nagyobb index `NA` értéket eredményez.
- Ha az index negatív egészekből álló vektor, akkor azokat az elemeket adja meg, melyek nem lesznek kiválasztva, az összes többi igen.
  - A vektor hosszánál nagyobb abszolút értékű index esetén hiba.
- Hiba, ha az index vektorban pozitív és negatív értékek is vannak.

Vektorok indexelése [ ] operátorral (4)

- Példa:

```
#  az első 10 elem kiválasztása:
x[1:10]

#  az első, az ötödik, majd ismét az első elem kiválasztása:
x[c(1, 5, 1)]

#  minden páros indexű elem kiválasztása:
x[(1:(length(x) / 2)) * 2]

#  az összes elem kiválasztása minden 5. elhagyásával:
x[-((1:(length(x) / 5)) * 5)]
```

Vektorok indexelése [ ] operátorral (5)

- Ha az index a vektorral azonos elemszámú logikai vektor, akkor a TRUE értékű elemekkel megegyező indexű elemek kiválasztása.
  - Ha az index a vektornál rövidebb logikai vektor, akkor az index vektor elemeinek ismétlése.
  - Ha az index a vektornál hosszabb logikai vektor, akkor az indexelt vektor kiegészítése NA értékekkel.
- Példa:

```
#  a negatív elemek kiválasztása:
x[x < 0]

#  ugyancsak a páros indexű elemeket választja ki:
x[c(FALSE, TRUE)]
```

Vektorok indexelése [ ] operátorral (6)

- Ha az index karakter vektor, akkor a `names` attribútum alapján történik az elemek kiválasztása.
- Teljes névegyezés kell egy elem kiválasztáshoz.
  - Ha több elem esetén is teljes névegyezés van, a sorrendben első kiválasztása.
- Ha elemek kinyerésnél nincs teljes egyezés egyetlen elem nevével sem, akkor az eredmény `NA`.
- Elemek kinyerésekor az indexben megjelenő üres karakterlánc (`""`) és a karakter típusú `NA` érték egyetlen elemet sem választanak ki.

Vektorok indexelése [ ] operátorral (7)

- Példa:

```
area <- c(9596961, 357386, 93030, 377973)
names(area) <- c("China", "Germany", "Hungary", "Japan")

area["Hungary"] # 93030
area[c("Hungary", "Germany")]# 93030 357386
area["Spain"]   # NA
```

Listák (1)

- Olyan vektorok, melyek elemei tetszőleges objektumok lehetnek.
  - Az atomi vektoroktól eltérően az elemek különböző típusúak lehetnek.
  - A listák rekurzív jellegű objektumok, az elemeik lehetnek további listák.
- Az elemekhez történő hozzáférés indexeléssel történik, a vektorokhoz hasonlóan.

Listák (2)

- Listákat a `list()` függvénnyel lehet létrehozni.
  - Tetszőleges számú argumentumot kaphat a függvény, melyek az elemeket adják meg.
  - Az elemeket meg lehet adni *név=érték* módon is, ahol *név* azonosító vagy karakterlánc.
  - A függvényt argumentum nélkül meghívva az eredmény egy üres lista.
- Listákat a `c()` függvénnyel lehet összefűzni.
  - Az argumentumok között egyaránt szerepelhetnek atomi vektorok és listák, ezek valamennyi eleme egy-egy komponense lesz az eredmény listának.

Listák indexelés (1)

- A vektorok indexelésénél elmondottak vonatkoznak a listákra is.
- Azonban használható indexelésre a $ operátor is.
- Az index operátorok révén törölhetők is elemek a listákból.
  - Ha értékadásban az elemek helyettesítésére használjuk az indexelést, akkor a `NULL` érték megadásával lehet törölni a lista megfelelő elemeit.

Listák indexelése (2)

- Listákat indexelni lehet *x*$*név* módon is.
  - A kifejezésben *név* azonosító vagy karakterlánc lehet.
  - Ilyenkor a megfelelő nevű elem kiválasztása történik.
  - Elemek értékének kinyerésénél nem szükséges teljes egyezés.
    - Ha nincs teljes egyezés egyetlen elem nevével sem, akkor elég prefix egyezés, amennyiben az egyértelmű.
    - Teljes egyezés hiányában nem egyértelmű prefix egyezés esetén az eredmény NULL.
  - Elemek kinyerésekor tehát az indexben az elemek nevét rövidíteni lehet, ha a rövidítés egyértelmű.

Listák indexelése (3)

- Az `[ ]` operátor használata esetén az eredmény mindig lista.
  - A megfelelő elemekből álló részlista.
  - Akkor is, ha az index egyelemű vektor (ilyenkor az eredmény egyelemű lista).
- A lista egyetlen elemét választják ki a `[ [ ] ]` és `$` operátorok.

Példa lista használatára (1)

```
x <- list(latitude = 47.49835, longitude = 19.04045)
x$lat                          # 47.49835
x$long                         # 19.04045
y <- list("Debrecen", coordinates = x)
y[[1]] <- "Budapest"
y[["coordinates"]]$lat         # 47.49835
y$coordinates$long             # 19.04045
names(y)                       # "" "coordinates"
names(y)[1] <- "city"
y$city                         # "Budapest"
```

Példa lista használatára (2)

```
z <- list(
   name = list(first.name = "John", last.name = "Cleese"),
   birth = as.Date("1939-10-27"),
   is.married = TRUE,
   children = 2,
   occupation = c("actor", "comedian", "writer")
)
z$name
z$name$first.name     # "John"
z$name$last           # "Cleese"
z[c("name", "birth")]
```

Faktorok (1)

- A faktorok olyan speciális vektor jellegű objektumok, melyek velük azonos hosszú vektorok elemeinek egy osztályozását határozzák meg.
  - A fakort alkotó véges számú különböző érték reprezentálja az osztályokat vagy kategóriákat.

Faktorok (2)

- Faktorokat a `factor(`*x*`)` függvénnyel lehet létrehozni tetszőleges típusú vektorokból.
  - Az *x* argumentum egy olyan vektor, mely általában kevés számú különböző értéket tartalmaz, a különböző értékek reprezentálják az egyes kategóriákat.
  - További információk: `?factor`

Példa faktorok használatára

```
area <- c(9596961, 357386, 93030, 377973)
names(area) <- c("China", "Germany", "Hungary", "Japan")

region <- c("Asia", "Europe", "Europe", "Asia")
regionf <- factor(region)
nlevels(regionf)   # 2
levels(regionf)    # "Asia" "Europe"
tapply(area, regionf, max) # az area vektor maximuma régiónként

drivingside <- c("right", "right", "right", "left")
drivingsidef <- factor(drivingside)
nlevels(drivingsidef)   # 2
levels(drivingsidef)    # "left" "right"

table(regionf, drivingsidef)  #  kontingenciatáblázat
```

Adatkeretek

- Olyan speciális listák, melyek komponensei azonos hosszúak.
  - A komponensek vektorok, listák (más adatkeret objektumok is), faktorok és mátrixok lehetnek.
  - Mátrixok esetében hossz alatt a sorok számát kell érteni).
- Olyan mátrix-szerű struktúrákként tekinthetünk rájuk, melyekben az oszlopok különböző típusúak lehetnek, és amelyekben az oszlopokat a lista komponensei adják meg.
  - A sorok megfigyelések, az oszlopok pedig változók.

Adatkeretek létrehozása

- A `data.frame()` függvénnyel lehet létrehozni adatkeret objektumokat.
- Alkalmas objektumokat (például mátrixokat és listákat) az `as.data.frame()` függvénnyel lehet adatkeret objektumokká konvertálni.
- Számos olyan függvény van, melyekkel külső forrásból lehet adatkereteket létrehozni.

Adatkeretek megjelenítése és szerkesztése

- Adatkeretek megjelenítésére szolgál a `View()` függvény.
  - További információk: `?View`
- Adatkeretek szerkesztésére szolgál a `data.entry()` függvény.
  - Ezt hívja meg az `edit()` függvény is, ha egy adatkeretet kap argumentumként.
  - További információk: `?data.entry`

A data.frame() függvény (1)

- A függvénynek tetszőleges sok argumentumot meg lehet adni, ezekből a komponensekből képződnek az adatkeret oszlopai.
  - Ha egy argumentumot *név=érték* módon adunk meg, ahol *név* egy azonosító vagy karakterlánc, akkor a rendszer ezt használja a megfelelő oszlop(ok) elnevezéséhez.
  - Egyébként az argumentum alapján automatikusan kerül(nek) meghatározásra az oszlopnév (oszlopnevek).

A data.frame() függvény (2)

- Az oszlopokat szolgáltató argumentumok azonos hosszúak kell, hogy legyenek.
  - Kivételt képeznek az atomi vektorok, faktorok és az `I(`*x*`)` függvénnyel levédett karakter vektorok (utóbbit lásd a következő oldalon).
  - Szükség esetén ezek azonos hosszra normalizálása az elemek ismétlésével.

A data.frame() függvény (3)

- A függvénynek átadott karakter vektorok automatikusan faktorokká lesznek konvertálva.
  - Ezt úgy lehet megakadályozni, hogy a vektort az `I(`*x*`)` függvénnyel levédve adjuk át a `data.frame()` függvénynek.
- Ha mátrixokat, listákat vagy adatkereteket adunk át argumentumokként, akkor ezek oszlopai illetve komponensei egyenként adódnak hozzá a létrehozandó adatkerethez.
  - Ezt meg lehet akadályozni úgy, hogy az `I(`*x*`)` függvényt használjuk az argumentum levédésére.

Példa adatkeret létrehozására

```
countries <- data.frame(
   name = c("Hungary", "Kenya", "Japan"),
   area = c(93030, 580367, 377873),
   oecd.member = c(TRUE, FALSE, TRUE),
   population = c(9797561, 49364325, 126440000),
   capital = I(c("Budapest", "Nairobi", "Tokyo")),
   row.names = "name"
)
```

Adatkeretek indexelése (1)

- Az indexelés hasonlóan történik a mátrixok és listák indexeléséhez.
  - A `[]`, `[[]]` és `$` operátorokat lehet használni.
- Indexelés nem csupán elemek értékének kinyerésére használható, hanem értékadásban is az adatkeret elemeinek helyettesítésére.
  - A viselkedés esetenként eltérő lehet elemek kinyerése és értékadás esetében!

Adatkeretek indexelése (2)

- A $ operátorral valamint a [ ] és [[ ]] operátoroknál egyetlen indexet megadva listaként lehet indexelni az adatkereteket.
  - Így oszlopokat lehet kiválasztani.
- A [ ] és [[ ]] operátoroknál két indexet (sor, oszlop) megadva úgy indexelhetjük az adatkereteket, mint a mátrixokat.
  - A [[ ]] operátorral csak egyetlen elemet lehet kiválasztani, a [ ] operátorral egyidejűleg többet is.

Példa adatkeret indexelésére (1)

```
countries[[1]]          #  az 1. oszlop, mint vektor
countries[["area"]] #  ua.
countries$area          #  ua.

countries[1]            #  az 1. oszlopból álló adatkeret
countries["area"]   #  ua.

countries[2:3]          #  a 2. és 3. oszlopból álló adatkeret
countries[c("oecd.member", "population")] #  ua.
```

Példa adatkeret indexelésére (2)

```
countries[[1, 2]]       #  az 1. oszlop 2. oszlopértéke
countries[["Hungary", "oecd.member"]]   #  ua.
countries[1, 2]                         #  ua.
countries["Hungary", "oecd.member"]     #  ua.

countries[1,]           #  az 1. sorból álló adatkeret
countries["Hungary",] #  ua.

countries[,2]                   #  a 2. oszlopból álló adatkeret
countries[, "oecd.member"] #  ua.

countries[1:2, 3:4] #  az 1-2. sor és a 3-4. oszlop által
                    #  meghatározott adatkeret
countries[1:2, 4:3] #  ua., de az oszlopok felcserélésével
```