Az R használata
(tárgyalt R verzió: 2.12.1)

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

Futtatás

- Tipikus futtatás Linux környezetben:

```
$ mkdir work
$ cd work
$ R
```

  - Minden egyes feladathoz célszerű egy külön alkönyvtárat létrehozni és abban elindítani a programot
    - Kilépéskor automatikusan elmenthető ide a felhasználó munkaterülete
    - Így nem vész el az elvégzett munka, később bármikor ott folytatható, ahol kilépésnél abbahagytuk

Inicializálás az R indításakor (1)

1. Felhasználói profil betöltése:
   - Ha be van állítva az `R_PROFILE_USER` környezeti változó, akkor ennek értéke adja meg a felhasználói profilt tartalmazó állományt
     - Relatív elérési útvonal az aktuális könyvtárhoz relatív
   - Egyébként a felhasználói profilt tartalmazó állományt az aktuális könyvtárban keresi a program `.Rprofile` néven, ha ott nincs ilyen, akkor a felhasználó HOME könyvtárában
   - A felhasználói profil a `source` függvénnyel kerül beolvasásra és végrehajtásra
     - R kódot tartalmaz, tipikusan beállításokat

Inicializálás az R indításakor (2)

2. Ha az aktuális könyvtár tartalmaz `.RData` nevű állományt, akkor ennek beolvasása a `load` függvénnyel
   - Az állomány tipikusan egy korábbi R munkamenet végén, a kilépésnél automatikusan kerül létrehozásra és a felhasználó által a munkamenet során létrehozott objektumokat tárolja

Inicializálás az R indításakor (3)

3. Ha van `.First` nevű függvény definiálva, akkor ennek meghívása
   - A függvény definícióját tartalmazhatja a felhasználói profil és az `.RData` állomány is
   - A függvényt argumentum nélküli függvényként így kell definiálni:

     `.First <- function()` *kifejezés*

Inicializálás az R indításakor (4)

4. Az előzményeket, azaz egy korábbi munkamenet során végrehajtott parancsokat tartalmazó állomány betöltése, ha van ilyen
   - Ha be van állítva az `R_HISTFILE` környezeti változó, akkor ennek értéke adja meg az állományt (az elérési útvonal az aktuális könyvtárhoz relatív)
   - Egyébként az állományt az aktuális könyvtárban keresi a program `.Rhistory` néven

Inicializálás az R indításakor (5)

- Példa felhasználói profilra (Linux):

```
options(browser="/usr/bin/firefox")
options(editor="/usr/bin/gedit")
options(digits=5, help.try.all.packages=TRUE,
  help_type="html")
.First <- function() {
  cat("\nWelcome, ", Sys.getenv("LOGNAME"),
    "!\n", sep="")
}
```

- Az inicializálás menetének részletes leírása megjeleníthető a `?Startup` paranccsal

Kilépés (1)

- Kilépéshez használjuk a `quit()` vagy `q()` függvényhívásokat
- A platformtól és a grafikus felhasználói interfésztől is függhet, hogy pontosan mi történik egy munkamenet befejeződésekor
- Használjuk a `?quit` parancsot részletes leírás megjelenítéséhez

Kilépés (2)

- A program rá fog kérdezni, hogy elmentse-e a felhasználó munkaterületét, amely a munkamenet során a felhasználó által létrehozott objektumokat tárolja
  - Igen válasz esetén a `save.image` függvénnyel az aktuális könyvtárba `.RData` néven kerül elmentésre a munkaterület
  - Az `.RData` állományt tartalmazó könyvtárból elindítva újra a programot automatikusan beolvasásra kerül az inicializálás során az állomány, így ott folytathatjuk a munkát, ahol az előző kilépésnél abbahagytuk

Kilépés (3)

- Ha van `.Last` nevű függvény definiálva, akkor automatikusan meghívásra kerül a kilépés során
  - A függvényt argumentum nélküli függvényként így kell definiálni:

    `.Last <- function()` *kifejezés*
- A `savehistory` függvénnyel automatikusan elmentésre kerülnek egy állományba a munkamenet során végrehajtott parancsok
  - Ha be van állítva az `R_HISTFILE` környezeti változó, akkor ennek értéke adja meg az állományt (alapértelmezés az aktuális könyvtár `.Rhistory` állománya)

Interaktív parancsvégrehajtás (1)

- Vegyük figyelembe, hogy az R megkülönbözteti a kis- és nagybetűket!
- Egy sorba akár több parancsot is gépelhetünk `';'` karakterekkel elválasztva
- Parancs gépelése közben a parancsot alkotó tokenek között elhelyezhető újsor karakter, így a parancs több sorba tördelhető
  - A végrehajtás a teljes parancs beolvasása után történik csak meg

Interaktív parancsvégrehajtás (2)

- A konzolon begépelt parancsok maximális hossza 4095 byte lehet (nem karakter!)
- A legtöbb platformon a lefelé és felfelé nyíl billentyűkkel lehet a korábban végrehajtott parancsokat visszahívni
- Rendelkezésre állhat automatikus parancskiegészítés funkció is, amelyet a TAB billentyűvel lehet elérni

Állományokban tárolt R kód végrehajtása

- Állományokban tárolt R kód beolvasásához és végrehajtásához használjuk a `source(`*file*`)` függvényhívást
  - Nem csupán állománynév adható meg, hanem akár URI is (a `http`, `ftp` és `file` URI sémák támogatottak)
  - Ha az R kód szintaktikai hibá(ka)t tartalmaz, akkor egyáltalán nem történik végrehajtás
  - A végrehajtás során nem jelenik meg a kiértékelt kifejezések értéke, ezért a kódban megfelelően gondoskodjunk a megjelenítésről
  - Ha megadjuk a `chdir=TRUE` argumentumot, akkor állomány elérési útvonal használata esetén a végrehajtás során ideiglenesen az állományt tartalmazó könyvtár lesz az aktuális könyvtár
  - Van néhány további hasznos argumentum, ezeket lásd a dokumentációban

Kimenet átirányítása

- A `sink(`*file*`)` függvényhívással lehet a kimenetet az adott állományba átirányítani
  - Ha nem adjuk meg a *file* argumentumot, akkor a legutóbbi átirányítás megszüntetése történik
  - Ha megadjuk az `append=TRUE` argumentum, akkor a kimenet az állomány végéhez lesz hozzáfűzve (az alapértelmezés az állomány felülírása)

Dokumentáció (1)

- Az Rd („R documentation”) az R saját dokumentációs formátuma
  - Egy egyszerű, a LaTeX-hez hasonló jelölőnyelv
  - Az ebben megírt dokumentációt több különböző formátumba lehet alakítani (LaTeX, HTML, sima szöveg)
  - Az R disztribúció több mint 1200 ilyen dokumentációs állományt tartalmaz
  - Az Rd állományok feldolgozásával kapcsolatos dokumentáció megjelenítéséhez használjuk a `?Rdconv` parancsot

Dokumentáció (2)

- A csomagok dokumentációját ugyan tipikusan Rd formátumban írják, de tetszőleges egyéb dokumentációs formátum is használható
  - Ajánlott a platformfüggetlen PDF formátum használata
- Egy speciális dokumentációs formátum az SWeave, az ebben írt dokumentációt vignettáknak nevezik
  - A vignettákat PDF formátumban lehet megjeleníteni

Segítség!

- A dokumentáció eléréséhez, kereséséhez az alábbi függvényeket használhatjuk:
  - `help`
  - `help.start`
  - `help.search`
  - `vignette`
  - `RSiteSearch`

A `help` függvény (1)

- Az Rd formátumban készült dokumentációt `help(`*téma*`)` függvényhívásokkal érhetjük el
  - A *téma* argumentum lehet azonosító és karakterlánc is
    - Operátorokat, függvényekhez és vezérlési szerkezetekhez kapcsolódó kulcsszavakat (`break`, `else`, `for`, `function`, `if`, `in`, `next`, `repeat`, `while`) karakterláncként kell megadni
  - Vannak további argumentumok is, ezeket lásd a dokumentációban

A `help` függvény (2)

- A `help_type` argumentumban lehet megadni a dokumentáció megjelenítési formátumát
  - A lehetséges értékek `"text"`, `"html"`, `"postscript"`, `"ps"` és `"pdf"`
    - `"html"` esetén a megjelenítéshez használt böngésző megadható a `browser` opcióval
    - Az utóbbi három esetén egy PostScript vagy PDF állomány jön létre az állományrendszerben, amelynek előállításához szükséges, hogy TeX illetve pdfTeX megfelelően telepítve legyen

A `help` függvény (3)

- A témához tartozó súgó oldalak keresése alapértelmezésben a keresési útvonalban lévő (betöltött) csomagokban történik
  - Megadható a `package` argumentum értékeként egy csomagnév, mint azonosító vagy egy csomagneveket tartalmazó karakter vektor
  - Így az adott csomag(ok)ra korlátozható a keresés

A `help` függvény (4)

- Ha megadjuk a `try.all.packages=TRUE` argumentumot, de nem adjuk meg a keresést az adott csomagokra korlátozó `package` argumentumot, akkor az összes telepített és rendelkezésre álló csomagban történik keresés, ha a keresési útvonalban lévő csomagoknál sikertelen volt
  - Az így megtalált súgó oldalak nem kerülnek megjelenítésre, csupán az jelenik meg, hogy mely csomagokban talált a program a témához súgó oldalakat

A `help` függvény (5)

- Használható röviden ?*téma* a `help(`*téma*`)` függvényhívás helyett
- Hasonlóan használható ?*csomag*`::`*téma* a `help(`*téma*`, package=`*csomag*`)` függvényhívás helyett
  - Ilyenkor *csomag* csak karakterlánc vagy azonosító lehet

Példa a `help` függvény használatára

- `help("if")`
- `?"if"`
- `help()`
- `help(on.exit)`
- `?on.exit`
- `help(permutations, package="e1071")`
- `?e1071::permutations`
- `help(roots, try.all.packages=TRUE)`

A `help.start` függvény

- A `help.start()` függvényhívással érhetjük el a dokumentációt HTML-ben
- Hatására egy olyan oldal jelenik meg egy böngészőablakban, ahonnan elérhető többek között valamennyi R kézikönyv és egy keresőmotor a dokumentáció kereséséhez
  - A keresőmotor használatához a böngészőben Java és JavaScript támogatás szükséges
- A használt böngésző megadható a `browser` opcióval

A `help.search` függvény (1)

- A `help.search(`*minta*`)` függvényhívással kereshető az Rd formátumban készült dokumentáció
  - A *minta* argumentum kötelezően egy karakterlánc
  - A további argumentumokat lásd a dokumentációban
- A keresés alapértelmezésben az összes telepített és rendelkezésre álló csomagban történik
- A keresés az dokumentációs oldalak különböző metaadat mezőiben történhet (például állománynév, cím, kulcsszavak)
- A kis- és nagybetűk alapértelmezésben nincsenek megkülönböztetve a keresésnél
- A keresés során fuzzy mintaillesztést használhat a program
- A keresés eredményeként a dokumentációs oldalak neve és címe jelenik meg

A `help.search` függvény (2)

- Használható röviden `??`*minta* a `help.search(`*minta*`)` függvényhívás helyett
- Hasonlóan használható `??`*csomag*`::`*minta* a `help.search(`*minta*`, package=`*csomag*`)` függvényhívás helyett
  - Ilyenkor *csomag* csak karakterlánc vagy azonosító lehet

Példa a `help.search` függvény használatára

- `help.search("normal distribution")`
- `help.search(keyword="optimize")`
- `help.search(keyword="IO")`
- `help.search("^read", fields="name", ignore.case=FALSE))`

A `vignette` függvény

- A `vignette(`*`név`*`)` függvényhívással lehet megjeleníteni az adott nevű vignettát
  - A *név* argumentum egy karakterlánc lehet, ez a vignetta lesz megjelenítve
  - A *package* argumentum értéke egy olyan karakter vektor lehet, amelyben felsoroljuk azoknak a csomagoknak a nevét, amelyekben a vignetták keresése történik
    - Alapértelmezésben az összes telepített csomagban történik a keresés
- A függvényt meghívhatjuk argumentum nélkül is, ekkor a rendelkezésre álló vignettákat listázza

Példa a `vignette` függvény használatára

- `vignette(package="grid")`
- `vignette("kernlab")`
- `vignette("TSP")`
- `v <- vignette("viewports")`
  `edit(v)`

Az `RsiteSearch` függvény

- A `RSiteSearch(`*`x`*`)` függvényhívással a http://search.r-project.org/ címen működő keresőmotort érhetjük el
  - Argumentumként egy karakterláncban adhatjuk meg a keresendő kifejezés(eke)t
    - Ha egy több szóból álló kifejezésre akarunk pontosan keresni, akkor azt `'{'` és `'}'` karakterek között adjuk meg a karakterláncban
    - A keresőkifejezések szintaxisának leírása megtalálható a következő címen: http://www.namazu.org/doc/manual.html#query
  - Vannak további argumentumok, ezeket lásd a dokumentációban
  - A keresés a levelezési lista archívumokban, súgó oldalakban és vignettákban történhet
  - A találatok webböngészőben jelennek meg

Példa az `RSiteSearch` függvény használatára

- `RSiteSearch("eigenvalue or eigenvector")`
- `RSiteSearch("{hidden Markov model}")`
- `RSiteSearch("RODBC", restrict="Rhelp08")`
- `RSiteSearch("bug nlm", restrict="R-devel")`

Az `example` függvény

- Az `example(`*téma*`)` függvényhívással lehet végrehajtani a `help(`*téma*`)` módon elérhető oldalak *Examples:* cím alatt feltüntetett R kódját
  - Ezek a függvények használatára adnak példát
- Az ún. *dontrun* részek nem kerülnek végrehajtásra, amelyeket a dokumentációban `"## Not run:"` szöveg jelez
- A kód végrehajtása alapértelmezésben a felhasználó munkaterületén történik, így a végrehajtás során létrehozott objektumok a végrehajtás után is rendelkezésre állnak
  - Ez felülbírálható a `local=TRUE` argumentum megadásával

Csomagok telepítésének helye

- `A .Library` globális változó értéke egy karakterlánc, amely annak a könyvtárnak az elérési útvonalát tartalmazza, amelyben alapértelmezésben történik a csomagok keresése
  - Tipikusan rendszergazdai jogosultság birtokában lehet csupán ebbe a könyvtárba csomagokat telepíteni
- Csomagok telepítése történhet az állományrendszerben ettől eltérő tetszőleges olyan könyvtár(ak)ba, amely(ek)re a felhasználó az ehhez szükséges jogosultságokkal rendelkezik
  - Megfelelő beállítás szükséges ahhoz, hogy ezekben a könyvtárakban is történjen a csomagok keresése (lásd a `.libPaths()` függvényél leírtakat)

A telepített csomagok tartalmazó könyvtárak megadása

- A `.libPaths(`*x*`)` függvényhívás használható a telepített csomagokat tartalmazó könyvtárak megadására
  - Az argumentum egy könyvtár elérési útvonalakat tartalmazó karakter vektor lehet, amely azt adja meg, hogy a továbbiakban az alapértelmezett hely mellett mely könyvtárakban történjen még a csomagok keresése
  - A függvény meghívható argumentum nélkül is, ekkor egy karakter vektorban adja vissza azoknak a könyvtáraknak az elérési útvonalait, amelyekben történik a csomagok keresése
- A telepített csomagokat tartalmazó könyvtárak elérési útvonalait meg lehet adni az `R_LIBS_USER` környezeti változóval is
  - Több elérési útvonal esetén használjuk a `':'` elválasztó karaktert a környezeti változó értékében

Csomagok telepítése (1)

- Az `install.packages(`*x*`)` függvénnyhívással lehet csomagokat telepíteni
  - Az argumentum a telepítendő csomagok neveit tartalmazó karakter vektor (vannak további argumentumok is, ezeket lásd a dokumentációban)
  - Grafikus felhasználói felület rendelkezésre állása esetén a függvényt argumentum nélkül is meghívhatjuk, ekkor egy listából interaktív módon lehet kiválasztani a telepítendő csomagokat
  - A telepítés alapértelmezésben egy olyan könyvtárba történik, amelynek írásához tipikusan rendszergazdai jogosultság szükséges
    - Ez felülbírálható a `lib` argumentum megadásával, amelynek értéke egy könyvtár elérési útvonalat tartalmazó karakterlánc lehet

Csomagok telepítése (2)

- A függvény automatikusan letölti a csomagok telepítéséhez szükséges további csomagokat

További függvények csomagok kezeléséhez

- Csomagok telepítésével kapcsolatosak az alábbi függvények (a részletes leírást lásd a dokumentációban):
  - Az `installed.packages` függvény a lokálisan telepített csomagokat szolgáltatja
  - Az `available.packages` függvény az elérhető telepíthető csomagokat szolgáltatja
  - Az `old.packages` függvény azokat a csomagokat szolgáltatja, amelyeknek újabb verziója érhető el a lokálisan telepítettnél
  - Utóbbihoz hasonlóan működik a `new.packages` függvény, amely azonban nem csupán az elavult, hanem az elérhető, de lokálisan nem telepített csomagokat is kijelzi
  - Az `update.packages` függvény a lokálisan telepített csomagokat frissíti az elérhető legfrissebb verzióra

Csomagok betöltése

- Csomagok betöltésére szolgálnak a `library(x)` és `require(`*x*`)` függvényhívások
  - Az *x* argumentum a csomag nevét megadó azonosító vagy karakterlánc
  - Az elsőt tipikusan interaktív parancsvégrehajtásnál, a másodikat pedig függvények törzsében használjuk

Csomagok betöltése a `library` függvénnyel

- A függvény meghívható argumentum nélkül, ilyenkor a telepített csomagokat listázza
- Meghívható `library(help=`*csomagnév*`)` módon, ahol *csomagnév* a csomag nevét megadó azonosító vagy karakterlánc
  - Ilyenkor a csomagról szolgáltat információkat (metaadatok, tartalom)
- A `lib.loc` argumentum értéke egy könyvtár elérési útvonalakat tartalmazó karakter vektor lehet
  - Az adott könyvtárakban történik a csomag keresése
  - Lásd a `.libPaths()` függvénynél leírtakat

Csomagok betöltése a `require` függvénnyel

- A függvény logikai visszatérési értékű
  - A visszaadott érték `TRUE` akkor, ha az argumentumként adott csomag lokálisan elérhető, egyébként pedig `FALSE`
- Nem telepített csomag megadása a `library` függvénytől eltérően nem okoz hibát, csak figyelmeztetést
- A `lib.loc` argumentum ugyanúgy használható, mint a library függvénynél

Csomagok leválasztása

- Betöltött csomagok leválasztásához használjuk a `detach(`*x*`)` függvényhívást
  - Az argumentumot `package:`*csomagnév* vagy `"package:`*csomagnév*`"` módon adhatjuk meg, ahol *csomagnév* a leválasztandó csomag neve

Példa csomagok használatára

```
p <- available.packages()
invisible(edit(p))
install.packages("fortunes", lib="/tmp")
library(help="fortunes", lib.loc="/tmp")
library(fortunes, lib.loc="/tmp")
fortune()
…
detach("package:fortunes")
.libPaths()
.libPaths("/tmp")
.libPaths()
library(fortunes)
fortune()
```

Opciók (1)

- Az R globális beállításai
- A használható opciókat megtaláljuk az `?options` paranccsal megjeleníthető súgó oldalon
- Opciók kezelésére szolgálnak az `options` és `getOptions` függvények

Opciók (2)

- Az `options(...)` függvényhívás segítségével lehet lekérdezni és beállítani az opciókat
  - A függvény meghívható argumentum nélkül, ekkor az összes beállított opciót adja vissza egy listában
    - A listában az opciók név szerint ábécé sorrendbe rendezettek
  - Opciók beállításához tetszőleges számú *név* = *érték* alakú kifejezést adhatunk meg argumentumként, vagy pedig egy ilyen komponensekből álló listát
    - *név* az opció nevét megadó azonosító vagy karakterlánc
      - Tetszőleges név megadható, hatása csak az R által kezelteknek lesz
    - *érték* az opciót értéket szolgáltató kifejezés
      - Az R által kezelt opciók értékére ellenőrzést végezhet a program (nem megengedett érték hibát okoz)

Opciók (3)

- A `getOption(`*x*`)` függvényhívás az argumentumként adott nevű opció értékét szolgáltatja
  - Az argumentum kötelezően egy karakterlánc
  - Be nem állított opció esetén `NULL` a visszaadott érték

Példa opciók használatára

```
options()
getOption("digits")
options(digits=getOption("digits") + 1)
oldOptions <- options()
options(help.try.all.packages=TRUE, verbose=TRUE)
options(editor="/usr/bin/emacs")
options(pdfviewer="/usr/bin/evince")
…
options(oldOptions) # a korábbi beállítások visszaállítása
```

Aktuális könyvtár

- Az aktuális könyvtár lekérdezéséhez használjuk a `getwd()` függvényhívást
  - A visszatérési érték az aktuális könyvtár elérési útvonalát tartalmazó karakterlánc, illetve `NULL` akkor, ha nem áll rendelkezésre az aktuális könyvtár
- Az aktuális könyvtárat a `setwd(`*könyvtár*`)` függvényhívással lehet beállítani
  - A argumentum értékeként az elérési útvonalat tartalmazó karakterláncot kell megadni