SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

R - PROGRAMMING LANGUAGE
SEMINÁRNA PRÁCA

2022Peter Andrejko, Marek Dráb, Dávid Gavenda, Marek Klimo

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

R - PROGRAMMING LANGUAGE

SEMINÁRNA PRÁCA

Študijný program:	Aplikovaná informatika
Predmet:	I–ASOS – Architektúra softvérových systémov
Prednášajúci:	RNDr. Igor Kossaczký, CSc.
Cvičiaci:	Ing. Stanislav Marochok

Bratislava 2022 **Peter Andrejko, Marek Dráb, Dávid Gavenda, Marek Klimo**

Obsah

Zoznam obrázkov a tabuliek

1 Čo je R project?

R je jazyk a prostredie pre štatistické výpočty a grafiku. R je vytvorený ako GNU project vychádzajúci z jazyka S vytvoreného v Bell Laboratories. Veľa vecí je teda nezmenených od S.

1.1 Prostredie R

R umožňuje efektívne spracovanie dát s priestormi na ukladanie, matematické operácie na prácu hlavne s maticami. Obsahuje rozsiahlu, ucelenú a integrovanú zbierku sprostredkovateľských nástrojov na analýzu údajov, grafické prostriedky na analýzu a zobrazenie údajov na obrazovke alebo na papieri a dobre vyvinutý, jednoduchý a efektívny programovací jazyk, ktorý obsahuje podmienky, cykly, rekurzívne funkcie definované používateľom a vstupné a výstupné zariadenia.

R, tak ako S, je dizajnovaný okolo „pravdivého počítačového jazyka“ a umožňuje pridávať používateľovi dodatočné funkcionality definovaním nových funkcií. Dokáže pracovať s C, C++ a Fortran počas behu procesu. Náročnejší užívatelia môžu napísať C kód na manipulovanie priamo s objektami R.

R má taktiež svoj dokumentový formát podobný LaTeX, ktorý sa používa na poskytovanie komplexnej dokumentácie, či už online alebo ako tlačivo.

1.2 Výhody jazyka R

- Open source
- Vzorová podpora pre prácu s údajmi
- Zoznam packages: 10000 packages v repozitári
- Kvalitné vykresľovanie grafov
- Kompatibilnosť
- Nezávislosť softvéru- R je nezávislý jazyk, umožňuje cross-platform programovanie, čiže môže bežať na Windows, Linuxe aj Macu
- Machine learning- klasifikácia, regresia
- Štatistika
- Frekvencia aktualizácií

1.3 Nevýhody jazyka R

- Slabý základ
- Data handling
- Bezpečnosť
- Komplikovaný jazyk
- Rýchlosť jazyka
- Algoritmy v rôznych packages

1.4 Popularita

Vďaka svojej všestrannosti, veľa odvetví priemyslu používa R, ako napr:

- Finančné služby
- Akademický výskum
- Vláda
- Sociálne média
- Dátový žurnalizmus
- Zdravotníctvo

V konkrétnych prípadoch sú to firmy ako napríklad:

- Airbnb
- Microsoft
- Uber
- Facebook
- Ford
- Google
- Twitter
- IBM

- American Express
- HP

2 História

S je jazyk, ktorý bol vyvinutý v Bell Telephone Laboratories vďaka Johna Chambers-a. S bolo vytvorené ako prostredie pre štatistickú analýzu v 1976 ako knižnica pre Fortran. Prvotné verzie nemali ani funkcie pre štatistické modely. V 1988 bol jazyk prerobený v C, pričom najnovšia verzia tohto jazyka je z roku 1998. Kvôli viacerým problémom s licenciami bol vývoj náročný. Preto Insightful predal implementáciu pod názvom S-PLUS. Myšlienka za S je, že základom bola analýza dát. Investori boli zameraní na ľahšiu analýzu dát. Potrebovali vytvoriť jazyk, ktorý by bol vhodný pre interaktívnu analýzu dát a taktiež aj pre písanie väčších programov.

Na jazyku R pracovali Robert Gentleman Ross Ihaka, pričom prvú kópiu binárky poslali do S-news v auguste 1993. Martin Mächler z ETH Zurich im odporučil vydať ich kód na R ako free software. V júni 1995 teda vydali R pod licenciou Free Software Foundation's GNU. V roku 1997 vytvorili tvorcovia „core group", ktorí pracovali na zdrojovom kóde. Medzi nich patrí Doug Bates, Peter Dalgaard, Robert Gentleman, Kurt Hornik, Ross Ihaka, Friedrich Leisch, Thomas Lumley, Martin Mächler, Paul Murrell, Heiner Schwarte, a Luke Tierney. Práca na jazyku R bola čisto dobrovoľná, organizácia je voľnejšia. Plná verzia 1.0.0 bola vydaná v roku 2000. V roku 2003 bolo vytvorené R Foundation z dôvodu financovania projektu.[1]

3 Syntax jazyka R

Hlavný zmysel pre R na začiatku bola náhradu za komerčnú verziu jazyka S, nazývanú S-PLUS. Syntax je pri oboch prípadoch podobná, avšak sémantika pre R je odlišná, pričom sa dá R považovať bližšie k jazyku Scheme. V dnešnej dobe je R kompatibilné s väčšinou populárnych operačných systémov. Dokáže pracovať s Windows, Linux, Apple, Android, ale aj s hernými konzolami. R vďaka svojej popularite a dostupnosti má časté aktualizácie, väčšinou v októbri, kedy vychádza hlavná aktualizácia každý rok, pričom inokedy v roku vychádzajú len patch-e. Ďalšia výhoda pre R sú balíčky pre štatistiku a sofistikované grafické rozhranie. [2]

R je rozdelené na 2 časti: základný balík z CRAN a potom všetko ostatné. Funkcionalita R je rozdelená do balíčkov, ako napr. utils, stats, datasets, graphics, grDevices, grid, methods, tools, parllel, compiler, splines, tcltk, stats4.

Všetky tieto balíčky sú dostupné pri prvotnej inštalácii jazyka z CRAN, ďalšie balíčky sa avšak už sťahujú dodatočne samotným užívateľom.

Avšak ani R nie je dokonalé, pričom tiež má svoje limity. Keďže je tento jazyk založený na 50-ročnej technológii, je tam nízka podpora pre dynamickú a 3D grafiku. Ďalším problémom je ukladanie objektov do fyzickej pamäte. Limitácie R sú závislé na požiadavkách používateľoch. Ak nikto nechce implementovať danú metódu, avšak je potrebná vo vašej práci, tak si ju musíte implementovať sami. Požiadavky komunity sa avšak zhodujú vo viacerých bodoch, pričom sa tieto požiadavky rozšírili aj do oblastí ako fyzika, astronómia a iné. [3]

4 Základná práca s R

4.1 Inštalácia

R je dostupné na všetkých dostupných platformách vrátane Windowsu, Mac OS X a systémov Linux. Tutoriály na inštaláciu sú dostupné:

- inštalácia R na Windows
- inštalácia R na Mac

Dostupné je aj vývojárske prostredie pre R postavené RStudiom. Dostupné je na webe RStudio. [3] [4]

4.2 Vstupy

Na priradenie hodnoty slúži symbol <- .

```
> x <- 1
> print(x)
[1] 1
> x
[1] 1
> msg <- "hello"
```

Obr. 1: Vstupy

4.3 Komentáre

Znak # označuje komentáre. Všetko napravo od # (vrátane samotnej #) je odignorované. Je to jediný znak použitý na komentovanie v R. Oproti iným jazykom, R nepodporuje viac riadkové komentáre alebo bloky.

```
x <-  ## Incomplete expression
```

Obr. 2: Komentáre

4.4 Vyhodnocovanie

Po úplnom zadaní príkazu v konzole sa príkaz vyhodnotí a vráti sa výsledok. Výsledok môže byť auto-printed - automaticky vypísaný.

```
> x <- 5  ## nothing printed
> x       ## auto-printing occurs
[1] 5
> print(x)  ## explicit printing
[1] 5
```

Obr. 3: Vyhodnocovanie

[1] vo výpise indikuje, že x je vektor a 5 je jeho prvý element.

V interaktívnom priestore nemusíme explicítne vypisovať objekt cez print funkciu; je oveľa jednoducšie robiť automatický výpis napísaním názvu objektu a následna stlačením enteru. Keď ale píšeme skripty, funkcie alebo dlhšie programy, tak niekedy je potrebné explicitne tlačiť objekty, keďže automatický výpis nefunguje s týmito možnosťami.

Keď vektor je vypísaný, je možné si všimnúť index vektore vypísaný v hranatých zátvorkách.

```
> x <- 11:30
> x
 [1] 11 12 13 14 15 16 17 18 19 20 21 22
[13] 23 24 25 26 27 28 29 30
```

Obr. 4: Vyhodnocovanie 2

Čísla v hranatých zátvorkách nie sú ale súčasťou vektoru, sú iba súčasťou vypísaného výstupu.

V R je rozdiel medzi skutočným R objektom a R objektom vypísaným v konzole. Väčšinou vypísaný výstup môže mať dodatky pre ľahšiu vizualizáciu pre používateľa.

4.5 Objekty

R má 5 základných tried objektov:

- character
- numeric
- integer
- complex

- logical (True/False)

Najzákladnejší typ v R je vector. Prázdne vektory môžu byť vytvorené pomocou vector() funkcie. Pri vektoroch je jedno pravidlo a to, že vektor sme obsahovať len objekty rovnakej triedy.

Výnimkou v tomto pravidle je list, ktorý je reprezentovaný ako vektor ale môže obsahovať objekty s rôznymi triedami.

Existuje aj trieda pre raw objekty. Tá ale nie je priamo používaná pri dátovej analýze a nebudú ďalej spomenuté.

4.6 Čísla

S číslami je všeobecne zaobchádzané ako s objektami typu numeric (reálne čísla zaokrúhlené na 2 desatinné miesta). To znamená, že aj keď sú použité čísla ako "1" alebo "2" v R, ktoré považujeme za integery, pravdepodobne sú reprezentované ako numeric objekty (niečo ako "1.00" alebo "2.00").

Ak explicitne chceme integer, potrebujeme špecifikovať L suffix. Zadanie 1 v R vráti objekt typu numeric, zadanie 1L vráti integer objekt.

V R existuje aj špeciálne číslo Inf, ktoré reprezentuje nekonečno. To nám umožňuje reprezentovať entity ako 1/0. Takýmto spôsobom môže byť Inf použité v klasický operáciách ako napr. 1 / Inf = 0.

4.7 Atribúty

R objekty môžu mať atribúty, ktoré sú ako metadata pre objekty. Tieto metadata môžu byť užitočné pri opisovaní objektu. Napr. názvy stĺpcov v data frame pomôžu určiť aké dáta sa nachádzajú v danom stĺpci. Niektoré prípady:

- mená, názvy dimenzií
- dimenzie (napr. matice, polia)
- triedy (napr. integer, numeric)
- dĺžka
- iné používateľom definované atribúty/metadata

K atribútom objektu pristupujeme pomocou funkcie attributes(). Nie všetky R objekty obsahujú atribúty. V takom prípade attributes() vráti NULL.

4.8 Vektory

Na vytvorenie vektorov objektov ich spájaním je možné použiť funkciu c().

```
> x <- c(0.5, 0.6)       ## numeric
> x <- c(TRUE, FALSE)    ## logical
> x <- c(T, F)           ## logical
> x <- c("a", "b", "c")  ## character
> x <- 9:29              ## integer
> x <- c(1+0i, 2+4i)     ## complex
```

Obr. 5: Vektory

Treba si všimnúť, že T a F sú skrátený zápis pre TRUE a FALSE. Je odporúčané používať explicitne TRUE a FALSE hodnoty. T a F hodnoty sú primárne na použitie, keď sa vám nechce.

Vektory je tiež možné inicializovať pomocou funkcie vector().

```
> x <- vector("numeric", length = 10)
> x
 [1] 0 0 0 0 0 0 0 0 0 0
```

Obr. 6: Vektory 2

4.9 Rozdielne objekty

V nasledujúcom príklade sú vytvárané vektory z 2 rôznych tried. Jediné pravidlo o vektoroch hovorí, že práve toto nie je dovolené. Ak sú vo vektore rôzne typy, nastáva coercion, teda každý element vektora je rovnakej triedy.

```
> y <- c(1.7, "a")   ## character
> y <- c(TRUE, 2)    ## numeric
> y <- c("a", TRUE)  ## character
```

Obr. 7: Rozdielne objekty

V uvedenom príklade môžeme vidieť efekt implicit coercion. R sa snaží nájsť spôsob ako reprezentovať všetky objekty vo vektore vhodným spôsobom. Napr. kombinovanie

numeric objektu a character objektu vytvorí character vector, pretože čísla vieme môžu byť jednoducho reprezentované ako stringy.

4.10 Explicítne volanie a pretypovanie

Objekty môžu byť explicitne pretypované pomocou funkcií as.*.

```
> x <- 0:6
> class(x)
[1] "integer"
> as.numeric(x)
[1] 0 1 2 3 4 5 6
> as.logical(x)
[1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
> as.character(x)
[1] "0" "1" "2" "3" "4" "5" "6"
```

Obr. 8: Explicítne volanie a pretypovanie

Niekedy R nevie rozoznať ako zmeniť objekt a toto môže vytvoriť chybu typu NA.

```
> x <- c("a", "b", "c")
> as.numeric(x)
Warning: NAs introduced by coercion
[1] NA NA NA
> as.logical(x)
[1] NA NA NA
> as.complex(x)
Warning: NAs introduced by coercion
[1] NA NA NA
```

Obr. 9: Explicítne volanie a pretypovanie 2

Väčšinou sa pri chybnom pretypovaní zobrazí chyba v R.

4.11 Matice

Matice sú vektory s dimenzionálnym atribútom. Dimenzionálny atribút je integer vektorov s dĺžkou 2 (počet riadkov, počet stĺpcov)

Matice sú konštruované po stĺpcoch, čiže začína sa vľavo hore.

```
> m <- matrix(nrow = 2, ncol = 3)
> m
     [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA
> dim(m)
[1] 2 3
> attributes(m)
$dim
[1] 2 3
```

Obr. 10: Matice 1

```
> m <- matrix(1:6, nrow = 2, ncol = 3)
> m
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Obr. 11: Matice 2

Matice taktiež vieme skonštruovať priamo z vektorov, ktorým pridáme dimenzionálny atribút.

```
> m <- 1:10
> m
 [1]  1  2  3  4  5  6  7  8  9 10
> dim(m) <- c(2, 5)
> m
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
```

Obr. 12: Matice 3

A ako posledné matice dokážu byť vytvorené spájaním stĺpcov alebo spájaním riadkov cez cbind() a rbind().

```
> x <- 1:3
> y <- 10:12
> cbind(x, y)
     x  y
[1,] 1 10
[2,] 2 11
[3,] 3 12
> rbind(x, y)
  [,1] [,2] [,3]
x    1    2    3
y   10   11   12
```

Obr. 13: Matice 4

4.12 Listy

Listy sú objekty, ktoré obsahujú elementy rôzneho typu vo vnútri. K elementom sa môže pristupovať pomocou mien v liste:

```
# Create a list containing a vector, a matrix and a list.
list_data <- list(c("Jan","Feb","Mar"), matrix(c(3,9,5,1,-2,8), nrow = 2),
   list("green",12.3))

# Give names to the elements in the list.
names(list_data) <- c("1st Quarter", "A_Matrix", "A Inner list")

# Show the list.
print(list_data)
```

Obr. 14: Listy 1

Výstup tohto kódu vyzerá nasledovne:

```
$`1st_Quarter`
[1] "Jan" "Feb" "Mar"

$A_Matrix
     [,1] [,2] [,3]
[1,]    3    5   -2
[2,]    9    1    8

$A_Inner_list
$A_Inner_list[[1]]
[1] "green"

$A_Inner_list[[2]]
[1] 12.3
```

Obr. 15: Listy 2

Listy sa taktiež dokážu zjednotiť:

```
# Create two lists.
list1 <- list(1,2,3)
list2 <- list("Sun","Mon","Tue")

# Merge the two lists.
merged.list <- c(list1,list2)

# Print the merged list.
print(merged.list)
```

Obr. 16: Listy 3

Ako posledná vec je, že listy sa dokážu meniť na vektory, pričom sa použije funkcia unlist():

```r
# Create lists.
list1 <- list(1:5)
print(list1)

list2 <-list(10:14)
print(list2)

# Convert the lists to vectors.
v1 <- unlist(list1)
v2 <- unlist(list2)

print(v1)
print(v2)

# Now add the vectors
result <- v1+v2
print(result)
```

Obr. 17: Listy 4

4.13 Faktory

Faktor je dátový objekt, ktorý sa používa na kategorizáciu a ukladanie dát do levelov. Môže ukladať aj reťazce aj čísla. Sú vhodné pri stĺpcoch, ktoré majú malé množstvo unikátnych hodnôt. Je možné rozdeliť takto napr. Muž, Žena, alebo pravda, nepravda.

Faktory sú vytvorené pomocou funkcie faktor(), pričom do vstupu ako parameter berú vektor.

```r
# Create a vector as input.
data <- c("East","West","East","North","North","East","West","West","West","East","North")

print(data)
print(is.factor(data))

# Apply the factor function.
factor_data <- factor(data)

print(factor_data)
print(is.factor(factor_data))
```

Obr. 18: Faktory

```
[1] "East"  "West"  "East"  "North" "North" "East"  "West"  "West"  "West"  "East" "North"
[1] FALSE
[1] East  West  East  North North East  West  West  West  East  North
Levels: East North West
[1] TRUE
```

Obr. 19: Faktory 2

Keď sa avšak pracuje s data frame, tak R berie stĺpce ako kategorické dáta a automaticky zfaktoruje.

```
# Create the vectors for data frame.
height <- c(132,151,162,139,166,147,122)
weight <- c(48,49,66,53,67,52,40)
gender <- c("male","male","female","female","male","female","male")

# Create the data frame.
input_data <- data.frame(height,weight,gender)
print(input_data)

# Test if the gender column is a factor.
print(is.factor(input_data$gender))

# Print the gender column so see the levels.
print(input_data$gender)
```

Obr. 20: Faktory 3

```
  height weight gender
1    132     48   male
2    151     49   male
3    162     66 female
4    139     53 female
5    166     67   male
6    147     52 female
7    122     40   male
[1] TRUE
[1] male   male   female female male   female male
Levels: female male
```

Obr. 21: Faktory 4

4.14 Chýbajúce hodnoty

Chýbajúce hodnoty sa zapisujú ako Na alebo NaN pre nezadefinované matematické operácie.

- is.na() je použité na testovanie objektov ak sú NA
- is.nan() sa používa na testovanie pre NaN
- NA hodnoty majú taktiež svoju triedu, takže existuje číselné NA, znakové NA, atď.
- NaN hodnoty sú NA, avšak nefunguje to naopak.

```
> ## Create a vector with NAs in it
> x <- c(1, 2, NA, 10, 3)
> ## Return a logical vector indicating which elements are NA
> is.na(x)
[1] FALSE FALSE  TRUE FALSE FALSE
> ## Return a logical vector indicating which elements are NaN
> is.nan(x)
[1] FALSE FALSE FALSE FALSE FALSE
```

Obr. 22: Chýbajúce hodnoty

```
> ## Now create a vector with both NA and NaN values
> x <- c(1, 2, NaN, NA, 4)
> is.na(x)
[1] FALSE FALSE  TRUE  TRUE FALSE
> is.nan(x)
[1] FALSE FALSE  TRUE FALSE FALSE
```

Obr. 23: Chýbajúce hodnoty 2

4.15 Data frame

Data frame sa používa na ukladanie tabuľkových dát v R. Sú doležitou súčasťou pre R a používajú sa na rôzne aplikácie štatistických modelov. Pre prácu s nimi existuje balík dplyr, ktorý vytvoril Hadley Wickham a má funkcie vytvorené pre efektívnu prácu s nimi.

Data frame reprezentuje špeciálny typ listu, kde každý prvok listu musí mať rovnakú dĺžku. Každý prvok listu sa môže považovať ako samostatný stĺpec a dĺžka každého prvku ako počet riadkov.

Narozdiel od matíc, data frame môže ukladať rôzne triedy objektov v každom stĺpci. Matice musia mať každý prvok v spoločnej triede (všetky číselné).

Názvy stĺpcov, názvy premenných môžu zobrazovať informácie v každom stĺpci data frame pomocou špeciálneho atribútu row.names.

Data frame sa často tvorí pomocou čítania datasetu pomocou read.table() alebo read.csv(). Data frame avšak dokáže byť explicitne vytvorené pomocou data.frame().

Data frame môže byť konvertovaný do matice pomocou funkcie data.matrix().

```
> x <- data.frame(foo = 1:4, bar = c(T, T, F, F))
> x
  foo   bar
1   1  TRUE
2   2  TRUE
3   3 FALSE
4   4 FALSE
> nrow(x)
[1] 4
> ncol(x)
[1] 2
```

Obr. 24: Dataframe

4.16 Názvy

R objekty môžu mať názvy, ktoré sú užitočné pre písanie čitateľného kódu a samopopisných objektov.

```
> x <- 1:3
> names(x)
NULL
> names(x) <- c("New York", "Seattle", "Los Angeles")
> x
   New York     Seattle Los Angeles
          1           2           3
> names(x)
[1] "New York"    "Seattle"     "Los Angeles"
```

Obr. 25: Názvy

Aj listy dokážu mať názvy, ktoré sú veľmi užitočné.

```
> x <- list("Los Angeles" = 1, Boston = 2, London = 3)
> x
$`Los Angeles`
[1] 1

$Boston
[1] 2

$London
[1] 3
> names(x)
[1] "Los Angeles" "Boston"      "London"
```

Obr. 26: Názvy 2

Matice môžu mať ako názvy aj stĺpce aj riadky.

```
> m <- matrix(1:4, nrow = 2, ncol = 2)
> dimnames(m) <- list(c("a", "b"), c("c", "d"))
> m
  c d
a 1 3
b 2 4
```

Obr. 27: Názvy 3

Stĺpce a riadky môžu byť zadané oddelene funkciami colnames() a rownames().

```
> colnames(m) <- c("h", "f")
> rownames(m) <- c("x", "z")
> m
  h f
x 1 3
z 2 4
```

Obr. 28: Názvy 4

4.17 Práca s dátami

Medzi pár základných funkcii na čítanie dát v R sú:

- read.table, read.csv: čítanie tabuľkových dát
- readLines: pre čítanie riadkov z textového súboru
- dget: taktiež pre čítanie v súboroch R kódu
- load: pre čítanie v uložených pracovných prostrediach
- unserialize: pre čítanie objektov v binárnej forme

Pričom pre písanie sa používajú následné funkcie:

- write.table: pre písanie tabuľkových dát do súboru
- writeLines: pre písanie dát riadok po riadku do súboru
- dump:na výpis textovej reprezentácie viacerých R objektov

- dput: pre výpis textovej reprezentácie R objektov
- save: ukladanie ľubovoľného počtu R objektov v binárnom formáte
- seralize: pre konvertovanie R objektov do binárneho formátu pre vpisovanie do súboru

Zoznam použitej literatúry

1. IHAKA, Ross (ed.). *R : Past and Future History* [online]. 1998. [cit. 2022-12-12]. Dostupné z : https://www.stat.auckland.ac.nz/~ihaka/downloads/Interface98.pdf.
2. PENG, Roger D. (ed.). *R Programming for Data Science* [online]. 2022-05-31. [cit. 2022-05-31]. Dostupné z : https://bookdown.org/rdpeng/rprogdatascience/r-nuts-and-bolts.html.
3. SIMPLILEARN. What is R: Overview, its Applications and what is R used for? [online]. 2022 [cit. 2022-11-23]. Dostupné z : https://www.simplilearn.com/what-is-r-article#what_is_r_used_for.
4. *Pros and Cons of R Programming Language – Unveil the Essential Aspects!* [online]. [cit. 2022-12-14]. Dostupné z : https://data-flair.training/blogs/pros-and-cons-of-r-programming-language/.