Intro to R - 1. Introduction

OIT/SMU Libraries Data Science Workshop Series

Michael Hahsler

OIT, SMU

World Changers Shaped Here

SMU®

1. What is R?

2. RStudio and a First R Session

3. R Basics: Vectors and Subsetting

4. Exercises

Section 1

What is R?

What is R?

- R is “GNU S’’. S is a language for statisticians developed at Bell Laboratories by John Chambers et al.
- R is designed by John Chambers and developed by the R Foundation.
- R is a language and environment for statistical computing and graphics
- R is the de facto standard to develop statistical software
- R implements variety of statistical and graphical techniques (linear and nonlinear modeling, statistical tests, time series analysis, classification, clustering, . . .)

What is R? (cont.)

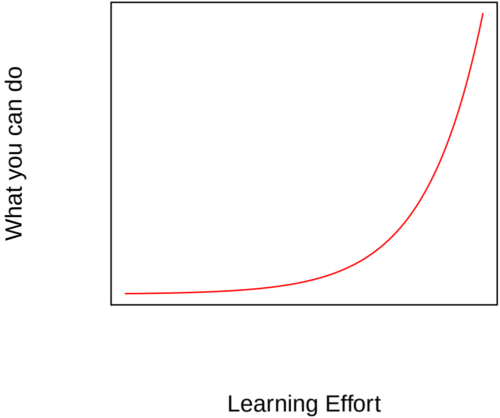
R provides

- data handling and storage
- operators for calculations on arrays (matrices)
- a large, coherent, integrated collection of intermediate tools for data analysis
- graphical facilities for data analysis and display
- simple and effective programming language (conditionals, loops, user defined recursive functions)
- extension mechanism with a large collection of packages

Why R?

- R is open-source and free to use.
- R has a large and active community.
- R is used widely in industry. Microsoft offers commercial solutions.
- R provides state-of-the-art algorithm in 15,000+ extension packages on CRAN (2019).
- R easily interfaces with other environments (C++, Python, Tensor Flow, . . .)
- R creates beautiful interactive visualizations (as seen in the New York Times and The Economist)
- RStudio makes creating reports and dash boards easy.

Issue?



Reason: R is confusing!

- Functional programming
- Vectorization
- Many competing ways to do things (base R, tidyverse, ggplot, grid, Sweave, markdown, . . .) and they all can be mixed.

Section 2

RStudio and a First R Session

A first session

Start RStudio and type the code in the gray box into the Console:

```
x <- 1:10
x
```

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

```
y <- x + 1
y
```

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

Look at the objects in the environment (Environment tab in RStudio)

```
ls()
```

```
## [1] "x" "y"
```

Ending an R session (Session menu in RStudio)

```
q()
```

How to get help

R comes with detailed online help

```
? ls            # get help on ls
help("ls")      # same as above
?? solve        # keyword search
```

Important information on http://cran.r-project.org/

- Manuals section (read: “An Introduction to R’ ’).
- Task Views section to find packages.
- Many packages have a vignette (see package page).

Other ways to find information

- Search https://stackoverflow.com/
- Just google

The R language

- R is a (mostly) functional programming language.
- Expressions are evaluated, printed and the result is lost unless assigned with `<-` (*Note:* Don't use `=` for assignments in R!)
- R is case sensitive!
- Commands are separated by a line break and rarely by a semi-colon (`;`)
- Expressions are grouped by braces (`{` and `}`)
- Comments start with a number sign (`#`)

Hint

R and RStudio provide very convenient auto-completion by hitting Tab.

Working Directory and Files

R sessions use a working directory to read and write files.

```
getwd()
setwd()
```

RStudio Recommendation

1. Start with a new R-script file.
2. Save it in the folder you want to work.
3. Go to Session and select `Set Working Directory -> To Source File Location`
4. Write all your code into the file (not the Console!) and execute with `CTRL-Enter`.

Data permanency (a.k.a. Workspace and Global Environment)

During an R session, objects are created and stored in the workspace by name. List objects with:

```r
ls()
```

```
## [1] "x" "y"
```

Objects can be removed from the workspace.

```r
rm(x)
ls()
```

```
## [1] "y"
```

Objects can be kept over several sessions by saving the workspace (to file called `.RData`).

Recommendation

Avoid saving the “Workspace” at the end of each session.
Reason: Your sessions will get messy and starting R may slow down if `.RData` gets very big. You can also remove the `.RData` file manually.

Section 3

R Basics: Vectors and Subsetting

Vectors

Vectors are the basic data structure in R. Scalars do not exist! Almost all numbers are seen as “numeric’’ (double).

```
42                      # this is a vector of length one!
```

```
## [1] 42
```

```
x <- c(10.4, 5.6, 3.1, 6.4, 21.7)   # c combines values
x
```

```
## [1] 10.4  5.6  3.1  6.4 21.7
```

```
1/x                                 # element-wise division
```

```
## [1] 0.0962 0.1786 0.3226 0.1562 0.0461
```

```
y <- c(x, 0, 0, 0, 0, 0, x)          # more combination
y
```

```
##  [1] 10.4  5.6  3.1  6.4 21.7  0.0  0.0  0.0  0.0
## [10]  0.0 10.4  5.6  3.1  6.4 21.7
```

Vector arithmetic

```r
x
```

```
## [1] 10.4  5.6  3.1  6.4 21.7
```

```r
y
```

```
##  [1] 10.4  5.6  3.1  6.4 21.7  0.0  0.0  0.0  0.0
## [10]  0.0 10.4  5.6  3.1  6.4 21.7
```

```r
x+y  # elements of the shorter array are recycled!
```

```
##  [1] 20.8 11.2  6.2 12.8 43.4 10.4  5.6  3.1  6.4
## [10] 21.7 20.8 11.2  6.2 12.8 43.4
```

```r
sum(x)
```

```
## [1] 47.2
```

```r
length(x)
```

```
## [1] 5
```

Sequences

```r
s1 <- 1:5          # sequence of integers
s1
```

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

```r
s2 <- seq(-1, 1, by = .2)   # using seq()
s2
```

```
##  [1] -1.0 -0.8 -0.6 -0.4 -0.2  0.0  0.2  0.4  0.6
## [10]  0.8  1.0
```

```r
rep(s1, times = 2)
```

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

```r
rep(s1, each = 2)
```

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

Try “`? seq`” and “`? rep`”

Logical vectors

```
x
```

```
## [1] 10.4  5.6  3.1  6.4 21.7
```

```
l <- x > 13   # compare each value in x
l
```

```
## [1] FALSE FALSE FALSE FALSE  TRUE
```

```
mode(l)
```

```
## [1] "logical"
```

```
as.numeric(l)   # coerce l into a numeric vector
```

```
## [1] 0 0 0 0 1
```

The usual relational operators are available (e.g., `<`, `<=`, `>`, `>=`, `==`, `!=`, `&` `|`). See `?"<"` and `?"&"` (quotation marks are necessary!)

Missing Values/Infinity

```
z <- c(1:3,NA)
z
```

```
## [1]  1  2  3 NA
```

```
ind <- is.na(z)     # find missing values
ind
```

```
## [1] FALSE FALSE FALSE  TRUE
```

```
0/0         # creates a NaN (not a number)
```

```
## [1] NaN
```

```
1 + NA
```

```
## [1] NA
```

```
2^5000          # (close to) infinity
```

```
## [1] Inf
```

See `?NA` and `?Inf`

Character vectors

```r
string <- c("Hello", "Ola")
string
```

```
## [1] "Hello" "Ola"
```

Combining string vectors

```r
paste(string, "World!")
```

```
## [1] "Hello World!" "Ola World!"
```

```r
labs <- paste(c("X","Y"), 1:10, sep = "")
labs
```

```
##  [1] "X1"  "Y2"  "X3"  "Y4"  "X5"  "Y6"  "X7"
##  [8] "Y8"  "X9"  "Y10"
```

See `?paste`

Factors

Used for categorical data. Strings are encoded as numbers with a look-up table.

```
(sex <- c("male", "female", "female", "male", "male"))
```

```
## [1] "male"   "female" "female" "male"   "male"
```

```
(sex <- factor(sex))
```

```
## [1] male   female female male   male
## Levels: female male
```

Look-up table

```
levels(sex)
```

```
## [1] "female" "male"
```

R stores an index into the look-up table

```
as.integer(sex)
```

```
## [1] 2 1 1 2 2
```

Warning

R sometimes converts strings into factors (e.g., for data tables) which can lead to problems. To get the strings back use `as.character()`.

Selecting and modifying subsets

```
x
```

```
## [1] 10.4  5.6  3.1  6.4 21.7
```

```
# select the first element (index starts with 1!)
x[1]
```

```
## [1] 10.4
```

```
# remove the first element
x[-1]
```

```
## [1]  5.6  3.1  6.4 21.7
```

```
# select elements (integer vector)
x[2:4]
```

```
## [1] 5.6 3.1 6.4
```

```
# select elements (logical vector)
x[x > 7]
```

```
## [1] 10.4 21.7
```

```
# replace elements
x[x > 7] <- NA
x
```

```
## [1]  NA 5.6 3.1 6.4  NA
```

Selecting and modifying subsets II

```
# using names
fruit <- c(5, 10, 1, 20)
names(fruit) <- c("orange", "banana", "apple", "peach")
fruit
```

```
## orange banana  apple  peach
##      5     10      1     20
```

```
lunch <- fruit[c("apple","orange")]
lunch
```

```
##  apple orange
##      1      5
```

See ?"["

Section 4

Exercises

Exercises

1. Create a vector with 10 numbers `(3, 12, 6, -5, 0, 8, 15, 1, -10, 7)` and assign it to `x`.
2. What is the ‘`data type'' ofx`‘? How can you find out?
3. Subtract 5 from the 2nd, 4th, 6th, etc. element in `x`.
4. Compute the sum and the average for `x` (there are functions for that).
5. Reverse the order of the elements in `x`.
6. Find out which numbers in `x` are negative.
7. Remove all entries with negative numbers from `x`.
8. How long is `x` now (there is a function).
9. Remove `x` from the environment/workspace (session).
10. Create the a vector of strings containing “CSE 8001”, “CSE 8002”, . . . , “CSE 8100” using paste.