Advanced R Programming - Lecture 1

Krzysztof Bartoszek
(slides by Leif Jonsson and Måns Magnusson)

Linköping University

krzysztof.bartoszek@liu.se

27 August 2018 (I101)

Today

Learn to

- Write R programs and packages
- Write performant code
- Learn basic software engineering practices

But most important...

But most important...

Your primary tool for (at least) the next two years

Course Plan

Part 1: R Syntax

Period: Week 1–2

Students work: Individually

Lab: Documented R file

Computer lab

Topics

- Basic R Syntax
- Basic data structures
- Program control
- R packages

Part 2: Advanced topics

Period: Weeks 3–7
Students work: In groups
Turn in: R package on GitHub
Seminar (**OBLIGATORY**),
1 Lab to help GitHub+Travis,

Topics

- Performant code: Writing quality code
- Linear algebra, Object orientation, Graphics
- Advanced I/O
- Performant code: Writing fast code
- Computational complexity
  (with exercise session, **BONUS** point possibility)

Today

Presentation(s)

Teaching staff for course

Me: Krzysztof Bartoszek, background

1. MEng in Computer Science, Gdańsk Univ. of Technology 2007
2. MPhil in Computational Biology, Univ. of Cambridge 2008
3. PhD in Statistics, Univ. of Gothenburg 2013
4. Postdoc, Dept. Mathematics Uppsala Univ. 2013–2017
5. Lecturer, STIMA LiU 2017–

Héctor Rodriguez Déniz

1. Labs
2. Grading

You

- Backgound?
- Why this course?
- Expectations?

Course Practicals...

Course Practicals...

- Course code: 732A94
- https://github.com/STIMALiU/AdvRCourse (materials)
- LISAM (submission, materials, messages, **exam information**)
- https://www.ida.liu.se/~732A94/index.en.shtml (2016 material, course reading)
- https://www.rstudio.com/
- https://cran.r-project.org/
- https://git-scm.com/

Course literature...

Course literature...

- Matloff, N. The art of R programming [online]
- Wickham, H. Advanced R [online]
- Wickham, H. R packages [online]
- Gillespie, C. and Lovelace, Efficient R programming [online]
- Google search, fora, . . .
- ...and articles.

Examination

Weekly mandatory labs/projects
– deadline: After corresponding lecture and seminar (for labs 3–6) stated on lab/LISAM

R package turn–in

Computer exam Points A: $[19, 20]$, B: $[17, 19)$, C: $[12, 17)$, D: $[10, 12)$, E: $[8, 9)$, F: $[0, 8)$

Computational complexity session bonus point can be obtained by correctly solving exercise on the board

Why R?

The One main reason

Choose the right tool for the job!

The One main reason

Choose the right tool for the job!

Your main job will be statistics and data analysis...
R is (nearly always) the right tool for that job!

Pros

- Popular (among statisticians)
- Good graphics support
- Open source - all major platforms!
- High-level language - focus on data analysis
- Strong community - vast amount of packages
- Powerful for communicating results
- API's to high-performance languages as C/C++ and Java

Cons

- "Ad hoc", complex, language (Compare Perl, Awk, Sh...)
- Can be slooooow
- Can be memory inefficient
- (Still) Hard'ish to troubleshoot (but . . .)
- (Still) Inferior IDE support compared to state of the art (but . . .)

Pros/Cons

- Niche language
- Specialized syntax
- Very permissive (changing for packages on CRAN)
- Troubleshooting: no (?) need to investigate memory
- (Still) Inferior IDE support compared to state of the art

Variable types

Variable type	Short	typeof()	R example
Boolean	logi	logical	TRUE
Integer	int	integer	1L
Real	num	double	1.2
Complex	cplx	complex	0+1i
Character	chr	character	"I <3 R"

Variable types

	Variable type	Short	typeof()	R example	
⇓	Boolean	logi	logical	TRUE	⇓
	Integer	int	integer	1L	
Coersion	Real	num	double	1.2	Coersion
	Complex	cplx	complex	0+1i	
⇓	Character	chr	character	"I <3 R"	⇓

Data structures

Dimension	Homogeneous data	Heterogeneous data
1	vector	list
2	matrix	data.frame
n	array	

- Constructors: vector() list() ...
- Name dimensions: dimnames()

Arithmetics

- Vectorized operations (element wise)
- Recycling
- Statistical functions

See reference card...

Logic operators

In symbols	A	B	¬A	A∧B	A∨B
In R	A	B	!A	A&B	$A \vert B$
	TRUE	FALSE	?	?	?
	TRUE	TRUE	?	?	?
	FALSE	FALSE	?	?	?
	FALSE	TRUE	?	?	?

Logic operators

In symbols	A	B	$\neg$A	A$\wedge$B	A$\vee$B
In R	*A*	*B*	!*A*	A&B	*A*|*B*
	TRUE	FALSE	FALSE	?	?
	TRUE	TRUE	?	?	?
	FALSE	FALSE	?	?	?
	FALSE	TRUE	?	?	?

Logic operators

In symbols	A	B	¬A	A∧B	A∨B
In R	A	B	$!A$	A&B	$A\|B$
	TRUE	FALSE	FALSE	FALSE	?
	TRUE	TRUE	?	?	?
	FALSE	FALSE	?	?	?
	FALSE	TRUE	?	?	?

Logic operators

In symbols	A	B	¬A	A∧B	A∨B
In R	*A*	*B*	!*A*	A&B	*A*|*B*
	TRUE	FALSE	FALSE	FALSE	TRUE
	TRUE	TRUE	?	?	?
	FALSE	FALSE	?	?	?
	FALSE	TRUE	?	?	?

Logic operators

In symbols	A	B	$\neg$A	A$\wedge$B	A$\vee$B
In R	A	B	$!A$	A&B	$A \vert B$
	TRUE	FALSE	FALSE	FALSE	TRUE
	TRUE	TRUE	FALSE	TRUE	TRUE
	FALSE	FALSE	TRUE	FALSE	FALSE
	FALSE	TRUE	TRUE	FALSE	TRUE

Logic operators

In symbols	$\wedge_{i=1}^{N} a_i$	$\vee_{i=1}^{N} a_i$	$\{j : a_j == TRUE\}$
In R	$all(A)$	$any(A)$	$which(A)$

Relational operators

In symbols	$a<b$	$a \leq b$	$a \neq b$	$a = b$	$a \in b$
In R	$a < b$	$a <= b$	$a! = b$	a == b	a %in% b

Vectors: Use []

- index by:
  - positive integers: include element(s)
  - negative integers: exclude element(s)
  - logical: include TRUEs

```
vect <- c(6,7,8,9)
> vect[vect>7];vect[which(vect>7)]  ##difference?
[1] 8 9
[1] 8 9
> vect[1:2]
[1] 6 7
> vect[c(1,2)]
[1] 6 7
> vect[c(-1,-2)]
[1] 8 9
```

Matrices

- Use [,]
- Two dimensions
- Index as vectors
- Can reduce (drop class) to vector
- Use [,,drop=FALSE]

Matrices

```
> mat <- matrix(c(1,2,3,4,5,6),nrow=2)
> mat
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> mat[c(1,2),c(1,2)]
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> mat[c(1,2),]
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> mat[mat>4]
[1] 5 6
```

Lists

- Use [] to access list elements
- Use [[]] to access list content
- Index as vectors
- Use $ to access list element by name
- Not like typical lists in other programming languages
- What if name of element sits inside a variable?

Lists

```
> lst <- list(a=47,b=11)
> lst[1]
$a
[1] 47

> lst[[1]]
[1] 47
> lst$b
[1] 11

> x<-"a";lst[which(names(lst)==x)]
$a
[1] 47

> lst[[which(names(lst)==x)]]
[1] 47
```

Data frames

- Very powerful data structure
- Can roughly think about it as the R representation of a CSV file
- Can be loaded from a CSV file
- Can be accessed both as a matrix and a list

- Be careful: picky data structure

Assigning subsets

- Change values in data structures
- Works for all above mentioned data types

Assigning subsets

```
> mat
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> mat[mat>4] = 75
> mat
     [,1] [,2] [,3]
[1,]    1    3   75
[2,]    2    4   75
```

Functions

```
my_function_name <- function(x, y){
        z <- x^2 + y^2
        return(z)
}
```

Unlike in many languages, `return` in R is a **function**. In other languages, `return` is usually a **reserved word** (like `if`). This means you must use `return` as a function call with parenthesis. By default R returns the last computed value of the function, so return is not strictly necessary in simple cases. What if you have a bunch of nested `if`s?

HELP!

?

help(function_name)

help("+")

?"-"

The End... for today.
Questions?
See you next time!