Introdution to R

Working with R

Franziska Faupel, Clara Filet & PD Dr. Oliver Nakoinz

October 2018

An Introdution to R

- R - What is it?!
  - *Why using R?*
  - *History of R*
  - *7 Levels of Learning*
  - *R Studio*
  - *Scripts*
- Working with R
  - *Function vs Objects*
  - *Vectors*
  - *Functions*
  - *Operators*
  - *Data Frames and Matrices*
- First Calculations and Plots in R
  - *Package Management*
  - *Loading Data*
  - *Data Management*
  - *Loops and Restrictions*
  - *Some Statistics*
  - *Digression: Tidy verse*
  - *Plotting in R*
- R and Spatial Data
  - *Load Spatial Objects*
  - *Create Spatial Objects*
  - *Manipulate Spatial Objects*
  - *Plot Spatial Objects*

Objects & Functions?!

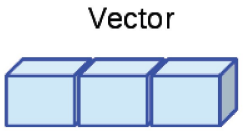
To understand computations in R, two slogans are helpful:

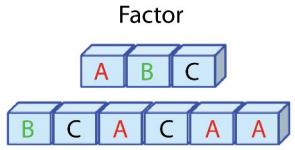
- **Everything that exists is an object.**
- **Everything that happens is a function call.**

John Chambers

R Data Structure

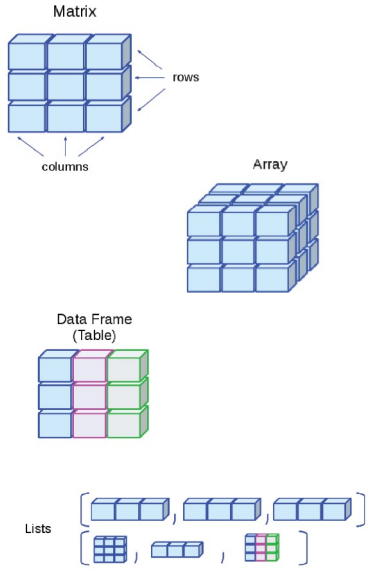
- **atomic vector**:
  - ***one-dimensional*** *arrays used to store collection data of the same mode*
    - numeric vectors (mode: numeric)
    - complex vectors (mode: complex)
    - logical vectors (mode: logical)
    - character vector or text strings (mode: character)
- **factors**:
  - *vectors containing pre-defined categorical values*
- **functions**:
  - *objects created by the user and reused to make specific operations.*



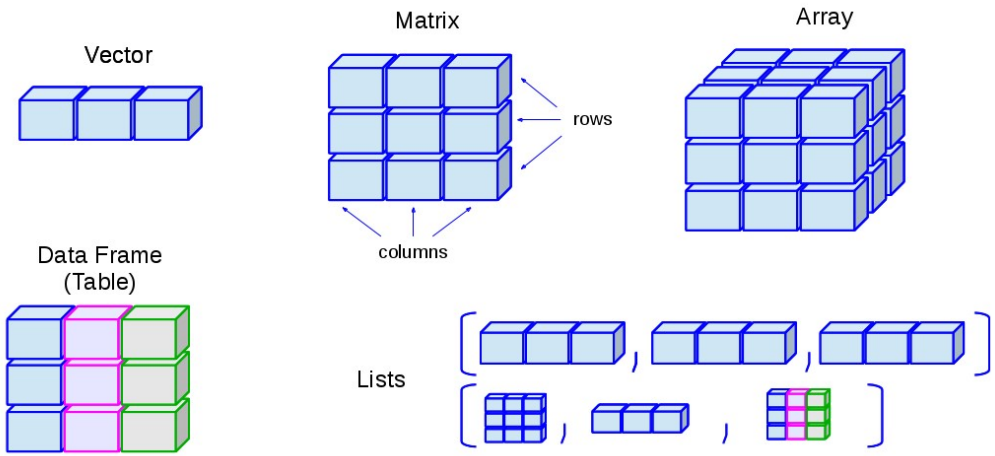


R Data Structure

- **matrix**:
  - ***two-dimensional*** *arrays to store collections of data of the same mode. They are accessed by two integer indices.*
- **arrays**:
  - *similar to matrices but they can be multi-dimensional (more than two dimensions)*
- **data frame**:
  - *generalization of matrices where different columns can store different mode data.*
- **list**:
  - *ordered collection of objects, where the elements can be of different types*



R Data Structure



Functions

- In programming, a named section of a program that performs a specific task.
- In this sense, a function is a procedure or routine.
- Some programming languages make a distinction between a function, which returns a value, and a procedure, which performs some operation but does not return a value.
- Most programming languages come with a pre-written set of functions that are kept in a library.
- In R you can download specific packages containing more specific functions.
- You can also write your own functions to perform specialized tasks.

http://www.webopedia.com/TERM/F/function.html

How to adapt an unkown function?

- Use implemented help function

```
help(round)
?round
```

- Read manual and vignette
- Try it and play around to see it operating

How to develop your own function?

```
name_func <- function(parameter1,parameter2){
  output <- "What ever needs to be done with
             parameter1 and parameter2"
  return (output)
}

name_func(parameter1,parameter2)
```

Operators

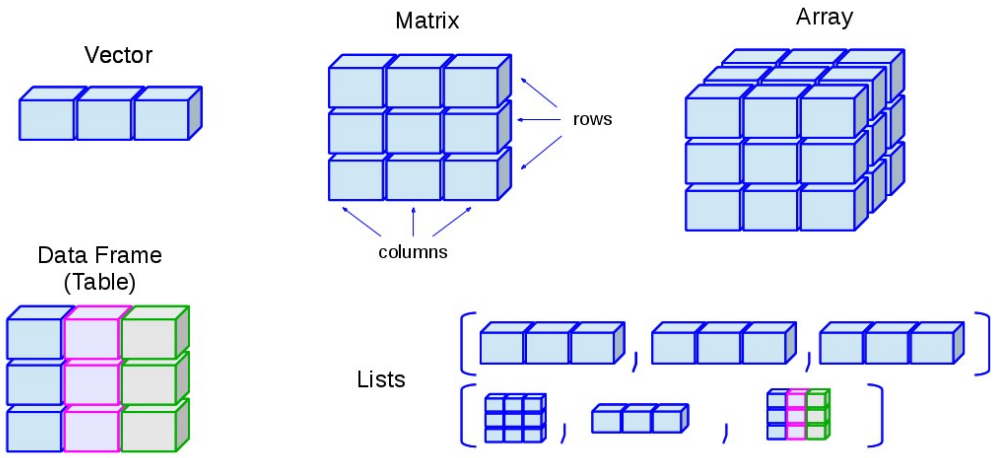
Arithmetic Operators

Operator	Description
+	addition
-	subtraction
*	multiplication
/	division
^ or **	exponentiation
x %% y	modulus (x mod y) 5%%2 is 1
x %/% y	integer division 5%/%2 is 2

Logical Operators

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	Not x
x	y
x & y	x AND y
isTRUE(x)	test if X is TRUE

Data Frames, Matrices & Arrays



Factors

Factors are special vectors that represent categorical/nominal data.

- Factors can only contain pre-defined values.
- Factors are pretty much integers that have labels on them.
  - *While factors look (and often behave) like character vectors, they are actually integers under the hood, and you need to be careful when treating them like strings.*
- Factors can be ordered or unordered and are important for modelling functions such as lm() and glm() and also in plot methods.
- When loading data, you should check, whether everything loaded correctly.

Summary: Objects in R

Obejct	Create	change to	check	get names	get dimensions
vector	`c()` or `vector()`	`as.vector`	`is.vector`	`names()`	`lenght()`
matrix	`matrix()`	`as.matrix`	`is.matrix`	`rownames()` or `colnames()`	`dim()`, `nrow()` or `ncol()`
array	`array()`	`as.array`	`is.array`	`dimnames()`	`dim()`
list	`list()`	`as.list`	`is.list`	`names()`	`lenght()`
data frame	`data.frame()`	`as.data.frame()`	`is.data.frame`	`names()`	`dim()`, `nrow()` or `ncol()`

Summary: Atomic Vectors

- numeric
  - *logical*
  - *integer*
  - *double*
- character
- complex
- raw
- date

```
dbl_var <- c(4,6,7,2)

# With the L suffix, you get an integer
int_var <- c(1L, 6L, 10L)

# Use TRUE and FALSE (or T and F)
# to create logical vectors
log_var <- c(TRUE, FALSE, T, F)

# Use " " for character strings
chr_var <- c("these are", "some strings")
```

Summary: Atomic Vectors

- numeric
  - *logical*
  - *integer*
  - *double*
- character
- complex
- raw
- date

```
dbl_var <- c(4,6,7,2)
dbl_var
# With the L suffix, you get an integer
int_var <- c(1L, 6L, 10L)
int_var
# Use TRUE and FALSE (or T and F)
# to create logical vectors
log_var <- c(TRUE, FALSE, T, F)
log_var
# Use " " for character strings
chr_var <- c("these are", "some strings")
chr_var
```

Summary: Atomic Vectors

- numeric
  - *logical*
  - *integer*
  - *double*
- character
- complex
- raw
- date

```r
# complex vector
cpl_var <-2i+2

# use 'as.Date()' to create a vector of type Date
d <- as.Date("2016-09-06")

# Raw vectors are used to store
# fixed-length sequences of bytes.
raw_var <- raw(10)
```

Summary: Atomic Vectors

- numeric
  - *logical*
  - *integer*
  - *double*
- character
- complex
- raw
- date

```
# complex vector
cpl_var <-2i+2
cpl_var
## [1] 2+2i
# use 'as.Date()' to create a vector of type Date
d <- as.Date("2016-09-06")
d
## [1] "2016-09-06"
# Raw vectors are used to store
# fixed-length sequences of bytes.
raw_var <- raw(10)
raw_var
##  [1] 00 00 00 00 00 00 00 00 00 00
```

Summary: Operators

Arithmetic Operators

Operator	Description
+	addition
-	subtraction
*	multiplication
/	division
^ or **	exponentiation
x %% y	modulus (x mod y) 5%%2 is 1
x %/% y	integer division 5%/%2 is 2

Logical Operators

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	Not x
x	y
x & y	x AND y
isTRUE(x)	test if X is TRUE