BST02: Using R for Statistics in Medical Research

Part C: Functions and Loops

Nicole Erler

Department of Biostatistics, Erasmus Medical Center

n.erler@erasmusmc.nl

24 - 28 February 2020

Recap Part B

Objects

- `vector`
- `matrix`
- `data.frame`
- `list`

Operators

- +, −, ∗, /
- <-, =
- <, >, ==

Data Structures

- `numeric`
- `character`
- `integer`
- `logical`
- `factor`

Special Values

- `NA`
- `NaN`
- `Inf`, `-Inf`

Data Transformations

- rounding (`format()`)
- convert to factor (`factor()`)

Data Exploration

- `mean()`, `median()`, `sd()`, `IQR()`, ...

Data Visualizations

- plotting packages
- plot types (`plot()`, `barplot()`, ...)

Subsetting

- `[[...]]`, `[...]`, ...

In this Section

- What are functions?
- Useful functions for data exploration
  - Demo
  - Practical
- Useful functions for data manipulations
  - Demo
  - Practical
- Writing functions
  - Demo
  - Practical
- Control-flow constructs
  - Demo
  - Practical

Functions

Sometimes we want to perform the same action / manipulation on several objects.

- Option 1: copy & paste
    - a lot of work
    - susceptible to mistakes
- Option 2: **functions**

Functions

Sometimes we want to perform the same action / manipulation on several objects.

- Option 1: copy & paste
  - a lot of work
  - susceptible to mistakes
- Option 2: **functions**

What are functions?

- a group of (organized) R commands
- a (small) programm with flexible (= not pre-specified) input

Almost all commands in R are functions!

Functions

Some examples:

- `mean()`
- `sum()`
- `plot()`
- ...

```r
class(mean)
## [1] "function"
class(sum)
## [1] "function"
class(plot)
## [1] "function"
```

Functions

Some examples:

- `mean()`
- `sum()`
- `plot()`
- ...

```
class(mean)
## [1] "function"
class(sum)
## [1] "function"
class(plot)
## [1] "function"
```

Even `class()` is a function:

```
class(class)
```

```
## [1] "function"
```

Useful Functions for Data Exploration

Link to Demo: Functions_DataExploration.R

Link to Practical:

Useful Functions for Data Exploration

Dimension

- `dim()`
- `nrow()`, `ncol()`
- `length()`

Data Structure

- `str()`
- `names()`,
- `head()`, `tail()`
- `is.data.frame()`, `is.list()`, `is.matrix()` `is.numeric()`, `is.ordered()`, ...

Descriptives for Continuous Variables

- `summary()`
- `min()`, `max()`, `range()`
- `mean()`, `median()`, `quantile()`, `IQR()`
- `sd()`, `var()`
- `ave()`

Tables

- `table()`, `prop.table()`
- `addmargins()`, `ftable()`

for `matrix` & `data.frame`

- `summary()`
- `var()`, `cor()`, `cov2cor()`
- `colSums()`, `colMeans()`, `rowSums()`, `rowMeans()`

Duplicates & Comparison

- `duplicated()`, `unique()`
- `all.equal()`, `identical()`

Useful functions for Data Manipulation

Link to Demo: Functions_DataManipulation.R

Link to practical

Useful functions for Data Manipulation

Transformations

- `log()`, `log2()`, `log10()`
- `exp()`, `sqrt()`, `plogis()`

Splitting & Combining

- `split()`, `cut()`
- `cbind()`, `rbind()`
- `merge()`
- `subset()`
- `c()`
- `paste()`

repetition & sequence

- `rep()`, `seq()`
- `expand.grid()`

Transformation for objects

- `t()`
- `unlist()`, `unname()`
- `as.numeric()`, `as.matrix()`, `as.data.frame()`

Sorting

- `sort()`, `order()`, `rev()`, `rank()`

matrices

- `%*%`
- `diag()`, `det()`, `solve()`
- `upper.tri()`, `lower.tri()`

Writing Functions

To write your own function:

```
myfun <- function(arguments) {
  syntax
}
```

For example:

```
square <- function(x) {
  x^2
}
```

```
square(3)
```

```
## [1] 9
```

Writing Functions

Functions do not always need an argument:

```r
random <- function() {
  rnorm(1)
}
```

```r
random()
## [1] 0.2338801
random()
## [1] -2.069272
random()
## [1] -0.4163516
```

Writing Functions

Functions can use multiple arguments:

```
subtract <- function(x, y) {
  x - y
}
```

```
subtract(x = 5.2, y = 3.3)
```

```
## [1] 1.9
```

Writing Functions

Multiple arguments are interpretet in the pre-defined order, unless they are named:

```
subtract(5.2, 1.2)
```

```
## [1] 4
```

```
subtract(y = 5.2, x = 1.2)
```

```
## [1] -4
```

Writing Functions

We can also define default values for arguments.

```
multiply <- function(x, y = 2) {
  x * y
}
```

The default value is used when the user does not specify a value for that argument:

```
multiply(x = 3, y = 3)
```

```
## [1] 9
```

```
multiply(x = 3)
```

```
## [1] 6
```

Writing Functions

Link to Demo

Link to Practical

Control-flow constructs

- `if(cond) expr`
- `if(cond) cons.expr else (alt.expr)`
- `ifelse()`
- `for`
- `while`
- `repeat`
- `break`
- `next`