Introduction to R

Patrick Toche[1]

[1]contact@patricktoche.com
https://github.com/ptoche/

April 2017

Introduction to R

- R is an open-source programming language, it is free for any use, personal, academic, commercial.
- The community of R users is one of the largest and growing. Thousands of users are ready to help.
- Check `http://stackoverflow.com/` for questions and answers. Before you post questions, be sure to check the forum's etiquette and how to write a good question: provide a minimal example (stripped of unnecessary detail) with reproducible data (either a sample of the original data or artificial data generated for the question), relevant bits of code you've tried, and the problems you've had, e.g. error messages. Stay focused.
- R has hundreds of packages that may be used to extend basic functionalities.
- `RStudio` is a great interface to R .

Get Started

- One of the first things you want to do is to set your current directory, so that you may be able to locate any data or plots you save to it.
  - On Windows:

    ```
    setwd("c:/R/introduction/")
    ```

  - On MacOS:

    ```
    setwd("~/R/introduction/")
    ```

- Find out what the current working directory is:

```
getwd()
```

Play around:

```
R>   2 + 2
## [1] 4
R>   2 * 2
## [1] 4
R>   2^2
## [1] 4
R>   pi
## [1] 3.141593
R>   Pi
## Error in eval(expr, envir, enclos):  object 'Pi' not found
R>   e
## Error in eval(expr, envir, enclos):  object 'e' not found
R>   exp(1)
## [1] 2.718282
```

Play around:

```
R>   x = 1
R>   str(x)

##  num 1

R>   x <- 1
R>   str(x)

##  num 1

R>   x <- "1"
R>   str(x)

##  chr "1"

R>   x <- 1L
R>   str(x)

##  int 1

R>   x <- c(1)
R>   str(x)

##  num 1
```

Play around:

```
R>   1 = 1
## Error in 1 = 1:  invalid (do_set) left-hand side to assignment
R>   1 == 1
## [1] TRUE
R>   is(1 == 1)
## [1] "logical" "vector"
R>   TRUE
## [1] TRUE
R>   TRUE == 1
## [1] TRUE
R>   is(TRUE == 1)
## [1] "logical" "vector"
R>   True == 1
## Error in eval(expr, envir, enclos):  object 'True' not found
```

R Objects

R objects include:

- Special values: `NA`, `NaN`, `Inf`, `-Inf`, `NULL`.
- Number types: integer (`int`), double-precision values (`num`), binary values that stand for True or False (`logi`), complex numbers (`cplx`). Double-precision values vary from machine to machine, but typically range from about $2e-308$ to $2e+308$.
- “container” objects (data structures): list, list of list, vector, matrix, array, table, data.frame, data.table (an efficient data.frame available from package `data.table`)
- “transforming” objects: function
- “graphical” objects: plot, ggplot, lattice, etc. May be subjected to further transformations for display or printing.
- “Date“ objects (`Date`).

R Commands

R frequently used commands include:

- `getwd()`, `setwd()`.
- `str()`, `typeof()`, `?` and `??`.
- `View()` to view a data.frame or matrix, e.g. `View(df)`. `head()` and `tail()` to quickly view parts of a data structure.
- `dput()` to convert a data.frame to R code.
- `rm()` to remove an object from the environment, e.g. `rm(df)`
- `install.packages("")` and `library()`, e.g. to use package `ggplot2`, install it with `install.packages("ggplot2")` (with quotes) and load it with `library("ggplot2")` (with or without quotes).

Vectors & Lists

```
R>   x <- list(1, 2, 3)
R>   str(x)

## List of 3
##  $ : num 1
##  $ : num 2
##  $ : num 3

R>   x <- c(1, 2, 3)
R>   str(x)

##  num [1:3] 1 2 3

R>   x <- c(a = 1, b = 2, c = 3)
R>   is.vector(x)

## [1] TRUE

R>   is.list(x)

## [1] FALSE

R>   typeof(x)

## [1] "double"
```

Vectors & Lists

- A list is a vector of mode “list.” The other modes are “logical”, “character”, “numeric”, “integer.”
- Lists are of recursive type, whereas atomic vectors are not. This means that they can contain values of different types, even other lists.
- In atomic vectors, all elements are of the same type, so manipulating them can be faster.

```
R>   x <- vector("list", 3)
R>   is.list(x)
## [1] TRUE
R>   is.vector(x)
## [1] TRUE
R>   is.recursive(x)
## [1] TRUE
R>   is.atomic(x)
## [1] FALSE
```

Subsetting Vectors

```
R>   x <- 1:10
R>   x[x > 4]
## [1]  5  6  7  8  9 10
R>   x[x > 9 | x < 2]
## [1]  1 10
R>   x[x > 4 & x < 6]
## [1] 5
R>   x[x == 5]
## [1] 5
R>   x[x %in% 3:5]
## [1] 3 4 5
R>   intersect(x, 3:5)
## [1] 3 4 5
```

Subsetting Vectors

```
R>   x <- rep("a", 10)
R>   x
##  [1] "a" "a" "a" "a" "a" "a" "a" "a" "a" "a"
```

The function `intersect` wraps the result with `unique`:

```
R>   unique(x)
## [1] "a"
R>   x <- c(1:10, 1:10)
R>   x[x %in% 3:5]
## [1] 3 4 5 3 4 5
R>   intersect(x, 3:5)
## [1] 3 4 5
```

Subsetting Vectors

Vectors can be subset by index.

```
R>   x <- letters[1:3]
R>   x
## [1] "a" "b" "c"
R>   x[2]
## [1] "b"
R>   x[-2]
## [1] "a" "c"
R>   x <- x[-2]
R>   x
## [1] "a" "c"
```

Subsetting Lists

Lists can hold lists:

```
R>   x <- list(ids = letters[1:3], values = sin(1:3),
+              info = list(x = 1, y = 2, z = 3))
R>   x$values
## [1] 0.8414710 0.9092974 0.1411200
R>   x[["ids"]]
## [1] "a" "b" "c"
R>   x$info$y
## [1] 2
R>   x[[3]] <- NULL
R>   x
## $ids
## [1] "a" "b" "c"
##
## $values
## [1] 0.8414710 0.9092974 0.1411200
```

Dates

R understands dates in the form “year-month-day“ or “year/month/day.” Other formats can be specified explicitly.

```
R>   as.Date("2017-12-13")

## [1] "2017-12-13"

R>   as.Date("2017-13-12")

## Error in charToDate(x):  character string is not in a standard
unambiguous format

R>   as.Date("2017-13-12", "%Y-%d-%m")

## [1] "2017-12-13"

R>   start <- as.Date("2000-7-1")
R>   end <- as.Date("2003-1-7")
R>   seq(start, end, by = "1 year")

## [1] "2000-07-01" "2001-07-01" "2002-07-01"

R>   seq(end, start, by = "-1 year") -> x
R>   x[x > start & x < end]

## [1] "2002-01-07" "2001-01-07"
```

Sequences

```
R>   0:10
##  [1]  0  1  2  3  4  5  6  7  8  9 10
R>   10:0
##  [1] 10  9  8  7  6  5  4  3  2  1  0
R>   rev(0:10)
##  [1] 10  9  8  7  6  5  4  3  2  1  0
R>   seq(0, 10)
##  [1]  0  1  2  3  4  5  6  7  8  9 10
R>   seq(0, 10, by = 2)
## [1]  0  2  4  6  8 10
R>   seq(0, 10, length.out = 2)
## [1]  0 10
R>   seq(0.1, 0.5, by = 0.05)
## [1] 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50
R>   seq(as.Date("2001-01-01"), as.Date("2003-01-01"), "years")
## [1] "2001-01-01" "2002-01-01" "2003-01-01"
```

Types & Conversions

Convert objects from one type to another:

```
R>   x <- c(1L, TRUE, 1.0, "1", 1)
R>   typeof(x)

## [1] "character"

R>   as.double(x)

## Warning:  NAs introduced by coercion
## [1]  1 NA  1  1  1

R>   as.integer(x)

## Warning:  NAs introduced by coercion
## [1]  1 NA  1  1  1

R>   as.numeric(x)

## Warning:  NAs introduced by coercion
## [1]  1 NA  1  1  1

R>   as.character(x)

## [1] "1"    "TRUE" "1"    "1"    "1"
```

Math Commands

- `mean()`, `median()`, `mode()`.
- `max()`, `min()`, `which.max()`, `which.min()`.
- `sum()`, `colSums()`, `rowSums()`, `colMeans()`, `rowMeans()`,
- `length()`, `unique()`, `na.omit()`, `is.na()`.
- `round()`, `ceiling()`, `floor()`.
- `var()`, `sd()`, `mad()`, `sample()`
- `factorial()`, `choose(n, k)`, `rbinom()`, `dnorm()`, `pnorm()`, `qnorm()`, `rnorm()`, `dunif()`, `punif()`, etc..
- `table()`, `cumsum()`, `cumprod()`, `cummax()`, `cummin()`

Math Commands

```
R>   x <- c(NA, 0L, 1L, 2L, 3L, 4L, 4L)
R>   mean(x)
## [1] NA
R>   mean(x, na.rm = TRUE)
## [1] 2.333333
R>   median(x, na.rm = TRUE)
## [1] 2.5
R>   max(x, na.rm = TRUE)
## [1] 4
R>   min(x, na.rm = TRUE)
## [1] 0
R>   na.omit(x)
## [1] 0 1 2 3 4 4
## attr(,"na.action")
## [1] 1
## attr(,"class")
## [1] "omit"
```

Math Commands

```
R>   x <- c(1L, 2L, 3L, 4L, 4L)
R>   quantile(x)
##   0%  25%  50%  75% 100%
##    1    2    3    4    4
R>   summary(x)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0     2.0     3.0     2.8     4.0     4.0
```

Math Commands

```
R>   x <- c(4L, 1L, 2L, 3L, 4L)
R>   sort(x)

## [1] 1 2 3 4 4

R>   rev(x)

## [1] 4 3 2 1 4

R>   sum(x)

## [1] 14

R>   prod(x)

## [1] 96

R>   cumsum(x)

## [1]  4  5  7 10 14

R>   cumprod(x)

## [1]  4  4  8 24 96

R>   length(x)

## [1] 5

R>   cumprod(x)[length(x)]

## [1] 96
```

More On Subsetting

```
R>   x <- c("c","ab","B","bba","c",NA,"@","bla","a","Ba","%")
R>   x[x %in% c(letters, LETTERS)]  # select single letters
## [1] "c" "B" "c" "a"
```

Functions

```
f <- function(x) {
  y <- x^2
  return(y)
}
```

```
R>   f <- function(x) {
+      y <- x^2
+      return(y)
+    }
R>   f(2)

## [1] 4
```

Functions

```
R>   "%w/o%" <- function(x, y) x[!x %in% y]  # x without y
R>   c(1:5) %w/o% 3

## [1] 1 2 4 5

R>   setdiff(c(1:5), 3)  # built-in function

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