Beginings

Al

2026-01-06

R programming Course notes

Following the first course beginings. They are telling me how to begin to understand the syntax of R. For example

```
x <- 1 ## <- assignment of value to a variable. This line is assigning the value 1 to the variable x
print(x) ## outputs in the terminal the value of x
```

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

```
x ## Auto-prints the value of x
```

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

```
msg <- "Hello" ## Assigning a string
msg
```

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

```
x <- 1:20 ## Creating a vector populated with the values from 1 to 20 in their respective positions.
x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

Data Types

R has 5 basic or "atomic" classes of objects:

- character
- numeric
- integer
- complex
- logical (boolean)

Objects

The most basic object is a vector

- A vector can only contain objects of the same class
- *list* is a vector that can contain different classes

Empty vector is created by the vector() function.

Numbers

Numbers in R are always treated as doubles. If you want an integer specifically you need to use the L suffix (ex. 1 is double, 1L is an integer). Inf is a special number which represents infinity. Also represented as (1/0). NaN represents an undefined value or missing number.

Attributes

R objeccts can have attributes - names,dimnames
- dimensions
- class
- length
- other user-defined attributes/metadata

Attributes of an objeect can be accessed using the attributes() functions.

Creating Vectors

The c() function can be used to create vectors of objects

```
x <- c(.5,.6) ## numeric
x
```

```
## [1] 0.5 0.6
```

```
x<- c(TRUE, TRUE) ## boolean
x
```

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

```
x <- c(T,F)
x
```

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

```
x <- c("a","b","c")
x
```

```
## [1] "a" "b" "c"
```

```
x <- 9:29 ## integer
x
```

```
##  [1]  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
```

```
x <- c(1+0i,2+4i) ## Complex
x
```

```
## [1] 1+0i 2+4i
```

Or using the vector() function

```
x <- vector("numeric", length = 10)
x
```

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

What about mixing objects?

```
y <- c(1.7,"a") ## char
y
```

```
## [1] "1.7" "a"
```

```
y <- c(T,2) ##numeric
y
```

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

```
y<- c("a", TRUE) ##Char
y
```

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

They get coerced to the common denominator class

Explicit coercion

Whay if you want to coerced something instead of letting the machine assume it? You can use the as.” ” () function.

```
x <- 0:6
class(x)
```

```
## [1] "integer"
```

```
as.numeric(x)
```

```
## [1] 0 1 2 3 4 5 6
```

```
class(as.numeric(x))
```

```
## [1] "numeric"
```

```
as.logical(x)
```

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

```
as.character(x)
```

```
## [1] "0" "1" "2" "3" "4" "5" "6"
```

Nonsensical coercion results in NA s

List

Vectors with different classes of opbjects

```
x <- list(1,"a",TRUE,1+4i,2+0i)
x
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] "a"
##
## [[3]]
## [1] TRUE
##
## [[4]]
## [1] 1+4i
##
## [[5]]
## [1] 2+0i
```