Object\_classes

Martin Andrzejak

November 2, 2021

Objects in R are the things you store in the environmental, there is an increasing amount of them and each packages has the potential to add different, new object classes. In this document I will go over the most commonly used classes (mainly base R). Each one comes with its own limitations and advantages.

For every object type, you can name the entries. So, you can name each vector entry, each column or row, or each list entry. If you do so, you can call the specific entry by using its name:

the\_vector["name\_of\_entry"]

the\_matrix[,"name\_of\_column"]

the\_dataframe[,"name\_of\_column"] or the\_dataframe$name\_of\_column the\_list[["name\_of\_list\_entry"]] or the\_list$name\_of\_list\_entry

1. Vector: A vector is a collection of data points and one dimensional. It can have only one data class. Usually a vector is indexed (that’s how you call single or multiple entries in it) like this:

the\_vector[position] (note the positions are usually a number)

to call the first entry of the vector you would write:

the\_vector[1]

to call the first three entries you would write:

the\_vector[1:3]

## different ways of creating a vector  
the\_vector1 <- as.vector(1:4)  
the\_vector2 <- c(1:4)  
the\_vector3 <- 1:4  
  
## creating an emtpy vector  
empty\_vec <- c()  
  
## note that it only tells you that it is an integer  
class(the\_vector1)

## [1] "integer"

## call the vector entries  
the\_vector2[1]

## [1] 1

the\_vector3[2:4]

## [1] 2 3 4

## name the vector entries and call them, this will change the class to named integer  
names(the\_vector1) <- c("entry1", "entry2", "entry3", "entry4")  
  
the\_vector1["entry1"]

## entry1   
## 1

1. Matrix: A matrix is a table consisting of rows and columns. The data within a matrix can be only of one class. matrixes are indexed like this: the\_matrix[row\_position, column\_position]; call the entire first row: the\_matrix[1,]; call the entire first column: the\_matrix[,1]; call the first cell (first position in row and column): the matrix[1,1]. A matrix is two dimensional (rows + columns)

## create a matrix, note that I added a column with characters, changing every cell of the matrix to an character  
matrix1 <- matrix(c(1:3, letters[1:3]), ncol = 2)  
  
## create an empty matrix   
matrix2 <- matrix()  
  
## call different entries of the matrix  
matrix1[1,] ## call first row

## [1] "1" "a"

matrix1[,1] ## call first column

## [1] "1" "2" "3"

matrix1[1,1] ## call first cell

## [1] "1"

##see what happens if we change the first column to numeric and try to save it  
matrix1[,1] <- as.numeric(matrix1[,1])  
  
matrix1 ## nothing changed, however you can call the first row as numeric:

## [,1] [,2]  
## [1,] "1" "a"   
## [2,] "2" "b"   
## [3,] "3" "c"

as.numeric(matrix1[,1])

## [1] 1 2 3

## name entries and call them  
rownames(matrix1) <- c("row1", "row2", "row3")  
colnames(matrix1) <- c("col1", "col2")  
  
matrix1[,"col1"]

## row1 row2 row3   
## "1" "2" "3"

matrix1["row1",]

## col1 col2   
## "1" "a"

matrix1["row1", "col1"]

## [1] "1"

1. data frame: Comparable to a matrix, but can consists of different data types. One column can be numeric, while another one consists of characters. Indexing is same as for a matrix as they are also two dimensional.

## create a data frame  
dataframe1 <- data.frame(col1 = 1:3, col2 = letters[1:3])  
dataframe1

## col1 col2  
## 1 1 a  
## 2 2 b  
## 3 3 c

## note how now the first column is an integer column and the second row a character column  
  
## we can also translate a matrix into a data frame  
mat <- matrix(c(1:3, letters[1:3]), ncol = 2)  
dataframe2 <- as.data.frame(mat)  
dataframe2

## V1 V2  
## 1 1 a  
## 2 2 b  
## 3 3 c

## in this case the data frame took the character class from the matrix in each column + the column names are V1 and V2, we can change both pretty easily and I am sure you can figure it out yourself  
  
##call a data frame, you can call it like the matrix, however there is one new way to call a column:   
dataframe1$col1

## [1] 1 2 3

1. lists: lists are like vectors but can store at each position a data frame, a matrix, a vector or what ever you want. Indexing: the\_list[[position\_of\_list\_entry]]. Once you call the list entry the indexing happens as what it was stored as. Lists can be multidimensional.

## create lists with different entry types:   
## 1. vector  
vec\_list <- list(c(1:3), c(4:6), c(7:9))  
vec\_list

## [[1]]  
## [1] 1 2 3  
##   
## [[2]]  
## [1] 4 5 6  
##   
## [[3]]  
## [1] 7 8 9

##call the first vector  
vec\_list[[1]]

## [1] 1 2 3

## call the second vector on the 2nd position  
vec\_list[[2]][2]

## [1] 5

## call the third vector on position 2 and 3  
vec\_list[[2]][2:3]

## [1] 5 6

## create a list of data frames  
df\_list <- list(mtcars, cars)  
df\_list[[1]]

## mpg cyl disp hp drat wt qsec vs am gear carb  
## Mazda RX4 21.0 6 160.0 110 3.90 2.620 16.46 0 1 4 4  
## Mazda RX4 Wag 21.0 6 160.0 110 3.90 2.875 17.02 0 1 4 4  
## Datsun 710 22.8 4 108.0 93 3.85 2.320 18.61 1 1 4 1  
## Hornet 4 Drive 21.4 6 258.0 110 3.08 3.215 19.44 1 0 3 1  
## Hornet Sportabout 18.7 8 360.0 175 3.15 3.440 17.02 0 0 3 2  
## Valiant 18.1 6 225.0 105 2.76 3.460 20.22 1 0 3 1  
## Duster 360 14.3 8 360.0 245 3.21 3.570 15.84 0 0 3 4  
## Merc 240D 24.4 4 146.7 62 3.69 3.190 20.00 1 0 4 2  
## Merc 230 22.8 4 140.8 95 3.92 3.150 22.90 1 0 4 2  
## Merc 280 19.2 6 167.6 123 3.92 3.440 18.30 1 0 4 4  
## Merc 280C 17.8 6 167.6 123 3.92 3.440 18.90 1 0 4 4  
## Merc 450SE 16.4 8 275.8 180 3.07 4.070 17.40 0 0 3 3  
## Merc 450SL 17.3 8 275.8 180 3.07 3.730 17.60 0 0 3 3  
## Merc 450SLC 15.2 8 275.8 180 3.07 3.780 18.00 0 0 3 3  
## Cadillac Fleetwood 10.4 8 472.0 205 2.93 5.250 17.98 0 0 3 4  
## Lincoln Continental 10.4 8 460.0 215 3.00 5.424 17.82 0 0 3 4  
## Chrysler Imperial 14.7 8 440.0 230 3.23 5.345 17.42 0 0 3 4  
## Fiat 128 32.4 4 78.7 66 4.08 2.200 19.47 1 1 4 1  
## Honda Civic 30.4 4 75.7 52 4.93 1.615 18.52 1 1 4 2  
## Toyota Corolla 33.9 4 71.1 65 4.22 1.835 19.90 1 1 4 1  
## Toyota Corona 21.5 4 120.1 97 3.70 2.465 20.01 1 0 3 1  
## Dodge Challenger 15.5 8 318.0 150 2.76 3.520 16.87 0 0 3 2  
## AMC Javelin 15.2 8 304.0 150 3.15 3.435 17.30 0 0 3 2  
## Camaro Z28 13.3 8 350.0 245 3.73 3.840 15.41 0 0 3 4  
## Pontiac Firebird 19.2 8 400.0 175 3.08 3.845 17.05 0 0 3 2  
## Fiat X1-9 27.3 4 79.0 66 4.08 1.935 18.90 1 1 4 1  
## Porsche 914-2 26.0 4 120.3 91 4.43 2.140 16.70 0 1 5 2  
## Lotus Europa 30.4 4 95.1 113 3.77 1.513 16.90 1 1 5 2  
## Ford Pantera L 15.8 8 351.0 264 4.22 3.170 14.50 0 1 5 4  
## Ferrari Dino 19.7 6 145.0 175 3.62 2.770 15.50 0 1 5 6  
## Maserati Bora 15.0 8 301.0 335 3.54 3.570 14.60 0 1 5 8  
## Volvo 142E 21.4 4 121.0 109 4.11 2.780 18.60 1 1 4 2

## name list entries, no matter what you store in the list (vectors,

data frames ...) this works the same  
  
names(df\_list) <- c("mtcars", "cars")  
df\_list$cars

## speed dist  
## 1 4 2  
## 2 4 10  
## 3 7 4  
## 4 7 22  
## 5 8 16  
## 6 9 10  
## 7 10 18  
## 8 10 26  
## 9 10 34  
## 10 11 17  
## 11 11 28  
## 12 12 14  
## 13 12 20  
## 14 12 24  
## 15 12 28  
## 16 13 26  
## 17 13 34  
## 18 13 34  
## 19 13 46  
## 20 14 26  
## 21 14 36  
## 22 14 60  
## 23 14 80  
## 24 15 20  
## 25 15 26  
## 26 15 54  
## 27 16 32  
## 28 16 40  
## 29 17 32  
## 30 17 40  
## 31 17 50  
## 32 18 42  
## 33 18 56  
## 34 18 76  
## 35 18 84  
## 36 19 36  
## 37 19 46  
## 38 19 68  
## 39 20 32  
## 40 20 48  
## 41 20 52  
## 42 20 56  
## 43 20 64  
## 44 22 66  
## 45 23 54  
## 46 24 70  
## 47 24 92  
## 48 24 93  
## 49 24 120  
## 50 25 85

## now you can do:   
df\_list$mtcars$mpg

## [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4  
## [16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7  
## [31] 15.0 21.4

1. Arrays­: Can be used to store more than 2 dimensions. Consist of only one data type. They are indexed like: the array[position\_row, position\_column, position\_in\_array]. I usually don’t use arrays as lists do pretty much the same thing. If you want to get into them: <https://www.tutorialspoint.com/r/r_arrays.htm>

There are many more data classes but those are probably the most important for the beginning. Important is that you understand how each one works and what the conditions of every single object class are.