Data Structures

2020

In this section we are going to review 5 data structures:

- vector
- list
- matrix
- array
- data frame

Vector

Vectors are collections of elements from the same type. They keep track of the order that the data is entered in. Their indexing starts from 1. You can create a vector in R using the combine function c.

```
> x <- c(1, 2, 3); x
[1] 1 2 3
> y <- c(5, 6, 7, 8, 9, 10); y
[1]  5  6  7  8  9 10
> z <- c(TRUE, FALSE); z
[1]  TRUE FALSE
> s <- c("S", "t", "a", "t", "i", "s", "t", "i", "c", "s"); s
[1] "S" "t" "a" "t" "i" "s" "t" "i" "c" "s"
```

Note that the vector has a type and can only store data from one type.

```
> typeof(x)
[1] "double"
> typeof(y)
[1] "double"
> typeof(z)
[1] "logical"
> typeof(s)
[1] "character"
> p <- c(5, 6.4, "string"); p
[1] "5"      "6.4"    "string"
> typeof(s)
[1] "character"
```

Note that in such situations, R will convert the element types to the highest common type following the order

NULL < logical < integer < double < character

R is a vectorised language, so operations are applied to each element of the vector automatically, without the need to loop through the vector.

```
> x <- c(1, 2, 3); x
[1] 1 2 3
> x + 2
[1] 3 4 5
```

```
> 5 * x
[1]  5 10 15
> x^2
[1] 1 4 9
> sqrt(x)
[1] 1.000000 1.414214 1.732051
> length(x)
[1] 3
```

Operations on vectors with the same length

```
> x <- c(1, 2, 3)
> y <- c(5, 6, 7)
> x + y
[1]  6  8 10
> y - x
[1] 4 4 4
> x * y
[1]  5 12 21
> x / y
[1] 0.2000000 0.3333333 0.4285714
```

If the two vectors are with different length, to make the operation we repeat the elements of the shorter vector until the vectors has the same length and then perform the operation.

```
> x <- c(1, 2, 3)
> y <- c(5, 6, 7, 8, 9, 10)
> x + y
[1]  6  8 10  9 11 13
> y - x
[1] 4 4 4 7 7 7
> x * y
[1]  5 12 21  8 18 30
> x / y
[1] 0.2000000 0.3333333 0.4285714 0.1250000 0.2222222 0.3000000
```

Taking elements from the vector

```
> y <- c(5, 6, 7, 8, 9, 10)
> y[3]
[1] 7
> y[c(3, 4)]
[1] 7 8
> y[c(4, 2, 3)]
[1] 8 6 7
> y[-3]
[1]  5  6  8  9 10
> y[8]
[1] NA
```

You can also assign value to a specific index

```
> y[9] = 11
> y[1:9]
[1]  5  6  7  8  9 10 NA NA 11
```

Different ways to generate vectors

```
> 1:12
[1]  1  2  3  4  5  6  7  8  9 10 11 12
> 1:-9
[1]  1  0 -1 -2 -3 -4 -5 -6 -7 -8 -9
> seq(1, 17, by = 2)
[1]  1  3  5  7  9 11 13 15 17
> rep(1:8, times = 2)
[1] 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8
> rep(1:8, each = 2)
[1] 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8
> rep(1:8, each = 2, times = 2)
[1] 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8
```

Some more examples

```
> x <- 5:7; x
[1] 5 6 7
> y <- 2:5; y
[1] 2 3 4 5
> y <= 3
[1]  TRUE  TRUE FALSE FALSE
> x[y <= 3]
[1] 5 6
> sum(y <= 3)
[1] 2
> sum(x[y <= 3])
[1] 11
> which(y >= 3)
[1] 2 3 4
> x[which(y >= 3)]
[1]  6  7 NA
> z <- 5:21; z
[1]  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
> z[y <= 3] # Different vectors length!
[1]  5  6  9 10 13 14 17 18 21
```

Lists

Lists are ordered sets of components stored in a vector. The objects in the list are not necessarily from the same type, the same data structure or the same length. To create a list in R we use the function list.

```
> x <- list(1, 2, 3); x
[[1]]
```

```
[1] 1

[[2]]
[1] 2

[[3]]
[1] 3

> y <- list(a = 1, b = 2, c = 3); y

$a
[1] 1

$b
[1] 2

$c
[1] 3

> z <- list(a = c(1, 2, 4, 5),
+          b = c(TRUE, FALSE, TRUE),
+          c = c("Statistics")); z

$a
[1] 1 2 4 5

$b
[1]  TRUE FALSE  TRUE

$c
[1] "Statistics"

> p <- list(a = c(1, 2, 4, 5),
+          b = list(c = c(TRUE, FALSE), d = c(1, 2, 4), e = c("string")),
+          c = data.frame(x = 1:10, y = rep(c(TRUE, FALSE), 5))); p

$a
[1] 1 2 4 5

$b
$b$c
[1]  TRUE FALSE

$b$d
[1] 1 2 4

$b$e
[1] "string"


$c
    x     y
```

```
1   1  TRUE
2   2 FALSE
3   3  TRUE
4   4 FALSE
5   5  TRUE
6   6 FALSE
7   7  TRUE
8   8 FALSE
9   9  TRUE
10 10 FALSE
```

We can see each component's structure using the str function

```
> str(p)
List of 3
 $ a: num [1:4] 1 2 4 5
 $ b:List of 3
  ..$ c: logi [1:2] TRUE FALSE
  ..$ d: num [1:3] 1 2 4
  ..$ e: chr "string"
 $ c:'data.frame':	10 obs. of  2 variables:
  ..$ x: int [1:10] 1 2 3 4 5 6 7 8 9 10
  ..$ y: logi [1:10] TRUE FALSE TRUE FALSE TRUE FALSE ...
```

Each component from the list can be extracted using the $ sign followed by the component's name or [[<index>]]

```
> p$a
[1] 1 2 4 5
> p[[1]]
[1] 1 2 4 5
> p$b$c
[1]  TRUE FALSE
> p[[2]][[1]]
[1]  TRUE FALSE
```

We can list the names of the components in the list using names function

```
> names(p)
[1] "a" "b" "c"
> names(p$b)
[1] "c" "d" "e"
```

Matrix
All of the elements in a matrix must be of the same type. To create a matrix in R we use the function matrix.

```
> A <- matrix(c(2, 4, 1, 5, 7, 6),
+           nrow = 2,
+           ncol = 3,
+           byrow = TRUE)
```

```
> A

     [,1] [,2] [,3]
[1,]    2    4    1
[2,]    5    7    6

> B <- matrix(c(2, 4, 3, 1, 5, 7),
+           nrow = 3,
+           ncol = 2)
> B

     [,1] [,2]
[1,]    2    1
[2,]    4    5
[3,]    3    7
```

Dimensions of the matrix

```
> dim(B)
[1] 3 2
```

Taking elements of the matrix

```
> A[2, 3]
[1] 6
> A[2, ]
[1] 5 7 6
> A[, 3]
[1] 1 6
> A[, c(1, 3)]
     [,1] [,2]
[1,]    2    1
[2,]    5    6
```

Operations on matrices

```
> A
     [,1] [,2] [,3]
[1,]    2    4    1
[2,]    5    7    6
> A + 3
     [,1] [,2] [,3]
[1,]    5    7    4
[2,]    8   10    9
> G <- matrix(c(rep(1:2, 3)), nrow = 2); G
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    2    2    2
> A + G
     [,1] [,2] [,3]
[1,]    3    5    2
[2,]    7    9    8
> A*G
```

```
     [,1] [,2] [,3]
[1,]    2    4    1
[2,]   10   14   12
```

t transpose the matrix

```
> t(A)
     [,1] [,2]
[1,]    2    5
[2,]    4    7
[3,]    1    6
```

%*% algebraic multiplication of two matrices

```
> G %*% t(A)
     [,1] [,2]
[1,]    7   18
[2,]   14   36
```

cbind combine a sequence of vector, matrix or data frame by columns

```
> cbind(c(1, 2, 3), c(3, 4, 5))
     [,1] [,2]
[1,]    1    3
[2,]    2    4
[3,]    3    5
> cbind(B, c(3, 4, 5))
     [,1] [,2] [,3]
[1,]    2    1    3
[2,]    4    5    4
[3,]    3    7    5
```

rbind combine a sequence of vector, matrix or data frame by rows

```
> rbind(c(1, 2, 3), c(3, 4, 5))
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    3    4    5
> C = rbind(A, c(3, 4, 5)); C
     [,1] [,2] [,3]
[1,]    2    4    1
[2,]    5    7    6
[3,]    3    4    5
```

dimnames set names to the dimensions of the object

```
> dimnames(C) <- list(c("r1", "r2", "r3"), c("c1", "c2", "c3"))
> C
   c1 c2 c3
r1  2  4  1
r2  5  7  6
r3  3  4  5
```

You can also use colnames and rownames functions to set the names of the object

```
> colnames(C) <- c("col1", "col2", "col3")
> C
  col1 col2 col3
r1    2    4    1
r2    5    7    6
r3    3    4    5
> rownames(C) <- c("row1", "row2", "row3")
> C
     col1 col2 col3
row1    2    4    1
row2    5    7    6
row3    3    4    5
```

Array

Arrays are multidimensional vectors. All the elements in the array must be of the same type. To create an array in R we use the function array.

```
> x <- array(1:24, dim = c(4, 3, 2)); x

, , 1

     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12

, , 2

     [,1] [,2] [,3]
[1,]   13   17   21
[2,]   14   18   22
[3,]   15   19   23
[4,]   16   20   24
```

Taking elements of the array

```
> x[1, , ]

     [,1] [,2]
[1,]    1   13
[2,]    5   17
[3,]    9   21

> x[1, , 1]

[1] 1 5 9

> x[, , 1]
```

```
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
```

dimnames set names to the dimensions of the object

```
> dimnames(x) <- list(c("x1", "x2", "x3", "x4"), c("y1", "y2", "y3"), c("z1", "z2"))
> x
, , z1

   y1 y2 y3
x1  1  5  9
x2  2  6 10
x3  3  7 11
x4  4  8 12

, , z2

   y1 y2 y3
x1 13 17 21
x2 14 18 22
x3 15 19 23
x4 16 20 24
```

Data Frame
Data frames hold vectors not necessarily from the same type but with the same length. To create a data frame in R we use the function data.frame.

```
> first <- 1:10
> second <- c("aa", "bb", "cc", "dd", "ee", "ff", "gg", "hh", "ii", "jj")
> third <- c(TRUE, FALSE, TRUE, TRUE, FALSE, TRUE, FALSE, FALSE, TRUE, FALSE)
> df <- data.frame(first, second, third); df
   first second third
1      1     aa  TRUE
2      2     bb FALSE
3      3     cc  TRUE
4      4     dd  TRUE
5      5     ee FALSE
6      6     ff  TRUE
7      7     gg FALSE
8      8     hh FALSE
9      9     ii  TRUE
10    10     jj FALSE
> class(df)
[1] "data.frame"
```

You can use View function to visualize the data frame in a separate window.
head and tail visualize the first respectively the last set of rows of the data frame

```
> head(df)
  first second third
```

```
1     1     aa  TRUE
2     2     bb FALSE
3     3     cc  TRUE
4     4     dd  TRUE
5     5     ee FALSE
6     6     ff  TRUE
> head(df, n = 3)
  first second third
1     1     aa  TRUE
2     2     bb FALSE
3     3     cc  TRUE
> tail(df)
   first second third
5      5     ee FALSE
6      6     ff  TRUE
7      7     gg FALSE
8      8     hh FALSE
9      9     ii  TRUE
10    10     jj FALSE
> tail(df, n = 3)
   first second third
8      8     hh FALSE
9      9     ii  TRUE
10    10     jj FALSE
```

dim, nrow and ncol shows the dimensions of the data frame

```
> dim(df)
[1] 10  3
> dim(df)[1]
[1] 10
> dim(df)[2]
[1] 3
> nrow(df)
[1] 10
> ncol(df)
[1] 3
```

To view the column's data types use str function

```
> str(df)
'data.frame':   10 obs. of  3 variables:
 $ first : int  1 2 3 4 5 6 7 8 9 10
 $ second: chr  "aa" "bb" "cc" "dd" ...
 $ third : logi  TRUE FALSE TRUE TRUE FALSE TRUE ...
```

It is possible to give or change the names of the columns of the data frame

```
> df <- data.frame(A = first, B = second, C = third); df
    A  B     C
1   1 aa  TRUE
2   2 bb FALSE
```

```
3   3 cc  TRUE
4   4 dd  TRUE
5   5 ee FALSE
6   6 ff  TRUE
7   7 gg FALSE
8   8 hh FALSE
9   9 ii  TRUE
10 10 jj FALSE
```

names shows the names of the columns of the data frame

```
> names(df)
[1] "A" "B" "C"
```

You can give names to the rows and columns of the data frame using the colnames and rownames functions

```
> colnames(df) <- c("f", "s", "t"); df
    f  s     t
1   1 aa  TRUE
2   2 bb FALSE
3   3 cc  TRUE
4   4 dd  TRUE
5   5 ee FALSE
6   6 ff  TRUE
7   7 gg FALSE
8   8 hh FALSE
9   9 ii  TRUE
10 10 jj FALSE
> rownames(df) <- c("one", "two", "three", "four", "five", "six", "seven", "eight", "nine",
"ten"); df
       f  s     t
one    1 aa  TRUE
two    2 bb FALSE
three  3 cc  TRUE
four   4 dd  TRUE
five   5 ee FALSE
six    6 ff  TRUE
seven  7 gg FALSE
eight  8 hh FALSE
nine   9 ii  TRUE
ten   10 jj FALSE
```

or we can use predefined constants for the names

```
> colnames(df) <- LETTERS[1:3]; df
       A  B     C
one    1 aa  TRUE
two    2 bb FALSE
three  3 cc  TRUE
four   4 dd  TRUE
five   5 ee FALSE
```

```
six     6 ff  TRUE
seven  7 gg FALSE
eight  8 hh FALSE
nine   9 ii  TRUE
ten   10 jj FALSE
> rownames(df) <- month.name[1:10]; df
           A  B     C
January    1 aa  TRUE
February   2 bb FALSE
March      3 cc  TRUE
April      4 dd  TRUE
May        5 ee FALSE
June       6 ff  TRUE
July       7 gg FALSE
August     8 hh FALSE
September  9 ii  TRUE
October   10 jj FALSE
```

Data frames also has attributes. Where names lists the names of the columns and row.names lists the names of the rows.

```
> attributes(df)
$names
[1] "A" "B" "C"

$class
[1] "data.frame"

$row.names
 [1] "January"   "February"  "March"     "April"     "May"       "June"     
 [7] "July"      "August"    "September" "October"  
> names(df)
[1] "A" "B" "C"
> colnames(df)
[1] "A" "B" "C"
> rownames(df)
 [1] "January"   "February"  "March"     "April"     "May"       "June"     
 [7] "July"      "August"    "September" "October"  
```

Taking elements of a data frame

- Taking the element from the 2 row, 3 column

```
> df[2, 3]
[1] FALSE
```

- Taking the elements from the 2 row

```
> df[2, ]
         A  B     C
February 2 bb FALSE
```

- Taking the elements from the 3 column

```
> df[, 3]
 [1]  TRUE FALSE  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE
```

- Taking the elements from the 2 row, 2 and 3 columns

```
> df[2, 2:3]
          B     C
February bb FALSE
```

- Taking the elements from the 2 row, columns “A” and “B”

```
> df[2, c("A" , "B")]
         A  B
February 2 bb
```

- Taking the elements from the column “A”

```
> df$A
 [1]  1  2  3  4  5  6  7  8  9 10
```

- Taking the elements from the 2 column

```
> df[[2]]
 [1] "aa" "bb" "cc" "dd" "ee" "ff" "gg" "hh" "ii" "jj"
```

Sources

[1] Monika Petkova's notes on R programming language @ FMI, Sofia University