Data structures in R

Andrew Ba Tran

Contents

This is from the first chapter of learn.r-journalism.com.

Vectors

A **vector** is a sequence of data elements of the same basic type. The parts that consist of a vector are called **components** or **elements**.

```
vec1 <- c(1,4,6,8,10)
vec1
```

```
## [1]  1  4  6  8 10
```

A vector `vec` is explicitly constructed by the concatenation function `c()`.

```
vec1[5]
```

```
## [1] 10
```

Elements in vectors can be addressed by standard `[i]` indexing

```
vec1[3] <- 12
vec1
```

```
## [1]  1  4 12  8 10
```

One of the elements in the array is replaced with a new number.

```
vec2 <- seq(from=0, to=1, by=0.25)
vec2
```

```
## [1] 0.00 0.25 0.50 0.75 1.00
```

This shows another useful way of creating a vector: the `seq()` or sequence function.

```
sum(vec1)
```

```
## [1] 35
```

Some calculations.

```
vec1 + vec2
```

```
## [1]  1.00  4.25 12.50  8.75 11.00
```

If you add up two vectors of the same length, the first elements of both vectors are summed, and the second elements, etc., leading to a new vector length of 5.

Matrices

Matrices are two-dimensional vectors.

It looks like this

```
mat <- matrix(data=c(9,2,3,4,5,6), ncol=3)
mat
```

```
##      [,1] [,2] [,3]
## [1,]    9    3    5
## [2,]    2    4    6
```

The argument `data` specifies which numbers should be in the matrix.

Use either `ncol` to specify the number of columns or `nrow` to specify the number of rows.

Matrix-operations are similar to vector operations

```
mat[1,2]
```

```
## [1] 3
```

Elements of a matrix can be addressed in the usual way

```
mat[2,1]
```

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

When you want to select a whole row, you leave the spot for the column number empty and vice versa for columns.

```
mean(mat)
```

```
## [1] 4.833333
```

This is how a function would work with a matrix as an argument.

Data frames

If you're used to working with spreadsheets, then *data frames* will make the most sense to you in R.

It's more general than a matrix - different columns can contain different modes of data (numeric, character, etc.)

Basically, it's a matrix with names above the columns for headers.

This means you can call and use one of the columns without knowing in which position it is.

- Rectangular array of data
- More general than a matrix - different columns can contain different modes of data (numeric, character, etc.)
- Similar to data sets in SAS, SPSS, and Stata

Creating a data frame:

```
patientID <- c(111, 208, 113, 408)
age <- c(25, 34, 28, 52)
sex <- c(1,2,1,1)
diabetes <- c("Type1", "Type2", "Type1", "Type1")
status <- c(1,2,3,1)
```

```
patientdata <- data.frame(patientID, age, sex, diabetes, status)
patientdata
```

```
##   patientID age sex diabetes status
## 1       111  25   1    Type1      1
## 2       208  34   2    Type2      2
## 3       113  28   1    Type1      3
## 4       408  52   1    Type1      1
```

This is a typical data frame built from arrays.

Specifying elements of a data frame:

```
patientdata[1:2]
```

```
##   patientID age
## 1       111  25
## 2       208  34
## 3       113  28
## 4       408  52
```

```
patientdata[c("diabetes", "status")]
```

```
##   diabetes status
## 1    Type1      1
## 2    Type2      2
## 3    Type1      3
## 4    Type1      1
```

```
patientdata$age
```

```
## [1] 25 34 28 52
```

```
patientdata[1:2]
```

```
##   patientID age
## 1       111  25
## 2       208  34
## 3       113  28
## 4       408  52
```

```
patientdata[c(1,3),1:2]
```

```
##   patientID age
## 1       111  25
## 3       113  28
```

```
patientdata[2:3, 1:2]
```

```
##   patientID age
## 2       208  34
## 3       113  28
```

```
mean(patientdata$age)
```

```
## [1] 34.75
```

Instead of using `mean(patientdata[,2])` like you would with a matrix, you can select the column `age` from the `patientdata` data frame with the `$` sign.

```r
mean(patientdata[["age"]])
```

```
## [1] 34.75
```

Here's an alternative way to refer to the `age` column of the `patientdata` data frame. But you will rarely use this method.

Lists

Another basic structure in R is a *list*.

The main advantage of lists is that the "columns" they're not really ordered in columns any more, but are more of a collection of vectors) don't have to be of the same length, unlike matrices and data frames.

Kind of like JSON files are structured.

```r
g <- "My First List"
h <- c(25, 26, 18, 39)
j <- matrix(1:10, nrow = 5)
k <- c("one", "two", "three")
mylist <- list(title = g, ages = h, j, k)
```

This is how a list would appear in the work space

```r
names(mylist)
```

```
## [1] "title" "ages"  ""      ""
```

How to find out what's in the list

```r
mylist[[2]]
```

```
## [1] 25 26 18 39
```

```r
mylist[["ages"]][[1]]
```

```
## [1] 25
```

The code above extracts data from the list

```r
mylist$age + 10
```

```
## [1] 35 36 28 49
```

How to refer to and use the numbers in the example list

Functions for working with objects

Let's start with the sample_df data frame below.

```r
# Run the lines of code below
sample_df <- data.frame(id=c(1001,1002,1003,1004), name=c("Steve", "Pam", "Jim", "Dwight"),
                        age=c(26, 65, 15, 7), race=c("White", "Black", "White", "Hispanic"))
sample_df$name <- as.character(sample_df$name)
sample_df
```

```
##     id   name age     race
## 1 1001  Steve  26    White
## 2 1002    Pam  65    Black
```

```
## 3 1003    Jim  15    White
## 4 1004 Dwight   7 Hispanic
```

`length(x)` - Find out how many things there are in an object or array

```
length(sample_df$name)
```

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

`nchar(x)` - If **x** is a string, finds how how many characters there are

```
sample_df$name[1]
```

```
## [1] "Steve"
```

```
nchar(sample_df$name[1])
```

```
## [1] 5
```

`dim(x)` - Gives the dimensions of **x**

```
dim(sample_df)
```

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

`ncol(x)` - Counts the number of columns

```
ncol(sample_df)
```

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

`nrow(x)` - Returns the number of rows of **x**

```
nrow(sample_df)
```

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

`str(x)` - Returns the structure of **x**

```
str(sample_df)
```

```
## 'data.frame':    4 obs. of  4 variables:
##  $ id  : num  1001 1002 1003 1004
##  $ name: chr  "Steve" "Pam" "Jim" "Dwight"
##  $ age : num  26 65 15 7
##  $ race: Factor w/ 3 levels "Black","Hispanic",..: 3 1 3 2
```

`summary(x)` - Summarizes the object as understood by R

```
summary(sample_df)
```

```
##        id           name                age              race
##  Min.   :1001   Length:4           Min.   : 7.00   Black   :1
##  1st Qu.:1002   Class :character   1st Qu.:13.00   Hispanic:1
##  Median :1002   Mode  :character   Median :20.50   White   :2
##  Mean   :1002                      Mean   :28.25
##  3rd Qu.:1003                      3rd Qu.:35.75
##  Max.   :1004                      Max.   :65.00
```

`View(x)` - A command to open the object to browse in RStudio

```
View(sample_df)
```

`rm(x)` - Removes **x**

```r
rm(sample_df)
sample_df
```

```
## Error in eval(expr, envir, enclos): object 'sample_df' not found
```