Create, Append and Modify List Components

Create, Append and Modify List Components

In this article, you will learn to work with lists in R programming. You will learn to create, access, modify and delete list components.

List is a data structure having components of mixed data types.

A vector having all elements of the same type is called atomic vector but a vector having elements of different type is called list.

We can check if it's a list with `typeof()` function and find its length using `length()`. Here is an example of a list having three components each of different data type.

```
> x
$a
[1] 2.5
$b
[1] TRUE
$c
[1] 1 2 3
> typeof(x)
[1] "list"
> length(x)
[1] 3
```

How to create a list in R programming?

List can be created using the `list()` function.

```
> x <- list("a" = 2.5, "b" = TRUE, "c" = 1:3)
```

Here, we create a list `x`, of three components with data types `double`, `logical` and `integer` vector respectively.

Its structure can be examined with the `str()` function.

```
> str(x)
List of 3
$ a: num 2.5
$ b: logi TRUE
$ c: int [1:3] 1 2 3
```

In this example, `a`, `b` and `c` are called tags which makes it easier to reference the components of the list.

However, tags are optional. We can create the same list without the tags as follows. In such scenario, numeric indices are used by default.

```
> x <- list(2.5,TRUE,1:3)
> x
[[1]]
[1] 2.5
[[2]]
[1] TRUE
[[3]]
[1] 1 2 3
```

How to access components of a list?

Lists can be accessed in similar fashion to vectors. Integer, logical or character vectors can be used for indexing. Let us consider a list as follows.

```
> x
$name
[1] "John"
$age
[1] 19
$speaks
[1] "English" "French"
> x[c(1:2)]    # index using integer vector
$name
[1] "John"
$age
[1] 19
> x[-2]        # using negative integer to exclude second component
$name
[1] "John"
$speaks
[1] "English" "French"
> x[c(T,F,F)]  # index using logical vector
$name
[1] "John"
> x[c("age","speaks")]    # index using character vector
$age
[1] 19
$speaks
[1] "English" "French"
```

Indexing with `[` as shown above will give us sublist not the content inside the component. To retrieve the content, we need to use `[[`.

However, this approach will allow us to access only a single component at a time.

```
> x["age"]
$age
[1] 19
> typeof(x["age"])    # single [ returns a list
[1] "list"
> x[["age"]]    # double [[ returns the content
[1] 19
> typeof(x[["age"]])
[1] "double"
```

An alternative to `[[`, which is used often while accessing content of a list is the `$` operator. They are both the same except that `$` can do partial matching on tags.

```
> x$name    # same as x[["name"]]
[1] "John"
> x$a              # partial matching, same as x$ag or x$age
[1] 19
> x[["a"]]         # cannot do partial match with [[
NULL
> # indexing can be done recursively
> x$speaks[1]
[1] "English"
> x[["speaks"]][2]
[1] "French"
```

How to modify a list in R?

We can change components of a list through reassignment. We can choose any of the component accessing techniques discussed above to modify it.

Notice below that modification causes reordering of components.

```
> x[["name"]] <- "Clair"; x
$age
[1] 19
$speaks
[1] "English" "French"
$name
[1] "Clair"
```

How to add components to a list?

Adding new components is easy. We simply assign values using new tags and it will pop into action.

```
> x[["married"]] <- FALSE
> x
$age
[1] 19
$speaks
[1] "English" "French"
$name
[1] "Clair"
$married
[1] FALSE
```

How to delete components from a list?

We can delete a component by assigning `NULL` to it.

```
> x[["age"]] <- NULL
> str(x)
List of 3
$ speaks : chr [1:2] "English" "French"
$ name   : chr "Clair"
$ married: logi FALSE
> x$married <- NULL
> str(x)
List of 2
$ speaks: chr [1:2] "English" "French"
$ name  : chr "Clair"
```