Expressions are encapsulated operations that can be executed by R. This may sound complicated, but using expressions allows you manipulate code with code! You can create an expression using thequote() function. For that function's argument, just type whatever you would normally type into the R console. For example:

```
two_plus_two <- quote(2 + 2)
two_plus_two
2 + 2
```

You can execute this expressions using the eval() function:

```
eval(two_plus_two)
[1] 4
```

You might encounter R code that is stored as a string that you want to evaluate with eval(). You can use parse() to transform a string into an expression:

```
tpt_string <- "2 + 2"

tpt_expression <- parse(text = tpt_string)

eval(tpt_expression)
[1] 4
```

You can reverse this process and transform an expression into a string using deparse():

```
deparse(two_plus_two)
[1] "2 + 2"
```

One interesting feature about expressions is that you can access and modify their contents like you a list(). This means that you can change the values in an expression, or even the function being executed in the expression before it is evaluated:

```
sum_expr <- quote(sum(1, 5))
eval(sum_expr)
[1] 6
sum_expr[[1]]
sum
sum_expr[[2]]
[1] 1
sum_expr[[3]]
[1] 5
sum_expr[[1]] <- quote(paste0)
sum_expr[[2]] <- quote(4)
sum_expr[[3]] <- quote(6)
eval(sum_expr)
[1] "46"
```

You can compose expressions using the call() function. The first argument is a string containing the name of a function, followed by the arguments that will be provided to that function.

```
sum_40_50_expr <- call("sum", 40, 50)
sum_40_50_expr
sum(40, 50)
eval(sum_40_50_expr)
[1] 90
```