In R, objects are allowed to have ***attributes***, which is a way for users to tag additional information to an R object.

There are a few reasons why one might want to use attributes. One reason that I encountered recently was to ensure that the type of object returned from a function remains consistent across a range of function options.

For example, imagine that you have a function that does a lot of complicated work to get an intermediate result `res1`, and just a little bit more work to get the final result `res2`. In some cases you might want to just return `res2`, while in other cases you might want to return `res1` as well to save you computation in the future.

Without attributes, you might do something like this:

```
f <- function(keep_intermediate) {
    ...
    if (keep_intermediate) {
        return(list(res1 = res1, res2 = res2))
    } else {
        return(res2)
    }
}
```

The tricky thing here is that the return value is a list if `keep_intermediate = TRUE`, but may not be if `keep_intermediate = FALSE`. With attributes, you can avoid this issue:

```
f <- function(keep_intermediate) {
    ...
    if (keep_intermediate) {
        attr(res2, "res1") <- res1
    }
    return(res2)
}
```

Working with attributes in R

Use the `attributes()` function to look at all the attributes an object has (returned as a list). The code below shows that by default a matrix will have a `dim` attribute, and the output of the `lm()` function will have `names` and `class` attributes.

```
x <- matrix(rnorm(10), ncol = 2)
attributes(x)
# $dim
# [1] 5 2

fit <- lm(rnorm(5) ~ x)
attributes(fit)
# $names
# [1] "coefficients" "residuals" "effects"
# [4] "rank" "fitted.values" "assign"
# [7] "qr" "df.residual" "xlevels"
# [10] "call" "terms" "model"
```

```
#
# $class
# [1] "lm"
```

Use the `attr()` function to set an attribute for an object:

```
x <- 1:3
attr(x, "test") <- "this is a test"
x
# [1] 1 2 3
# attr(,"test")
# [1] "this is a test"
```

Note that the method above will not work when the object is `NULL`:

```
y <- NULL
attr(y, "attr1") <- "test"
# Error in attr(y, "attr1") <- "test" : attempt to set an attribute on
NULL
```

If the object is `NULL`, we can do the following instead:

```
attributes(y)$attr1 <- "this works"
attributes(y)["attr2"] <- "this also works"
str(y)
# list()
# - attr(*, "attr1")= chr "this works"
# - attr(*, "attr2")= chr "this also works"
```

Checking if an object has a particular attribute doesn't seem that easy, the code below is what I have (perhaps there is an easier way!). Note that

```
attributes(attributes(y))
# $names
# [1] "attr1" "attr2"
```

Hence, the code below checks if `y` has the `"attr1"` and `"attr3"` attribute:

```
"attr1" %in% attributes(attributes(y))$names
# [1] TRUE
"attr3" %in% attributes(attributes(y))$names
# [1] FALSE
```

You can use the following code to remove an attribute:

```
attr(y, "attr1") <- NULL
str(y)
# list()
# - attr(*, "attr2")= chr "this also works"…
```