Chapter 8: Classes

The class of a data-object determines which functions will process its contents. The `class`-attribute is a character vector, and objects can have zero, one or more classes. If there is no class-attribute, there will still be an implicit class determined by an object's `mode`. The class can be inspected with the function `class` and it can be set or modified by the `class<-` function. The S3 class system was established early in S's history. The more complex S4 class system was established later

Section 8.1: Inspect classes

Every object in R is assigned a class. You can use `class()` to find the object's class and `str()` to see its structure, including the classes it contains. For example:

```
class(iris)
[1] "data.frame"

str(iris)
'data.frame':    150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 ...

class(iris$Species)
[1] "factor"
```

We see that iris has the class `data.frame` and using `str()` allows us to examine the data inside. The variable Species in the iris data frame is of class factor, in contrast to the other variables which are of class numeric. The `str()` function also provides the length of the variables and shows the first couple of observations, while the `class()` function only provides the object's class.

Section 8.2: Vectors and lists

Data in R are stored in vectors. A typical vector is a sequence of values all having the same storage mode (e.g., characters vectors, numeric vectors). See `?atomic` for details on the atomic implicit classes and their corresponding storage modes: `"logical"`, `"integer"`, `"numeric"` (synonym `"double"`), `"complex"`, `"character"` and `"raw"`. Many classes are simply an atomic vector with a `class` attribute on top:

```
x <- 1826
class(x) <- "Date"
x
# [1] "1975-01-01"
 x <- as.Date("1970-01-01")
 class(x)
#[1] "Date"
 is(x,"Date")
#[1] TRUE
 is(x,"integer")
#[1] FALSE
 is(x,"numeric")
#[1] FALSE
  mode(x)
#[1] "numeric"
```

Lists are a special type of vector where each element can be anything, even another list, hence the R term for lists: "recursive vectors":

```
mylist <- list( A = c(5,6,7,8), B = letters[1:10], CC = list( 5, "Z") )
```

Lists have two very important uses:

- Since functions can only return a single value, it is common to return complicated results in a list:

```
f <- function(x) list(xplus = x + 10, xsq = x^2)

f(7)
# $xplus
# [1] 17
#
# $xsq
# [1] 49
```

- Lists are also the underlying fundamental class for data frames. Under the hood, a data frame is a list of vectors all having the same length:

```
L <- list(x = 1:2, y = c("A","B"))
DF <- data.frame(L)
DF
#   x y
# 1 1 A
# 2 2 B
is.list(DF)
# [1] TRUE
```

The other class of recursive vectors is R expressions, which are "language"- objects

Section 8.3: Vectors

The most simple data structure available in R is a vector. You can make vectors of numeric values, logical values, and character strings using the `c()` function. For example:

```
c(1, 2, 3)
## [1] 1 2 3
c(TRUE, TRUE, FALSE)
## [1]  TRUE  TRUE FALSE
c("a", "b", "c")
## [1] "a" "b" "c"
```

You can also join to vectors using the `c()` function.

```
x <- c(1, 2, 5)
y <- c(3, 4, 6)
z <- c(x, y)
z
## [1] 1 2 5 3 4 6
```

A more elaborate treatment of how to create vectors can be found in the *"Creating vectors"* topic