17.10: Coercing Data from One Class to Another

Sometimes you want to change the variable class. This can happen for all sorts of reasons. Sometimes when you import data from files, it can come to you in the wrong format: numbers sometimes get imported as text, dates usually get imported as text, and many other possibilities besides. Regardless of how you've ended up in this situation, there's a very good chance that sometimes you'll want to convert a variable from one class into another one. Or, to use the correct term, you want to ***coerce*** the variable from one class into another. Coercion is a little tricky, and so I'll only discuss the very basics here, using a few simple examples.

Firstly, let's suppose we have a variable `x` that is *supposed* to be representing a number, but the data file that you've been given has encoded it as text. Let's imagine that the variable is something like this:

```
x <- "100"  # the variable
class(x)    # what class is it?
```

```
## [1] "character"
```

Obviously, if I want to do calculations using `x` in its current state, R is going to get very annoyed at me. It thinks that `x` is text, so it's not going to allow me to try to do mathematics using it! Obviously, we need to coerce `x` from character to numeric. We can do that in a straightforward way by using the `as.numeric()` function:

```
x <- as.numeric(x)  # coerce the variable
class(x)            # what class is it?
```

```
## [1] "numeric"
```

```
x + 1               # hey, addition works!
```

```
## [1] 101
```

Not surprisingly, we can also convert it back again if we need to. The function that we use to do this is the `as.character()` function:

```
x <- as.character(x)   # coerce back to text
class(x)               # check the class:
```

```
## [1] "character"
```

However, there's some fairly obvious limitations: you can't coerce the string `"hello world"` into a number because, well, there's isn't a number that corresponds to it. Or, at least, you can't do anything useful:

```
as.numeric( "hello world" )  # this isn't going to work.
```

```
## Warning: NAs introduced by coercion
```

```
## [1] NA
```

In this case R doesn't give you an error message; it just gives you a warning, and then says that the data is missing (see Section 4.6.1 for the interpretation of `NA`).

That gives you a feel for how to change between numeric and character data. What about logical data? To cover this briefly, coercing text to logical data is pretty intuitive: you use the `as.logical()` function, and the character strings `"T"`, `"TRUE"`, `"True"` and `"true"` all convert to the logical value of `TRUE`. Similarly `"F"`, `"FALSE"`, `"False"`, and `"false"` all become `FALSE`. All other strings convert to `NA`. When you go back the other way using `as.character()`, `TRUE` converts to `"TRUE"` and `FALSE` converts to `"FALSE"`. Converting numbers to logicals – again using `as.logical()` – is straightforward. Following the convention in the study of logic, the number `0` converts to `FALSE`. Everything else is `TRUE`. Going back using `as.numeric()`, `FALSE` converts to `0` and `TRUE` converts to `1`.
