3.8: Storing Text Data

A lot of the time your data will be numeric in nature, but not always. Sometimes your data really needs to be described using text, not using numbers. To address this, we need to consider the situation where our variables store text. To create a variable that stores the word "hello", we can type this:

```
greeting <"hello"
greeting
```

```
## [1] "hello"
```

When interpreting this, it's important to recognise that the quote marks here *aren't* part of the string itself. They're just something that we use to make sure that R knows to treat the characters that they enclose as a piece of text data, known as a ***character string***. In other words, R treats `"hello"` as a string containing the word "hello"; but if I had typed `hello` instead, R would go looking for a variable by that name! You can also use `'hello'` to specify a character string.

Okay, so that's how we store the text. Next, it's important to recognise that when we do this, R stores the entire word `"hello"` as a *single* element: our `greeting` variable is *not* a vector of five different letters. Rather, it has only the one element, and that element corresponds to the entire character string `"hello"`. To illustrate this, if I actually ask R to find the first element of `greeting`, it prints the whole string:

```
greeting[1]
```

```
## [1] "hello"
```

Of course, there's no reason why I can't create a vector of character strings. For instance, if we were to continue with the example of my attempts to look at the monthly sales data for my book, one variable I might want would include the names of all 12 `months`.^[Though actually there's no real need to do this, since R has an inbuilt variable called `month.name`] that you can use for this purpose. To do so, I could type in a command like this

```
months <c("January", "February", "March", "April", "May", "June",
          "July", "August", "September", "October", "November",
          "December")
```

This is a ***character vector*** containing 12 elements, each of which is the name of a month. So if I wanted R to tell me the name of the fourth month, all I would do is this:

```
months[4]
```

```
## [1] "April"
```

3.8.1 Working with text

Working with text data is somewhat more complicated than working with numeric data, and I discuss some of the basic ideas in Section 7.8, but for purposes of the current chapter we only need this bare bones sketch. The only other thing I want to do before moving on is show you an example of a function that can be applied to text data. So far, most of the functions that we have seen (i.e., `sqrt()`, `abs()` and `round()`) only make sense when applied to numeric data (e.g., you can't calculate the square root of "hello"), and we've seen one function that can be applied to pretty much any variable or vector (i.e., `length()`). So it might be nice to see an example of a function that can be applied to text.

The function I'm going to introduce you to is called `nchar()`, and what it does is count the number of individual characters that make up a string. Recall earlier that when we tried to calculate the `length()` of our `greeting` variable it returned a

value of `1` : the `greeting` variable contains only the one string, which happens to be `"hello"` . But what if I want to know how many letters there are in the word? Sure, I could *count* them, but that's boring, and more to the point it's a terrible strategy if what I wanted to know was the number of letters in *War and Peace*. That's where the `nchar()` function is helpful:

```
nchar( x = greeting )
```

```
## [1] 5
```

That makes sense, since there are in fact 5 letters in the string `"hello"` . Better yet, you can apply `nchar()` to whole vectors. So, for instance, if I want R to tell me how many letters there are in the names of each of the 12 months, I can do this:

```
nchar( x = months )
```

```
##  [1] 7 8 5 5 3 4 4 6 9 7 8 8
```

So that's nice to know. The `nchar()` function can do a bit more than this, and there's a lot of other functions that you can do to extract more information from text or do all sorts of fancy things. However, the goal here is not to teach any of that! The goal right now is just to see an example of a function that actually does work when applied to text.
