4.9: Lists

The next kind of data I want to mention are ***lists***. Lists are an extremely fundamental data structure in R, and as you start making the transition from a novice to a savvy R user you will use lists all the time. I don't use lists very often in this book – not directly – but most of the advanced data structures in R are built from lists (e.g., data frames are actually a specific type of list). Because lists are so important to how R stores things, it's useful to have a basic understanding of them. Okay, so what is a list, exactly? Like data frames, lists are just "collections of variables." However, unlike data frames – which are basically supposed to look like a nice "rectangular" table of data – there are no constraints on what kinds of variables we include, and no requirement that the variables have any particular relationship to one another. In order to understand what this actually *means*, the best thing to do is create a list, which we can do using the `list()` function. If I type this as my command:

```
Dan <- list( age = 34,
             nerd = TRUE,
             parents = c("Joe","Liz")
)
```

R creates a new list variable called `Dan`, which is a bundle of three different variables: `age`, `nerd` and `parents`. Notice, that the `parents` variable is longer than the others. This is perfectly acceptable for a list, but it wouldn't be for a data frame. If we now print out the variable, you can see the way that R stores the list:

```
print( Dan )
```

```
## $age
## [1] 34
##
## $nerd
## [1] TRUE
##
## $parents
## [1] "Joe" "Liz"
```

As you might have guessed from those `$` symbols everywhere, the variables are stored in exactly the same way that they are for a data frame (again, this is not surprising: data frames *are* a type of list). So you will (I hope) be entirely unsurprised and probably quite bored when I tell you that you can extract the variables from the list using the `$` operator, like so:

```
Dan$nerd
```

```
## [1] TRUE
```

If you need to add new entries to the list, the easiest way to do so is to again use `$`, as the following example illustrates. If I type a command like this

```
Dan$children <- "Alex"
```

then R creates a new entry to the end of the list called `children`, and assigns it a value of `"Alex"`. If I were now to `print()` this list out, you'd see a new entry at the bottom of the printout. Finally, it's actually possible for lists to contain other lists, so it's quite possible that I would end up using a command like `Dan$children$age` to find out how old my son is. Or I could try to remember it myself I suppose.