CSC 120: Computer Science for the Sciences (R section)

Radford M. Neal, University of Toronto, 2015

http://www.cs.utoronto.ca/∼radford/csc120/

Week 7

Adding Attributes to R Objects

An R object can have one or more "attributes", that record extra information. They are mostly ignored if you don't look at them, but are there if you look.

An example:

```
> x <- 123                      # Set x to a plain number
> x
[1] 123
> attr(x,"fred") <- "abc"       # Add a "fred" attribute to x
> x
[1] 123
attr(,"fred")
[1] "abc"
> attr(x,"fred")                # We can get just the attribute if we like
[1] "abc"
> x + 1000                      # The attribute (usually) gets passed on
[1] 1123
attr(,"fred")
[1] "abc"
```

Attributes for Dimensions and Names

You can attach attributes to objects for your own purposes, but R also has some standard uses for attributes.

R uses a `dim` attribute to mark an object as a matrix, and hold how many rows and columns it has. This attribute is not usually shown explicitly, be we can see it if we look using `attr`:

```
> M <- matrix(0,nrow=3,ncol=5)
> attr(M,"dim")
[1] 3 5
```

R uses a `names` attribute to hold the names of elements in a list:

```
> L <- list (abc=9, def=10, xyz="ha")
> attr(L,"names")
[1] "abc" "def" "xyz"
```

The Class Attribute

The special `class` attribute tells R that some operations on the object should be done in a special way. We'll cover more about how this works later — and about how it can be used to program in a style known as 'object-oriented programming".

For the moment, here's a brief illustration of what can be done:

```
> g <- 123
> attr(g,"class") <- "gobbler"
> print.gobbler <- function (what) {
+     cat ("I'm a gobbler with value", unclass(what), "\n")
+ }
> g
I'm a gobbler with value 123
> g+1000
I'm a gobbler with value 1123
```

We've used the `class` attribute to tell R that objects in our "gobbler" class should be printed in a different way than ordinary numbers. Note that `unclass` gets rid of the class attribute, which lets us handle the number inside a gobbler object in the usual way (though using `unclass` is not strictly necessary here).

Data Frames

One major use of classes is for R's `data.frame` objects, which are the most common way that data is represented in R.

A data frame is sort of like a list and sort of like a matrix. Each "row" of a data frame holds information on some individual, object, case, or whatever. The "columns" of a data frame correspond to variables whose values have been measured for each case. These variables can be numbers, logical (`TRUE/FALSE`) values, or character strings (but all values for one variable have the same type).

For example, here's how R prints a small data frame containing the heights and weights of three people:

```
> heights_and_weights
  name height weight
1 Fred     62    144
2 Mary     60    131
3  Joe     71    182
```

A data frame is really a list, with named elements that are the columns of the data frame, but with a `data.frame` class attribute that makes R do things like printing and subscripting differently from an ordinary list.

Getting Data Out of a Data Frame

You can get data from a data frame using subscripting operations similar to those for a matrix (by row and column index), or by operations similar to a list (using names of variables). For example:

```
> heights_and_weights           # The data frame from the last slide
  name height weight
1 Fred     62    144
2 Mary     60    131
3  Joe     71    182
> heights_and_weights$height    # All values of the "height" variable
[1] 62 60 71
> heights_and_weights[2,]       # All values for the 2nd person
  name height weight
2 Mary     60    131
> heights_and_weights[2,3]      # Value of 3rd variable for 2nd person
[1] 131
> heights_and_weights$weight[2] # ... and the same, by variable name
[1] 131
```

Creating a Data Frame

Using `as.data.frame`, you can create a data frame from a list (it just adds the `data.frame` class attribute) or from a matrix (it has to split it up into columns). If you don't provide variable names, R uses `V1`, `V2`, etc.

Examples:

```
> as.data.frame (list (abc=c(1,3,2),
+                      pqr=c(TRUE,FALSE,FALSE),
+                      xyz=c("a","bb","c")))
  abc   pqr xyz
1   1  TRUE   a
2   3 FALSE  bb
3   2 FALSE   c
>
> as.data.frame (matrix (1:12, nrow=3, ncol=4))
  V1 V2 V3 V4
1  1  4  7 10
2  2  5  8 11
3  3  6  9 12
```

Reading Data Into a Data Frame

The `read.table` function creates a data frame using data it reads from a text file.

The file has to contain one line for each row of the data frame, containing a value (eg, a number, `TRUE/FALSE`, a string) for each variable for the case corresponding to that row.

If a `header=TRUE` argument is given to `read.table`, the names of the variables will be taken from the first line of the file.

Here's how we could read the heights and weights data frame from a file on the course web page:

```
heights_and_weights <-
  read.table ("http://www.cs.utoronto.ca/~radford/csc120/data7",
              header=TRUE)
```

The contents of the file read are as below:

```
name height weight
Fred 62 144
Mary 60 131
Joe 71 182
```