4.3: Managing the Workspace

Let's suppose that you're reading through this book, and what you're doing is sitting down with it once a week and working through a whole chapter in each sitting. Not only that, you've been following my advice and typing in all these commands into R. So far during this chapter, you'd have typed quite a few commands, although the only ones that actually involved creating variables were the ones you typed during Section 4.1. As a result, you currently have three variables; `seeker` , `lover` , and `keeper` . These three variables are the contents of your ***workspace***, also referred to as the **global environment**. The workspace is a key concept in R, so in this section we'll talk a lot about what it is and how to manage its contents.

4.3.1 Listing the contents of the workspace

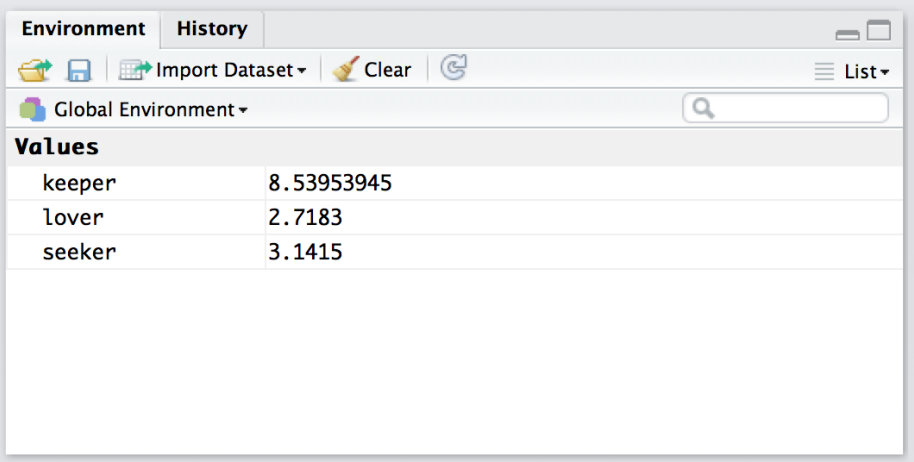

Figure 4.5: The Rstudio Environment panel shows you the contents of the workspace. The view shown above is the list view. To switch to the grid view, click on the menu item on the top right that currently reads list. Select grid from the dropdown menu, and then it will switch to a view like the one shown in the other workspace figure

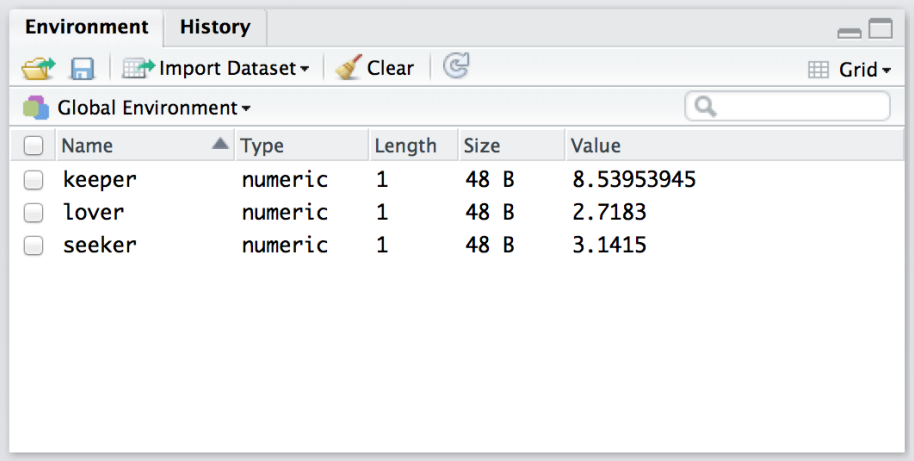

Figure 4.6: The Rstudio "Environment" panel shows you the contents of the workspace. Compare this "grid" view to the "list" earlier

The first thing that you need to know how to do is examine the contents of the workspace. If you're using Rstudio, you will probably find that the easiest way to do this is to use the "Environment" panel in the top right hand corner. Click on that, and you'll see a list that looks very much like the one shown in Figures 4.5 and 4.6. If you're using the commmand line, then the `objects()` function may come in handy:

```
objects()
```

```
## [1] "keeper" "lover"  "seeker"
```

Of course, in the true R tradition, the `objects()` function has a lot of fancy capabilities that I'm glossing over in this example. Moreover there are also several other functions that you can use, including `ls()` which is pretty much identical to `objects()`, and `ls.str()` which you can use to get a fairly detailed description of all the variables in the workspace. In fact, the `lsr` package actually includes its own function that you can use for this purpose, called `who()`. The reason for using the `who()` function is pretty straightforward: in my everyday work I find that the output produced by the `objects()` command isn't *quite* informative enough, because the only thing it prints out is the name of each variable; but the `ls.str()` function is *too* informative, because it prints out a lot of additional information that I really don't like to look at. The `who()` function is a compromise between the two. First, now that we've got the `lsr` package installed, we need to load it:

```
library(lsr)
```

```
## Warning: package 'lsr' was built under R version 3.5.2
```

and now we can use the `who()` function:

```
who()
```

```
##    -Name -   -Class -   -Size --
##    keeper       numeric       1
##    lover        numeric       1
##    seeker       numeric       1
```

As you can see, the `who()` function lists all the variables and provides some basic information about what kind of variable each one is and how many elements it contains. Personally, I find this output much easier more useful than the very compact output of the `objects()` function, but less overwhelming than the extremely verbose `ls.str()` function. Throughout this book you'll see me using the `who()` function a lot. You don't have to use it yourself: in fact, I suspect you'll find it easier to look at the Rstudio environment panel. But for the purposes of writing a textbook I found it handy to have a nice text based description: otherwise there would be about another 100 or so screenshots added to the book.[48]

4.3.2 Removing variables from the workspace

Looking over that list of variables, it occurs to me that I really don't need them any more. I created them originally just to make a point, but they don't serve any useful purpose anymore, and now I want to get rid of them. I'll show you how to do this, but first I want to warn you – there's no "undo" option for variable removal. Once a variable is removed, it's gone forever unless you save it to disk. I'll show you how to do *that* in Section 4.5, but quite clearly we have no need for these variables at all, so we can safely get rid of them.

In Rstudio, the easiest way to remove variables is to use the environment panel. Assuming that you're in grid view (i.e., Figure 4.6), check the boxes next to the variables that you want to delete, then click on the "Clear" button at the top of the panel. When you do this, Rstudio will show a dialog box asking you to confirm that you really do want to delete the variables. It's always worth checking that you really do, because as Rstudio is at pains to point out, you can't undo this. Once a variable is deleted, it's gone.[49] In any case, if you click "yes", that variable will disappear from the workspace: it will no longer appear in the environment panel, and it won't show up when you use the `who()` command.

Suppose you don't access to Rstudio, and you still want to remove variables. This is where the ***remove*** function `rm()` comes in handy. The simplest way to use `rm()` is just to type in a (comma separated) list of all the variables you want to remove. Let's say I want to get rid of `seeker` and `lover`, but I would like to keep `keeper`. To do this, all I have to do is type:

```
rm( seeker, lover )
```

There's no visible output, but if I now inspect the workspace

```
who()
```

```
##    -Name -   -Class -   -Size --
##    keeper        numeric       1
```

I see that there's only the `keeper` variable left. As you can see, `rm()` can be very handy for keeping the workspace tidy.