R: Basics

Signal Data Science

This presents the basics of programming in R.

Some of the information on this page might not be immediately relevant, but you should at least skim all the content regardless just to know that it exists. When the need arises, you'll know where to look.

Installing R

We recommend using RStudio, a very full-featured IDE for programming in R—you can download the binaries on its website if you run Windows, OS X, Ubuntu/Debian, or Fedora/Red Hat/openSUSE. Since RStudio is open source, unofficial binaries exist for other Linux distributions too, like Arch Linux.

Running code in RStudio

By default, RStudio has four collapsible panels: code files (top left), environment information (bottom left), console (top right), and documentation/plots (bottom right). I like to collapse the bottom left panel with the _ icon in the top right so I have more space to see my code.

It's possible to type code directly into the console and to press `Enter` to evaluate a line of code. However, it's usually best to save all of your work in a R script file.

- Create a new script file called `test.R` in any directory of your choice. Type some arithmetic into it like `1+1` and `2*4^2` on different lines and click the `Run` button in the top right.

You can press `Ctrl+Enter` to run just the line that your cursor is on or the code block that you have selected. When your scripts get large and cumbersome, rerunning just the parts that changed will be much faster than rerunning the entire file (which could take a very long time depending on what computations you're doing).

Unless your script is very short, you should *almost never* be rerunning the entire file every time you make a change to your code. If you are, it might be a sign that your code needs to be structured differently.

Assignment operators

Let's go over variable assignment. In R, you can assign values to variables with both the `=` and the `<-` operators, like so:

```
> x = 100
> x
[1] 100
> x <- 100
> x
[1] 100
```

There is a good StackOverflow discussion of the differences between the two operators. The main points are:

1. The `<-` operator is generally preferred in the R community for "compatibility with (very) old versions of S-Plus".
2. But the `<-` operator is whitespace dependent, because the syntax is equivalent to the comparison `x < -y` (`x` is less than negative `y`). The `=` operator doesn't have this problem.
3. The scoping of the two assignment operators is different. Here, `=` has more intuitive behavior: if you evaluate, say, `median(x=1:10)`, it doesn't change the value of `x` in the outside environment, but if you evaluate `median(x<-1:10)`, the value of `x` outside the function call is also set to `1:10`.
4. **Bottom line:** Just use `=` instead of `<-`. It's faster, clearer, and there are almost no downsides, ever.

Installing packages

To install a package, run `install.packages('packagename')`. If it doesn't work, check that you spelled the package name correctly!

Now, to load a package, run `library('packagename')`. You'll find that in other people's R code, they might use `library(packagename)` or `require('packagename')`, which are both inferior! Yihui Xie has a post about why `library('packagename')` is *almost always better*.

If your installation or compilation of R packages is taking an inordinately long time and you're on a Unix/Unix-like system, you can speed up the compilation process by changing your compilation options.

Changing the current directory

Sometimes, you'll want to change the current (working) directory. For example, suppose that you're running `/tmp/scripts/analysis.R` which depends on some functions defined in `/tmp/scripts/functions.R` and a data file called `/tmp/scripts/data.csv`. Now, when loading the files, you can type out the full paths, but you might want to be more concise and portable by just calling `functions.R` and `data.csv`. If you do that, R has to know where to look for those files; if your current directory is set to, say, `/home/andrew/`, then it'll look for `/home/andrew/functions.R`, which doesn't exist. What you would want to do in this case is to set the current directory to `/tmp/scripts/`.

You can use the `getwd()` (**get w**orking **d**irectory) command to see what the current directory is set to. You can also use the `setwd()` (**set w**orking **d**irectory) function to set the working directory to whatever path you pass in to the function.