

## 9.1: How to make your first R script

Script is a core tool for reproducible, evaluable data analysis. Every R user must know *how to make scripts*.

This is a short instruction for unexperienced user:

1. Save your history of commands, just in case.
2. Then copy-paste all necessary commands from your R console into the text editor (e.g., open blank file in R editor with `file.edit()` command<sup>[1]</sup>).

Notes:

- (a) *Interactive commands* which need user input, like `help()`, `identify()`, `install.packages()`, or `url.show()` should *not* go into the script.
  - (b) All *plotting commands* should be within `pdf(...)` / `dev.off()` or similar.
  - (c) It is also a good idea to place your package/script *loading commands* first, then your data loading commands like `read.table()` and finally actual calculations and plotting.
  - (d) To add the single function, you may (1) *type* function name without parentheses, (2) *copy-paste* function name and output into the script and (3) after the name of function, *insert* assignment operator.
  - (e) Try to *optimize* your script (Figure 9.1.1), e.g., to remove all unnecessary commands. For example, pay attention to those which do not assign or plot anything. Some of them, however, might be useful to show your results on the screen.
  - (f) To learn how to write your scripts better, read style guides, e.g., Google's R Style Guide on [google.github.io/styleguide/...xml](https://google.github.io/styleguide/...xml)<sup>[2]</sup>.
3. Save your script. We recommend `.r` extension, there are also other opinions like `.R` or `.Rscript`. Anyway, please do not forget to tell your OS to *show file extensions*, this could be really important.
  4. Close R, *do not save workspace*.
  5. Make a `test` directory inside your working directory, or (if it already exists) *delete* it (with all contents) and then *make again* from scratch.
  6. *Copy* your script into `test` directory. Note that the master version of script (were you will insert changes) should stay outside of `test` directory.
  7. Start R, make `test` the working directory.
  8. Run your script from within R via `source(script_name.r, echo=TRUE)`.

Note that: (a) R runs your script *two times*, first it checks for errors, second performs commands; and (b) all warnings will concentrate at the end of output (so please do not worry).

It is really important to check your script exactly as described above, because in this case commands and objects saved in a previous session will not interfere with your script commands. Alternatively you can use non-interactive way with `Rresults` shell script (see below).

If everything is well (please check especially if all plot files exist and open correctly in your independent viewer), then script is ready.

If not, open script in the editor and try to find a mistake (see below), then correct, close R, re-create (delete old and make new) `test` directory and repeat.

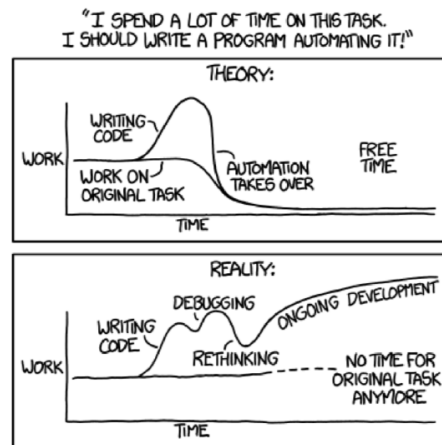


Figure 9.1.1 Automation (taken from XKCD, <http://xkcd.com/1319/>).

When your script is ready, you may use it as the most convenient way to protocol and even to report your work. The most important is that your script is self-sufficient, downloads all data, loads all packages and makes all plots itself.

Actually, this book is the one giant R script. When I run it, all R plots are re-created. This is the first plus: the exact correspondence between code and plots. Second plus is that all code is checked with R, and if there is a mistake, it will simply stop. I do not control textual output because I want to modify it, e.g., to make it fit better with the text.

. Can you find it?

## References

1. Linux users might want to add option editor=.
2. Package lintr contains lint() command which checks R scripts.

This page titled [9.1: How to make your first R script](#) is shared under a [Public Domain](#) license and was authored, remixed, and/or curated by [Alexey Shipunov](#) via [source content](#) that was edited to the style and standards of the LibreTexts platform.