

## 9.2: My R script does not work!

What if your script does not work?

Most likely, there is some message (which you probably do not understand) but outputs nothing or something inappropriate. You will need to *debug* your script!

- First is to find *where exactly* your script fails. If you run `source()` command with `echo=TRUE` option, this is possible just by looking into output. If this is still not clear, *run the script piece by piece*: open it in any simple text editor and copy-paste pieces of the script from the beginning to the end into the R window.
  - Above mentioned is related with one of the most important principles of debugging: minimize your code as much as possible, and find the *minimal example which still does not work*. It is likely that you will see the mistake after minimization. If not, that minimal example will be appropriate to post somewhere with a question.
  - Related with the above is that if you want to ask somebody else about your R problem, make not only minimal, but *minimal self-contained* example. If your script loads some data, attach it to your question, or use some *embedded* R data (like `trees` or `iris`), or *generate* data with `sample()`, `runif()`, `seq()`, `rep()`, `rnorm()` or other command. Even R experts are unable to answer questions without data.
  - Back to the script. In R, many expressions are “Russian dolls” so to understand how they work (or why they do not work), you will need to take them to pieces, “undress”, removing parentheses from the outside and going deeper to the core of expression like:
- To make smaller script, do not remove pieces forever. Use *commenting* instead, both one-line and multi-line. The last is not defined in R directly but one can use:
    - If your problem is likely within the large function, especially within the cycle, use some way to “look inside”. For example, with `print()`:
    - The most common problems are mismatched parentheses or square brackets, and missing commas. Using a text editor with syntax highlighting can eliminate many of these problems. One of useful precautions is always count open and close brackets. These counts should be equal.
    - Scripts or command sets downloaded from Internet could suffer from automatic tools which, for example, convert *quotes* into quotes-like (but not readable in R) symbols. The only solution is to carefully replace them with the correct R quotes. By the way, this is another reason why not to use office document editors for R.
    - Sometimes, your script does not work because *your data changed* and now conflicts with your script. This should not happen if your script was made using “paranoid mode”, commands which are generally safe for all kinds of data, like `mat.or.vec()` which makes vector if only one column is specified, and matrix otherwise. Another useful “paranoid” custom is to make checks like `if(is.matrix) { ... }` everywhere. These precautions allow to avoid situations when you updated data start to be of another type, for example, you had in the past one column, and now you have two. Of course, something always should be left to chance, but this means that you should be ready to conflicts of this sort.
    - Sometimes, script does not work because there were *changes* in R. For example, in the beginning of its history, R used underscore (`_`) for the left assignment, together with `<-`. The story is when S language was in development, on some keyboards underscore was located where on other keyboards there was *left arrow* (as one symbol). These two assignment operators were inherited in R. Later, R team decided to get rid of underscore as an assignment. Therefore, older scripts might not work in newer R. Another, more recent example was to change clustering `method="ward"` to `method="ward.D"`. This was because initial implementation of Ward’s method worked well but did not reflect the original description. Consequently, in older versions of R newer scripts might stop to work. Fortunately, in R cases like first (broken *backward compatibility*) or second (broken *forward compatibility*) are rare. They are more frequent in R packages though.
  - If you downloaded the script and do not understand what it is doing, use minimization and other principles explained above. But even more important is to *play* with a script, change options, change order of commands, feed it with different data, and so on. Remember that (almost) everything what is made by one human could be deciphered by another one.

This page titled 9.2: My R script does not work! 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.