

## 9.4: Good, Bad, and Not-too-bad

This last section is even more practical. Let us discuss several R scripts.

### Good

This is an example of (almost) ideal R script:

Its main features:

- clearly separated parts: loading of external material (lines 1–12) and processing of the data itself (lines 13–26)
- package(s) first (line 3), then custom functions (line 5–7), then data (line 9)
- data is checked (lines 16–18) with `str()` and then checked for normality
- after checks, data was plotted first (lines 21–23), then analyzed (line 26)
- acceptable style
- every step is commented

To see how it works, change working directory to where script is located, then load this script into R with:

Another variant is non-interactive and therefore faster and cleaner. Use [Rresults](#) script (works on macOS and Linux) like:

### Bad

Now consider the following script:

It is really bad, it simply does not work. Problems start on the first line, and both interactive (with `source()`) and non-interactive (with [Rresults](#)) ways will show it like:

Something is really wrong and you will need to find and correct (debug) it. And since code was not commented, you have to guess what author(s) actually wanted.

Other negative features:

- no parts, no proper order of loading, checking and plotting
- interactive `url.show()` will block non-interactive applications and therefore is potentially harmful (not a mistake though)
- bad style: in particular, no spaces around assignments and no spaces after commas
- very long object names (hard to type)

Debugging process will consist of multiple tries until we make the working (preferably in the sensible way), “not-too-bad” script. This could be prettified later, most important is to make it work.

There are many ways to debug. For example, you can open (1) R in the terminal, (2) text editor<sup>[2]</sup> with your script and probably also some advanced (3) file manager. Run the script first to see the problem. Then copy-paste from R to editor and back again.

Let us go to the first line problem first. Message is cryptic, but likely this is some conflict between `read.table()` and the actual data. Therefore, you need to look on data and if you do, you will find that data contains both spaces and tabs. This is why R was confused. You should tell it to use tabs:

First line starts to work. This way, step by step, you will come to the next stage.

### Not too bad

This is result of debugging. It is not yet fully prettified, there are no chapters and comments. However, it works and likely in the way implied by authors.

What was changed

- custom commands moved up to line 3 (not to the proper place, better would be line 1, but this position guarantees work)
- `url.show()` commented out
- checks added (lines 5–6)
- names shortened a bit and style improved (not very important but useful)
- plotting now plots to file, not just to screen device
- object `willows` appeared out of nowhere, therefore we had to guess what is it, why it was used, and then somehow recreate it (lines 8–9)

object but it is not the same as in initial script. What is different? Is it possible to make them the same?

## References

1. There is, by the way, a life-hack for lazy reader: all plots which you need to make yourself are actually present in the output PDF file.
2. Among text editors, Geany is one of the most universal, fast, free and works on most operation sys- tems.

---

This page titled 9.4: Good, Bad, and Not-too-bad 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.