Table of Contents

How to prepare reproducible and portable R script

These are simple instructions on how to prepare a reproducible R script, the one that you can share with your colleagues/friends/teachers in a way which will make it easy to co-operate. You can find many similar suggestions online, so take this just as my personal recommendation and google further to see more advanced suggestions.

Reproducible script is the one which you can open, run, and get the same result as you got when you created it. Reproducible *and portable* is a script which also the others can open, run and get the same result as you did, with no (or minimal) need for interference with the code itself. Remember that script could be fully reproducible on your computer (with your structure of file folders, your R version, and all libraries already installed), but it may not run on computers of others. If you make the script just for yourself, it's fine (you are likely to open it on the same computer with the same files stored on the same file paths, so it should run; just make sure that your *future you* will know what the script is doing). But if you want to share the script with others, you need to make sure that it is also *portable*, i.e. the others can run it on their computer. If your script starts with `setwd ('c:\\Users\\Me\\This\\Folder\\Only\\I\\Have')`, than it is most likely not portable.

Below are a few issues you may need to pay attention to when producing *reproducible and portable* R script.

Quick summary:

- Call all needed libraries (e.g. `library (myLovelyPackage)`), but do not include installation script (or comment it, e.g. `#install.packages ('myLovelyPackage')`) - check here what is the difference between `install.packages` and `library`.
- Use annotations (comments, starts with #), but not necessarily for every trivial action.
- Do not start your script by `setwd` referring to local directories on your computer (e.g. by `setwd ('c:\\Users\\Me\\This\\Folder\\Only\\I\\Have')`); this makes it not portable (probability that someone else has the same directory structure is close to 0%). If you run the code on your computer, but also share it with others, you can include `setwd` to your folder structure, but again comment it (by adding # at the beginning of that line).
- Think twice how to get data into the script. The worst option is `myData <- read.delim ('clipboard')` - you don't disclose what was on the clipboard when you run the script, and next time you may copy there something else. Better is `myData <- read.delim ('c:\\Users\\Me\\This\\Folder\\Only\\I\\Have\\myData.txt')`, which is reproducible on your computer, but not portable to other computers. To make script portable, best is to load data from somewhere (dataset in R, Dropbox, Gist on Github), or directly include data into the script (e.g. using `dput` function). If using Dropbox, do not forget to change the end of the link from dl=0 into dl=1 (otherwise you load nonsense, see below why). Always safe is to include data as a file together with the script (perhaps zipped together into an archive), but make sure that the script contains the part telling the user how to load the data from that file;
- make the script clean and tidy, so as to *the future you* and your colleagues/friends/teachers can navigate in it without too much hassle (see here what clean and tidy is).

Some terminology first

- *comments* or *annotations* - are starting with the hashtag #; what is written on the line after # is ignored by R.
- *to comment* something means to add the hashtag in front of the function to deactivate it from running automatically - if needed, this function or the whole line could be *uncommented*, i.e. the hashtag removed to make it active.
- *to source the file* - to apply the function `source` on your file instead of running it *line-by-line* via the command line. In RStudio, this can also be done by pressing the *Source* button in the head of the upper-left panel (when the file you want to source is open).

File format

Preferably *.r file, which can be directly opened in R or RStudio and sent to command line. For more advanced users, consider using a commented script with comments written in R Markdown and saved into *.rmd format (make sure, however, that the file can also *knit* on other computers, or include a pdf version of the file for sure).

Annotations

Use plenty of annotations, but at the same time be concise - do not comment on EVERYTHING, just things which are not immediately apparent or which require the attention of the user. It is good to start the script with a title and your name, to make it clear who is the author of the script (also include an email address in case you hope for feedback). If the script produces some files (e.g. figure files), it may be good to mention this in the script header, so as users know what to expect.

Data import

In most cases you are using some data on which you do editing/plotting/analysis or whatever else. There are several options how to share data, depending how large are the data and whether they can be made public or need to remain private:

- **Short data** (e.g. if you script is done to illustrate some function/problem and the dataset contains few elements), you may include directly into the script. Example“ `myData <- data.frame (spe = c(2,45,2), ele = c(345, 456, 678))`. Alternatively, use the function `dput` to store R objects in text representation which can be resourced into R. Example: `dput (myData)` will return `structure(list(spe = c(2, 45, 2), ele = c(345, 456, 678)), .Names = c(“spe”, “ele”), row.names = c(NA, -3L), class = “data.frame”)`, and you can use this to create dataset `myData_2 <- structure(list(spe = c(2, 45, 2), ele = c(345, 456, 678)), .Names = c(“spe”, “ele”), row.names = c(NA, -3L), class = “data.frame”)`.
- **Larger data** which can be **public** (e.g. data which are so as so publicly available and does not carry any copyright) - consider storing them in some easy to access online repository, such as gist on GitHub (see fast tutorial how to do it here; you need to create an account - a free version can store many public Gists) or https://pastebin.com/. Store data in (preferable) *.txt format

(separated by tabulator) and read it into R using `read.delim` or `readr::read_delim` functions.
- **Larger data** which should be **private** (e.g. your experimental data or data which are copyrighted and should not be publicly available) - consider using Dropbox or other cloud services (Drive) which allow sharing links. Again, store data preferably as a *.txt file with cells separated by tabulator, and read them into R using `read.delim` or `readr::read_delim` functions. **If you use the Dropbox share functionality, you need to replace `dl=0` at the end of the link by `dl=1`** (this triggers the direct download of the file instead of opening it in the website browser). Example: `temp <- read.delim (file = 'https://www.dropbox.com/s/8svpdyq959a8bx6/temperature.txt?dl=1')` (note that there is `dl=1` at the end of the dropbox link).
- **Any data** - include the file with the R code and make sure that the R script contains the command to read them from the folder and save into the right object. This is always the option, although it means that the one who opens your script needs to change the directory (`setwd`) in the same directory where she/he saved your data. This may need to be mentioned in the R code. Do not include files of uncommon formats, which may require installation of special software; e.g. reading Excel file is not recommended, since most R libraries doing this job require installation of third-party software on users computer (like Perl), and not everybody is happy/willing/able to do that. Preferably store data as plain text and as *.txt file with cells separated by tabs.

Body of the R script

Few suggestions for good practice:

- Include the call of any library you used to produce your code (`library (thatPackage)`), but do not include `install.packages ("thatPackage")` command - and if yes, please comment it (by adding # at the beginning of the command). You don't want the other always to reinstall all the libraries which may be already installed and running well on their computer.
- Do not include any R function which is not directly related to what the script should do. It is common that when you e.g. import data into R, you have a look on them using `View` or `fix` or `edit` or `head` or `str` commands. But these commands do not need to be included in the reproducible script you are going to share with the others, and if yes, deactivate them by *commenting them*.
- Make the script tidy and clean (see here to know what I mean by that).