How to prepare a reproducible and portable R script

These are simple instructions for preparing a reproducible R script you can share with colleagues/friends/teachers in a way that will make it easy to collaborate on.

A reproducible script is one you can open and run to get the same result you got when you created it. Reproducible //and portable is a script which others can also open, run, and get the same result as you did, with no (or minimal) need for interference with the code. Remember that a 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 others' computers. 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, make sure it is also *portable*, i.e., 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 a *reproducible and portable* R script.

Quick summary:

- Call all needed libraries (e.g. `library (myLovelyPackage)`), but do not include an installation script (or comment it, e.g. `#install.packages ('myLovelyPackage')`).
- 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 definitely not portable. 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 about 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 something else there. 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 the script portable, the best is to load data from somewhere (a dataset in R, Dropbox, a Gist on GitHub), or directly include data in the script (e.g., using the `dput` function). If using Dropbox, do not forget to change the end of the link from dl=0 to 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 #; R ignores everything what is written on the line after #;
- *to comment the code* - to add a hashtag in front of the code to deactivate it (it can be later *uncommented* again by removing the hashtag);
- *to source the file* - to apply the function `source` on your R script 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 code panel (when the file you want to source is open).

File format

Preferably a *.r file, which can be directly opened in R or RStudio and sent to the command line. For more advanced users, consider using a commented script with comments written in R Markdown and saved into *.rmd format (ensure 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 don't comment on everything, just things that aren't immediately apparent or require the user's attention. Start the script with a header including a brief title and your name (also include an email address if you want feedback). If the script produces some files (e.g., figure files), it may be good to mention this in the script header, so that 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 for how to share data, depending on how large the data are and whether they can be made public or need to remain private:

- **Short data** (e.g. if your script is done to illustrate some function/problem and the dataset contains few elements), you may include it 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 a text representation that can be sourced back 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 publicly available and do not carry any copyright) - you may store them in some easy-to-access online repository, such as a gist on GitHub (see fast tutorial on how to do it here; you need to create an account - a free version can store many public Gists). Store data in (preferably) *.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 some cloud service which allows sharing links (Dropbox, Drive). Again, store data preferably as a *.txt file with cells separated by tabs, 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 with `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 them into the right object. This is always an option, although it means the person who opens your script needs to change the directory (`setwd`) to the same directory where s/he saved your data. This may need to be mentioned in the R code. Do not include files in uncommon formats that may require special software; e.g., reading Excel files is not recommended, since it is a commercial format and can change anytime. Preferably store data as plain text in a *.txt file with cells separated by tabs.

Body of the R script

A 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 others to reinstall libraries that may already be installed and working on their computer.
- Do not include any R function that is not directly related to what the script should do (e.g., `View` or `fix` or `edit` or `head` or `str` commands).
- Make the script tidy and clean (see here to know what we mean by that).