Automating script start-up

How do you open up R to start working on a script file?

-

Efficient method to accomplish this
Why - What - How - Pos/Neg

Why?

Best practices of writing in R

1) Describe your code – When you start coding describe what the R code does in the very first line. For subsequent blocks of codes follow the same method of describing the blocks. This makes it easy for other people to understand and use the code.

Example:

```
# This code captures the 52-week high effect in stocks
# Code developed by Milind Paradkar
```

2) Load Packages – After describing your code in the first line, use the library function to list and load all the relevant packages needed to execute your code.

Example:

```
library(quantmod);  library(zoo); library(xts);
library(PerformanceAnalytics); library(timeSeries); library(lubridate);
```

https://www.quantinsti.com/blog/r-best-practices-r-you-writing-the-r-way

Why?

Best practices of writing in R

1) Describe your code – When you start coding describe what the R code does in the very first line. For subsequent blocks of codes follow the same method of describing the blocks. This makes it easy for other people to understand and use the code.

Example:

```
# This code captures the 52-week high effect in stocks
# Code developed by Milind Paradkar
```

2) Load Packages – After describing your code in the first line, use the library function to list and load all the relevant packages needed to execute your code.

Example:

```
library(quantmod);  library(zoo); library(xts);
library(PerformanceAnalytics); library(timeSeries); library(lubridate);
```

https://www.quantinsti.com/blog/r-best-practices-r-you-writing-the-r-way

Why?

Structuring a script:

- Always load packages at the beginning; if it's fairly self-evident why the package is needed, at least to me, I just load and move on. If it's a specialty or convenience package, then I remind myself what function(s) I'm going to use; this way, if I don't have the package down the road, I can get a sense of how hard it's going to be to fix up and run the code.

```
library(car)              # recode()
```

https://www.stat.ubc.ca/~jenny/STAT545A/block19_codeFormattingOrganization.html

Why?

General Layout and Ordering

If everyone uses the same general ordering, we'll be able to read and understand each other's scripts faster and more easily.

1. Author comment
2. File description comment, including purpose of program, inputs, and outputs
3. `source()` and `library()` statements
4. Function definitions
5. Executed statements, if applicable (e.g., `print`, `plot`)

https://msu.edu/~idworkin/ZOL851_style_guide.html

Why?

Be explicit about the requirements and dependencies of your code

Loading all of the packages that will be necessary to run your code (using `library`) is a nice way of indicating which packages are necessary to run your code. It can be frustrating to make it two-thirds of the way through a long-running script only to find out that a dependency hasn't been installed.

```
library(ggplot2)
library(reshape)
library(vegan)
```

https://swcarpentry.github.io/r-novice-inflammation/06-best-practices-R/

Challenge

- Capitalize on that structure
- Automation
- Without hiding essential information to maintain reproducibility

One solution: generic but specific .Rprofile

https://github.com/karl-cottenie/kc-.Rprofile

One more thing: snippets

Advantages/disadvantages

- +
  - Automatic
  - Reproducible
  - Generic
  - Cool/fun
- -
  - Some overhead (but only once)
  - Startup is longer (but you can adjust number of files/lines read)
  - If lots of .R files, startup longer (but you can adjust the profile if you want)

Efficient method to accomplish this
Why - What - How - Pos/Neg

How many want to try and install this?