R Coding Best Practices

Florian M. Hollenbach

2/22/2019

Some best practices for writing r-code

(To Be Continuously Updated)

No use of `rm(list = ls())` at beginning of files

- Do not empty the workspace with `rm()`.
- `rm()` keeps many previous settings .
- For example, all the packages that were already loaded remain or options like `options(stringsAsFactors = FALSE)` remain set.
- *Instead*: always start with a fresh r-session.

No use of `setwd()`

- Instead: *use the here package*, see: https://github.com/jennybc/here_here
- If you use r-studio, create an R-project for the project in the main folder.
- When loading `library(here)` within the project in rstudio, the main folder will become the basis from which to declare any paths.
- If you use other editors, you can create a .here file in the main folder.
- Use `here()` to declare paths from main folder.
- This means no more changing of paths, etc.
- The code will run on different machines or at different points in time. Even if we rename the main folder at some point or move it to a different location.

No massive package loading

- Instead: *only load those packages that are actually used in the file.*
- Also, pay attention to function conflicts when loading, i.e., which functions are masked when loading a new package.
- For example, if tidyverse is loaded the lag functions within plm, lfe will not work!
- Instead: *use the conflicted package*, see: https://github.com/r-lib/conflicted
- The conflicted package will alert/throw an error when using any function that has conflict.
- Also, when loading a package that has a conflicted function, immediately after loading the package declare which package to use the function from, e.g.: `conflict_prefer("filter", "dplyr") ### use filter from dplyr`.