The Good Coder Commandments

Léa Dousset[*]

September 2021

Foreword

Always keep in mind that you write code for multiple audiences: your computer, your future self, your colleagues, your research assistants, and people who want to replicate your results.

The following commandments are meant to make coding easier and more efficient. They will help to ensure reproductibility and preserve your sanity.

The Commandments

1. *Comment your code*
   - Explain in comments what your code is doing. This makes your code easier to understand for other people, but also your future self.
2. *Use a clean and consistent coding style*
   - There is not only one way to write code, but pick one that is clean and stick to it. Do not forget to:
     (a) indent
     (b) leave spaces: between blocks of code and between logical statements and computation
     (c) structure your code: you can use block commenting to separate different sections
     (d) keep it short in terms of length and width
     (e) keep it simple: break complicated algebraic calculation into pieces, do one step at the time.
3. *Choose meaningful and consistent names*
   - Good descriptive variable and function names could almost replace comments because they make your code self-documenting.
4. *Automate your code*
   - Reduce the things you do by hand (e.g. copy pasting some statistics, exporting a result) to the bare minimum. Ideally, if someone else than you were to use your code, they should only have to change one line of code: the path to the directory.
5. *Check the accuracy of your code*
   - First and foremost, you write code to your computer. To make sure that you managed to send the proper message to your computer, you should test your code. Two options occur:
     (a) The code is not running. You probably made a mistake in your code and the computer cannot understand you. Use the compiling error messages to fix your code.

[*]Disclaimer: I am not a computer science expert. This note is my attempt to gather useful tips for coding I wish I would have been taught earlier in my economics studies.

   (b) The code runs. You still want to check whether your computer is exactly performing what you want it to do. To do so, you can have a look at your data directly, inspect some descriptive statistics (e.g. use tabulate or summarize functions), or even better, run some tests (e.g. use the assert functions or code some "unit test").

6. *Abstract your code*

   - If you notice that several code blocks are implementing the same abstract idea, then turn them into a general-purpose tool, such as a loop or a function. This will eliminate redundancy of your code, which reduces the scope for error, and could also increase the efficiency of your code.

7. *Have a clear and organized work environment*

   - Your directory structure should be easy to understand and similar across your different projects. This will preserve a clear organization you will always know where things are. The usual structure is the following:
     - Input
       - Raw
       - Temp
       - Clean
     - Code
     - Output
       - Tables
       - Figures
     - Documentation