A Bit of Style

FCA Collin, Ph.D.

Thursday, April 15, 2021

Contents

2021-03-11, FCA Collin

Style Advises

Use Comment Appropriately

Main reason to comment:

- Code Readability
- Explanation of the code or Metadata of the project
- Temporarily prevent execution of code
- To include resources

https://style.tidyverse.org/syntax.html (210415):

In data analysis code, use comments to record important findings and analysis decisions. If you need comments to explain what your code is doing, consider rewriting your code to be clearer. If you discover that you have more comments than code, consider switching to R Markdown.

How to comment

https://style.tidyverse.org/functions.html (210415):

- Explain the *why* and not the *what* or *how*
- Comments should be in sentence case, and only end with a full stop if they contain at least two sentences

Down-side of (bad) comment practice.

- Focus and cognitive load: stops screening code because *maybe that is important* (and 99% of case is not).
- Hides what's important: increase the length of the code. If, eventually, it is understood by the programmer who commented it, it is noise to every one else. It is better to keep the code to the necessary piece and as, along with other measures, it will improve code maintenance by preventing to have to look for the needle in the haystack.
  - Alternative: if all this comment is necessary, use literate programming such as rmarkdown.
- The commented code does not evolve and may become out of date: and this is a problem as all the code should work.
- Unfinished job: the comment can be an alternative code as the programmer thinks "*maybe that could be another valid approach*". The programmer is still the better placed to decided what is better, or may ask advises, but the job will be finish when finally ruling the decision. Alternative:
  - if it is hard to decide, take the time to weight the decision, eventually ask another colleague.
  - if the decision must be documented: consider literate programming with rmarkdown.
- Someone else will delete it. As ambiguous, someone may decide to delete the commented code. If this was really informative, that will be a loss which could have been prevented (i.e. literate programming, unit tests). Some programmers think:
  - *I'll Delete Your Commented Code Without Reading It and I'm Not Sorry* https://blog.submain.com/delete-commented-code-without-reading/

Around the Web:

- https://kentcdodds.com/blog/please-dont-commit-commented-out-code
- https://agiletribe.wordpress.com/2015/12/26/never-leave-commented-code-in-the-source/
- https://blog.submain.com/delete-commented-code-without-reading/

Few quotes:

- *The cost of that commented code always outweighs the benefit.*
- *Deleting commented code is an easy and effective way to improve the code base, without any risk of negative consequences.*

```
sessionInfo()
```

```
## R version 4.0.4 (2021-02-15)
## Platform: x86_64-pc-linux-gnu (64-bit)
```

```
## Running under: Debian GNU/Linux 10 (buster)
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/openblas/libblas.so.3
## LAPACK: /usr/lib/x86_64-linux-gnu/libopenblasp-r0.3.5.so
##
## locale:
##  [1] LC_CTYPE=en_GB.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_GB.UTF-8        LC_COLLATE=en_GB.UTF-8
##  [5] LC_MONETARY=en_GB.UTF-8    LC_MESSAGES=en_GB.UTF-8
##  [7] LC_PAPER=en_GB.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_GB.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets
methods   base
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.0.4    magrittr_2.0.1    tools_4.0.4
htmltools_0.5.1.1
##  [5] yaml_2.2.1        stringi_1.5.3     rmarkdown_2.6
knitr_1.31
##  [9] stringr_1.4.0     xfun_0.22         digest_0.6.27
rlang_0.4.10
## [13] evaluate_0.14
```