R Coding - Style Guide

STATS 4380 - Data Science
Villanova University
Spring 2022
Jacob Rozran
January 07, 2022

Preface

This document is created from a RMarkdown document. We'll learn more about these soon. They'll be used extensively in class.

This document is mostly my opinion. It is how I code and how I'd like to see your code written. Being that it is my opinion, I am open to suggestions and it is not the law of the land. But also, I am your professor and you want a good grade from me, so it is probably a good idea to stick to this...

To be honest, I used the tidyverse style guide as my starting point. Much of this is copied/pasted directly from the online book. I got rid of a lot of things that I thought were overkill and modified things that I didn't agree with. So, while this document is wholly my preference, not every idea is original.

File Names

File names should be meaningful and end in .R. Avoid using special characters in file names - stick with numbers, letters, `-`, and `_`.

It is also best practice to stay all lowercase.

Also - note - no spaces. I'm not sure I totally hate spaces, but it is certainly best practice to avoid them. Things get tricky when you try to reference a file with spaces from the command line.

```
# Good names
fit_models.R
utility_functions.R
style-guide.Rmd

# Bad names
fit models.R
foo.r
stuff.r
```

f files should be run in a particular order, prefix them with numbers. If it seems likely you'll have more than 10 files, left pad with zero:

```
00_download.R
01_explore.R
...
09_model.R
10_visualize.R
```

Variable Names

In this class, we prefer snake case to camel case.

Variable and function names should use only lowercase letters, numbers, and `_`.

Variables should be nouns and functions should be verbs. It should also be pretty self explanatory what they are or what they do.

```
# Good
day_one
day_1

# Bad
DayOne
dayone
```

Avoid reusing names of common functions or variables (actually... avoid reusing all variable names in your scripts). Also, make sure that what you assign a thing makes sense for its name.

```
# Bad
T <- FALSE
c <- 10
mean <- function(x) sum(x)
```

Spacing

Always put a space after a comma, never before, just like in regular English.

```
# Good
x[, 1]

# Bad
x[,1]
x[ ,1]
x[ , 1]
```

Do not put spaces inside or outside parentheses for regular function calls.

```
# Good
mean(x, na.rm = TRUE)

# Bad
mean (x, na.rm = TRUE)
mean( x, na.rm = TRUE )
```

Place a space before and after `()` when used with `if`, `for`, or `while`.

```
# Good
if (debug) {
    show(x)
}

# Bad
if(debug){
    show(x)
}
```

Place a space after `()` used for function arguments:

```
# Good
function(x) {}

# Bad
function (x) {}
function(x){}
```

Most operators (`==`, `=`, `+`, `-`, `<-`, etc.) should always be surrounded by spaces:

```
# Good
height <- (feet * 12) + inches
mean(x, na.rm = TRUE)

# Bad
height<-feet*12+inches
mean(x, na.rm=TRUE)
```

One important exception to the space rule are these: `::`, `$`, `[`, `[[`, `^`.

```
# Good
sqrt(x^2 + y^2)
df$z
x <- 1:10

# Bad
sqrt(x ^ 2 + y ^ 2)
df $ z
x <- 1 : 10
```

Code Blocks

Curly braces, `{}`, define the most important hierarchy of R code. To make this hierarchy easy to see:

- `{` should be the last character on the line. Related code (e.g., an if clause, a function declaration, a trailing comma, ...) must be on the same line as the opening brace.
- The contents should be indented by four spaces.
- `}` should be the first character on the line.

```
# Good
if (y < 0 && debug) {
    message("y is negative")
}

if (y == 0) {
    if (x > 0) {
        log(x)
    } else {
        message("x is negative or zero")
    }
} else {
    y^x
}

# Bad
if (y < 0 && debug) {
```

```
 message("y is negative")
}

if (y == 0)
{
    if (x > 0) {
      log(x)
    } else {
  message("x is negative or zero")
    }
} else { y^x }
```

Long Lines

On my screen, I have a line that shows me when my code gets to be 80 characters wide and I try my best to never go past it (sometimes it is hard).

You can add this yourself: `preferences > code > display > show margin > margin column = 80`

Then, while you are coding, you can always add extra returns to keep things within the boundaries.

```
# Good
do_something_very_complicated(
    something = "that",
    requires = many,
    arguments = "some of which may be long"
)

# Bad
do_something_very_complicated("that", requires, many, arguments, "some of which may be long")
```

Semicolons

Don't put `;` at the end of a line, and don't use `;` to put multiple commands on one line.

Assignment

Use `<-`, not `=`, for assignment.

```
# Good
x <- 5

# Bad
x = 5
```

Character vectors

Use `"`, not `'`, for quoting text. The only exception is when the text already contains double quotes and no single quotes.

```
# Good
"Text"
'Text with "quotes"'
```

```
'<a href="http://style.tidyverse.org">A link</a>'

# Bad
'Text'
'Text with "double" and \'single\' quotes'
```

Logical vectors

Prefer `TRUE` and `FALSE` over `T` and `F`.

Comments

Each line of a comment should begin with the comment symbol and a single space: `#`

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.