STATS 4380 - Data Science

Style Guide Presentation

Jake Rozran

Villanova University

September 1, 2022

File Names

File names should be meaningful and end in `.R` (or `.Rmd`)

- Avoid using special characters in file names
- Stick with numbers, letters, `-`, and `_`
- All lowercase letters
- No spaces

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

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

If 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

- Snake case GOOD
- Camel case BAD
- Variable and function names should use only lowercase letters, numbers, and _
- Variables should be nouns and functions should be verbs
    - It should also be self explanatory what they are or what they do

```
# Good
day_one
day_1

# Bad
Day0ne
dayone
```

- Avoid reusing names of common functions or variables
  - actually... avoid reusing all variable names in your scripts
- 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.

```r
# 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.

- `{` should be the last character on the line
- The contents should be indented by four spaces
- `}` should be the first character on the line

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

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

Long Lines

- Code should [almost] never go past 80 characters
- You can add a line that tells you how wide 80 characters is
  - You can add this yourself: `preferences > code > display > show margin > margin column = 80`
- 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", requ
```

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.

```r
# 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