Coding Standards

Biostat 140.776

Coding Standards

- Enhance readability
- Allow for interoperability, inclusion in other programs
- Sometimes required by the language itself
- Readability
- Readability

Coding Standards

- Program code files should **always** be ASCII text files. You should be able to source() the file into R directly without any conversion.
- Always use a monospace font when displaying code; others alter structure of code and limit readability
- Indent your code; use large enough indent so that old people like me can see it; I use 8 spaces
- Limit to 80 columns (or so) in width; this is important!
- One page per function; otherwise split

Linux Kernel Coding Style

Rationale: The whole idea behind indentation is to clearly define where a block of control starts and ends. Especially when you've been looking at your screen for 20 straight hours, you'll find it a lot easier to see how the indentation works if you have large indentations.

--Linus Torvalds

https://www.kernel.org/doc/Documentation/CodingStyle

Linux Kernel Coding Style

“Now, some people will claim that having 8-character indentations makes the code move too far to the right, and makes it hard to read on a 80-character terminal screen. The answer to that is that **if you need more than 3 levels of indentation, you're screwed anyway, and should fix your program**.

“In short, 8-char indents make things easier to read, and have the added benefit of warning you when you're nesting your functions too deep. Heed that warning.”

--Linus Torvalds

https://www.kernel.org/doc/Documentation/CodingStyle

Indenting

```
nalines <- function(x, y, NAcol = gray(0.6), ...) {
use <- complete.cases(x, y)
idx <- which(use)
n <- length(idx)

if(n < 2)
return(invisible())
for(i in seq_len(n - 1)) {
j <- idx[i]
k <- idx[i+1]
col <- if((k - j) > 1)
NAcol
else
"black"
lines(c(x[j], x[k]), c(y[j], y[k]), col = col)
}
invisible()
}
```

Indenting

```
nalines <- function(x, y, NAcol = gray(0.6), ...) {
  use <- complete.cases(x, y)
  idx <- which(use)
  n <- length(idx)

  if(n < 2)
    return(invisible())
  for(i in seq_len(n - 1)) {
    j <- idx[i]
    k <- idx[i+1]
    col <- if((k - j) > 1)
      NAcol
    else
      "black"
    lines(c(x[j], x[k]), c(y[j], y[k]), col = col)
  }
  invisible()
}
```

Indenting

```
nalines <- function(x, y, NAcol = gray(0.6), ...) {
        use <- complete.cases(x, y)
        idx <- which(use)
        n <- length(idx)

        if(n < 2)
                return(invisible())
        for(i in seq_len(n - 1)) {
                j <- idx[i]
                k <- idx[i+1]
                col <- if((k - j) > 1)
                        NAcol
                else
                        "black"
                lines(c(x[j], x[k]), c(y[j], y[k]), col = col)
        }
        invisible()
}
```

Functions

- Functions should do one thing well
- ~1 screen full of text
- 5-10 local variables
- Informative but succinct names
- If functions get long break them up into helper functions

Commenting

- Commenting your code is good in general
- There *is* such a thing as over-commenting
- Never try to explain how your code works in a comment
- Write the code so that the it is obvious how it works
- Comments should explain *what* the code does, not how it works

Summary

- Indenting improves readability
- Fixing line length (80 columns) prevents lots of nesting and very long functions
- Indents of 4 spaces at minimum; 8 spaces ideal
- BUT: Always follow the established style of an existing project. Don’t make trouble for the maintainer!
- Functions should do one thing well
- Comments should be as simple as possible but no simpler