Developing a Consistent R Style Statistics 506

Code Styles

It is important to develop a consistent style when coding to make your code more readable. Consider the following two snippets of code:

```r
x<-read.csv("file.csv")
x[,1]=round(x[,1],2)
for(i in 2:5) x[,i]<-as.numeric(x[,i])
m=apply(x,2,mean)
for(i in 1:5) if(m[i]<0) x[,i]<--x[,i]
```

```r
measures <- read.csv("measures_data.csv")

# The first column has too many decimals; the remaining columns are read in
# as strings for some reason. Fix both.
measures[,1] <-  round(measures[, 1], 2)
for (i in 2:5) {
  measures[, i] <- as.numeric(measures[, i])
}

# Check the average measures.
mean.measures <- apply(measures, 2, mean)

for (i in 1:5) {
  if (mean.measures[i] < 0) {
    # If any average measure is negative, swap the sign to ensure it's positive.
    measures[, i] <- -measures[, i]
  }
}
```

These two code snippets do identical tasks, but the second is vastly easier to read. It uses informative variable names, has spacing that clearly distinguishes code inside control statements, is consistent in terms of `=` versus `<-` and is well commented.

There is no "correct" code style for R. Is it important to develop a personal style that strives for **consistentcy** and **readability**. (Some languages enforce different levels of consistency, such as python's indenting rules.)

Consistency refers to picking a style and sticking with it. Imagine reading a novel where the author changes the spelling of the characters names every few chapters, and switches randomly from first-person to third-person narrative. You could read it, but every time something changed you'd be caught off-guard for a second. When reading well styled code, you quickly gain an understanding for the programmers style and changes from that can make reading difficult.

Readability refers to the ability to share your code with others and their ability to understand it. Obfuscated code, that is difficult to read, unintentially or intentionally, may sometimes be very clever, but unless you are working under conditions where you need to optimize the code to a high degree, readable code is preferred over clever code.

Important Rules

Below I link to several different style guides you might base your own personal style on (or develop one entirely on your own). Here I want to highlight a few style rules that are either very important or very easy to overlook. Note that these are not mandated for work submitted in this class, but if you have a different style, make sure it is just as **readable** and **consistent**.

Control statements

Control statements (`if`, `for`, `while`, etc) in R can be written extremely succinctly or verbose. It is better to write them verbosely. For example,

Bad:

```
if (x > 5) y <- 3

for (i in 1:5) { x[i] <- y[i]; r[i] <- r[i]^2 }

x <- if (is.numeric(y)) 3 else 4
```

Good:

```
if (x > 5) {
  y <- 3
}

for (i in 1:5) {
  x[i] <- y[i]
  r[i] <- r[i]^2
}

if (is.numeric(y)) {
  x <- 3
} else {
  x <- 4
}
```

Always surround the expression following the control statement by curly braces (`{` and `}`), and the expression should be on a line separate from the control statement. `ifelse` is a suitable alternative *if the two possible entries are short*. E.g., the last example could be

```
x <- ifelse(is.numeric(y), 3, 4)
```

attach

Never use `attach`. It is far too easy to have serious issues when using it. If you really want to avoid using the `$` notation, use `with` instead:

```
with(mydata, {
  plot(y ~ x)
  mod <- lm(y ~ x)
})
```

Line length

Long function calls can lead to extremely long lines. Split the lines up at natural points to improve readability. If you do so, indent appropriately.

Bad:

```
mod <- lm(mpg ~ price + weight + headroom + trunk + length + Turn + displacement + gear_ra
```

Good:

```
mod <- lm(mpg ~ price + weight + headroom + trunk + length +
                turn + displacement + gear_ratio + foreign,
          data = cars,
          subset = (make = "Ford"),
          weights = carwts,
          na.action = na.exclude)
```

Developing your own style

If you have a style which you've already developed, feel free to stick with it. If you've not developed a style yet, or are looking to clean up your style, here's a few example styles to get you started.

- Tidyverse Style Guide: Has a lot of stuff specific to the tidyverse, but does offer general style recommendations.
- Google's R Style Guide: Based on tidyverse, with a few tweaks.
- R Coding Conventions: A much more in depth guide.
- The R Inferno: While not a style guide, it is a list of oddities in R's code that can confuse users. For example, what do you think `sqrt(2)^2 == 2` will return? Go ahead, give it a try.