: Coding etiquette

How to write code that is clean, clear, and reproducible

Style

Coding, like any other writing type, is dependent upon clear and consistent style. As the tidyverse style guide so eloquently put it, "Good coding style is like correct punctuation: you can manage without it, butitsuremakesthingseasiertoread." Here, we can clearly see how a simple phrase becomes exponentially more challenging to read and understand. The same goes for coding.

It is important to remember that R **cannot** handle spaces between words. Because of this, we must be creative in how we name things. It is a good idea to follow the **BigCamelCase** naming method. Let's start by naming a vector

```
#GOOD
MyNewVector <-c(1,2,3,4,5,6)

#BAD
mynewvector <- c(7,8,9,10,11,12)
```

While in this example it is fairly easy to read both, we see how the name following the **BigCamelCase** format is easier to follow.

Let's look at another example of naming, this time getting more specific with our vector name.

```
#GOOD
SlugDensityData_Spring2023 <- c(1,3,5,7)

#BAD
slugdensitydataspring2023 <-c(2,4,6,8)
```

I incorporated an underscore in the first name to make it even more distinct. We can clearly see now with increasing complexity of our names, the first is much easier to read.

Annotations

When taking notes in a lecture, do you think it wise to take poorly written and hard to understand notes? Or, would we rather take clear, concise, and methodical notes to ensure we can return to them and understand exactly what the lecture was about? If you choose the former, then please, continue reading.

Annotation, like note taking, is very important within our code. We must be able to return to each line and know exactly what we did and why we did it. Along with this, if we wish to share this code with anyone, they too must be able to understand the methodology without needing you by their side. The habit of good code annotation is one that should be adopted immediately and practiced throughout the duration of your programming days.

Let's look at some examples of both good, and bad annotations.

```
#GOOD
##
#I am creating a vector to practice running different functions

PracticeVector <- c(11,3,4,5,6,7)

#Trying out the mean function here
mean(PracticeVector)
#This works. I will leave this code here to reference in the future
##
```

```
#BAD
practicevector <- c(11,2,3,4,5,11,2)
mean(practicevector)
```

While these examples are very simple in their nature, we can imagine how scrolling through 500+ lines of un-annotated code can be a nightmare. Along with this, to reiterate my naming point, we can see how with poor naming practices and a lack of annotation, the bad example is doubly hard to follow.

Along with annotating what you are doing, it can also be helpful to write out your thought process for an action. Let's say you are writing code for a project on a Friday, and since you are great at managing your workload, you plan to not work this weekend. When Monday rolls around, you open your R script up and have completely forgotten why you were running a specific test or structuring your code a specific way.

With proper annotation, this hiccup can be avoided.

Let's take a look at some examples.

```r
#GOOD
##
#I am trying to create a fake data set to practice some functions on
#Not to be used for analyses, simply for me

bugs <- data.frame( #naming this 'bugs' and using the data.frame function to build this
  spiders = rnorm(200), #naming this column 'spiders' and using the rnorm function. This f
  beetles = rnorm(250), #Same as above, but with 250 values
  wasps = rnorm(1000) #Same as above, but with 1000 values
)
##
```

In this example, I clearly noted what I was doing and why I was doing it. For my sake, I can return to this easily. If someone else was to come upon this, they too would be able to understand what my process was.

```r
#BAD
notbugs <- data.frame(
  clover = rnorm(200),
  shrubberies = rnorm(200),
  elderberry = rnorm(2000)
)
```

In this example, it is unclear what the purpose of this data set is. Along with that, if someone is not familiar with this script, they may find it very challenging to follow.