Style Guide Challenge

## Instructions

As the great Hadley Wickham states: “Good coding style is like correct punctuation: you can manage without it, butitsuremakesthingseasiertoread.”

Making things “easy” to read isn’t essential for R. R has some finicky rules for how you tell it do things, and abiding by these rules is of course important. When you don’t, R will always complain in the form of an error or warning message.

But even if your code is perfectly sound and readable for R, that doesn’t guarantee that it’s readable to *other human beings*. Data analytics is inherently collaborative, and that often means writing code in teams and sharing your code with others. So, following consistent stylistic rules is essential for making teamwork in data analytics all-the-easier.

Think of it this way… I asked ChatGPT to write me a paragraph in American English and then again in Shakespearian English. This is what it gave me:

**American English paragraph:**

“The United States is a diverse country with people from all walks of life. From big cities to small towns, you can find a variety of cultures and traditions. Each state has its own unique history and geography, making the country a fascinating place to explore. Whether you’re interested in art, music, or sports, there’s something for everyone here.”

**Shakespearian English:**

“Hark! The United States doth be a diverse realm, wherein folk of all stripes doth reside. From cities grand to hamlets small, a multiplicity of cultures and traditions be present. Each province doth bear its own distinct history and geography, rendering this land a most fascinating place to peruse. Whither thou art moved by the arts, music, or sport, there doth be somewhat for every soul here.”

Both of these are written in English and convey the same basic message. But the styling, and hence intelligibility, are different. It’s the same with writing code. Whether your code conveys the right message to R is not the appropriate bar for judging the quality of your code. Instead, the quality of your code should be judged both on whether it works *and* whether it’s legible to other people.

## Style Guides

There are lots of style guides out there with recommendations for how to write your code for maximum intelligibility. My go-to source is the [Tidyverse Style Guide](https://style.tidyverse.org/index.html). I highly recommend using this as a resource. I also recommend putting its guidelines into practice in your own coding. The sooner you put its recommendations into practice, the better your coding habits will be.

To help introduce you to some of the finer stylistic points in the guidance, I’ve given you some examples of poorly written code below. By “poorly written” I don’t mean that the code doesn’t work. Provided the necessary objects are available in R, it would all work just fine. However, it is poorly written from a human legibility standpoint.

I’d like you to rewrite the example code below so that it conforms to the Tidyverse Style Guide and, hence, is more reader-friendly. For each section, I’ve given you the relevant link in the Style Guide.

## Fix This Code

### Syntax

Helpful link: [Syntax](https://style.tidyverse.org/syntax.html)

Fix these object names:

dataOne  
economic\_data\_for\_project  
my.mean

Fix the spacing:

x[,4]  
mean ( x )  
height<-feet\*12+inches  
sqrt(df $ z)

Is the extra spacing after mean here okay?

list(  
 total = a + b + c,  
 mean = (a + b + c) / n  
)

### Functions

Helpful link: [Functions](https://style.tidyverse.org/functions.html)

Rewrite this for clarity:

long\_function\_name <- function(a = "a long argument",  
 b = "another argument",  
 c = "another long argument") {  
 # Here's where the code for the function would go  
}

### Pipes

Helpful link: [Pipes](https://style.tidyverse.org/pipes.html)

Fix this code:

iris %>% group\_by(Species) %>% summarize\_all(mean) %>%  
 ungroup %>% gather(measure, value, -Species) %>%  
 arrange(value)

Is any of these best?

# Way 1:  
iris\_long <-  
 iris %>%  
 gather(measure, value, -Species) %>%  
 arrange(-value)  
   
# Way 2:  
iris\_long <- iris %>%  
 gather(measure, value, -Species) %>%  
 arrange(-value)  
   
# Way 3:  
iris %>%  
 gather(measure, value, -Species) %>%  
 arrange(-value) ->  
 iris\_long

### ggplot

Helpful link: [ggplot](https://style.tidyverse.org/ggplot.html)

Fix these code chunks:

data %>%  
 filter(var == "cat1") %>%  
 ggplot(aes(x = height, y = weight)) + geom\_point()

ggplot(aes(x = height, y = weight, color = var)) +  
 geom\_point() +  
 labs(x = "Height in Inches", y = "Weight in lbs", title = "Weight per Height")

ggplot(filter(data, var == "cat1"), aes(x = height, y = weight)) +  
 geom\_point()