Using apply, purrr and Advanced Functions

Zach Ginder

Task 1: Conceptual Questions

Question 1: What is the purpose of the lapply() function? What is the equivalent purrr function?

The lapply() function allows us to apply a function of our choice to a list object and return a list object. The equivalent purrr function to lapply() is map().

Question 2: Suppose we have a list called my_list. Each element of the list is a numeric data frame (all columns are numeric). We want to use lapply() to run the code cor(numeric_matrix, method="kendall") on each element of the list. Write code to do this.

The code would be as follows:

```
#cor_of_my_list<- lapply(my_list, FUN=cor, method="kendall")
```

Question 3: What are two advantages of using purrr functions instead of BaseR apply functions?

Two advantages are as follows:

1. With purrr functions you can predict the output type exclusively from the function name but this is not always the case for the BaseR apply functions.
2. Purr functions have helpers which allow you to write compact code for common special cases, giving us a shorthand way to make anonymous functions.

Question 4: What is a side-effect function

A side-effect function [like print(), write_csv(), plot()] does not change the data it just tries to produce something, therefore, it does not naturally return the modified argument. This is in contrast to say transformation functions.

Question 5: Why can you name a variable sd in a function and not cause any issues with the sd function?

This is because of the environment nature of R. When you call a function, it creates a temporary function environment allowing variables in the function to exist but only in the temporary environment not overwriting the sd() function.