The Map Function in R: Takeaways

Syntax

- Creating a custom function and using map() to vectorize it:

```
format_score <- function(score) {
  fmt_string <- str_replace(score, "%", "")
  num <- as.numeric(fmt_string)
  return(num)
}
example_scores <- c("19%", "81%", "100%")
map_result <- map(example_scores, format_score)
```

- Using map2() to vectorize a function that takes in two inputs:

```
first_inputs <- c(1, 2, 3)
second_inputs <- c(4, 5, 6)
add_inputs <- function(x, y) {
  return(x + y)
}
output <- map2(first_inputs, second_inputs, add_inputs)
```

- Using the map() and mutate() functions to create a new column in your dataset:

```
format_score <- function(score) {
  fmt_string <- str_replace(score, "%", "")
  num <- as.numeric(fmt_string)
  return(num)
}
scores <- scores %>%
  mutate(
    new_writing_score = unlist(map(writing_score, format_score))
  )
```

- Using lists as an input to the map() function:

```
input_list <- list(
  c(1, 2),
  c(3, 4),
  c(5, 6),
  c(7, 8),
  c(9, 10)
)
output <- map(input_list, sum)
```

- Using group_by() and summarize() together to vectorize a summary function across groups in a dataset:

```
avg_score_by_student <- student_scores %>%
  group_by(names) %>%
```

```
  summarize(
    avg_writing = mean(new_writing_score)
  )
```

Concepts

- The `purrr` package has a family of functions that can accommodate any number of inputs and data types. Each of these functions work out similarly, so learning one can make learning the others easier.
- We use the `map()` function to vectorize a given function. You can use any function, whether it's one from R itself or a custom function that you've made yourself.
- We use the `map2()` function to vectorize a given function that uses two inputs.
- We can use the `group_by()` and `summarize()` functions together to create powerful analyses. These two work well when a dataset has two or more groups that we would like to compare against each other.