- We can change any set of R code into a function in a similar way.

- The code for generating 100 samples, running the t-tests, and then plotting the results of the t-tests is turned into a function on the next slide - I've called it `simulate()`

- It takes one parameter input which specifies the sample size you want to simulate.

```
simulate <- function(sample_size) {
  total_samples <- 100
  participant <- rep(1:sample_size, times = 2)
  condition <- c(rep("fast", times = sample_size), rep("slow", times = sample_size))
  all_data <- NULL

  for  (i in 1:total_samples) {
    sample <- i
    set.seed(1233 + i)
    dv <- c(rnorm(sample_size, 1000, 50), rnorm(sample_size, 1020, 50))
    data <- as.tibble(cbind(participant, condition, dv, sample))
    all_data <- rbind(data, all_data)
  }

  all_data$condition <- as.factor(all_data$condition)
  all_data$dv <- as.integer(all_data$dv)
  print(ggplot(all_data, aes(x = condition, y = dv, fill = condition)) + geom_violin() +
  geom_jitter(alpha = .3, width = .05) + guides(fill = FALSE) + facet_wrap(~sample))
  result <- NULL

  for  (i in 1:total_samples) {
    result <- rbind(tidy(t.test(filter(all_data, condition == "fast" & sample == i)$dv,
                                filter(all_data, condition == "slow" & sample == i)$dv,
                                paired = FALSE)), result)
  }

  print(ggplot(result, aes(x = p.value)) + geom_histogram(bins = 50))
  print(ggplot(filter(result, p.value < .05), aes(x = p.value)) + geom_histogram(bins = 50))
  return(count(filter(result, p.value < .05)))

}
```