assignment_02

2023-09-05

Infinite Monkeys Need Infinite Bananas

Here is the code for the function for simulating keystrokes:

```
simulate_key_strike <- function() {
  sample(c(LETTERS, letters, " ", 0:9), size = 1)
}
```

Testing our function,

```
## [1] "F"
```

Seems to work! Let's move on.

Part 2

Now, we want to simulate multiple keystrokes. Let's write a function for it:

```
simulate_several_key_strikes <- function(x) {
  store <- numeric(x)
  string <- ""
  #concatenating the individual keystrokes into one full string
  for (i in 1:x) {
    store[i] <- simulate_key_strike()
    string <- paste(string, store[i], sep = "")
  }
  return(string)
}
#this seems a bit inefficient, but I can't figure out how else to count the keystrokes
```

It seems to be working:

```
## [1] "QsOOV"
```

Monkey See Monkey Do

Here, we withhold bananas until the monkey types up 1000 strings of length 12:

```
wrapper <- function(x) {
  simulate_several_key_strikes(12L)
}
monkey_test <- sapply(1:1000, wrapper)
```

Using the `stringr` package, we calculate the required data proportion here:

```
library(stringr)
data_proportion <- sum(str_count(monkey_test, "Data Science"))/1000
```

Unsurprisingly, "Data Science" doesn't actually appear at all:

```
## [1] 0
```

Monkey Tee Monkey To

Here, we write a loop for counting the number of times the letter "t" or "T" appears at least once in the 12 strikes.

```
monkey_t <- sapply(1:10000, wrapper)

#I wanted to write a reusable function but couldn't figure out how to make the count work,
#so I used a for loop instead:

t_count_loop <- function() {
  count <- 0
  for (i in 1:10000) {
  if (grepl("t", monkey_t[i]) | grepl("T", monkey_t[i])) {
        count = count + 1
   } else {
       count = count
     }
  }
  return(count)
}

t_or_T_proportion <- t_count_loop()/10000
```

Here is the required proportion:

```
## [1] 0.3291
```

I'm Like t_T (T_T)

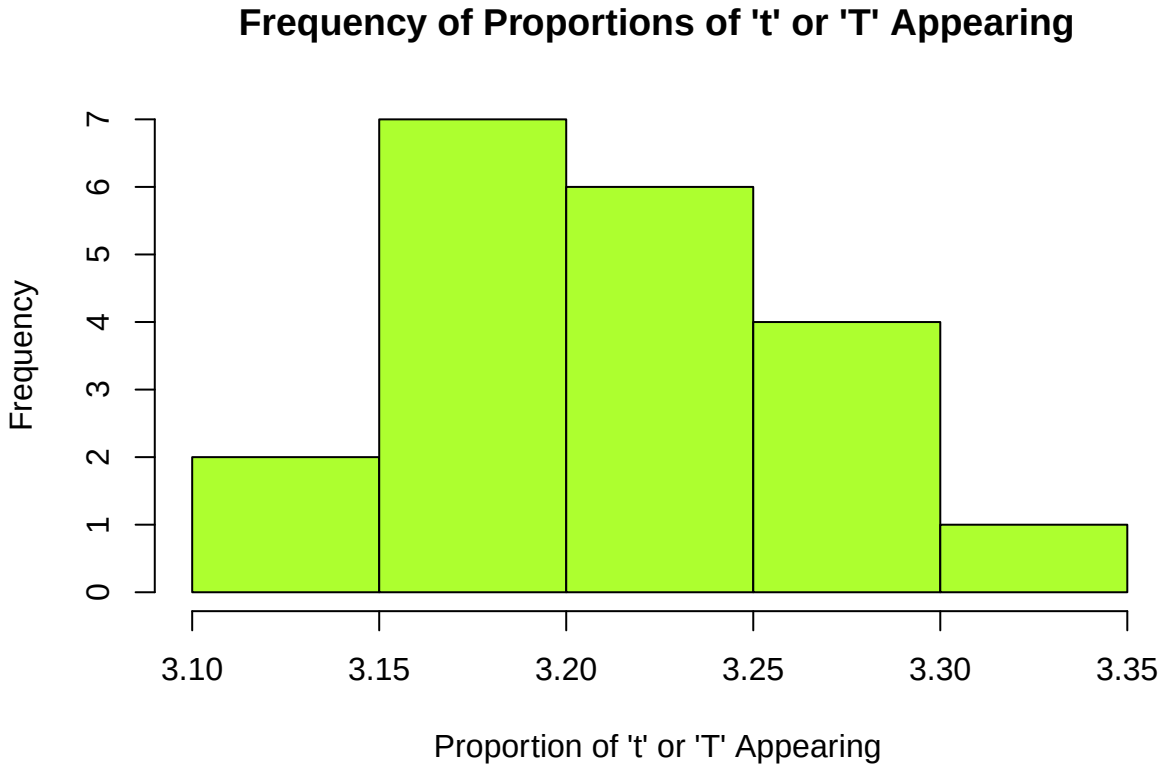
If we just ended the simulation here, then I think a computer simulation would not be effective in estimating the chance of a "t" or a "T" appearing in these random key strikes. However, I think a computer simulation would be effective if we repeated this experiment over maybe 1000 iterations and plotted a histogram of the proportions we get. Through plotting a histogram, we'd be able to see the proportion that appears with the highest frequency. That should be the expected chance of "t" or "T" appearing.

So, basically, if we did this:

```
t_T_proportions <- numeric(20)

for (i in 1:20) {
  monkey_t <- sapply(1:10000, wrapper)
  t_T_proportions[i] <- t_count_loop()/1000
}
```

I only ran it 20 times as it was taking forever to loop through. Not sure if there's a way to make it more efficient. But this is the general idea. Plotting the proportions in a histogram:



It would be a lot better if we could run it a higher number of times, but alas, my computer's capabilities are only so much.

The chances of typing "Data Science" are probably close to `0`. I'd already be surprised if we see any legible word or phrase in a string a monkey types up. If we could run the simulation forever, then maybe. The actual chance is actually `(1/63)^12`, which is very very close to `0`.