

Loopy for Loops, Crazy for Conditionals, Silly for Samples - Hands-on!

YOUR NAME

2023-10-10

Outline

Hello R students! Your silly student coder has returned to make your assignments notably worse than if Dr. B had made them. This week you will learn:

- intro to loops: `for` and `while`.¹
 - conditional statement: `if`, `then`, `else`, and `ifelse`.
 - a soft intro to sampling using mostly base R.
 - as much dplyr as I can include to drive Levi crazy².
-

What is a loop?

Loops are a programming element that repeat a portion of code a set number of times until the desired process is complete.³

All programming languages have loops. Their syntax might differ, but they are always there, just like element types, data structures, etc.

We are going to be focusing on “for loops”. A for loop applies a command to each value provided then stops.

Basic example:

```
for (i in 1:5) {  
  print(i)  
}
```

```
## [1] 1  
## [1] 2  
## [1] 3  
## [1] 4  
## [1] 5
```

¹R also has a loop called `repeat` that I only learned about when I googled “types of loops in R. I don’t think you’ll need to know about it

²Ask Levi about his beef with `tidyverse` if you are unaware

³See ref 1 at the end of this document.

- You don't actually have to use `i` for the value there but most people do. `i` is presumably short for "index", which makes sense. But you could just as well call it `banana`.

Ex.1) run the same command above replacing `i` with `banana`.

```
#type your code in here
```

So this loop printed each value for `i` (or `banana`). How would you write code that printed the values 6-10?

```
for (i in 6:10) {
  print(i)
}
```

```
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

It's often useful to define a placeholder variable before running your loop. For example, this loop calculates the mean of a given data set.

```
numbers <- c(4, 22, 6, 13, 19, 2, 11)
sum <- 0
for(num in numbers) {
  sum <- sum + num
}
mean <- sum / length(numbers)
print(mean)
```

```
## [1] 11
```

You can also write loops with character vectors. Here's a character vector: `days<-c("Mon", "Tues", "Wednes", "Thurs", "Fri", "Satur", "Sun")`

Ex.2) Write a loop that adds day to each day of the week so that the output of `days` gives you "Monday", "Tuesday", etc, one in each line.

Hint: if you add the argument `sep = ""` to paste you don't get a space in between)

```
#First create the vector days as show above
```

```
#now write a for loop that adds "day" to each of the elements and print each one as you  
↪ go
```

You can do a lot more with loops once you learn about functions. But alas, that's for next lab. For now that's probably enough for loops.

Recommended: This guide someone wrote on the internet is really good if you want to also read it. <https://intro2r.com/loops.html>

Conditional statements

Sampling

There is a basic function in R for taking samples very appropriately named `sample`. You need three arguments for it to work (there is a 4th but don't worry about it yet). The first is the data set, the second is the number of samples, and the third is with or without replacement.

Read in the `human_genes.csv` file and name it `human_genes`. Play around with it using your strong exploratory data analysis skills. Get a good feel for the data set.

Overall goal of this exercise is to compare sample means at different `n` to the population mean.

Its big right? So let's take some samples. NOTE: Your output will look different to the pdf because your samples will be different values.

1. First filter it to only keep the size and name column.
2. create a second data frame with summary statistics mean, median and SD of length. (what is interesting about this data??)
3. Now let's take some samples. Take 3 samples of gene size with replacement on. Use `n = 10, 100, 1000`. Calculate the mean of these samples, and save each mean as `mean10`, `mean100`, and `mean1000`.
4. Compare these means to the population mean. Which is closest?

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##   filter, lag

## The following objects are masked from 'package:base':
##
##   intersect, setdiff, setequal, union
```

```
library(readr)
human_genes <- read_csv("human_genes.csv")
```

```
## Rows: 22385 Columns: 4

## -- Column specification -----
## Delimiter: ","
## chr (3): gene, name, description
## dbl (1): size
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```

human_genes<-human_genes %>% select(name, size)

human_genes_summ <- human_genes %>%
  summarise(
    MeanLength = mean(size),
    MedianLength = median(size),
    SDLength = sd(size)
  )

mean10<-mean(sample(human_genes$size, 10, replace = TRUE))
mean100<-mean(sample(human_genes$size, 100, replace = TRUE))
mean1000<-mean(sample(human_genes$size, 1000, replace = TRUE))

```

References

- [1] Korable. What are loops? Link: <http://support.korable.com/en/articles/417331-what-are-loops#:~:text=Definition%3A%20Loops%20are%20a%20programming,save%20time%20and%20minimize%20errors>.
- [2] Intro2r. For loops. <https://intro2r.com/loops.html>