Creating Functions

Why Build Functions?

Functions are the building blocks of code. Just as we break down tasks into smaller steps in everyday life, functions break down complex coding tasks into manageable units. But why should we bother with building your own functions when so many others exist?

- ***Modularity:*** Functions help us organize code by grouping related actions together. This makes our code more structured, easier to read, and simpler to maintain.
- ***Reusability:*** Once you've built a function to perform a specific task, you can use it as many times as you need without rewriting the same code. This not only saves time but also reduces the chances of errors.
- ***Efficiency:*** Functions promote efficient code development by enabling us to focus on solving specific problems without getting bogged down in larger program logic.
- ***Debugging:*** When an error occurs, functions help narrow down the problematic part of the code, making it easier to identify and fix issues.

Function Syntax

At its core, a function is a self-contained block of code designed to perform a specific task. Like a recipe, a function takes in ingredients (inputs or arguments) and produces a result (output). Let's break down the basic structure of a function in R:

Defining a Function

```
function_name <- function(argument1, argument2, ...) {
  Code to perform the task
  return(result)
}
```

- ***function_name***: Choose a meaningful name for your function, describing what it does.
- ***argument1, argument2, ...***: These are placeholders for the values (arguments) that the function will work with.
- ***return(result)***: After performing the task, the function uses the return statement to provide the result.

Example 1: Create a function that computes the predicted number of calories required by women.

BMR estimation formulas: The Mifflin St Jeor equation

$$predicted\ calories = 10 * weight(in\ kg) + 6.25 * height(in\ cm) - 5 * age(in\ years) + s$$

where s is a constant equal to -161 for women and 5 for men.

Codes:

```
calories_women <- function(weight, height, age){
    calories <- (10 * weight) + (6.25 * height) - (5 * age) - 161
    return(calories)
}
```

What is the output when I run this line after I ran the code that defines my function?

```
calories_women(weight = 65, height = 170, age = 35)
```

Output:

Note: `calories_women(65, 170, 35)` will give you the same output as above.

Using default values

```
calories_women_fixed <- function(weight, height, age = 30){
    calories <- (10 * weight) + (6.25 * height) - (5 * age) - 161
    return(calories)
}

calories_women_fixed(weight = 65, height = 170)
calories_women_fixed(weight = 65, height = 170, age = 35)
```

Example 2: Create a function for which the two arguments are two numbers a and b. Name this function “find_max”. Your function should print the maximum value between the two numbers or it should print that the two values are equal in the case that the two numbers are the same.

Code:

What would your function print if you ran these codes?

`find_max(1,2)` **Output:**

`find_max(5,3)` **Output:**

`find_max(6,6)` **Output:**

Example 3: Create a function that computes the mean of a vector. You have to build this function without the use of the default "mean" function provided by R. You should name this function mean_yourname. Hint: To sum the elements of a vector you can use `sum` and to calculate the number of elements in a vector, you can use `length`.

Codes:

What output would you get if you ran these lines:

```
v <- 1:4
mean_abdalla(v)
```

Output: