Functions in R

Sean Hellingman ©

Data Visualization and Manipulation through Scripting (ADSC1010)

shellingman@tru.ca

Fall 2024

THOMPSON RIVERS UNIVERSITY

Topics

Introduction

- We have spent some time using existing functions in R.
- Might be situations where you would like to create your own.
- Gain a better understanding of how functions work.
- Keep your code DRY (Don't Repeat Yourself)
  - *If at any time you catch yourself copying and pasting code, that's an indicator that you should be writing a function instead.*

Function Source Code

- We can look inside the internal workings of functions in R.
- Simply type the `function.name` without the `()`.
- R will return the source code for that function.
- Useful if you want to adjust existing functions for your own applications.

Structure

- In R functions are structured in the following way:
  - `fun.name <- function(in.var){`*Body*`}`
  - The *Body* part of the function actually does the work.
- The *Body* of the function must be enclosed in curly braces.
  - Everything outside of the braces will be ignored.
- The `in.var` is the input variable(s) into the function.
  - There may be more than one input variable into a function in R.

Example 1

- Write a function that takes an integer n as the input variable and computes $3(2^n - 1)$.

Question

- How do we use functions to assign values to variables?

Assigning Variables

- Variables that are assigned within a function body never make it into my workspace.
- Functions have their own workspaces, and those workspaces are hidden from us.
- Need to assign the values produced by functions to variables in our own workspace.

Example 2

- Consider the following functions:
  - `fun_one <- function(x){x^3}`
  - `fun_two <- function(x){y <- x^3}`
- i.e. does `fun_one(2)` produce the same output as `fun_two(2)`?
- Why or why not?

Multiple Inputs

- So far we have only considered functions with a single input variable.
- To allow for two input variables, assignment would look like

```
fun.name <- function(in.var.1,in.var.2){Body}
```

- Input variables must be separated by commas.
- Follow this format to extend to more than two input variables.

Example 3

- Write a function that takes numbers n, t, and x as the input variables and computes $2(n+t)^x$.

Function Body

- So far, all the functions we have looked at have had a single line in the *Body*.
- Functions can be created with multiple lines in the *Body*.
  - This will be especially useful when including conditions in your functions.

Example 4

- Write a function that takes numbers x, and y as the input variables and computes:
  - x+y
  - x*y
- Write a function that takes numbers x, and y as the input variables and computes:
  - x*y
  - x+y
- What happens when you use the functions?

`return()` Function

- Can use the `return()` function to output the desired value.
- Can include the `return()` function in all functions to avoid confusion.
  - This will be especially useful for increasingly complex functions.

Example 5

- Repeat Example 4 but ensure the output of both your functions is x+y.

Multiple Outputs

- The `return()` function will **not** allow for multiple outputs.
- Instead, we must create an object with our desired outputs and return that object:
  - Vectors
  - Lists
  - Matrices
  - Dataframes

Example 6

- Create a function that take a vector as an input and returns the maximum and minimum values.

Conditional Statements

- Used to add logic to code.
- *If X is true, do something, or else do something different.*

```
if (condition) {
Code executed when condition is TRUE
} else {
Code executed when condition is FALSE
}
```

`ifelse()` Statements

- Can be condensed into an `ifelse()` function.
- `ifelse(Condition, output when TRUE, output when FALSE)`

`ifelse()` Statements

- Can be condensed into an `ifelse()` function.
- `ifelse(Condition, output when TRUE, output when FALSE)`

Relational Operators

Operation	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
!=	Not equal to
&	Logical AND
|	Logical OR

Example 7

- Write a function that returns the minimum value in a vector only if the vector is of length less than 5.
- If the vector is longer than 5, the function should return the statement: "Vector is longer than 5".

Default Values

- We can set *default* values to some or all of the arguments of a user-defined function.

- Simply add the defaults into the function definition:

```
F1 <- function(a, b = 1, c = 2, d = NULL){ a*b*c + d }
```

Example 8

- Write a function with default values that either calculates the area of a rectangle or the volume of a rectangular prism depending on the input arguments.

Exercise 1

- Can you think of any functions that would help you with the materials we have learned so far?
- Examine the source code for some of the functions we have already used.

Exercise 2

- Write a function that takes a vector `x` as input, and returns its range (difference between largest and smallest values) as output.
    - Test your function on four different vectors.

Exercise 3

- Write a function that takes a square matrix `A` as input, and returns its trace (sum of its diagonal elements) as output.
  - Test your function on the matrices $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and $\begin{bmatrix} 1 & 0 & -1 \\ 2 & 1 & 7 \\ -2 & 8 & -10 \end{bmatrix}$.
- Write a function that takes a square matrix `A` as input, and returns the sum of its last column as output.
  - Test your function on the matrices $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and $\begin{bmatrix} 1 & 0 & -1 \\ 2 & 1 & 7 \\ -2 & 8 & -10 \end{bmatrix}$.

Exercise 4

- Write a function that takes a vector `x` as input, and returns a vector containing (in this order)
  1. the average of the values in `x`.
  2. the range of `x` (difference between largest and smallest values).
  3. the length of `x`.
- Use the tags `"mean"`, `"range"` and `"length"` (i.e. change the name of each element).
- Test your function on four different (short and simple) input vectors.

Exercise 5

- Write a function that takes a vector `x` as input, and returns its maximum value if the maximum value is greater than 10, otherwise it returns the vector's minimum value.
    - Make sure the function output indicates which value it is returning.
    - Test your function on four different vectors.

References & Resources

1. Douglas, A., Roos, D., Mancini, F., Couto, A., & Lusseau, D. (2023). *An introduction to R*. Retrieved from https://intro2r.com/

- https://www.r-bloggers.com/2022/04/how-to-create-your-own-functions-in-r/