PROG102: Functions

Writing your own functions in R

MARINCS 100B | Intro to Marine Data Science | Winter 2025

Key concepts

2 purposes of a function

Encapsulation and Reusability

Functions have a specific syntax, require parameters

Turns an essay into a recipe

Easy to read

Take a complex concept and make it easier to understand

Reusable

Functions have parameters. You can use the function multiple time to work with different sections of the code

Syntax

Name
Keyword
Parameters
Body
Return/output

```
function_name <- function(input1, input2, etc) {
    output_value <- DO SOMETHING TO INPUT
    return(output_value)

}
```

Demo in R

Recap

Functions encapsulate the details and makes code easier to repeat

New vocabulary and lingering questions

New vocabulary

Encapsulation

Lingering questions

Why do we do this

```
test_f <- function(test) {
    return_var <- test + 1
    return(return_var)
}
```

Instead of this

```
test_f <- function(test) {
    return(test + 1)
}
```

Exercises

Label the five parts of this function:

Name Keyword Parameters

```
first_and_last <- function(s) {
  first_char <- substr(s, 1, 1)
  last_char <- substr(s, nchar(s), 1)
  result <- paste(first_char, last_char)
  return(result)
}
```

Body

Output

Exercises

Match the function bodies on the left with the name that describes what they're doing on the right.

```
function(x) {
  result <- x + 1
  return(result)
}
```

increment

```
function(a) {
  result <- a * 2
  return(result)
}
```

double

```
function(a, b) {
  c_squared <- a^2 + b^2
  result <- sqrt(c_squared)
  return(result)
}
```

hypotenuse_length

Exercises

Write a function that turns a vector into a palindrome. For example, it should turn 1 2 3 into 1 2 3 3 2 1. Hint: you'll have to use a function called `rev()`. Choose a short but descriptive name for your function.

```
palindrize <- function(vec) {
  return_vec <- c(vec, rev(vec))
  return(return_vec)
}

palindrize(c(1,2,3,4))
```

PROG102: Functions

How functions execute

MARINCS 100B | Intro to Marine Data Science | Winter 2025

Key concepts

Functions act as "Black Boxes", or pocket dimensions
Parameters and returns are bridges in and out
Debugger helps us look at functions

The black box

Encapsulation

Demo in R

Recap

Calling browser() at the beginning of a function allows us to walk through it step by step

New vocabulary and lingering questions

New vocabulary

Debugger
Browser

Lingering questions

Is there any way to affect variables from inside a function? Or is it something you MUST do outside?

Exercises

- What value does the following code yield?

  11

- How could you change `fish_mass` so the code yields 12 instead?

  Change the initial mass to 6

- How could you change the body of the function so the code yields 12?

  Change 0.2 to 0.25

```
fish_mass <- 5
temperature <- 20
fish_growth <- function(mass, temp) {
  growth <- 2 + 0.2 * temp
  mass <- mass + growth
  return(mass)
}
fish_growth(fish_mass, temperature)
```

Exercises

In your own words, why does running this code generate an error?

The variable "area" is only defined inside the function, which makes it inaccessible to the rest of the program

```
calc_volume <- function(height, width, depth) {
  area <- height * width
  volume <- area * depth
  return(volume)
}
vol <- calc_volume(3, 5, 1)
area
```

PROG102: Functions

Default and named parameters

MARINCS 100B | Intro to Marine Data Science | Winter 2025

Key concepts

Parameters usually enter in order(by position)
Default parameters allow you to not specify a value(they default to something)
Named parameters can go wherever
Default and Named are options

Default and named parameters

```
round(x, digits = 0)
```

digits = 0That is default value

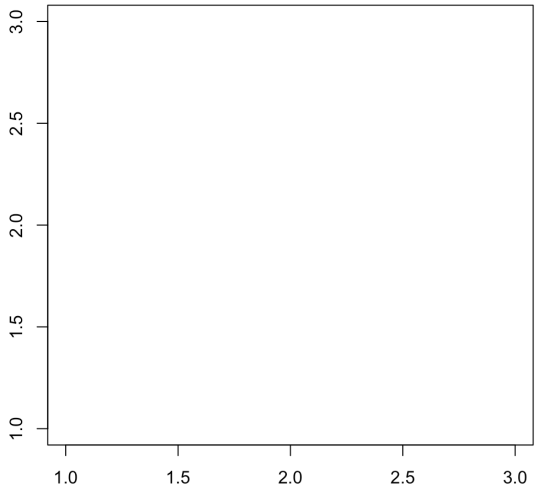
Pi rounded = 3
round(pi) = 3

round(digits = 2, pi) = 3.14

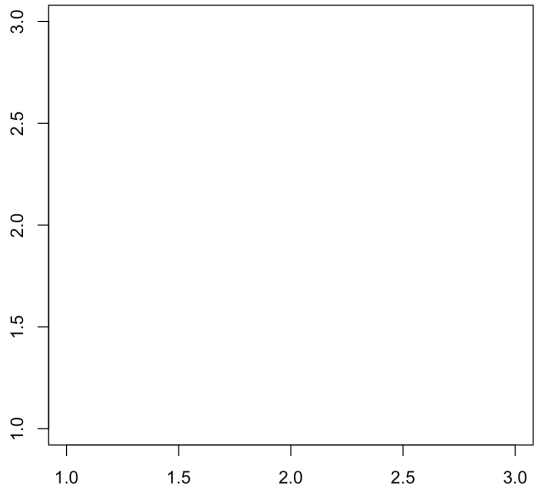
Long parameter lists

```
plot(x, y = NULL, type = "p",  xlim = NULL, ylim = NULL,
     log = "", main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
     ann = par("ann"), axes = TRUE, frame.plot = axes,
     panel.first = NULL, panel.last = NULL, asp = NA,
     xgap.axis = NA, ygap.axis = NA,
     ...)
```

```
plot(c(1, 2, 3), c(3, 2, 1))
```




```
plot(c(1, 2, 3), c(3, 2, 1),
     xlab = "x", ylab = "y")
```



Demo in R

SKIPPED

Triple dots

```
max(..., na.rm = FALSE)
paste(..., sep = " ", collapse = NULL, recycle0 = FALSE)
```

Ignore triple dots. Use intuition

Recap

Modify how functions work
Default values allows us to skip over parameters
Named values allows us to put the important stuff first

New vocabulary and lingering questions

New vocabulary

Default parameters
Named parameters

Lingering questions

What are those triple dots doing???

Exercises

R represents *missing* data with the value NA. Say you're doing an experiment and you miss the second observation. In R you can write that as `c(1, NA, 3, 4)`.

Most summary functions, like `mean()`, `max()`, and `median()`, have a parameter called `na.rm`. What does this parameter do? What is its default value? How would you get the maximum value of the vector `c(1, NA, 3, 4)`?

```
x <- c(1, NA, 3)

print(max(x))

print(max(x, na.rm = TRUE))
```

na.ra usually defaults to FALSE
This parameters removes any missing data, allowing us to call the max/median/mean functions