PROG102: Functions

Writing your own functions in R

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

Key concepts

Functions have two purposes

- hide code "encapsulation"
- Appy same code to new inputs

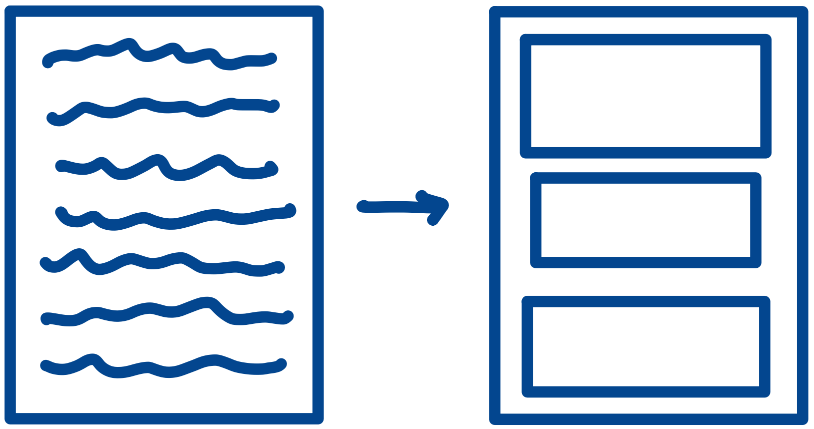
  reusablity

Know correct syntax

Easy to read

cognative load:

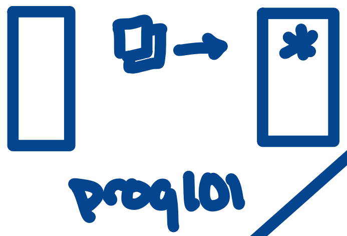
remember <7 items



grouping data

Reusable

Copy Paste:



Parameters

$$\begin{tikzcd} \text{inputs} \arrow[r] & \boxed{} \arrow[r] & \text{Outputs} \end{tikzcd}$$

same logic, new inputs, new outputs

Syntax

name

keyword

Parameter (parentheses)

body {curly braces}

return output

Demo in R

Recap

Functions make code readable by hiding the details : Encapsulation

Functions make code reusable by allowing different inputs : parameters

Syntax - every function definition has five parts

New vocabulary and lingering questions

New vocabulary

Encapsulation: a way of bundling data in your code

Parameter: a way to reference a single peice of data

Reusablity: the ability to make versatile code than can work for different things

Lingering questions

Exercises

Label the five parts of this function:

```
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)
}
```

Key word

Body

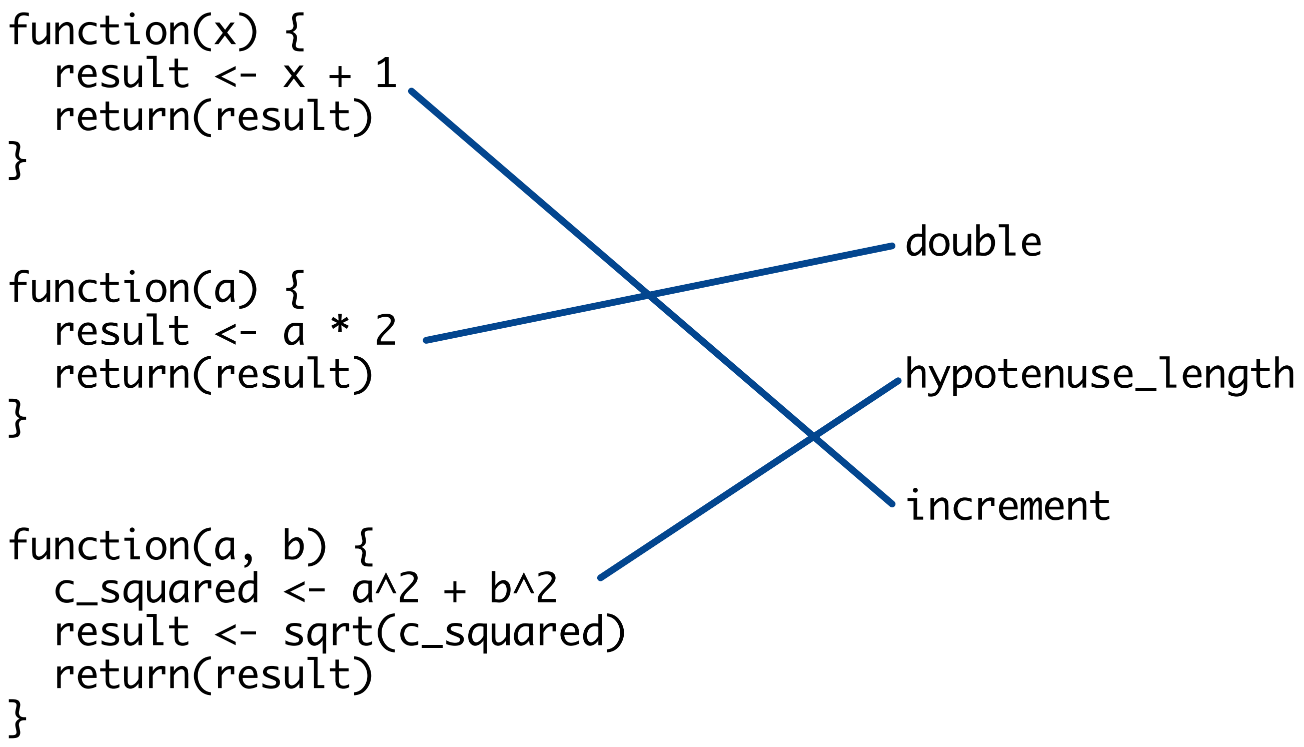
Parameters

Return output

Name

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)
}
```

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

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

```
double
hypotenuse_length
increment
```

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.

```
palin <- function(x) {
   var_1 <- c(1,2,3)
   var_2 <- rev(var_1)
   result <- (c(var_1, var_2))
   return(result)
 }
```

PROG102: Functions

How functions execute

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

Key concepts

Functions act as black boxes

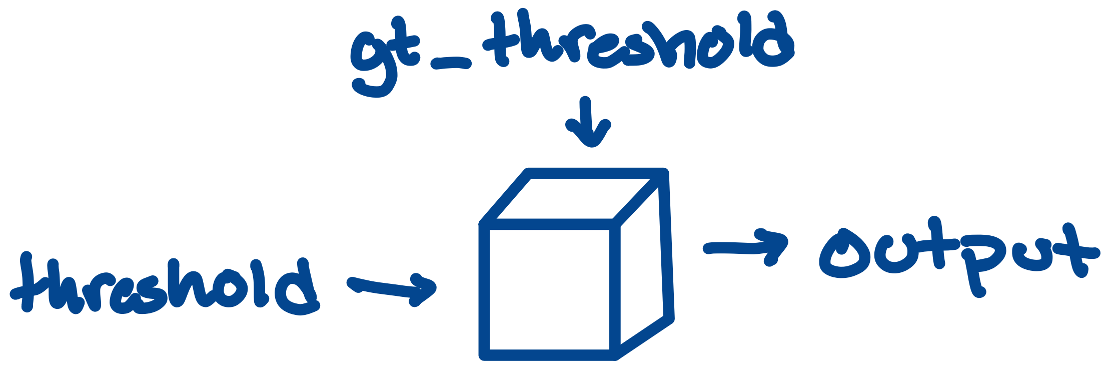
- seperate universe

Parameters and returns, those are bridges into and out of the black box

Debugger: a useful way to peak inside the box

The black box

Encapsulation:



Demo in R

Recap

Function opperate their own little universe the black box

Parameters are how we let information in

Return() is how we let info-rmation back out

New vocabulary and lingering questions

New vocabulary

Debugger: a way of looking into encapsulated code to see each step

Return: tells the computer which value it should spit back out to you

Lingering questions

Exercises

- What value does the following code yield?

  the code yeilds 11

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

  if you change 5 to 6

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

  if you change the 2 to a 3

```
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?

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

Area is encapsulated inside the box and therefore cannot be found seperately

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 parameter values allow you to omitt values

Named parameters let you skip in order

Defualt and Named parameters are usually options that modify function execution

Default and named parameters

$$\texttt{round(x, }\underbrace{\texttt{digits}}_{\text{parameter name}}\texttt{ = }\underbrace{\texttt{0}}_{\text{default}}\texttt{)}$$

round(pi) → 3 use the default

round(pi, 0) → 3 by position

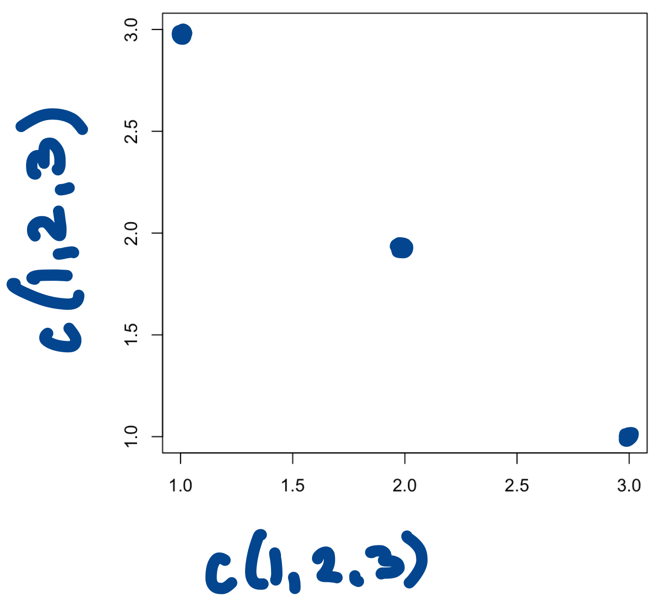
round(digits = 0, pi) → 3 named parameter

Long parameter lists

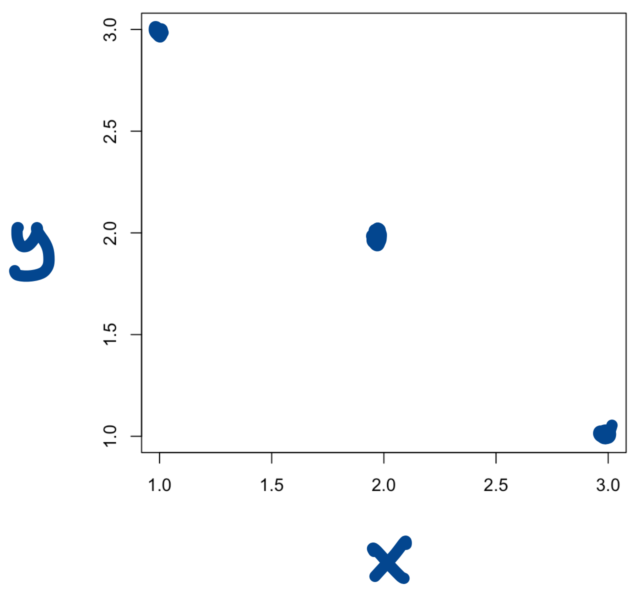
```
plot(x, y = NULL, type = "p",  xlim = NULL, ylim = NULL,
     log = "", main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
     ann = par("ann"), axes = TRUE, frame.plot = axes,
     panel.first = NULL, panel.last = NULL, asp = NA,
     xgap.axis = NA, ygap.axis = NA,
     ...)
```

x y

```
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

Triple dots

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

$\rightarrow$ ignore

$\rightarrow$ focus

max(1,2,3) $\rightarrow$

paste("water", "is", "wet") $\rightarrow$ water is wet

Recap

Named and default parameters are useful for modifying how functions work

Default values allow omission

Named parameters allow skip

New vocabulary and lingering questions

New vocabulary

Defualt Parameter: a pre-designated function of code the computer knows

Named Parameter: a parameter that you choose the name for, either making your own or renaming computers.

Lingering questions

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)`?

mean(c(1,NA,3,4), na.rm = TRUE)   output: $2.\overline{66}$

- this ignores the NA in the string of values

max(c(1,NA,3,4), na.rm = TRUE)   output: 4