Week 11: Functions

Elena & Willa

11/8/2022

What is a function?

What is a function?

A flexible, reusable chunk of code.

What is a function?

A flexible, reusable chunk of code.

When you find yourself using copy-paste or repeatedly changing one variable in your code you probably need a function!

What is a function?

A flexible, reusable chunk of code.

When you find yourself using copy-paste or repeatedly changing one variable in your code you probably need a function!

```
for (i in 1:100){

"do something"

}
```

```
for (i in 1:1000){

"do something"

}
```

```
for (i in 1:10000){

"do something"

}
```

Copy-pasting can get messy, increase the chance of errors, and makes your code harder to use and read.

Some scenarios where you might create a function

- Multiple people will be using your code on different data.

Some scenarios where you might create a function

- Multiple people will be using your code on different data.
- You will be using your code on different data in the future or later in the script

Some scenarios where you might create a function

- Multiple people will be using your code on different data.
- You will be using your code on different data in the future.
- You will be doing the same process multiple times.

Some scenarios where you might create a function

- Multiple people will be using your code on different data.
- You will be using your code on different data in the future.
- You will be doing the same process multiple times.

Can you think of other scenarios?

Can you think of a situation where you probably wouldn't want a function?

Functions in R

You are already familiar with a lot of built in functions in R and tidyverse functions mean(), print(), case_when() etc.

Functions in R

You are already familiar with a lot of built in functions in R and tidyverse functions mean(), print(), case_when() etc.

All functions take some kind of input (eg. your data), perform a process (eg. calculate the mean), and then return a value (eg. the mean of your data).

In built in functions we often don't see the code that performs the process. It happens "under the hood".

We can create our own functions in R

```
NAME <- function(ARGUMENTS) {

  ACTIONS

  return(OUTPUT)

}
```

R syntax for functions

What will you call your function (Can't use built-in function names)

```
NAME <- function(ARGUMENTS) {

  ACTIONS

  return(OUTPUT)

}
```

R syntax for functions

What will you call your function (Can't use built-in function names)

*What are the **inputs** to the function?*

```
NAME <- function(ARGUMENTS) {

  ACTIONS

  return(OUTPUT)

}
```

Things that are specific to the user or change each time.

R syntax for functions

What will you call your function (Can't use built-in function names)

*What are the **inputs** to the function?*

```
NAME <- function(ARGUMENTS) {

  ACTIONS

  return(OUTPUT)

}
```

Things that are specific to the user or change each time.

These will be fed into the body of the function

R syntax for functions

What will you call your function (Can't use built-in function names)

What are the *inputs* to the function?

```
NAME <- function(ARGUMENTS) {

  ACTIONS

  return(OUTPUT)

}
```

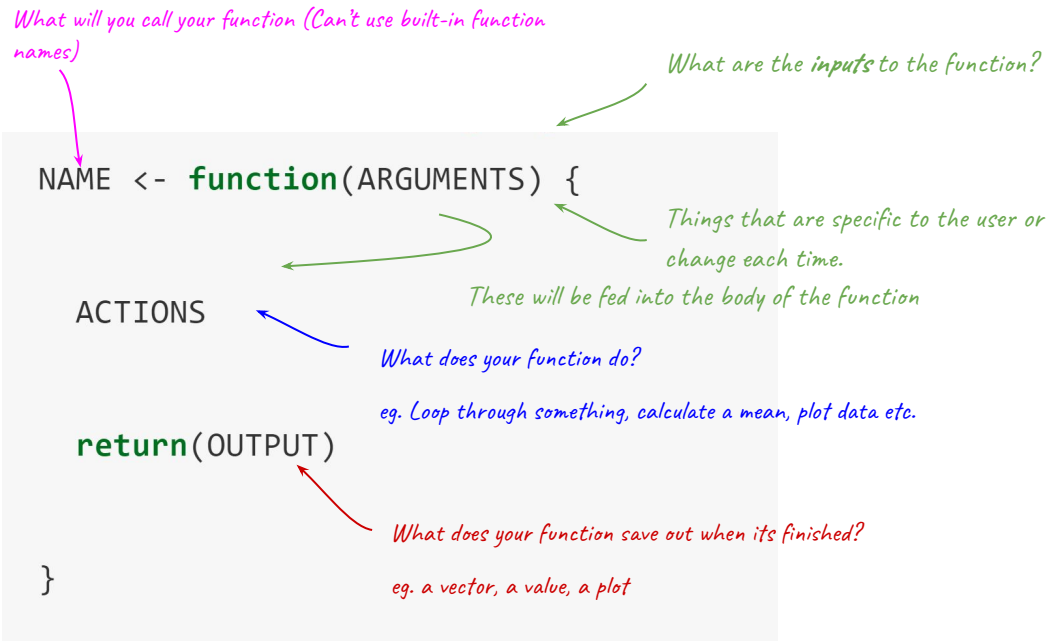
Things that are specific to the user or change each time.

These will be fed into the body of the function

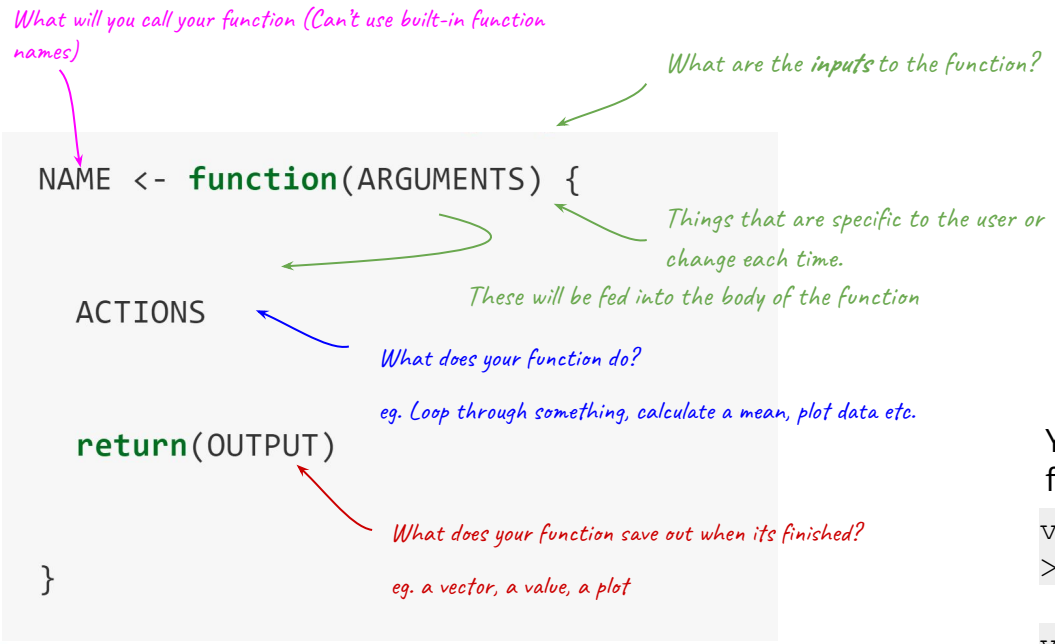
What does your function do?

eg. Loop through something, calculate a mean, plot data etc.

R syntax for functions



R syntax for functions



Using a function

You call your function the same way as any built in function.

```
var mean <- mean(my_data)
> var_mean
```

```
var f <- my_func(arg1)
> var_f
```

Let’s consider some built-in functions

Pick a function, and then describe its 4 attributes (in English words).

```
NAME <- function(ARGUMENTS) {

  ACTIONS

  return(OUTPUT)

}
```

sum()	mean()
paste()	print()

A little more on function arguments

When you write your own function you have to decide on the arguments.

- What information do you need from the user? (eg. their data)
- Do you want the user to have control over any additional info (eg. number of samples they take, additional parameters etc.)

A little more on function arguments

When you write your own function you have to decide on the arguments.

- What information do you need from the user? (eg. their data)
- Do you want the user to have control over any additional info (eg. number of samples they take, additional parameters etc.)

You can choose to have no inputs (this is less common)

```
> f <- function() {
+         cat("Hello, world!\n")
+ }
> f()
Hello, world!
```

A little more on function arguments

When you write your own function you have to decide on the arguments.

- What information do you need from the user? (eg. their data)
- Do you want the user to have control over any additional info (eg. number of samples they take, additional parameters etc.)

You can choose to have no inputs (this is less common)

```
> f <- function() {
+         cat("Hello, world!\n")
+ }
> f()
Hello, world!
```

You can set defaults. These become optional arguments.

```
f <- function(data, na.rm = TRUE){

ACTIONS

return (OUTPUT}
}
```

User must provide

if user doesn't provide NAs will be removed

```
Var1 <- f(df)
Var2 <- f(df, na.rm = FALSE)
```