# Example 1

The first thing to note is that the function receiving the ... does not itself need ... in its function signature. In the example below, f1 has ... in its function signature, and passes ... to f2. f2, does not have ... in its function signature but is able to interpret the call from f1 correctly.

f1 <- function(x, ...) { f2(...)

}

f2 <- function(y) { print(y)

}

f1(x = 1, y = 2)

# [1] 2

As one might expect, if f1 passes anything other than y, we get an error:

f1(x = 1, y = 2, z = 3)

# Error in f2(...) : unused argument (z = 3)

It’s interesting that if we do not specify y = in the f1 function call, R is smart enough to decipher what is going on and give the answer we expect. I don’t recommend writing such code though as it can be ambiguous to the reader.

f1(1, 2)

# [1] 2

# Example 2

This example is almost the same as the previous one except f2 now has ... in its function signature as well. The f1 function call with just x and y works as expected.

f1 <- function(x, ...) { f2(...)

}

f2 <- function(y, ...) { print(y)

}

f1(x = 1, y = 2)

# [1] 2

Note, however, that the call with x, y and z does not fail:

f1(x = 1, y = 2, z = 3)

# [1] 2

When f1 calls f2, the z argument goes into f2‘s It’s not used by the function, but it does

not throw an error because this is not an illegal input to f2. This may or may not be what you

want! I recently got burned by this because I was expecting f2 to throw an error but it didn’t, which made me think that my code was working when it wasn’t.

# Example 3

You can use list(...) to interpret the arguments passed through ... as a list. This can be useful if you want to amend the arguments before passing them on. Save the output of list(...) as a variable, amend this variable, then call the next function with the amended variable using do.call().

For example, in the code below, f1 checks to see if the argument y is passed. If so, it is doubled before being passed on to f2.

f1 <- function(x, ...) {

args <- list(...)

if ("y" %in% names(args)) { args$y <- 2 \* args$y

}

do.call(f2, args)

}

f2 <- function(y) { print(y)

}

f1(x = 1, y = 2)

# [1] 4