functions

September 27, 2020

Defining Functions in R
Paul Stey, Ph.D.
2020-09-21

1 1. Functions in R

- Small, re-usable code chuncks
- Take some input (arguments) and return some output
  - Thus, similar to functions in mathematical sense

1.1 1.1 R Built-in Functions we Have Used

- `print()`
- `c()`
- `mean()`
- `is.na()`

```
In [1]: print("hey buddy!")            # the "hey buddy!" is the argument passed to `print()`
```

```
[1] "hey buddy!"
```

```
In [2]: v <- c(4, 137, 151)            # pass 3 intergers to c() function

        print(v)                       # pass `v` to print()
```

```
[1]   4 137 151
```

1.2 1.1.1 Functions with Optional Arguments

```
In [3]: v <- c(2, 5, 7)

        mean(v)
```

4.66666666666667

```
In [4]: u <- c(4, 5, NA)

        mean(u)
```

<NA>

```
In [5]: mean(u, na.rm = TRUE)
```

4.5

2 2. Defining a Function

- Why?
  - Modularity, code clarity, re-usability, scope cleanliness

```
In [6]: # Below we define a function that takes a single argument
        # which we call `n` and then returns the sum of `n` and 2

        add_two <- function(n) {
            x <- n + 2
            return(x)
        }
```

```
In [7]: b <- 50

        a <- add_two(b)             # call our newly created function

        print(a)
```

```
[1] 52
```

```
In [8]: w <- add_two(137)

        print(w)
```

```
[1] 139
```

2.1 2.1 Defining a More Interesting Fucntion

- Functions can have many arguments
- And even a variable-number of arguments

```
In [9]: area_of_square <- function(len, wd) {

            area <- len * wd

            return(area)
        }
```

```
In [10]: area_of_square(4, 5)          # run our newly-defined function
```

20

```
In [11]: area                          # error
```

```
        Error in eval(expr, envir, enclos): object 'area' not found
    Traceback:
```

```
In [ ]: ls()                          # list elements in current "scope"
```