Exploring data 2

Functions

Functions

As you move to larger projects, you will find yourself using the same code a lot.

Examples include:

- Reading in data from a specific type of equipment (air pollution monitor, accelerometer)
- Running a specific type of analysis (e.g., fitting the same model format to many datasets)
- Creating a specific type of plot or map

If you find yourself cutting and pasting a lot, convert the code to a function.

Functions

Advantages of writing functions include:

- Coding is more efficient
- Easier to change your code (if you've cut and paste code and you want to change something, you have to change it everywhere)
- Easier to share code with others

Functions

You can name a function anything you want, as long as you follow the naming rules for all R objects (although try to avoid names of preexisting-existing functions). You then specify any inputs (arguments; separate multiple arguments with commas) and put the code to run in braces. You **define** a function as an R object just like you do with other R objects (<-).

Here is the basic structure of "where things go" in an R function definition.

```
## Note: this code will not run
[function name] <- function([any arguments]){
        [code to run]
}
```

Functions

Here is an example of a very basic function. This function takes a number as the input (`number`) and adds 1 to that number. An R function will only return one R object. By default, that object will be the last line of code in the function body.

```
add_one <- function(number){
        number + 1 # Value returned by the function
}

add_one(number = 1:3)
```

```
## [1] 2 3 4
```

```
add_one(number = -1)
```

```
## [1] 0
```

Functions

```
add_one <- function(number){
        number + 1 # Value returned by the function
}
```

- I picked the name of the function (`add_one`) (just like you pick what name you want to use with any R object)
- The only input is a numeric vector. I pick the name I want to use for the vector that is input to the function. I picked `number`.
- Within the code inside the function, the `number` refers to the numeric vector object that the user passed into the function.

Functions

As another example, you could write a small function to fit a specific model to a dataframe you input and return the model object:

```
fit_time_pos_mod <- function(df){
  lm(Tackles ~ Time + Position,
     data = df) # Returns result from this call
}
```

- I picked the name of the function (`fit_time_pos_mod`) (just like you pick what name you want to use with any R object)
- The only input is a dataframe. I pick the name I want to use for the dataframe that is input to the function. I picked `df` (I often use this as a default parameter name for a dataframe).
- Within the code inside the function, the `df` refers to the dataframe object that the user passed into the function.

Functions

Now you can apply that function within a tidy pipeline, for example to fit the model to a specific subset of the data (the top four teams):

```
data(worldcup)
worldcup %>%
  filter(Team %in% c("Spain", "Netherlands",
                     "Uruguay", "Germany")) %>%
  fit_time_pos_mod() %>%
  tidy()
```

```
## # A tibble: 5 x 5
##   term                estimate std.error statistic  p.value
##   <chr>                  <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)          -0.312    1.34       -0.232 8.17e- 1
## 2 Time                  0.0237   0.00268     8.85  3.60e-13
## 3 PositionForward      -3.31     1.49       -2.22  2.95e- 2
## 4 PositionGoalkeeper -12.6      2.66       -4.75  9.77e- 6
## 5 PositionMidfielder   2.23     1.32        1.68  9.68e- 2
```

Functions

- Functions can input any type of R object (for example, vectors, data frames, even other functions and ggplot objects)
- Similarly, functions can output any type of R object
- However, functions can only output one R object. If you have complex things you want to output, a list might be a good choice for the output object type.
- Functions can have "side effects". Examples include printing something or drawing a plot. Any action that a function takes *besides returning an R object* is a "side effect".