Functions

Functions in R

I've been denoting functions with parentheses: `func()`

We've seen functions such as:

- `mean()`
- `tbl_summary()`
- `init()`
- `create_github_token`

Functions take **arguments** and return **values**

Looking inside a function

If you want to see the code within a function, you can just type its name without the parentheses:

```
1 usethis::create_github_token
```

```
function (scopes = c("repo", "user", "gist", "workflow"), description = "DESCRIBE THE TOKEN'S USE
CASE",
    host = NULL)
{
    scopes <- glue_collapse(scopes, ",")
    host <- get_hosturl(host %||% default_api_url())
    url <- glue("{host}/settings/tokens/new?scopes={scopes}&description={description}")
    withr::defer(view_url(url))
    hint <- code_hint_with_host("gitcreds::gitcreds_set", host)
    message <- c(`_` = "Call {.run {hint}} to register this token in the local Git\n
credential store.")
    if (is_linux()) {
        message <- c(message, `!` = "On Linux, it can be tricky to store credentials
persistently.",
            i = "Read more in the {.href ['Managing Git(Hub) Credentials' article]
(https://usethis.r-lib.org/articles/articles/git-credentials.html)}.")
    }
    message <- c(message, i = "It is also a great idea to store this token in any\n
password-management software that you use.")
```

```
    ui_bullets(message)
    invisible()
}
<bytecode: 0x141ac26a0>
<environment: namespace:usethis>
```

Structure of a function

```
1  func <- function()
```

You can name your function like you do any other object

- Just avoid names of existing functions

Structure of a function

```
func <- function(arg1,
                 arg2 = default_val)
}
```

What objects/values do you need to make your function work?

- You can give them default values to use if the user doesn't specify others

Structure of a function

```
1 func <- function(arg1,
2                  arg2 = default_val) {
3
4 }
```

Everything else goes within curly braces

- Code in here will essentially look like any other R code, using any inputs to your functions

Structure of a function

```
1 func <- function(arg1,
2                  arg2 = default_val) {
3   new_val <- # do something with args
4 }
```

- One thing you'll likely want to do is make new objects along the way
- These aren't saved to your environment (i.e., you won't see them in the upper-right window) when you run the function
- You can think of them as being stored in a temporary environment within the function

Structure of a function

```
1 func <- function(arg1,
2                  arg2 = default_val) {
3   new_val <- # do something with args
4   return(new_val)
5 }
```

Return something new that the code has produced

- The `return()` statement is actually optional. If you don't put it, it will return the last object in the code. When you're starting out, it's safer to always explicitly write out what you want to return.

Example: a new function for the mean

Let's say we are not satisfied with the `mean()` function and want to write our own.

Here's the general structure we'll start with.

```
1 new_mean <- function() {
2
3 }
```

New mean: arguments

We'll want to take the mean of a vector of numbers.

It will help to make an example of such a vector to think about what the input might look like, and to test the function. We'll call it `x`:

```
1 x <- c(1, 3, 5, 7, 9)
```

We can add `x` as an argument to our function:

```
1 new_mean <- function(x) {
2
3 }
```

New mean: function body

Let's think about how we calculate a mean in math, and then translate it into code:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

So we need to sum the elements of `x` together, and then divide by the number of elements.

We can use the functions `sum()` and `length()` to help us.

We'll write the code with our test vector first, before inserting it into the function:

```
1  n <- length(x)
2  sum(x) / n
```

```
[1] 5
```

New mean: function body

Our code seems to be doing what we want, so let's insert it. To be explicit, I've stored the answer (within the function) as `mean_val`, then returned that value.

```
1 new_mean <- function(x) {
2   n <- length(x)
3   mean_val <- sum(x) / n
4   return(mean_val)
5 }
```

Testing a function

Let's plug in the vector that we created to test it:

```
new_mean(x = x)
```

```
[1] 5
```

And then try another one we create on the spot:

```
new_mean(x = c(100, 200, 300))
```

```
[1] 200
```

Exercises

Create some functions!

Create an R script in your project called `functions.R` to save your work!