

An introduction functions in R

https://bradduthie.github.com/talks/R_functions.pdf

Brad Duthie (alexander.duthie@stir.ac.uk)

06 December 2023

What is a function?

Functions are ubiquitous and highly useful in R

- ▶ Code that is organised to perform a specific task¹
- ▶ Can get functions in multiple ways
 1. R base functions
 2. R package functions
 3. Custom functions
- ▶ All functions have a similar structure
- ▶ **Today you will learn to write functions**

¹R-Functions. Tutorialspoint.

What is a function?

```
function_name <- function(arg1, arg2 = default){  
  # Code that can use arguments, arg1 & arg2  
  return(output);  
}
```

What is a function?

```
function_name <- function(arg1, arg2 = default){  
  # Code that can use arguments, arg1 & arg2  
  return(output);  
}
```

```
my_mean <- mean(x = 1:10);
```

```
## [1] 5.5
```

What is a function?

```
function_name <- function(arg1, arg2 = default){  
  # Code that can use arguments, arg1 & arg2  
  return(output);  
}
```

```
my_mean <- mean(x = 1:10);
```

```
## [1] 5.5
```

```
class(mean);
```

```
## [1] "function"
```

Base R includes hundreds of functions

- ▶ Most **base functions** not used¹
- ▶ Familiar functions `mean`, `plot`, `summary`
- ▶ Includes functions like `+`, `<-`, `"`, or `!`

Additional functions can be found in R packages, or custom made and read into the R console.

¹<https://stat.ethz.ch/R-manual/R-devel/library/base/html/00Index.html>

Non-base functions in R

Functions outwith base R available in packages

- ▶ [Comprehensive R Archive Network](#) includes 18000+ packages
- ▶ Packages include specialised functions
- ▶ Access with 'install.packages' and 'library'

```
install.packages("ggplot2");  
library("ggplot2");  
ggplot(data = dat, mapping = aes(x = wgt, y = totlen))  
  + geom_point();
```

Custom functions can be written in R too with the `function` function.

A custom function in R

Convert from Fahrenheit to Celsius

```
F_to_C <- function(F_temp){  
  C_temp <- (F_temp - 32) * 5/9;  
  return(C_temp);  
}
```

Highlight the whole function and run it, then you can use it.

```
F_to_C(F_temp = 70);
```

```
## [1] 21.11111
```

Now write a custom function for C to F!

Functions within functions

We can use a custom function within another custom function.

Convert from Fahrenheit to Kelvin.

```
F_to_K <- function(F_temp){  
  K_temp <- F_to_C(F_temp = F_temp) + 273.15;  
  return(K_temp);  
}
```

Because Kelvin equals degrees Celsius plus 273.15, we can call `F_to_C`, then add 273.15 to it.

Functions can go in functions

```
F_convert <- function(F_temp = 70,  
                      conversion = "Celsius"){  
  if(conversion == "Celsius"){  
    converted <- F_to_C(F_temp = F_temp);  
  }  
  if(conversion == "Kelvin"){  
    converted <- F_to_K(F_temp = F_temp);  
  }  
  return(converted);  
}
```

Convert to Kelvin again.

```
F_convert(F_temp = 70, conversion = "Kelvin");
```

```
## [1] 294.2611
```

Always good to add error messages

```
F_convert <- function(F_temp = 70,  
                      conversion = "Celsius"){  
  if(conversion != "Celsius" & conversion != "Kelvin"){  
    stop("'conversion' must be 'Celsius' or 'Kelvin'.");  
  }  
  if(is.numeric(F_temp) == FALSE){  
    stop("F_temp argument must be numeric");  
  }  
  if(conversion == "Celsius"){  
    converted <- F_to_C(F_temp = F_temp);  
  }else{  
    converted <- F_to_K(F_temp = F_temp);  
  }  
  return(converted);  
}
```

Some additional points

- ▶ Recursive functions
- ▶ Function environment
- ▶ Order of arguments
- ▶ Function returns
- ▶ `do.call` function