Lecture 18: Functions

STAT 450, Fall 2021

In this class we have already used many built-in functions in R. For example: `mean()`, `sd()`, `summary()`, `is.na()` and `plot()`.

In this lecture, we will discuss how you can write your own functions in R.

Writing Functions in R

Functions are created using the `function()` keyword and given a name using the `<-` assignment operator. The code in between the curly brackets `{}` defines the body of the function. The return value of a function is the very last expression that is evaluated. Here is a basic examples:

```
square <- function(x) {
  x^2
}
square(5)
```

```
## [1] 25
```

```
square(1:5)
```

```
## [1]  1  4  9 16 25
```

The inputs of a function are called **arguments**. The function `square()`, defined above, only has one argument: a numeric vector `x`. Here is an example of a function with two arguments:

```
pow <- function(x, n) {
  x^n
}
pow(5, 2)
```

```
## [1] 25
```

```
pow(5, 3)
```

```
## [1] 125
```

```
pow(1:5, 3)
```

```
## [1]   1   8  27  64 125
```

Function Arguments

Default values for arguments of a function can be specified using `=`

```
pow <- function(x, n, k = 1) {
  k * x^n
}
pow(5, 2) # uses default
```

```
## [1] 25
```

```
pow(5, 2, 3)
```

```
## [1] 75
```

When calling a function, if you specify arguments by name then the ordering does not matter:

```
pow(5, n=2, k=3)
```

```
## [1] 75
```

```
pow(5, k=3, n=2)
```

```
## [1] 75
```

You have already been using the defaults for common R functions. For example, the `sd()` function has two arguments: a numeric vector `x`, and a logical value `na.rm` which indicates whether missing data should be removed. The default is `na.rm = FALSE`.

```
x <- c(20, 1 , 1, 0, 10, 8 , 2)
sd(x)
```

```
## [1] 7.28011
```

```
y <- c(20, 1 , 1, NA, 0, 10, 8 , 2, NA)
sd(y, na.rm = TRUE)
```

```
## [1] 7.28011
```

Here's another example, using the base R function `plot()`, which shows that if you specify arguments by name, then the ordering of the arguments does not matter.

```
plot(mtcars$wt, mtcars$mpg, xlab = "weight", ylab = "mpg", col = "blue")
```

```
# this makes the exact same scatter plot:
plot(mtcars$wt, mtcars$mpg, col = "blue", xlab = "weight", ylab = "mpg")
```

Exercise: Consider the following code from last class which simulates a coin flip:

```
u <- runif(1) # generate random number between 0 and 1
if(u > 0.5) {
  print("Heads")
} else {
  print("Tails")
}
```

```
## [1] "Tails"
```

Write an R function called `flip_coin()`, which has one argument called `n` that specifies the number of flips [hint: use a `for` loop in your function]. Set the default `n=1`. Here's an example of what the output of your function should look like:

```
flip_coin()
```

```
## [1] "Heads"
```

```
flip_coin(5)
```

```
## [1] "Tails"
## [1] "Heads"
## [1] "Heads"
## [1] "Tails"
## [1] "Heads"
```

Exercise: The function `is.na()` can be used to check for missing data. For example:

```
x <- c(6, 21, NA, NA, 12, NA, 23, 15)
is.na(x)
```

```
## [1] FALSE FALSE  TRUE  TRUE FALSE  TRUE FALSE FALSE
```

```
sum(is.na(x)) # counts NA values
```

```
## [1] 3
```

Write a function called `count_na()` that counts the number of `NA` values in a vector. This is what the output of your function should look like:

```
x <- c(6, 21, NA, NA, 12, NA, 23, 15)
count_na(x)
```

```
## [1] 3
```

```
count_na(mtcars$mpg)
```

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

```
count_na(airquality$Ozone)
```

```
## [1] 37
```

Global versus local variables

Any variables defined within a function are local to that function. Here is an example:

```
x <- 0
n <- 1
f1 <- function(x, y) {
  n <- 2
  x^n + y^n
}
f1(x = 2, y = 3)
```

```
## [1] 13
```

```
x
```

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

```
n
```

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