STAT 33B Lab 5

Ziang Gao

This assignment is due **Mar 16, 2020** by 11:59pm.

Edit this file, knit to PDF, and:

- Submit the Rmd file on bCourses.
- Submit the PDF file on Gradescope.

If you think you'll need help with submission, please ask during the lab.

Answer all questions with complete sentences, and put code in code chunks. You can make as many new code chunks as you like. Please do not delete the exercises already in this notebook, because it may interfere with our grading tools.

As you work, you may find it helpful to be able to run your code. You can run a single line of code by pressing Ctrl + Enter. You can run an entire code chunk by clicking on the green arrow in the upper right corner of the code chunk.

Knit the document from time to time to make sure that your code runs without errors from top to bottom in a fresh R environment.

Writing Functions

In this lab, you'll practice writing functions.

Exercise 1

Write a function `to_kelvin()` to convert temperatures in degrees Celsius to temperatures in degrees Kelvin (you can find the formula online). Your function should have a parameter `celsius` for the temperature to convert.

Make sure that your function is vectorized in `celsius`, so that it can convert an entire vector of temperatures in one call.

Test your function to show that it works correctly.

```
# Your code goes here.
to_kelvin = function(celsius) {
  celsius + 273.15
}

to_kelvin(10)
```

```
## [1] 283.15
```

```
to_kelvin(c(10, 20, 30))
```

```
## [1] 283.15 293.15 303.15
```

Exercise 2

Extend your `to_kelvin()` function to be able to convert degrees Fahrenheit to degrees Kelvin as well. Add a parameter `fahrenheit` for the Fahrenheit temperature to convert.

Leave the code in the body of your function unchanged. Instead, provide a default argument for `celsius` that converts the `fahrenheit` argument.

Test your function to show that it works correctly. Include test cases for both Celsius and Fahrenheit temperatures (it's okay to reuse a test case from Exercise 1).

What happens if you call your function with arguments to the `celsius` parameter and the `fahrenheit` parameter at the same time?

```
# Your code goes here.
to_kelvin = function(celsius = (fahrenheit - 32) * 5 / 9, fahrenheit) {
  c(celsius + 273.15, (fahrenheit - 32) * 5 / 9 + 273.15)
}

to_kelvin(, 100)
```

```
## [1] 310.9278 310.9278
```

```
to_kelvin(100, 200)
```

```
## [1] 373.1500 366.4833
```

YOUR WRITTEN ANSWER GOES HERE: If I call my function with arguments to the celsius oarameter and the afhrenheit parameter the same time, it will out put two numbers where the first one is the kelvin representation of the input celsius, and the second one is the kelvin representation of the input fahrenheit.

Exercise 3

Rather than using a separate parameter in `to_kelvin()` for each source unit, a better strategy is to have a `temperature` parameter and a `unit` parameter. The function can then check `unit` and choose an appropriate conversion for `temperature`.

The `unit` parameter should be restricted to supported units, which in this case are `"celsius"` and `"fahrenheit"`. You can use the special `match.arg()` function to check that an argument matches against a set of candidate arguments. For instance, `match.arg(x, c("red", "blue"))` checks that the argument to `x` is either `"red"` or `"blue"`. The function also allows for partial matches, so for instance `"r"` matches `"red"`. See the documentation for full details.

Rewrite the `to_kelvin()` function with a `temperature` and `unit` parameter. Make sure the function is still vectorized in the `temperature` parameter.

Test your function to show that it works correctly. Include test cases for both Celsius and Fahrenheit temperatures.

```
# Your code goes here.

to_kelvin = function(temperature, unit) {
  u = match.arg(unit, c("celsius", "fahrenheit"), TRUE)
  switch(u,
         "celsius" = temperature + 273.15,
         "fahrenheit" = (temperature - 32) * 5 / 9 + 273.15)
}

to_kelvin(100, "celsius")
```

```
## [1] 373.15
```

```
to_kelvin(c(200, 100), "fahrenheit")
```

```
## [1] 366.4833 310.9278
```

Exercise 4

The `stop()` function prints an error message and stops evaluation (exiting any called functions). For example, `stop("Something went wrong!")` prints the error message `Something went wrong!` and then stops evaluation.

Rewrite your `to_kelvin()` function from Exercise 3 so that it is also vectorized in both the `temperature` and `unit` parameter. Specifically:

- If `unit` is a scalar, use it to convert each element of `temperature`.
- If `unit` is a vector with the same length as `temperature`, use each element of `unit` to convert the corresponding element of `temperature`.
- If `unit` is a vector with a different length from `temperature`, print an appropriate error message and stop evaluation.

Do this without using any explicit loops (vectorization is okay).

Hint: Use subsetting or `ifelse()` to convert Fahrenheit temperatures to Celsius before converting to Kelvin.

```
# Your code goes here.
to_kelvin = function(temperature, unit) {
  tl = length(temperature)
  ul = length(unit)
  if (ul == 1) {
    if (unit == "celsius") {
      return(temperature + 273.15)
    } else {
      return((temperature - 32) * 5 /9 + 273.15)
    }
  } else if (tl == ul) {
    index_c = which(unit == "celsius")
    index_f = which(unit == "fahrenheit")
    temperature[index_c] = temperature[index_c] + 273.15
    temperature[index_f] = (temperature[index_f] - 32) * 5/ 9 + 273.15
    return(temperature)
  } else {
    stop("Unmatched length")
  }
}

to_kelvin(c(1, 2,3), c("celsius", "celsius", "celsius"))
```

```
## [1] 274.15 275.15 276.15
```