STAT 33B Workbook 6

Ming Fong (3035619833)

Oct 8, 2020

This workbook is due **Oct 8, 2020** by 11:59pm PT.

The workbook is organized into sections that correspond to the lecture videos for the week. Watch a video, then do the corresponding exercises *before* moving on to the next video.

Workbooks are graded for completeness, so as long as you make a clear effort to solve each problem, you'll get full credit. That said, make sure you understand the concepts here, because they're likely to reappear in homeworks, quizzes, and later lectures.

As you work, write your answers in this notebook. Answer questions with complete sentences, and put code in code chunks. You can make as many new code chunks as you like.

In the notebook, you can run the line of code where the cursor is by pressing `Ctrl` + `Enter` on Windows or `Cmd` + `Enter` on Mac OS X. You can run an entire code chunk by clicking on the green arrow in the upper right corner of the code chunk.

Please do not delete the exercises already in this notebook, because it may interfere with our grading tools.

You need to submit your work in two places:

- Submit this Rmd file with your edits on bCourses.
- Knit and submit the generated PDF file on Gradescope.

If you have any last-minute trouble knitting, **DON'T PANIC**. Submit your Rmd file on time and follow up in office hours or on Piazza to sort out the PDF.

Functions

Watch the "Functions" lecture video.

No exercises for this section.

Default Arguments

Watch the "Default Arguments" lecture video.

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.

Show that your function works correctly for a few test cases.

YOUR ANSWER GOES HERE:

```
to_kelvin = function(celsius) {
    celsius + 273.15
}
to_kelvin(c(15, 0, 1, 100))
```

```
## [1] 288.15 273.15 274.15 373.15
```

```
to_kelvin(1234)
```

```
## [1] 1507.15
```

Function Example

The "Function Example" lecture video is optional. It provides a more realistic example of writing a function.

No exercises for this section.

Variable Scope & Lookup

Watch the "Variable Scope & Lookup" lecture video.

No exercises for this section.

Lazy Evaluation

Watch the "Lazy Evaluation" lecture video.

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.

Show that your function works correctly for a few test cases (for both Celsius and Fahrenheit temperatures).

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

YOUR ANSWER GOES HERE:

Passing both the `celsius` and `fahrenheit` parameters will only use the `celsius` parameter. This is because fahrenheit only works when it overrides the `celsius` default value.

```
to_kelvin = function(celsius = (fahrenheit - 32) * 5 / 9, fahrenheit) {
    celsius + 273.15
}
to_kelvin(celsius = 100)
```

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

```
to_kelvin(fahrenheit = 32)
```

```
## [1] 273.15
```

```
to_kelvin(fahrenheit = c(14, 32, 100))
```

```
## [1] 263.1500 273.1500 310.9278
```

```
to_kelvin(celsius = 0, fahrenheit = c(14, 32, 100))
```

```
## [1] 273.15
```

Exercise 3

Rather than using a separate parameter in `to_kelvin()` for each source unit, a better design 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.

As always, show that your function works correctly for a few test cases.

YOUR ANSWER GOES HERE:

```
to_kelvin = function(temperature, unit) {
    unit = match.arg(unit, c("celsius", "fahrenheit"))
    if (unit == "celsius") {
        temperature + 273.15
    } else if (unit == "fahrenheit") {
        (temperature - 32) * 5 / 9 + 273.15
    }
}
to_kelvin(0, "celsius")
```

```
## [1] 273.15
```

```
to_kelvin(c(32, 10, 100), unit = "f")
```

```
## [1] 273.1500 260.9278 310.9278
```

```
to_kelvin(c(23, 0, -100, 20), unit = "cel")
```

```
## [1] 296.15 273.15 173.15 293.15
```

The Dots Parameter

Watch the "The Dots Parameter" lecture video.

No exercises for this section.

Using Functions

Watch the "Using Functions" lecture video.

No exercises for this section.