Stats 102A - Homework 1 Instructions

Homework 1 Requirements

You will submit three files.

The files you submit will be:

1. `102a_hw_01_script_first_last.R` Your R script file containing the functions you write for the homework assignment. Write comments as necessary. **Submit this file to BruinLearn**
2. `102a_hw_01_output_first_last.qmd` Take the provided Quarto file and make the necessary edits so that it will generate the requested output. The first line of your Quarto file should be to `source()` the R script file you wrote. You will also need to go through the rest of the file and insert your answers to any questions. **Submit this file to BruinLearn**
3. `102a_hw_01_output_first_last.pdf` Your output PDF file. This is the primary file that will be graded. Make sure all requested output is visible in the output file. **Submit this file to Gradescope. Be sure to tag all pages properly on Gradescope. Output that is not tagged properly will not be graded.**

Failure to submit all 3 files will result in a penalty.

Academic Integrity

At the top of your Quarto file, be sure to modify the Academic Integrity statement with your name.

Part 1

Write a function called `by_type()` that takes an atomic vector as input and an optional argument 'sort' with default value FALSE.

The function looks at each element in the atomic vector and sees if it can coerce them into integer or floating-point values. The function outputs a list that has separated the values into a list with integers, doubles, and character values. If the optional argument `sort` is `TRUE`, then it will sort the results of each vector.

For example, if x is the following vector,

```
x <- c("house", "6", "2.2", "a", "3.4", "1")
```

Then the output of `by_type(x)` will be a list:

```
$integers
[1] 6 1

$doubles
[1] 2.2 3.4

$character
[1] "house"     "a"
```

If `sort = TRUE` then the output will sort each section and return:

```
by_type(x, sort = TRUE)

$integers
[1] 1 6

$doubles
[1] 2.2 3.4

$character
[1] "a"     "house"
```

Unlike Python, numbers such as `5.0` and `5` will both be classified as integers.

If the vector contains logical TRUE/FALSE values, those should be put into the character section.

If the input has no values of a certain type, it should show a zero-length vector of that type. For example, if there are no character values, the list should have `character(0)` in the `$character`

part of the list. `NA` values can be assumed to be of character type and when sorted will show up at the end after all other characters. `NA` should appear as `NA` (the value representing missing) and not `"NA"` (in quotes).

No warnings should be produced in your output. It is acceptable to use the command `suppressWarnings()`

Part 2

Write a function named `prime_factor(x)` that will find the prime factorization of an integer `x`. The function will output a numeric vector. All of the values in the output vector will be prime. The product of the numeric vector will be the original value x. There should be a check to make sure that the input value is a number greater than or equal to 2 with no decimal values with appropriate error messages.

If you're stuck on getting started with this, I suggest doing some prime factorizations yourself. For example, try to find the prime factors of 171 or 364. Note the steps you go through to find these prime factors and see if you can create code to do it.

You should not need to use the `is_prime()` function in your `prime_factor()` function.

(The point of the exercise is to give you practice writing code. Yes, I am aware that solutions to this problem already exist on the Internet or that you can ask ChatGPT to give you the solution. Please work out your own solution. While the problem is simple, it is complex enough that it is incredibly unlikely for students to create identical code solutions. You are free to *talk* about your approach and I fully expect many students to take identical approaches, but students writing identical code is an entirely different matter.)

Part 3

Write a function named `month_convert(x, from_lang, to_lang)` that will convert month names from one language to another. The function will accept three arguments: `x`, which is a factor with month information; the language from which we are translating `from_lang`; and the language to which we are translating `to_lang`.

For example:

```
x <- factor(c("March","March","February","June"))
month_convert(x, "English", "Spanish")
```

will output:

```
[1] marzo   marzo   febrero junio
Levels: febrero junio marzo
```

If the input contains a value that is not a real month, then it will be replaced with NA. Any values that are not real months should have the corresponding levels removed as well.

```
x <- factor(c("March","March","February","June","Jaly"))
month_convert(x, "English", "Spanish")
[1] marzo   marzo   febrero junio   <NA>
Levels: febrero junio marzo
```

The order of the levels should remain the same before and after conversion between languages (they should not be realphabetized).

```
x <- factor(c("August", "April", "December", "April"))
x
[1] August   April    December April
Levels: April August December

month_convert(x, "English", "Italian")
[1] agosto   aprile   dicembre aprile
Levels: aprile agosto dicembre
```

I have uploaded a file called "month_names.txt" on Canvas which has the month names in several European languages. Use `month_names <- read.delim("month_names.txt", encoding="UTF-8", row.names=1)` to import the file.

Part 4

There are several questions in the Quarto file that you will need to answer with words.