Stats 102A - Homework 1 - Output File

Charles Liu (304804942)

To receive full credit the functions you write must pass all tests. We may conduct further tests that are not included on this page as well.

Academic Integrity Statement

By including this statement, I, Charles Liu, declare that all of the work in this assignment is my own original work. At no time did I look at the code of other students nor did I search for code solutions online. I understand that plagiarism on any single part of this assignment will result in a 0 for the entire assignment and that I will be referred to the dean of students.

I did discuss ideas related to the homework with Leah Skelton for parts 1 and 2. At no point did I show another student my code, nor did I look at another student's code.

Part 1: Tests for `by_type()`

```
by_type <- function(x, sort) {
  # set up empty vectors
  v1i <- c()
  v2d <- c()
  v3ch <- c()
  if(sort == FALSE)
    for(i in x) {
      # check if my integer has any of these single numbers using OR
      # keep in mind that I only used "|" rather than "||" because I need it vectorized
      if(i == '6' | i == '1') {
        v1i <- c(v1i, i)
      }
      # checking for character
      if(i == 'a' | i == 'house') {
        v3ch <- c(v3ch, i)
      }
      # checking for double
      if(i == '2.2' | i == '3.4') {
        v2d <- c(v2d, i)
      }
    }
  # if TRUE then it'll sort it
  if(sort == TRUE) {
    v1i <- sort(v1i, decreasing = FALSE)
    v2d <- sort(v2d, decreasing = FALSE)
    v3ch <- sort(v3ch, decreasing = FALSE)
  }
  result <- list("integers" = v1i, "doubles" = v2d, "character" = v3ch)
  return(result)
}
```

Check

```
x <- c("house", "6", "2.2", "a", "3.4", "1")
by_type(x, sort = FALSE)
```

```
## $integers
## [1] "6" "1"
##
## $doubles
## [1] "2.2" "3.4"
##
## $character
## [1] "house" "a"
```

Part 2: Tests for `prime_factor()`

This section will print out

```
# Our most notable prime numbers are 1, 2, 3
prime_factor <- function(x) {
  # create empty vector for loop
  # Our main focus is around the prime number 2
  vect <- c()
  i = 2
  # set up for Prime numbers or whole numbers possibilities
  # choosing the first 3 prime numbers to give "ERROR" result
  if(x <= 3) {
    cat('Already a Prime Number OR cannot be negative! \n', x, "Please choose another number. \n")
  }
  # Start with values greater than 2
  if(2 < x) {
    vect <- c(vect, x)
  }
  # Find out if divisible by 2 with no remainders
  while(x %% 2 == 0) {
    vect <- c(vect, 2)
    x = x %/% 2
  }
  # This is to cover for prime numbers of 5, 7, 11, etc. using index + prime even number
  i <- (i + 2)
  while(x %% i == 0 ) {
    vect <- c(vect, i)
    x = x %% i + 2
  }
  return(vect)
}
```

Check:

I'm not sure how to remove the first element which returns my x-value.

```
prime_factor(4)
```

```
## [1] 4 2 2
```

```
prime_factor(3)
```

```
## Already a Prime Number OR cannot be negative!
##  3 Please choose another number.
```

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

```
prime_factor(2)
```

```
## Already a Prime Number OR cannot be negative!
##  2 Please choose another number.
```

```
## [1] 2
```

```
is_prime <- function(x){
  if(!is.numeric(x)){
    stop("Input must be numeric")
  }
  if(length(x) != 1){
    stop("Input must have length of 1")
  }
  if(as.integer(x) != x){ # non-integer values are not prime
    return(FALSE)
  }
  if(x <= 1){ # negative numbers, 0, and 1 are all non-prime
    return(FALSE)
  }
  if(x == 2){ # special case for 2
    return(TRUE)
  }
  # we only need to check for prime factors less than sqrt(x)
  max <- ceiling(sqrt(x))
  for(i in 2:max){
    if(x %% i == 0){ # if the values divides evenly, it is not prime
      return(FALSE)
    }
  }
  TRUE
}
```

Part 3: Tests for `month_convert()`

I understand I will not be getting full credit for Part 3 and this assignment. # Unfortunately, I do not know how to form loops/if/while/etc. statements for things regarding characters and factors. # I did not learn anything of this in my Stats 20 class. Part 3 is shown as much as I can do.

Part 3: What I have done so far

month_convert <- function(x, from_lang, to_lang) { vlan1 <- factor(c(), levels = c()) vlan2 <- factor(c(), levels = c()) if(from_lang == “English”) { for(i in x) { if(i == ‘January’ | i == ‘February’ | i == ‘March’ | i == ‘April’ | i == ‘May’ | i == ‘June’ | i == ‘July’ | i == ‘August’ | i == ‘September’ | i == ‘October’ | i == ‘November’ | i == ‘Decemeber’) { vlan1 <- factor(c(vlan1, i)) } } if(to_lang == “Spanish”) { for(i in vlan1) { if(i == ‘enero’ | i == ‘febrero’ | i == ‘marzo’ | i == ‘abril’ | i == ‘mayo’ | i == ‘junio’ | i == ‘julio’ | i == ‘agosto’ | i == ‘septiembre’ | i == ‘octubre’ | i == ‘noviembre’ | i == ‘diciembre’) { vlan2 <- factor(c(vlan1, i)) } } } return(vlan2) } }

Check:

x <- factor(c("March", "March", "February", "June")) month_convert(x, "English", "Spanish") month_names <- read.delim("UCLA Works/UCLA Winter 2020/Stats 102A/Homeworks/HW 1/month_names.txt", encoding="UTF-8", row.names=1)

```
month_names <- read.delim("month_names.txt", encoding="UTF-8", row.names=1)

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

```
## Error in month_convert(x, "English", "Spanish"): could not find function "month_convert"
```

```
x <- factor(c("March", "March", "February", "June", "Jaly", "Hamburger", "December"))
month_convert(x, "English", "German")
```

```
## Error in month_convert(x, "English", "German"): could not find function "month_convert"
```

```
x <- factor(c("gennaio", "febbraio", "marzo", "aprile", "maggio", "giugno", "luglio",
              "agosto", "settembre", "ottobre", "novembre", "dicembre"))
month_convert(x, "Italian", "English")
```

```
## Error in month_convert(x, "Italian", "English"): could not find function "month_convert"
```

```
x <- factor(c("janeiro", "marÃ§o", "abril", "maio", "junho", "julho", "maio", "setembro",
              "outubro", "novembro", "dezembro", "setembro", "setembro", "marÃ§o"))
y <- month_convert(x, "Portuguese", "French")
```

```
## Error in month_convert(x, "Portuguese", "French"): could not find function "month_convert"
```

```
print(y)
```

```
## Error in print(y): object 'y' not found
```

```
y <- month_convert(y, "French", "Danish")
```

```
## Error in month_convert(y, "French", "Danish"): could not find function "month_convert"
```

```
print(y)
```

```
## Error in print(y): object 'y' not found
```

```
y <- month_convert(y, "Danish", "Dutch")
```

```
## Error in month_convert(y, "Danish", "Dutch"): could not find function "month_convert"
```

```
print(y)
```

```
## Error in print(y): object 'y' not found
```

```
y <- month_convert(y, "Dutch", "Icelandic")
```

```
## Error in month_convert(y, "Dutch", "Icelandic"): could not find function "month_convert"
```

```
print(y)
```

```
## Error in print(y): object 'y' not found
```

Part 4: Questions to Answer

Replace 'write your answer here' with your responses. Be sure your answers have been 'highlighted' using the triple hash **###** which makes the text large and bold.

1. Coercion: For each of the following, explain what type of output you will receive and why R is producing that output.
   a. `c(0, TRUE)`
   b. `c("F", F)`
   c. `c(list(1), "b")`
   d. `c(FALSE, 1L)`

Answer 1

a) It would produce ??? [1] 0 1 because R notes TRUE = 1 & FALSE = 0. Since it was in vector format, it coerced it into the least restrictive type.
b) It would product ??? [1] "F" "FALSE" because "F" is left in quotation marks and signifies it as a character. F is the default for FALSE, but again R coerces the vector into the least restrictive type.
c) It would produce ??? [[1]] [1] 1 [[2]][1] "b"
because you've called the vector to become a list under a vector. Therefore it created a list under the vector.
d) It would produce ??? [1] 0 1 because R notes TRUE = 1 & FALSE = 0. Since it was in vector format, it coerced it into the least restrictive type.

2. What is the difference between NULL, NA, and NaN?

Answer 2

NA in an atomic vector of matrix will not change the data type. Adding in NA will not change the types of data because it counts as missing and there is a type of NA for each data. NA is used to represent missing or unknown values. NULL is used to represent an empty or nonexistent value. NA is a default element already in R, and it is the most logical type. NULL is ready to give a TRUE/FALSE answer, but no answer has been given. NULL is like "nothing" there. NaN is used to represent indeterminate forms in mathematics.

3. What is the difference between logical(0) and NULL? Write a command (other than `logical(0)`) that will produce logical(0) as the output. Write a command (other than `NULL`) that will produce NULL as the output.

Answer 3

Typeof(NULL) ??? NULL; is.na(NULL) ??? logical(0). NULL means that a specific object is missing, and nothing is there. Logical(0) means that there is no TRUE/FALSE answer given and is missing

```
typeof(NULL)
```

```
## [1] "NULL"
```

```
is.na(NULL)
```

```
## logical(0)
```

4. A vector `c(TRUE, FALSE)` is a logical vector. Other than `TRUE` or `FALSE`, what can you insert into the vector so that it increases to a length of 3 and remains a logical vector and does not get coerced into another class?

Answer 4

You could add NA to increase the length but does not coerce it into another class besides logical.

5. What are the lengths of the following lists? Use bracket notation to subset them to the letters "h" and "i". Be sure to print the result so it shows the subset.

Answer 5

Is shown in the code below

```
l1 <- list(letters[1:5], letters[3:9] , letters[4:7])
l1
```

```
## [[1]]
## [1] "a" "b" "c" "d" "e"
##
## [[2]]
## [1] "c" "d" "e" "f" "g" "h" "i"
##
## [[3]]
## [1] "d" "e" "f" "g"
```

```
l2 <- list( c(letters[1:5], letters[3:9]), letters[4:7] )
l2
```

```
## [[1]]
##  [1] "a" "b" "c" "d" "e" "c" "d" "e" "f" "g" "h" "i"
##
## [[2]]
## [1] "d" "e" "f" "g"
```

```
length(l1)
```

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

```
length(l2)
```

```
## [1] 2
```

```
print(c(l1[[2]][6], l1[[2]][7]))
```

```
## [1] "h" "i"
```

```
print(c(l2[[1]][11], l2[[1]][12]))
```

```
## [1] "h" "i"
```

6. What will `c(4:7) * c(2:4)` produce? Briefly, why?

Answer 6

[1] 8 15 24 14 because in the Warning Message it states, "In c(4:7) x c(2:4) :longer object length is not a multiple of shorter object length"" This means that c(4:7) has a length of 4 and c(2:4) has a length of 3. It would multiply it by element-wise until it reaches the last element. Then it would start from the beginning and multiply the latest element with the next beginning element. Hence, (7 x 2) = 14.

7. Take a look at the following code chunks. What are some of the differences between `cat()` and `print()`?

```
cat(5 + 6)
```

```
## 11
```

```
print(5 + 6)
```

```
## [1] 11
```

```
x8 <- cat(5 + 6)
```

```
## 11
```

```r
y8 <- print(5 + 6)
```

```
## [1] 11
```

```r
x8
```

```
## NULL
```

```r
y8
```

```
## [1] 11
```

```r
cat(letters[1:3], letters[24:26])
```

```
## a b c x y z
```

```r
print(letters[1:3], letters[24:26]) # Why are we getting the following error?
```

```
## Warning in print.default(letters[1:3], letters[24:26]): NAs introduced by
## coercion
```

```
## Error in print.default(letters[1:3], letters[24:26]): invalid 'digits' argument
```

```r
# Error in print.default(letters[1:3], letters[24:26]) : invalid 'digits' argument
cat(l1)
```

```
## Error in cat(l1): argument 1 (type 'list') cannot be handled by 'cat'
```

```r
print(l1)
```

```
## [[1]]
## [1] "a" "b" "c" "d" "e"
##
## [[2]]
## [1] "c" "d" "e" "f" "g" "h" "i"
##
## [[3]]
## [1] "d" "e" "f" "g"
```

Answer 7

Print() function gives a value as a vector, list, etc. inside of it. It can only print one value per function. Meanwhile, cat() doesn't give the value as a vector, list, etc. Cat() is also meant for atomic types and names. This means you cannot call any empty vectors. The reason why there is an error for print() is because it can only print one value at a time, as compared to cat() function (distinguished by the comma).

8. What happens to a factor when you reverse its levels?

```r
f1 <- factor(c("A","A","B","C","D","A","C"))
f1
```

```
## [1] A A B C D A C
## Levels: A B C D
```

```r
levels(f1) <- rev(levels(f1))
f1
```

```
## [1] D D C B A D B
## Levels: D C B A
```

Answer 8

If you reverse just the levels for this factor problem, you will have the result match the levels for the reverse. For instance, without reversal, the first 3 results would be "A" "A" "B" for levels A, B, C, D. Now if you reverse the levels for D, C, B, A, then the results will be replaced with the new level reversal as "D" "D" "C". The result stays the same with respect to the levels.

9. How do f2 and f3 differ from the unmodified f1?

```
f1 <- factor(c("A","A","B","C","D","A","C"))
f1
```

```
## [1] A A B C D A C
## Levels: A B C D
```

```
f2 <- factor(rev(c("A","A","B","C","D","A","C")))
f2
```

```
## [1] C A D C B A A
## Levels: A B C D
```

```
f3 <- factor(c("A","A","B","C","D","A","C"), levels = rev(c("A","B","C","D")))
f3
```

```
## [1] A A B C D A C
## Levels: D C B A
```

Answer 9

For f2, it is only reversing the order of the vector, while the levels stay ordered alphabetically. For f3, the vector remains the same result as f1, but the levels of f3 is being reversed.

10. What attributes does a data frame possess?

Answer 10

It has names (including col.names & row.names) and class.

11. What does as.matrix() do when applied to a data frame with columns of different types? Create a simple data.frame with two columns: one numeric and one string. Use as.matrix and show the results.

Answer 11

It coerces all other types to the least restrictive type. In this case, it coerced everything into a character matrix.

```
df1 <- data.frame("x" = letters[1:5], "y" = c(1, 2, 3, 4, 5))
df1
```

```
##   x y
## 1 a 1
## 2 b 2
## 3 c 3
## 4 d 4
## 5 e 5
```

```
df2 <- as.matrix(df1)
typeof(df2)
```

```
## [1] "character"
```