HW 2

Enter your name and EID here: Joseph Hendrix | jlh7459

You will submit this homework assignment as a pdf file on Gradescope.

For all questions, include the R commands/functions that you used to find your answer (show R chunk). Answers without supporting code will not receive credit. Write full sentences to describe your findings.

The goal of this assignment is to encode your name (or any other message) using a *cipher* function: We want to replace each letter of a given character vector with the letter of the alphabet that is *k* positions after it in the alphabet. For example, if the letter was `a` and `k = 3`, we would replace it with `d`. We will also want it to loop around, so if the letter was `y` and `k = 3`, we'd replace it with `b`. For example, with `k = 3`, the word `dog` would become `grj`. Let's take it step by step.

Question 1: (2 pts)

Type the word `letters` into the R chunk below. What does this predefined object in R contain? What is this object's data type/class? How many elements does it contain? *Include base R commands used to answer all three questions.*

```
# your code goes below (make sure to edit comment)
letters
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
## [20] "t" "u" "v" "w" "x" "y" "z"
```

```
class(letters)
```

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

```
length(letters)
```

```
## [1] 26
```

Letters is a vector of 26 character objects.

Question 2: (2 pts)

First, here is the code to split a word into a vector containing each letter.

```
test <- unlist(strsplit("test", split = ""))
# Note: the function strsplit() returns a list, use unlist() to return a vector
```

Remember that `A %in% B` returns a vector of the positions of matches of an object A in an object B (Worksheet 2):

```
letters %in% test
```

```
##  [1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE
```

How many elements are false in the resulting logical vector? *Include base R commands used to answer this question (recall: TRUE is equivalent to the value 1 and FALSE is 0.*

```
sum(!letters %in% test)
```

```
## [1] 23
```

23 elements are false!

Question 3: (2 pts)

Another function that will be useful is `which()` : it takes a logical vector as an input and returns the indices/positions that are TRUE. For example, run the following code:

```
# Note: T is shorthand for TRUE, F for FALSE
which(c(F,T,F,T,F,T))
```

```
## [1] 2 4 6
```

The output tells you that elements in position 2, 4, and 6 are true. Now, use the `which` function, along with `%in%` and `letters` , to find which positions in the alphabet the letters in the name `layla` occupy (saved as an object called `name_v` below). Would that combination of functions alone work to encode a name? Why/Why not?

```
# Define name as a vector
name_v <- unlist(strsplit("layla", split = ""))

logVec <- letters %in% name_v
which(logVec)
```

```
## [1]  1 12 25
```

It doesn’t work. Using a logical vector doesn’t account for multiples of the same letter. Each letter only counts one time.

Question 4: (2 pts)

How can we avoid this? For example, we can test each letter one at a time in their correct order! One approach would be to use a *for loop*. Write a for loop that goes through each element of the character vector `name_v` (i.e., each letter in `c("l","a","y","la")`) one at a time, finds its position in the alphabet, and saves each position in a vector called `positions` . Confirm that the positions are correct by using `positions` as an index to find the corresponding letters in the object `letters` .

For example, the name `ali` would give you the positions 1,12, and 9. You can grab the letters in those positions by doing `letters[c(1, 12, 9)]` .

```
name_v <- unlist(strsplit("layla", split = ""))
positions <- vector()

for (char in name_v) {
  logVec <- letters %in% char
  pos <- (which(logVec))
  positions <- append(positions, pos)
}

positions
```

```
## [1] 12  1 25 12  1
```

Question 5: (2 pts)

Let's encode the name `layla` ! Shift all the `positions` by 1 and index `letters` to obtain the encoded name. Is the encoded name a real name?

```
positions <- positions + 1
positions
```

```
## [1] 13  2 26 13  2
```

```
letters[positions]
```

```
## [1] "m" "b" "z" "m" "b"
```

Mbzmb isn't a real name that's just silly.

Question 6: (2 pts)

Now, what if you would like to get the positions in your name? Or any other name? We would have to repeat questions 2-5… Instead let's write a function to 1) split a name as a vector (i.e., a character vector whose elements contain single letters), 2) initialize the positions, and 3) report each position in a vector `positions` with a for loop for each new name we would like to encode. The function should take a `name` (for example, "layla") as the input and return the alphabetical positions each of those letters occupy. Call the function `get_position`. Once you have defined it, test it out with "layla". Did you get all positions?

```
get_position <- function(name) {
  nameVec <- unlist(strsplit(name, split = ''))
  positions <- vector()

  for (char in nameVec) {
    logVec <- letters %in% char
    pos <- which(logVec)
    positions <- append(positions, pos)
  }
  return(positions)
}

positions <- (get_position("layla"))
positions
```

```
## [1] 12  1 25 12  1
```

It works!

Question 7: (2 pts)

What happens when we shift the positions past `z`, the 26th and final letter of the alphabet? Shifting the positions in `layla` up by `k = 2` should give `ncanc`, but since there is no 27th element of letters, it will return `NA` instead of `a`. Try it in the code chunk below.

```
letters[get_position(name_v) + 2]
```

```
## [1] "n" "c" NA  "n" "c"
```

```
# returns NA instead of a :(
```

How do we make it loop around so that `z` shifted up 1 becomes `a`? In other words, how can we make 27 become 1, 28 become 2, 29 becomes 3, etc.? We will use a mathematical operator called modulo `%%` (which tells you the remainder when you divide one number by another). Try running the code below, `27 %% 26` (pronounced "27 modulo 26"). It returns 1, the remainder when the number on the left (27) is divided by the number on the right (26).

```
27 %% 26
```

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

We just need our shifted positions *modulo* 26. You can do this with `(positions + k) %% 26`. One last minor issue: `26 %% 26` is 0 (or any multiple of 26 %% 26 is 0) but we want it to return 26 (i.e., the letter `z`). We can fix this issue by using `ifelse` for example. Test if `positions + k %% 26` is 0: if it is, use 26, if it is not use `positions + k %% 26`. Use your function `get_position()` and the fix with modulo `%%` in `ifelse()` to encode the word `layla` by shifting every letter `k = 2` positions forward correctly. Is the encoded name a real name?

```
positions <- get_position("layla")
k <- 2
letters[ifelse((positions + k) %% 26 == 0, 26, (positions + k) %% 26)]
```

```
## [1] "n" "c" "a" "n" "c"
```

Ncanc isn't a real name.

Question 8: (2 pts)

Putting it all together: Write a function that incorporates all the work you have done to achieve the encoding task. Name the function `cipher`. This function should take two arguments: a `name` (a string) and how many positions to shift (`k`). Fill in the code below with what you have been using above. Check your code with `layla` shifted by 2 positions and test your code with your own name with the shift of your choice! Is the encoded name a real name?

```
# edit the code below (make sure to edit comment)
cipher <- function(name, k) {

  name_v <- unlist(strsplit(name, split = ""))

  positions <- c()
  for (char in name_v) {
    logVec <- letters %in% char
    pos <- which(logVec)
    positions <- append(positions, pos)
  }

  result <- letters[ifelse((positions + k) %% 26 == 0, 26, (positions + k) %% 26)]
  return(result)
}

# check
cipher("layla", 2)
```

```
## [1] "n" "c" "a" "n" "c"
```

```
# test your name!
cipher("joseph", 1)
```

```
## [1] "k" "p" "t" "f" "q" "i"
```

Neither of the encoded names are real!

Question 9: (2 pts)

A less guided question… You were given the code `oldp` . Can you decipher the code and find the name hidden behind it?

```
# brute force deciphering
codeVec <- unlist(strsplit("oldp", split = ""))
positions <- c()

for (char in codeVec) {    # get positions coded phrase
  logVec <- letters %in% char
  pos <- which(logVec)
  positions <- append(positions, pos)
}

for (i in c(1:26)) {    # decipher using every value for k
  k <- i
  res <- letters[ifelse((positions - k) %% 26 == 0, 26, (positions - k) %% 26)]
  print(paste(res, collapse = ""))
}
```

```
## [1] "nkco"
## [1] "mjbn"
## [1] "liam"
## [1] "khzl"
## [1] "jgyk"
## [1] "ifxj"
## [1] "hewi"
## [1] "gdvh"
## [1] "fcug"
## [1] "ebtf"
## [1] "dase"
## [1] "czrd"
## [1] "byqc"
## [1] "axpb"
## [1] "zwoa"
## [1] "yvnz"
## [1] "xumy"
## [1] "wtlx"
## [1] "vskw"
## [1] "urjv"
## [1] "tqiu"
## [1] "spht"
## [1] "rogs"
## [1] "qnfr"
## [1] "pmeq"
## [1] "oldp"
```

The encoded name is Liam!

Formatting: (2 pts)

Comment your code, write full sentences, and knit your file!

```
##                                                     sysname
##                                                     "Linux"
##                                                     release
##                                         "5.15.0-56-generic"
##                                                     version
## "#62~20.04.1-Ubuntu SMP Tue Nov 22 21:24:20 UTC 2022"
##                                                    nodename
##                                "educcomp01.ccbb.utexas.edu"
##                                                     machine
##                                                    "x86_64"
##                                                       login
##                                                   "unknown"
##                                                        user
##                                                   "jlh7459"
##                                              effective_user
##                                                   "jlh7459"
```