Problem Set 4

Stefanie Peschel, Philip Boustani, Philip Studener

14 November 2024

Resources

Read chapter 19 from *R for Data Science*. You can skip the exercises and references/links to other chapters or external materials.

Besides writing functions, in this exercise you should practice documenting your code, particularly functions that you write. There are more formal ways to do this, but for now you can simply use comments to explain what a function does, what arguments it takes and what they mean. A template of a function documentation is given for the function `my_sum` below. Please document all the functions you write in this exercise using this template.

```
### Sum function
###
### This function computes the sum of the elements in a vector.
### Arguments:
### x:          A numeric vector. The vector for which the sum should be
###             computed.
### na_rm:      A logical of length 1. Should NA's be removed when the sum is
###             computed? If FALSE and NA's are present the resulting sum
###             will be NA. Defaults to TRUE.
### Returns:    A numeric of length one: the computed sum.
my_sum <- function(x, na_rm = TRUE) {

  sum(x, na.rm = na_rm)

}
# application example
sum(1:10)
```

```
## [1] 55
```

```
my_sum(1:10)
```

```
## [1] 55
```

```
sum(c(1:10, NA))
```

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

```
my_sum(c(1:10, NA))
```

```
## [1] 55
```

1. What is the difference between `sum` and `my_sum`? Is this a useful function?
2. Write a function `my_mean` that calculates the mean of a numeric vector `x`. Let n be the number of elements in Vektor $\mathbf{x} = (x_1, \ldots, x_n)^T$. Then the mean is defined as the sum of the elements in $\mathbf{x}$ devided

by the number of elements `n`. $\frac{1}{n}\Sigma_{i=1}^{n} x_i$. When implementing the `my_mean` function, do not use the `mean` function from R. The *function signature* is given below. Replace the *function body* (everything between the `{` and `}`) with your code. Don't forget to also write the documentation for the function.

The expected output is given below:

```
my_mean(1:10)
```

```
## [1] 5.5
```

```
my_mean(c(1, 3))
```

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

```
my_mean(c(1:10, NA))
```

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

3. **Bonus**: As you can see from the last application of the `my_mean` function, the function returns `NA`, when the vector contains missing values. Rewrite your `my_mean` function such that it can handle `NA`s.

4. Consider once again the data frame from the previous problem set:

```
name   <- c("Alex", "Lilly", "Mark", "Oliver", "Martha", "Lucas", "Caroline")
age    <- c(25, 31, 23, 52, 76, 49, 26)
height <- c(177, 163, 190, 179, 163, 183, 164)
weight <- c(57, 69, 83, 75, 70, 83, 53)
gender <- c("D", "F", "M", "M", "F", "M", "F")
df     <- data.frame(name, age, height, weight, gender)
```

In the previous exercise, you were asked to calculate the BMI and store it as a new column in the data set. You will probably do this more often, so in order to avoid repeating yourself (*DRY* principle), write a function `calculate_bmi` that you can apply on demand, whenever you need to calculate the BMI. The function

- takes two arguments `height` (in cm) and `weight` (in kg)
- returns the BMI

The function *signature* is given below

```
### TODO
calculate_bmi <- function(weight_kg, height_cm) {

  # TODO

}
```

Use the function to calculate the BMI for subjects in data set `df` (defined above). The expected output is given below:

```
calculate_bmi(df$weight, df$height)
```

```
## [1] 18.19401 25.97012 22.99169 23.40751 26.34649 24.78426 19.70553
```

5. Write another function `mean_bmi` (signature below), that uses inputs weight and height to calculate the BMI and finally returns the mean of the BMI. Within the function body, use your previously defined functions `my_mean` and `calculate_bmi`.

```
### TODO
mean_bmi <- function(weight_kg, height_cm) {

  # TODO
```

```
}
```

Apply your function to calculate the mean BMI for all subjects in data set `df`. The expected output is given below.

```
mean_bmi(df$weight, df$height)
```

```
## [1] 23.05709
```

6. Calculate the mean BMI for the subset of females.
7. Extend your function `mean_bmi` such that it checks whether the inputs are numeric and that vectors `weight_kg` and `height_cm` have the same length. If this is not the case, return a useful error message (*Hint:* R4DS, Chp. 19.5.2)

The expected output is given below (Note: you have to write (the second half of) the error messages yourself):

```
mean_bmi(df$weight, df$height)
```

```
## [1] 23.05709
```

```
mean_bmi(df$weight[-1], df$height)
```

```
## Error in mean_bmi(df$weight[-1], df$height): Inputs do not have the same length
```

```
mean_bmi(df$gender, df$height)
```

```
## Error in mean_bmi(df$gender, df$height): One of the inputs is not numeric.
##  Please check your inputs!
```

8. Why is it good practice to use helpful function names and argument names?
9. Write a function `freq_cat` that categorizes a numeric input according to defined breaks and returns the frequency for each category (*Hint:* `?cut`). It should be possible to use all the arguments of the `cut` function without directly specifying them in the function signature. The expected output is given below.

```
freq_cat(df$height, breaks = c(150, 170, 180, 190, 200))
```

```
## x_cat
## (150,170] (170,180] (180,190] (190,200]
##         3         2         2         0
```

```
freq_cat(df$weight, breaks = c(50, 60, 70, 90), right = FALSE)
```

```
## x_cat
## [50,60) [60,70) [70,90)
##       2       1       4
```

```
freq_cat(df$gender, breaks = c("F", "M"))
```

```
## Error in freq_cat(df$gender, breaks = c("F", "M")): is.numeric(x) is not TRUE
```

10. Use the function to calculate the frequency of different categories of BMI according to the "Singapore" definition: https://en.wikipedia.org/wiki/Body_mass_index
11. Write a function that tests if an integer value is a prime number. Your function should return a logical. Document your function and give it a suitable name. Apply your function to some examples to check its functionality. *Hint:* Use the modulo operator `%%`.

Session Info

```
sessionInfo()
```

```
## R version 4.4.1 (2024-06-14)
## Platform: x86_64-apple-darwin20
## Running under: macOS Big Sur 11.7.10
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.4-x86_64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.4-x86_64/Resources/lib/libRlapack.dylib;  LAPACK v
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: Europe/Berlin
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.4.1    fastmap_1.2.0     cli_3.6.3         tools_4.4.1
##  [5] htmltools_0.5.8.1 rstudioapi_0.16.0 yaml_2.3.9        rmarkdown_2.27
##  [9] knitr_1.48        xfun_0.45         digest_0.6.36     rlang_1.1.4
## [13] evaluate_0.24.0
```