Class 6 R Functions

Nicole Alfonso

2024-01-25

This week we are introducing **R functions** and how to write our own R functions.

Questions to answer:

> Q1. Write a function grade() to determine an overall grade from a vector of student homework assignment scores dropping the lowest single score. If a student misses a homework (i.e. has an NA value) this can be used as a score to be potentially dropped. Your final function should be adquately explained with code comments and be able to work on an example class gradebook such as this one in CSV format: "https://tinyurl.com/gradeinput" [3 pts]

```
# Example input vectors to start with
student1 <- c(100, 100, 100, 100, 100, 100, 100, 90)
student2 <- c(100, NA, 90, 90, 90, 90, 97, 80)
student3 <- c(90, NA, NA, NA, NA, NA, NA, NA)
```

Follow the guidelines from class

- Write a working snippet of code that solves a simple problem

```
#Straight forward mean()
student1 <- c(100, 100, 100, 100, 100, 100, 100, 90)

mean(student1)
```

```
## [1] 98.75
```

But... we are allowed to drop the lowest assignment. How do we do that? First, we need to identify the lowest score.

```
min(student1)
```

```
## [1] 90
```

Using Help –> "See Also" to see related codes.

```
# Which element of the vector is the lowest
which.min(student1)
```

```
## [1] 8
```

What I want is to now drop (i.e. exclude) the lowest score from my mean() calculation

```r
# To get only the eighth value
student1[8]
```

```
## [1] 90
```

```r
#To get everything but the eighth element of the vector
student1[-8]
```

```
## [1] 100 100 100 100 100 100 100
```

Now we can use the answer from which.min() to return all other elements of the vector

```r
# Now, we will be combining all of the steps that we learned. First, we know that having a function[-X]

#This is our first working snippet

mean(student1[-which.min(student1)])
```

```
## [1] 100
```

Does this snippet work for the other students as well?

```r
student2 <- c(100, NA, 90, 90, 90, 90, 97, 80)

# Using the mean() function on student2 will return as NA, since there is a missing numerical value bec

mean(student2)
```

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

```r
# To fix that, we include within a parenthesis, (, na.rm=TRUE) to skip over the missing value

mean(student2, na.rm = TRUE)
```

```
## [1] 91
```

```r
#If the same function is applied to student 3, an average grade of 90 will return. However, it is not f

student3 <- c(90, NA, NA, NA, NA, NA, NA, NA)
mean(student3, na.rm = TRUE)
```

```
## [1] 90
```

Another approach is to mask (i.e. replace) all NA values with zero.

First we need to find the NA elements of the vector. How do we find the NA elements?

```r
student2 <- c(100, NA, 90, 90, 90, 90, 97, 80)
x <- student2

# Is there a function that can replace NA values with zero?
is.na(x)
```

```
## [1] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
which(is.na(x))
```

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

Now that we have identified the NA elements, we want to "mask" them with zero

```
x[is.na(x)] <- 0
x
```

```
## [1] 100   0  90  90  90  90  97  80
```

Now that we masked NA as 0, how do we drop the lowest score?

```
mean(x[-which.min(x)])
```

```
## [1] 91
```

```
# Student 3
student3 <- c(90, NA, NA, NA, NA, NA, NA, NA)
x <- student3 <- c(90, NA, NA, NA, NA, NA, NA, NA)
x[is.na(x)] <- 0
mean(x[-which.min(x)])
```

```
## [1] 12.85714
```

Now we make our function

Take the snippet and turn it into a function. Every function consists of 3 parts:

1. A name: `grade()`
2. Input arguments: a vector of student scores
3. The body: working snippet of code

Using RStudio, I will select `Code > Extract Function`

```
grade <- function(x) {
  x[is.na(x)] <- 0
  mean(x[-which.min(x)])
}
```

```
# Using the function to call out each student individually.
grade(student1)
```

```
## [1] 100
```

```
grade(student2)
```

```
## [1] 91
```

```
grade(student3)
```

```
## [1] 12.85714
```

This looks great! We now need to add comments to explain this to our future selves and others who want to use this function.

```
#' Calculate the average score for a vector student scores, dropping the lowest score.
#' Missing values will be treated as zero.
#'
#' @param x A numeric vector of homework scores
#'
#' @return Average score
#' @export
#'
#' @examples
#' student <- c(100, NA, 90, 97)
#' grade(student)
#'
grade <- function(x) {
  #Masks NA with zero, to treat missing homework as zero
  x[is.na(x)] <- 0
  #Takes the average score, which excluding the lowest score
  mean(x[-which.min(x)])
}
```

Now we can use our function on our 'real' whole class data (a CSV format file) "https://tinyurl.com/gradeinput"

```
url <- "https://tinyurl.com/gradeinput"
gradebook <- read.csv(url, row.names = 1)
```

```
apply(gradebook, 1, grade)
```

```
##  student-1  student-2  student-3  student-4  student-5  student-6  student-7
##      91.75      82.50      84.25      84.25      88.25      89.00      94.00
##  student-8  student-9 student-10 student-11 student-12 student-13 student-14
##      93.75      87.75      79.00      86.00      91.75      92.25      87.75
## student-15 student-16 student-17 student-18 student-19 student-20
##      78.75      89.50      88.00      94.50      82.75      82.75
```

> Q2. Using your grade() function and the supplied gradebook, Who is the top scoring student overall in the gradebook? [3pts]

```
results <- apply(gradebook, 1, grade)
which.max(results)
```

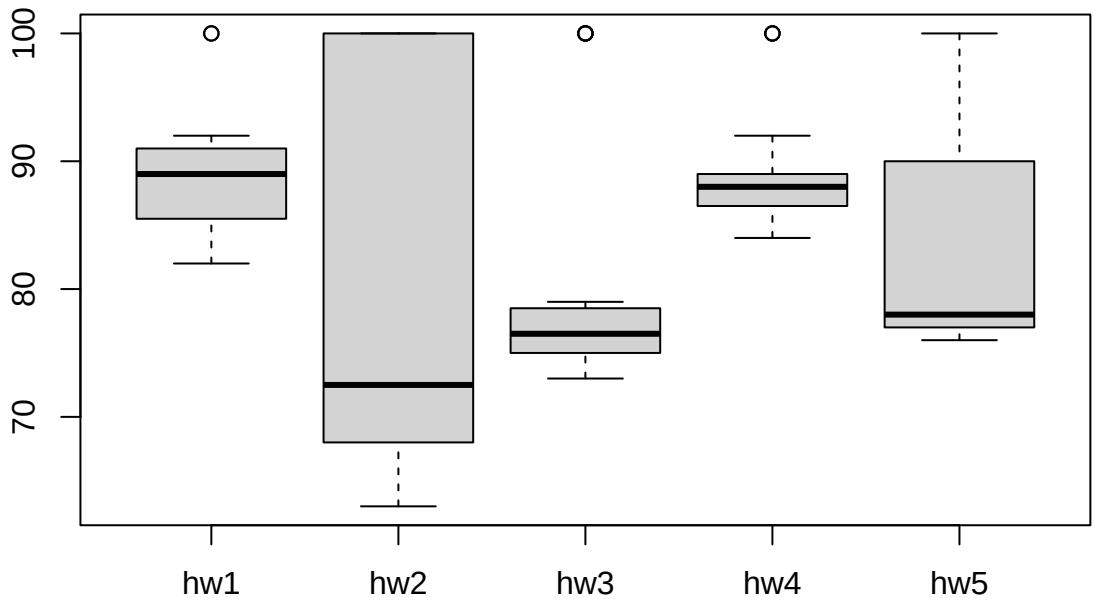
```
## student-18
##         18
```

Q3. From your analysis of the gradebook, which homework was toughest on students (i.e. obtained the lowest scores overall? [2pts]

```
avg.scores <- apply(gradebook, 2, mean, na.rm=TRUE)
which.min(avg.scores)
```

```
## hw3
##   3
```

```
boxplot(gradebook)
```



Q4. Optional Extension: From your analysis of the gradebook, which homework was most predictive of overall score (i.e. highest correlation with average grade score)? [1pt]

```
masked.gradebook <- gradebook
masked.gradebook[ is.na(masked.gradebook)] <- 0
masked.gradebook
```

```
##            hw1 hw2 hw3 hw4 hw5
## student-1  100  73 100  88  79
## student-2   85  64  78  89  78
## student-3   83  69  77 100  77
## student-4   88   0  73 100  76
## student-5   88 100  75  86  79
## student-6   89  78 100  89  77
## student-7   89 100  74  87 100
## student-8   89 100  76  86 100
## student-9   86 100  77  88  77
## student-10  89  72  79   0  76
## student-11  82  66  78  84 100
## student-12 100  70  75  92 100
## student-13  89 100  76 100  80
## student-14  85 100  77  89  76
## student-15  85  65  76  89   0
## student-16  92 100  74  89  77
## student-17  88  63 100  86  78
## student-18  91   0 100  87 100
## student-19  91  68  75  86  79
## student-20  91  68  76  88  76
```

```
cor(results, masked.gradebook)
```

```
##            hw1      hw2       hw3       hw4       hw5
## [1,] 0.4250204 0.176778 0.3042561 0.3810884 0.6325982
```

```
apply(masked.gradebook, 2, cor, x=results)
```

```
##       hw1       hw2       hw3       hw4       hw5
## 0.4250204 0.1767780 0.3042561 0.3810884 0.6325982
```

Q5. Make sure you save your Quarto document and can click the "Render" (or Rmarkdown"Knit") button to generate a PDF foramt report without errors. Finally, submit your PDF to gradescope. [1pt]