Class 6: R Functions

Taylor Forman (A59010460)

2/4/2022

In this class we are going to learn all about functions in R

First we will write a function to grade some student scores.

```
# 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)
```

```
mean(student3, na.rm = TRUE)
```

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

mean(x, na.rm = FALSE/TRUE) where NA values are removed from the vector

This does not help us calculate an average for student 3, as the only value taken into account is the first numerical value

To find NA values (not available, a.k.a. missing data) we found the `is.na()` function.

```
student2
```

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

```
is.na(student2)
```

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

`is.na()` is not what we want

We use logical vectors all the time in R like this

```
x <- 1:5
x > 2
```

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

We can search through a particular vector to search for a particular value

We can also use the `is.na()` function within a vector query to search for NA values in a vector

```
student2
```

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

```
student2[is.na(student2)]
```

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

Lastly, we can set these returned NA values to 0

We should work with a placeholder variable ('x') instead of changing the original student2

```
x <- student2
x[is.na(x)]
```

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

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

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

```
mean(x)
```

```
## [1] 79.625
```

This now replaces NA with 0 in the 'x' vector

We can also do this with student3

```
y <- student3
y[is.na(y)] <- 0
y
```

```
## [1] 90  0  0  0  0  0  0  0
```

```
mean(y)
```

```
## [1] 11.25
```

We now need to remove the lowest grade from each student's vector (after replacing NAs with 0s)

We can use the `min()` function to find the smallest value

We can use `which.min()` to find the location of the smallest value in the vector

```
student1
```

```
## [1] 100 100 100 100 100 100 100  90
```

```
min(student1)
```

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

```
which.min(student1)
```

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

But we still need to remove this value

Making a location in a vector negative removes this entry

```
student1[8]
```

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

```
student1[-8]
```

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

We can apply this technique in tandem with replacing NAs with 0s and then find the mean of this modified vector

```
z <- student1
z[-which.min(z)]
```

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

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

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

```
# Assigns a student2 vector to a new variable that we can change without worry
mod_student2 <- student2

# Finding where the NA values in the new vector (this is mostly just a checkpoint for me, not actually
mod_student2[is.na(mod_student2)]
```

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

```
# Assigning the value of 0 to any NA values in the vector
mod_student2[is.na(mod_student2)] <- 0

# Prints modified student2 vector without the lowest grade
mod_student2[-which.min(mod_student2)]
```

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

```r
# Calculates the mean of the modified student2 vector
mean(mod_student2[-which.min(mod_student2)])
```

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

We are now ready to make this entire process into a function called `grade()`

Every function has at least 3 components: - Name (grade) - Input arguments (student1, student2, etc.) - Body (working code snippet)

```r
grade <- function(x) {
  # Make NA values 0
  x[is.na(x)] <- 0
  # Exclude lowest score and calculate average
  mean(x[-which.min(x)])
}
```

Once this code is run (and appears in the environment), we can try it out

```r
grade(student1)
```

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

```r
grade(student2)
```

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

```r
grade(student3)
```

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

Grade the Class

To read a CSV file we can use the `read.csv()` function:

```r
gradebook <- read.csv("https://tinyurl.com/gradeinput", row.names = 1)
head(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  NA  73 100  76
## student-5  88 100  75  86  79
## student-6  89  78 100  89  77
```

We also want the function to be able to read values from an online database of grades

To do this, we will use the `apply()` function

Question 1: 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".

```
# Initiate the apply function, state the vector/matrix to be applied to, use 1 to apply the function to
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
```

Question 2: Using your grade() function and the supplied gradebook, Who is the top scoring student overall in the gradebook?

```
scores <- apply(gradebook, 1, grade)
scores
```

```
##  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
```

```
which.max(scores)
```

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

Question 3: From your analysis of the gradebook, which homework was toughest on students (i.e. obtained the lowest scores overall)?

```
# Checks the mean scores of the homework values when the NAs are removed
apply(gradebook, 2, mean, na.rm=TRUE)
```

```
##      hw1      hw2      hw3      hw4      hw5
## 89.00000 80.88889 80.80000 89.63158 83.42105
```

```
apply(gradebook, 2, median, na.rm=TRUE)
```

```
##  hw1  hw2  hw3  hw4  hw5
## 89.0 72.5 76.5 88.0 78.0
```

Homework 3 appears to have been the most difficult overall based on the mean, but we need to take into account the median to eliminate the outliers, which results in Homework 2

Question 4: Optional Extension - From your analysis of the gradebook, which homework was most predictive of overall score (i.e. highest correlation with average grade score)?

Question 5: Make sure you save your Rmarkdown document and can click the "Knit" button to generate a PDF foramt report without errors. Finally, submit your PDF to gradescope.