Class 6: R functions

Youn Soo Na (PID:A1704731)

Exploring R Functions

Starting simple and write our first function to add some numbers

Every function in R has at least 3 things

- a **name**, we pick this
- one or more input **arguments**
- the **body**, where the work gets done.

```
add <- function(x,y=1,z=0){
  x + y + z
}
```

Now let's try it out

```
add(x=c(10,1,1,10), y=1)
```

```
[1] 11  2  2 11
```

Q1

> 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 adequately 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"

```r
# 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)
```

Begin by calculating the average for student1

```r
student1
```

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

```r
mean(student1)
```

```
[1] 98.75
```

Try on student 2

```r
student2
```

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

```r
mean(student2, na.rm=T)
```

```
[1] 91
```

and student3

```r
student3
```

```
[1] 90 NA NA NA NA NA NA NA
```

```r
mean(student3, na.rm=T)
```

```
[1] 90
```

This is not fair for student1 and student2. I need to try something else and come back to this issue of missing values (NA)

We also want to drop the lowest score from a given students' set of scores.

```
student1[-8]
```

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

But not every student's lowest score will be the eighth score

We can try the `min()` function to find the lowest score

```
min(student1)
```

```
[1] 90
```

Not exactly what we want. Let's use HELP to see if it gives us what we exactly want.

From HELP, we find `which.min()`, let's try it out!

```
student1
```

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

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

```
[1] 8
```

Let's combine `which.min()` with `student1[]` and finally `mean()`!

```
student1[-8]
```

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

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

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

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

```
[1] 100
```

Now, let's figure out how to do the same thing with student2, which includes an NA value.

Utilise the `is.na()` function.

```
student2
```

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

```
student2[is.na(student2)] <- 0
student2[-which.min(student2)]
```

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

So far,

`x[is.na(x)] <- 0` will find NAs in the vector x and make them 0 (numerical)

`mean( x[-which.min(x)])` will find and remove the lowest score before calculating the average score from student x.

Set all students to x so that x can be used to calculate the average score for *any* student.

Now take it all and turn it into a function

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

Now, test it out

```
grade(student1)
```

```
[1] 100
```

```
grade(student2)
```

```
[1] 91
```

```
grade(student3)
```

```
[1] 12.85714
```

In order to easily and efficiently grade all the students at once,

use the function `apply(X, margin, fun)`

X is the input

Margin is the row and column (1 indicates rows, 2 indicates columns, c(1,2) indicates rows and columns)

Fun is the function to be applied

```
gradebook <- read.csv("https://tinyurl.com/gradeinput", row.names = 1)
# row.names sets the name of the rows to be column 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
```

To use `apply()` to the gradebook, I need to decide whether I want to "apply" the `grade()` function over the rows (1) or columns (2) of the gradebook dataset.

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

```
 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]

```
# Find which student scored the highest.
which.max(ans)
```

```
student-18
        18
```

Q3

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

```
masked_gradebook <- gradebook
masked_gradebook[ is.na(masked_gradebook)] <- 0
masked_gradebook_ans <- apply(masked_gradebook, 2, mean)
which.min(masked_gradebook_ans)
```

```
hw2
  2
```

I could modify the `grade()` fucntion to do this too - i.e. not drop the lowest options

```
# By using the drop.low=T and if(drop.low), I can
grade2 <- function(x, drop.low=T) {
  x[is.na(x)] <- 0

  if(drop.low) {
    cat("yes low")
    out <- mean(x[-which.min(x)])

  } else {
    out <- mean(x)
    cat("no low")
  }
  return(out)

}
grade2(student1, F)
```

```
no low
```

```
[1] 98.75
```

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]

The function to calculate correlations in R is called `cor()`

```
x <- c(100,90,80,100)
y <- c(100,90,80,100)
z <- c(80,90,100,10)

cor(x,z)
```

```
[1] -0.6822423
```

0 means there is no correlation at all

1 means it is perfectly correlated

-1 means it is perfectly anti-correlated

```
cor(ans, masked_gradebook$hw1)
```

```
[1] 0.4250204
```

```
apply(masked_gradebook, 2, cor, ans)
```

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