Class 06: R Functions

Tomi Lane Timmins

Writing Functions

In this class we will work through the process of developing our own function for calculating average grades for fictional students in a fictional class.

We will start with a simplified version of the problem. Grade some vectors of student scores. We want to drop the lowest score and find the average of each student.

#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)
```

To find the average score, we could just use mean function.

```
mean(student1)
```

```
[1] 98.75
```

We can find the smallest value with the min function.

```
min(student1)
```

```
[1] 90
```

THere. is also the 'which.min' function. Let's see if this can help. The minimum is the 8th element of the vector.

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

```
[1] 8
```

To return this value:

```
student1[8]
```

```
[1] 90
```

This function can be placed inside of the vector student1 to give the minimum value.

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

```
[1] 90
```

```
x<-1:5
x
```

```
[1] 1 2 3 4 5
```

```
x[4]
```

```
[1] 4
```

To get everything, but the fourth element:

```
x[-4]
```

```
[1] 1 2 3 5
```

Applying this knowledge:

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

```
[1] 100
```

What about student 2?

```
mean(student2[-which.min(student2)])
```

```
[1] NA
```

This does not work because student 2 has a missing score. So let's test which function did not process the NA. First the 'which.min'

```
which.min(student2)
```

```
[1] 8
```

It worked! So let's test again.

```
student2[-which.min(student2)]
```

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

It worked! So the problem must be the mean!

```
mean(student2)
```

```
[1] NA
```

```
mean( c(5, 5, 5, NA))
```

```
[1] NA
```

Test out hypothesis, does changing na.rm from FALSE to TRUE give an answer?

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

```
[1] 91
```

Does the change give a correct answer?

```
mean( c(5, 5, 5, NA), na.rm = TRUE)
```

```
[1] 5
```

```
mean(student2[-which.min(student2)], na.rm = TRUE)
```

```
[1] 92.83333
```

Let's see if we can use this on student 3, who has multiple NAs.

```
student3
```

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

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

```
[1] 90
```

Student 3 has a high mean even though they only did one assignment. So this code inflates grades by dropping all the Nas before determining the mean. The code needs to be revised.

To uninflate the grade, we could define NA by 0, so missing scores would be 0. To do this, we should be able to find all the NAs in a score.

Let's google it! Found 'is.na()' function. How does it work?

```
student3
```

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

```
is.na(student3)
```

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

The 'is.na' function returns TRUE for all the NA values.

```
student2
```

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

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

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

I can use a logical vector to index another vector.

```
x <- 1:5
x[x>3]
```

```
[1] 4 5
```

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

```
[1] NA
```

I can override this value with whatever I want (for example 0 for missing values)

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

student2
```

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

Let's test this on student3 (while assigning x to student3 to make typing easier)

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

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

Now let's get our final working snippet to properly grade the students.

```
x <- student3
x[is.na(x)] <- 0
mean(x[-which.min(x)])
```

```
[1] 12.85714
```

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

```
[1] 91
```

```
x <- student1
x[is.na(x)] <- 0
mean(x[-which.min(x)])
```

```
[1] 100
```

The code works for all the students!

We have our working snippet of code! This is now going to. be the body of our function!

All functions in R have at least 3 things: 1. name (we pick that) 2. input arguments 3. a body (code that does the work)

##Q1

```
grade <- function(x) {
  #mask NA to zero
  x[is.na(x)] <- 0
  #Drop the lowest score and get mean
  mean( x[ -which.min(x) ] )
  }
```

Let's try it out! Don't forget to press play button!

```
grade(student1)
```

```
[1] 100
```

```
grade(student2)
```

```
[1] 91
```

```
grade(student3)
```

```
[1] 12.85714
```

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" [3pts]

```
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
```

I can use super useful, but a bit more complicated 'apply()' function to use our existing 'grade()' function on the whole class gradebook.

How does this 'apply()' function work? We look to help page and find the arguments. The first is what array we want to apply a function over (in this case the gradebook), the second argument is the MARGIN (1=apply over rows, 2= apply down the columns), and the third argument is the function we want to use (in this case 'grade()')

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

```
 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]

```
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]

We should use apply with the sum function. We do not want to use mean because it is susceptible to outliers.

```
tough_hw <- apply(gradebook, 2, sum)
tough_hw
```

```
 hw1  hw2  hw3  hw4  hw5
1780   NA 1616   NA   NA
```

We should still mask NA to 0, so we don't get NAs.

```
tough_hw <- apply(gradebook, 2, sum, na.rm = TRUE)

tough_hw
```

```
 hw1  hw2  hw3  hw4  hw5
1780 1456 1616 1703 1585
```

```
which.min(tough_hw)
```

```
hw2
  2
```

Try the same with mean:

```
which.min(apply(gradebook, 2, mean, na.rm = TRUE))
```

```
hw3
  3
```

The sum function is more accurate of the homework that was the hardest for students.

Let's mask the NAs in the gradebook for future work.

```
mask <- gradebook
mask[is.na(mask)] <-0
cor(mask$hw5, results)
```

```
[1] 0.6325982
```

> 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]

A good assignment would produce grades that are correlated with what students get in the course overall.

We are going to look at the correlation of each homework results (ie the columns in the gradebook) with the overall grade of students from the course (in the 'results' object obtained from using our 'grade()' function)

```
results
```

```
 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
```

```
gradebook$hw4
```

```
 [1]  88  89 100 100  86  89  87  86  88  NA  84  92 100  89  89  89  86  87  86
[20]  88
```

```
mask$hw4
```

```
 [1]  88  89 100 100  86  89  87  86  88   0  84  92 100  89  89  89  86  87  86
[20]  88
```

I am going to use 'cor()' function

```
cor(results, mask$hw4)
```

```
[1] 0.3810884
```

```
apply(mask, 2, cor, y=results)
```

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