Class 06: R Functions

Andres Sandoval

R 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 get the average.

Example input vectors to start with

```r
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)
```

We can use the `mean()` function to get the average.

```r
mean(student1)
```

```
[1] 98.75
```

We can find the smallest value with the `min()` function

```r
min(student1)
```

```
[1] 90
```

Use Function key and F1 to pull up the help page

There is also the `which.min()` function. Let's see if this can help. You can also use the Fn + F1 key to seek help.

```r
student1
```

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

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

```
[1] 8
```

The following code will return the value of the minimum using the position function.

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

```
[1] 90
```

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

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

Using the `[-#]` will include all values in the vector besides that number.

```r
x[4]
```

```
[1] 4
```

```r
x[-4]
```

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

This code calculates the mean after dropping the lowest score. The `student1[-which.min(student1)]` will include all values but the lowest score.

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

```
[1] 100
```

Will this work for student2?

```r
student2
```

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

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

```
[1] NA
```

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

```
[1] 8
```

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

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

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

```
[1] NA
```

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

```
[1] NA
```

After evaluating the help page, decided to use the weighted.mean function in order to access the `na.rm`. This will remove the NA from mean calculation (from both the sum of values and the total amount of values).

Can I use this argument for student2?

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

```
[1] 92.83333
```

Does this work for student 3?

```r
student3
```

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

With this code, 90 gets dropped as it is both the lowest and the highest score.

```r
weighted.mean(student3[-which.min(student3)], na.rm = TRUE)
```

```
[1] NaN
```

This function inflates the score as well since only one score is calculated in the function. The NAs are not taken into consideration.

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

```
[1] 90
```

So this sucks. It inflates grades as it drops all the NAs before determining the mean…

After using a Google search, I want to know how does function `is.na()` work?

```r
student3
```

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

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

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

```r
student2
```

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

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

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

I can use a logical vector to index another vector. In other words, index means access the true elements of the other vector.

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

```
[1] 4 5
```

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

```
[1] NA
```

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

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

Combine new method with old method

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

```
[1] 12.85714
```

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: -A name (we pick that) -input arguments -a body (the code that does the work)

```
grade <- function(x) {
  # Mask NA to zero
  x[is.na(x)] <- 0
  # drop lowest value and get mean
  mean(x[-which.min(x)])
}
```

Let's try it out. Be sure to run code before the running it.

```
grade(student1)
```

```
[1] 100
```

```r
grade(student2)
```

```
[1] 91
```

```r
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]

```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
```

I can use the 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?

```r
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 
```

Apply(gradebook, Margin = 1 , grade) where input = gradebook margin = rows (if it was 2, it would be columns) function = grade

> 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]

Find the average of each column and find the lowest, or change the apply function to columns.

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

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

The correct answer is hw2.

```
#This is the correct answer
which.min(apply(gradebook, 2, sum, na.rm= TRUE))
```

```
hw2
  2
```

Try using Mean. This will spit out the wrong answer unless the NAs are masked.

```
#not a good way because the mean is taken down by the NA
apply(gradebook, 2, mean, na.rm=TRUE)
```

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

```
#not the correct answer
which.min(apply(gradebook, 2, mean, na.rm=TRUE))
```

```
hw3
  3
```

If I want to use the mean approach, I will need to mask the NA (missing homework to zero first):

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

```
           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
```

> 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]

Here we are going to look at the correlation of each homework results (i.e. 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
```

```
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 the `cor()` function:

```
cor(results, mask$hw1)
```

```
[1] 0.4250204
```

```
cor(results, mask$hw2)
```

```
[1] 0.176778
```

```
cor(results, mask$hw3)
```

```
[1] 0.3042561
```

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

```
[1] 0.3810884
```

```
cor(results, mask$hw5)
```

```
[1] 0.6325982
```

```r
#correlation function is comparing the results to the mask
cor(results, mask)
```

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

Use apply function for the correlation

```r
#the results section is the 'optional arguments' to the function. In this case we needed t
apply(mask, 2, cor, results)
```

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

The Homework with the highest correlation is...

```r
which.max(apply(mask, 2, cor, results))
```

```
hw5 
  5 
```

Homework 5