

# Week 5 R Functions

Yash Garodia

2/4/2022

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 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>” [3pts]

```
# 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

```
#which element is the lowest?
which.min(student1)
```

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

What I want is to now exclude the lowest score from my mean calculations.

```
#This will return 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.

```
#This is our first working snippet
mean(student1[-which.min(student1)])
```

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

However this wouldn't work for student 2 and 3. We could try using the `na.rm argument = TRUE` argument but this is not very useful:

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

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

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

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

This won't be fair as even though student 3 has only given one hw, he would have an average of 90 while student 2 has given all but one hw and he would have a score of 91. Not a good approach, this is unfair.

Another approach is to mask i.e replace all NA values with 0. First we need to find the NA elements of the vector. How do we find the NA elements?

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

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

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

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

Now we have identified the NA elements. We want to “mask them”. Replace them with zero?

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

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

This works well. We can simplify this:

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

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

```
mean(x)
```

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

Recall that we have to drop the lowest score now..

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

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

Applying this to student 3:

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

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

The answer is no longer 90, which makes perfect sense. Now we are essentially there with your working snippet!

##Now we make our function

Take the snippet and turn it into a function. Remember that every function has 3 parts

-A name, in our case `grade()` -Input arguments, a vector of student scores -The body i.e our 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)])
}
```

```
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 would like to use this function.

```
##' Calculate the average score for a vector of student scores, dropping the lowest score. Missing values
##'
##' @param x a numeric vector of homework scores
##'
##' @return Average score
##' @export
##'
##' @examples
##' student <- c(100, NA, 90, 97)
##' grade(student)

grade <- function(x) {
  #mask NA with zero. Treat missing values as 0
  x[is.na(x)] <- 0
  #exclude lowest score from mean
  mean( x[-which.min(x)])
}
```

Now finally we can use our function on our “real” whole class data from the CSV input file: “https://tinyurl.com/gradeinput”

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

We then apply the function to the gradebook, applying it to all rows (each row represents a student)

```
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]

To answer this question we want to use the apply function and save the results.

```
results <- apply(gradebook, 1, grade)
sort(results, decreasing = TRUE)
```

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

Alternatively we can also use the which.min function.

```
which.max(results)
```

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

Either ways, we get that **Student 18 is the highest scoring student**

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

We can try looking at mean as a way to identify the lowest scores

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

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

```
which.min(ave.scores)
```

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

We can try looking at median instead as well since it is less susceptible to fluctuations due to outliers:

```
med.scores <- apply(gradebook, 2, median, na.rm = TRUE)  
med.scores
```

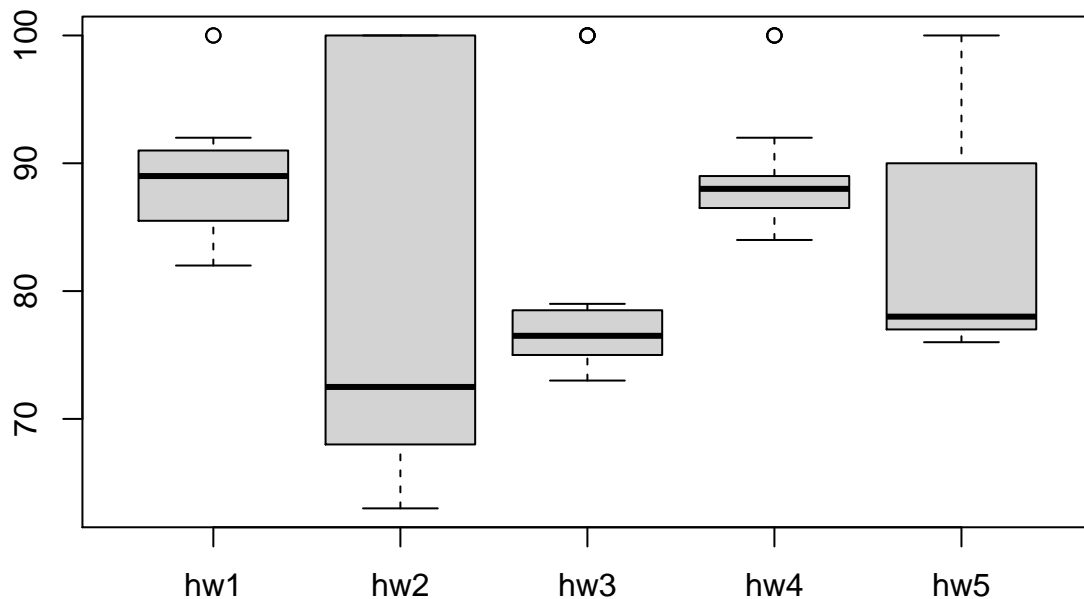
```
## hw1 hw2 hw3 hw4 hw5  
## 89.0 72.5 76.5 88.0 78.0
```

```
which.min(med.scores)
```

```
## hw2  
## 2
```

Both options show different results, so we should look at the data using a boxplot to determine which homework was the hardest:

```
boxplot(gradebook)
```



The graph suggests that **hw2 is the harder homework** because the median line is lower

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]

In order to do this, we must use the `cor()` function. Are the final results (i.e average score for each student) correlated with the results (i.e scores) for individual homeworks - the gradebook columns? First we convert the NAs to zeroes throughout the gradebook.

```
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
```

Then we look at correlation:

```
cor(results, masked.gradebook$hw5)
```

```
## [1] 0.6325982
```

However, we have to apply this across the whole vector to compare it across homeworks. Data for average of homeworks for each student is stored in `results`, and we are comparing the student's average to each homework score of the student.

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

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

From this, we see that homework 5 has the highest correlation and is the most predictive of overall student score.

Q5. 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. [1pt]

Knit the document into a PDF