

# Week 6 R Functions

Katie Chau

2/7/2022

This week we are introducing **R functions** and how to write our own.

Questions:

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)
```

Simple function

```
student1 <- c(100, 100, 100, 100, 100, 100, 100, 90)
mean(student1)
```

```
## [1] 98.75
```

But... need to drop lowest score. First identify lowest score.

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

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

What I want is to exclude the lowest value from the mean calculation.

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

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

Other students?

Na.rm = TRUE is unfair.

```
student2 <- c(100, NA, 90, 90, 90, 90, 97, 80)
mean(student2, na.rm = TRUE)
```

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

```
student3 <- c(90, NA, NA, NA, NA, NA, NA, NA)
mean(student3, na.rm = TRUE)
```

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

Mask and replace all NA with 0

Find NA elements.

```
student2 <- c(100, NA, 90, 90, 90, 90, 97, 80)
x <- student2
#is.na(x)
which(is.na(x))
```

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

Identified NA elements, need to override them

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

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

```
mean(x)
```

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

Drop lowest score.

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

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

```
student3 <- c(90, NA, NA, NA, NA, NA, NA, NA)
x<-student3
x[is.na(x)]<-0
mean(x[-which.min(x)])
```

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

**Now we need a function.**

Take the snippet and turn it into a function. Every function has 3 parts. - A name = “grade” - An input argument = a vector of student scores - A body = working snippet of code

Using R studio 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
```

Need to add comments to explain how to use this function.

```
##' Calculate average score for a vector of students scores dropping the lowest score. Missing values are
##'
##' @param x A numeric value of homework scores
##'
##' @return Average scores
##' @export
##'
##' @examples
##'
##' student <- c(100,NA,90,97)
##' grade(student)
##'
grade <- function(x) {
  #mask NA with 0
  #treat missing values as 0
  x[is.na(x)]<-0
  # exclude lowest score from mean
  mean(x[-which.min(x)])
}
```

Use the function on the class data using the CSV file format: “<https://tinyurl.com/gradeinput>”.

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

```
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]

We must run 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
```

```
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]

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

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

```
which.min(avg.scores)
```

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

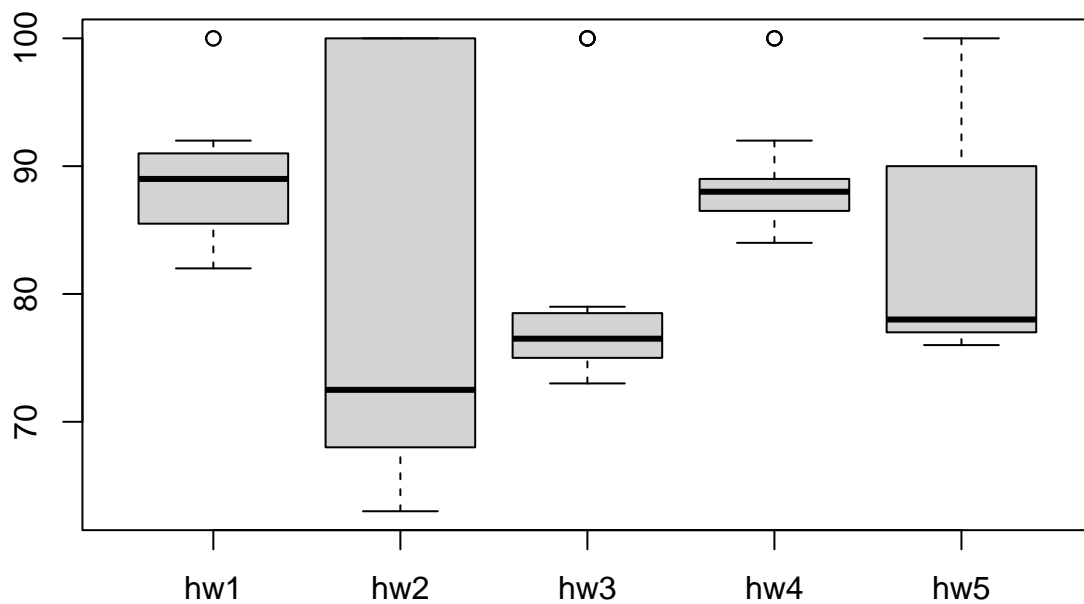
```
med.scores <- apply(gradebook, 2 , median, na.rm = TRUE)
med.scores
```

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

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

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

```
boxplot(gradebook)
```



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]

Are the final results (i.e. average score for each student) correlated with the results (i.e. scores) for individual homeworks - the gradebook columns?

```
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
```

And look at correlation

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

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

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

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

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 to make a PDF.