R Functions Lab (Class 06)

Matt Hashimoto

10/18/2021

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]"

To begin, let's define some sample vectors to work with before focusing on importing from a .csv file:

```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)
```

One way to write function is to first write working code, and then adapt it to fit a function format with the ability to accept arguements.

Thus, let's start with the min() function. It works for determining the lowest score:

```r
#Running student1 works, as it only contains integers
min(student1)
```

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

```r
#Running student2 or student3 only return NA, as they have at least one NA
min(student2)
```

```
## [1] NA
```

```r
min(student3)
```

```
## [1] NA
```

As you may notice, running student2 and student3 through the min() function returns NA. One potential way to solve this is to replace all the NAs in a vector with a 0, as missed homeworks result in that score (as well as making taking the average score in the future easier):

```r
#Position of NA values are checked with is.na() and stored in boolean vector
#naLocation
naLocation <- is.na(student2)

#Constructing new vector with same length/values as student2
student2Zeroes <- student2
#Loops through vector checking for NA and replaces with 0
for(x in c(1:length(student2))) {
  if(is.na(student2[x])) {
    student2Zeroes[x] <- 0
  }
}

#Print new vector
student2Zeroes
```

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

Now that we have a way to replace NAs with 0s, working with statistical analysis tools becomes much easier. min() and mean() now return values other than NA.

Next is to remove the lowest value. We can accomplish this by iterating through the new vector and checking for the first instance of the minimum score. Noting its position, it can then be removed.

```r
#Find the minimum score
lowScore <- min(student2Zeroes)

#Find index for minimum score
lowIndex <- 1
for(x in c(1:length(student2Zeroes))) {
  if(isTRUE(all.equal(student2Zeroes[x], lowScore))) {
    lowIndex <- x
    break
  }
}

#Reconstruct new vector without lowest value
student2Adj <- student2Zeroes[-lowIndex]

#Check new vector
student2Adj
```

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

As you can see, the new vector lacks the lowest score: a NA indicative of an incompleted homework.

At this point, all we need to do is return the mean of the adjusted grades:

```r
#Overall grade, adjusted to drop the lowest score
mean(student2Adj)
```

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

We now have all the pieces we need to construct a function that can return an adjusted average grade for a single student.

```
grade <- function(stu, na.rm=FALSE, plot=FALSE) {

  #Position of NA values are checked with is.na() and stored in boolean vector
  #naLocation
  naLocation <- is.na(stu)

  #Constructing new vector with same length/values as stu
  stuZeroes <- stu
  #Loops through vector checking for NA and replaces with 0
  for(x in c(1:length(stu))) {
    if(is.na(stu[x])) {
      stuZeroes[x] <- 0
    }
  }
  #Find the minimum score
  lowScore <- min(stuZeroes)

  #Find index for minimum score
  lowIndex <- 1
  for(x in c(1:length(stuZeroes))) {
    if(isTRUE(all.equal(stuZeroes[x], lowScore))) {
      lowIndex <- x
      break
    }
  }

  #Reconstruct new vector without lowest value
  stuAdj <- stuZeroes[-lowIndex]

  #Return the result
  return(mean(stuAdj))
}
```

We can check the function using student1 and student3:

```
#Check student1
grade(student1)
```

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

```
#Check student3
grade(student3)
```

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

A brief manual calculation yields the same results.

Next, we can find the average grades for all students in the gradebook:

```
#Create new
gradebook <- read.csv("student_homework.csv")

#Apply the function to the rows of the gradebook
avgGrades <- apply(gradebook[2:6], 1, grade)
avgGrades
```

```
##  [1] 85.00 77.25 80.75 62.25 85.00 86.00 90.25 90.50 85.50 56.75 82.00 84.25
## [13] 89.00 85.50 57.50 85.00 81.75 71.75 77.00 77.00
```

As you can see, this gives a list of the adjusted average grade for each student.

Q2

"Using your grade() function and the supplied gradebook, Who is the top scoring student overall in the gradebook?"

To answer this question, we can use a similar method for locating values as we used in the grade() function:

```
#Find and store the highest grade value
highGrade <- max(avgGrades)

#Iterate through and find the index of the highest grade
highIndex <- 1
for(x in c(1:length(avgGrades))) {
  if(isTRUE(all.equal(avgGrades[x], highGrade))) {
    highIndex <- x
    break
  }
}

#Print student number referenced by that index
gradebook[highIndex, 1]
```

```
## [1] "student-8"
```

Therefore, the student with the highest grade is Student 8.

Q3

For this question, the homework with the lowest average score should be the hardest (this is assuming that incomplete assignments don't contribute to difficulty assessment):

```
#Calculate average scores for homework 1-5
hw1 <- mean(gradebook$hw1, na.rm = TRUE)
hw2 <- mean(gradebook$hw2, na.rm = TRUE)
hw3 <- mean(gradebook$hw3, na.rm = TRUE)
hw4 <- mean(gradebook$hw4, na.rm = TRUE)
hw5 <- mean(gradebook$hw5, na.rm = TRUE)
```

```r
#Construct a data frame with all average homework scores
avgHW <- data.frame("homework 1" = hw1, "homework 2" = hw2, "homework 3" = hw3,
                    "homework 4" = hw4, "homework 5" = hw5)

#Iterate through the data frame and record the lowest average
lowHW <- 999
lowHWname <- ""
for (x in c(1:length(avgHW))){
  if (avgHW[x] < lowHW) {
    lowHW <- avgHW[x]
    lowHWname <- colnames(avgHW)[x]
  }
}

#Print the homework with the lowest average
lowHWname
```

```
## [1] "homework.3"
```

As shown, homework 3 appears to have the lowest average score for those that completed it, and thus was the toughest.