class06_RFunctionsLab

Anita Wang

10/14/2021

R Functions lab – introducing **R functions** and how to write our own R functions

Hands-on section worksheet 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 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]

```
# 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)
```

Guidelines to follow (from lecture slides):

- First, write a working snipet of code that solves a simple version of the problem

```
#Use R's mean() function
student1 <- c(100, 100, 100, 100, 100, 100, 100, 90)

mean(student1)
```

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

- But how do we drop the lowest score? First identify the lowest score...
- Look at the help page for min() and find the related functions at the bottom of the page
- Let's try which.min() -> returns the location(position) of the minimum score within the input vector

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

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

- Our goal: to drop(i.e. exclude) this lowest score from the mean() calculation
- We can use the answer from which.min() to return all elements in the vector besides the lowest score

```
#Adding a "-" will return everything
#but the specified element of the vector
student1[-which.min(student1)]
```

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

- Now we can calculate the mean for student1, after their lowest score has been dropped

```
#This is our first working snippet of code, but how can we apply it to the rest of the students in the
mean(student1[-which.min(student1)])
```

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

- Is this snippet of code robust enough? Will it work for the other example students?
- We need to find a way to deal with students who have missing assignments – Students 2 and 3 have NA values!
- We could try using na.rm=TRUE argument to modify mean(), but this is not a good approach because it would enable students with multiple NA's to receive falsely high scores. Ex.) Student 3 – having only done the first assignment, this approach would drop all their missed assignments and assign them with their highest score, not representative of their actual performance (unfair!)
- This approach is better: Mask (i.e replace) all NA values with 0
- But, how do we find the NA elements in the vector?

```
#Let's first assign student2 to "x"
x <- student2

#Now, what function in R has the capability of locating (finding the position of) NA/missing assignment

is.na(x)
```

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

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

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

- R has an is.na() function that returns a logical TRUE or FALSE for whether or not the input is an NA
- Then we can use which() function to return the location of NA
- Now we have identified the NA element(s) we want to "mask"/override with zero(s). How do we do that?

```
#Do this! If element of x is an NA, assign it a value of 0 and then return x, the vector student2
x[is.na(x)] <- 0
x
```

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

- Now we can drop the lowest score and take the mean
- Make sure to put all snippets within one code chunk

```
#Here is our revised working snippet that accounts for missed assignments
x[is.na(x)] <- 0
mean(x[-which.min(x)])
```

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

- Yay!
- Now try it for student 3:

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

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

- Success!

These two lines of code are the body of our function

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

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

Now we make the rest of our function

Take the snippet and turn it into a function

Remember, every function has 3 parts

- **A name**, in our case `grade()`
- **Input arguments**, a numeric vector of student scores
- **The body**, our working snippet of code

Let's use an RStudio helper to help us turn this snippet into a working function:

- Select `Code > Extract Function`
- Helps with formatting (inserts name and formatting for you)

Our Function

```
grade <- function(x) {
  x[is.na(x)] <- 0
  mean(x[-which.min(x)])
}
```

Let's try it out!

```
grade(student1)
```

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

```
grade(student2)
```

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

```
grade(student3)
```

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

Per Q1, let's now add code comments to adequately explain the code to others - Can use another RStudio helper: `code > insert Roxygen Skeleton`

```
#' Calculate the average score after
#' dropping the lowest score for a
#' numerical vector of student scores.
#' Missing (NA) values will be treated
#' as zeros.
#'
#' @param x A numeric vector of
#' homework/assignment scores
#'
#' @return Average score
#' @export
#'
#' @examples
#' student <- c(100, NA, 90, 97, 92)
#' grade(student)
#'

grade <- function(x) {
  # To treat missing assignments as scores of zero, we mask NA with the value zero
  x[is.na(x)] <- 0
  # Exclude the lowest score from mean
  mean(x[-which.min(x)])
}
```

Now lets use grade() to work on the example class gradebook found in this CSV formatted file: "https://tinyurl.com/gradeinput"

```
url <- "https://tinyurl.com/gradeinput"
#Default is to have student names in columns, but the row.name=1 argument makes student names become ro
gradebook <- read.csv(url, row.name=1)
```

Hint from worksheet:

- Once you have a working function for vector inputs (such as the student1, student2, and student3 vectors below) you can use the apply() function to work with data frame inputs such as those obtained from read.csv().

```
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]

```
scores <- apply(gradebook,1,grade)
which.max(scores)
```

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

- *Student 18 is the top scoring student overall in the gradebook*

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

Each column in the gradebook data frame is a HW assignment

Median is good to use as a gauge of difficulty as it is not sensitive to outlier values

However, let's first see what happens if we use other statistical points:

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

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

- Using the mean, it looks like HW 3 was the toughest.

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

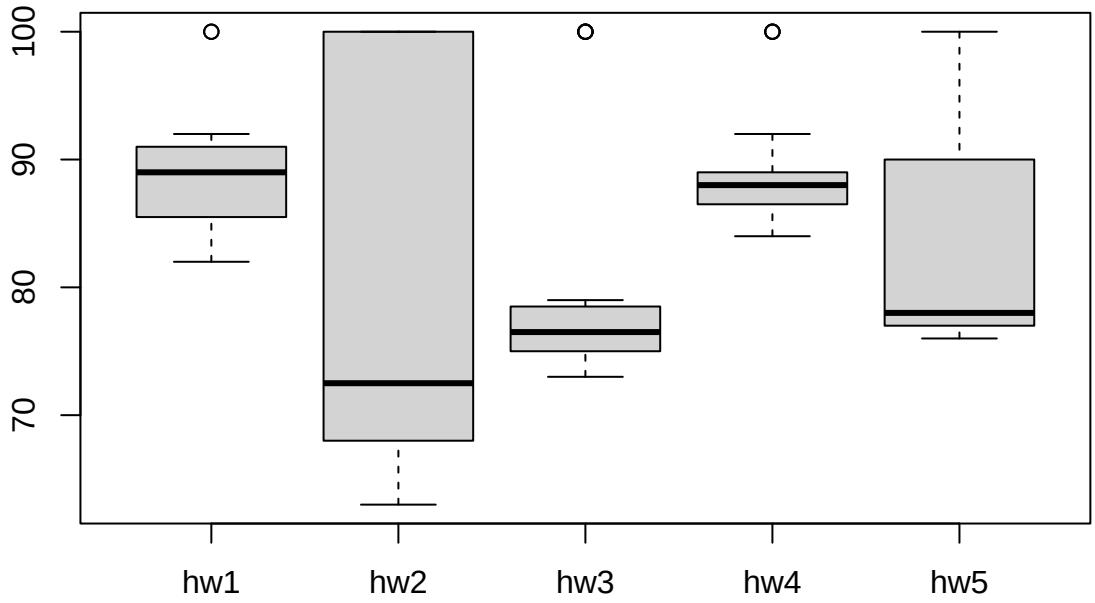
```
## hw2
##   2
```

- Using the median,it looks like HW 2 was the toughest.

Well which one is it?

Let's look at the data!

```
boxplot(gradebook)
```



- Looking at the boxplot, the largest spread in performance is in HW 2.
- This aligns with gauging difficulty via the *median*
- **From the analysis of the gradebook, homework 2 was toughest on students (i.e. obtained the lowest scores overall)**

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]

-Correlation Analysis: 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
#NA will be masked by 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
```

- We can now look at the correlation for HW5:

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

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

- So which HW has the best predictive value?

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

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

- **HW 5 is most predictive of overall score as it has the highest correlation value (values closer to 1 are more desirable)**

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]

- PDF generated