Class06

Sindy Chavez

Table of contents

Function basics 1

Q1. Write a function grade() to determine an overall grade from a vector of student homework assignment scores dropping the lowest single score 2

We need another way 3

Q2. Grade a class 5

Q3 Which homework was the toughest on students (aka lowest scores) 7

Q4 *Optional Extension* Which Homework was most predictive of overall score? 7

Q5 Can we render this page? 8

Function basics

All functions in R have at least 3 things:

- A **name** (we pick this),
- Input **arguments** (there can be loads comma separated)
- A **body** (the R code that does the work)

Write chunks of R script as you would write a paragraph. Don't write everything in one R script chunk

Q1. Write a function grade() to determine an overall grade from a vector of student homework assignment scores dropping the lowest single score

Write a function

```
# 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)
```

I can use the 'mean()' function to get the average

```
mean(student1)
```

```
[1] 98.75
```

I can find the lowest value with the 'min()' value

```
min(student1)
```

```
[1] 90
```

'min()' function on its own is not that useful. We looked up more related functions and found 'which.min'function, what does it do?

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

```
[1] 8
```

8 is the position of the lowest score in the vector

```
student1[1:7]
```

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

Can we use the minus 'student1[-8]' index trick?

```r
student1[-8]
```

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

Yes, we can use it. It can exclude whatever you tell it to.

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

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

To get the average, use the mean function on this.

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

```
[1] 100
```

What about student 2 (won't work because there is an NA in the vector)

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

```
[1] NA
```

What about student3?

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

```
[1] 90
```

The average doesn't actually give a real answer for student2 and student3. We need to try something else.

We need another way

Can I replace NA values with zero? No homework = 0

```r
student2
```

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

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

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

True is in the second position, where the NA is

```r
# Make NA zeros
student2[ is.na(student2)] <- 0
student2
```

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

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

```
[1] 91
```

```r
positions <- is.na(student3)
student3[ positions ] <- 0
student3
```

```
[1] 90  0  0  0  0  0  0  0
```

```r
mean(student3[ -which.min(student3)])
```

```
[1] 12.85714
```

Re-write my snippet to be more simple

Replace student with x, then tell R what you want x to be

```r
x <- student1

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

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

```
[1] 100
```

Now to make a true function

```r
grade <- function(x) {

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

mean(x[ -which.min(x)])
}
```

Highlight, Code, extract function, or Ctrl + Alt + X

Now use that to grade student1, etc. ?

```r
grade(student1)
```

```
[1] 100
```

Q2. Grade a class

CSV comma separated file

> Using your grade() function and the supplied gradebook . To tell you who is the top scorer in the gradebook.

```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
```

Now I want to introduce the 'apply()' function. Appplies functions to whatever data that you want to look at

apply(x,margin,fun) x=the thing we want to grade (here it's gradebook) margin=1 or 2 for either rows or columns (reads things either by row or column, row=you get the avg for each student, column=you get the average for each homework, dropping the lowest score each time because that's what our function does) fun=funtion

```r
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 
```

```r
apply(gradebook, 2, grade)
```

```
     hw1      hw2      hw3      hw4      hw5 
89.36842 76.63158 81.21053 89.63158 83.42105 
```

Question 2 answer

```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 
```

I can use the 'which.max()' to find where the largest/max value is in this results vector

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

```
student-18 
        18 
```

which.max only returns one highest score, unlike the sort function, which will return more than one if there was more than one high score

Q3 Which homework was the toughest on students (aka lowest scores)

We want to use 'apply()' and look at the columns, but without dropping the lowest grades

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

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

The lowest score

```
which.min(homeworks)
```

```
hw2 
  2 
```

Q4 *Optional Extension* Which Homework was most predictive of overall score?

The highest score

```
which.max(homeworks)
```

```
hw1 
  1 
```

Looking at the lowest and highest scoring homeworks can give the instructor insight as to which homeworks were good (students could score highly) and bad (all students did poorly)

Pearson correlation cor(x, y)

plots things x vs y

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

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

```
[1] 0.6325982
```

Can I apply the 'cor()'function ovver the masked gradebook? Sure!

```r
apply(mask,2, cor, y=results)
```

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

Q5 Can we render this page?

Aparently yes