lab 6

Lauren Waters

##All about functions in R

Functions are the way we get stuff done in R. We call a function to read data, compute stuff, plot stuff, …

R makes writing functions accesible but we should alwyas start by trying to get a working snippet of code first before we write our function.

##Today's lab

We will grade a whole class of student assignments. We will always try to start with a simplified version of the problem

```
# 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)
```

If we want the average we can use the `mean()` function

```
mean(student1)
```

```
[1] 98.75
```

Lets drop the lowest score. Can use the `min()` function to find the lowest value.

```
min(student1)
```

```
[1] 90
```

The `which.min` function may be useful here.

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

```
[1] 8
```

'-' will remove the value it's applied to (in this case the lowest score).

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

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

Can put everthing together to get the average grade (excluding the lowest value).

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

```
[1] 100
```

Now, lets test on student 2.

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

```
[1] NA
```

Where is the problem? The same code doesn't work because they missed a homework, and there is a gap in the grades. By adding `na.rm=` argument, it strips all the NA from the sequence.

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

```
[1] 91
```

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

```
[1] 90
```

This isn't fair!

I want to stop typing out student1, student2, …. So, lets work with an input called 'x'

```r
x <- student2
```

We want to overwrite the NA values with 0 - if you miss a homework, then you get 0% on the assignment.

Google and Chatgpt told me about the `is.na()` function.

```r
x
```

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

```r
is.na(x)
```

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

We can use logicals to index a vector. They pull out only the TRUE outputs

```r
y <-  1:5
y
```

```
[1] 1 2 3 4 5
```

```r
y > 3
```

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

```r
y[y > 3]
```

```
[1] 4 5
```

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

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

This is my working snippet of code that solces the problem for all my example student inputs. Can replace x with each student (just reassign x everytime you want to change the student)

```r
x <- student2
# Mask Na vlaues to zero
x[is.na(x)] <- 0
# Drop lowest score and get the mean
mean(x[-which.min(x)])
```

```
[1] 91
```

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]

```r
grade <- function(x) {
# Mask Na vlaues to zero
x[is.na(x)] <- 0
# Drop lowest score and get the mean
mean(x[-which.min(x)])
}
```

Use this function:

```r
grade(student1)
```

```
[1] 100
```

```r
grade(student2)
```

```
[1] 91
```

```r
grade(student3)
```

```
[1] 12.85714
```

We need to read the gradebook

```r
gradebook <- read.csv("https://tinyurl.com/gradeinput", row.names =1)
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
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  NA  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  NA
student-16  92 100  74  89  77
student-17  88  63 100  86  78
student-18  91  NA 100  87 100
student-19  91  68  75  86  79
student-20  91  68  76  88  76
```

Use `apply()` function to get the average grades for each student. For the second position (MARGIN), 1 = rows and 2 = colonms.

```r
ans <- apply(gradebook, 1, grade)
ans
```

```
 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]

```r
max(ans)
```

```
[1] 94.5
```

```r
which.max(ans)
```

```
student-18
        18
```

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

Use the `mean()` and `apply` functions. If you use the `grade` function, then it will drop the lowest score.

```r
mask <- gradebook

mask[is.na(mask)] <- 0
questions <- apply(mask, 2, mean)
questions
```

```
  hw1   hw2   hw3   hw4   hw5 
89.00 72.80 80.80 85.15 79.25 
```

```r
which.min(questions)
```

```
hw2 
  2 
```

We could also use the `sum()` function

```r
questions <- apply(mask, 2, sum)
questions
```

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

```
which.min(questions)
```

```
hw2
  2
```

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]

You can add the `cor()` function

```
opt <- apply(mask, 2, cor, y=ans)
which.max(opt)
```

```
hw5
  5
```