class06

Assael Madrigal (PID: A10179083)

Functions in R

Every function needs 3 things: name, arguments and body

The function of today is to grade a class of student assigment scores. **all students get to drop 1 of their lowest score**

But first I am going to work with a vector and make sure i know how it works because i know what the answer is.

as a side note, to have Quarto render a new line we need to give the line 2 spaces.

```
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)
```

to calculate average: mean()

```
mean(student1)
```

```
[1] 98.75
```

```
mean(student2)
```

```
[1] NA
```

```
mean(student3)
```

```
[1] NA
```

the function `which` can be used with `min()` to return the position where the lowest value is `which.min()`

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

```
[1] 8
```

```
student1[8]
```

```
[1] 90
```

We can use the - inside to return everything except what is inside

```
student1[-8]
```

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

We can put together to find the mean of student1 dropping their lowest score

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

```
[1] 100
```

Will this work for student 2? No because it has an NA

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

```
[1] NA
```

To switch i can assign x to variable to make it easier

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

```
[1] 100
```

One idea is to "mask" the NA and change them to be 0. So if you don't do HW you get 0 points. The `is.na()` returns a logical for every position The `replace()` takes 3 arguments 1. the vector 2. the condition 3. what to replace it with

```
  x <- student3
  replace(x, is.na(x), 0)
```

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

The long way to do it is to get the vector of na first, then assign each element to 0

```
  x<-student3
  is.na(x)
```

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

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

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

So then we can combine this with the code that removes the smallest value before calculating the mean. And we get our functional code

```
  x<-student2
  #Mask NA with 0
  x[ is.na(x)] <- 0
  #drop the lowest value and find the average
  mean(x[-(which.min(x))])
```

```
[1] 91
```

###Question 1 Let's turn it into a function

```
  grade <- function(x){
    #Mask NA with 0
    x[ is.na(x)] <- 0
    #drop the lowest value and find the average
    mean(x[-(which.min(x))])
  }
```

Now that it is loaded i can use it to confirm my function works

```
grade(student1)
```

```
[1] 100
```

Next I want to read a csv

```
gradebook <- read.csv("https://tinyurl.com/gradeinput")
gradebook
```

```
            X hw1 hw2 hw3 hw4 hw5
1   student-1 100  73 100  88  79
2   student-2  85  64  78  89  78
3   student-3  83  69  77 100  77
4   student-4  88  NA  73 100  76
5   student-5  88 100  75  86  79
6   student-6  89  78 100  89  77
7   student-7  89 100  74  87 100
8   student-8  89 100  76  86 100
9   student-9  86 100  77  88  77
10 student-10  89  72  79  NA  76
11 student-11  82  66  78  84 100
12 student-12 100  70  75  92 100
13 student-13  89 100  76 100  80
14 student-14  85 100  77  89  76
15 student-15  85  65  76  89  NA
16 student-16  92 100  74  89  77
17 student-17  88  63 100  86  78
18 student-18  91  NA 100  87 100
19 student-19  91  68  75  86  79
20 student-20  91  68  76  88  76
```

but i don't want x in my column so i use row.names=1 to read in the code from the 1st row instead of the 0th. So it made its own rows but i want it to use the first column as the name for the rows.

```
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
```

the `apply()` function is really important to learn

```
?apply()
```

We can `apply` the grade function to the gradebook. I need the array, the margin, and the function to be applied. So i need to find the margins i want

```
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
```

##Question 2 To find the student that scored the highest. I can just ask it to find the max value and which student it corresponded to

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

```
student-18
        18
```

##Question 3 To find the toughest homework i can find the average of the columns instead of the rows. But i should not do grade() because i dont want it to drop it. Then find which was the lowest from the ones that got turned in

```
  ans3 <-apply(gradebook, 2,mean, na.rm=TRUE)
  which.min(ans3)
```

```
hw3
  3
```

```
  ans3
```

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

Let's see if we mask the NA with 0 if that will change teh answer

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

```
           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
```

```
ans3<-apply(gradebook, 1,grade)
ans3
```

```
 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
```

lets see what happens if we use the sum

```
which.min(apply(mask,2,sum))
```

```
hw2
  2
```

##Quesion 4 From your analysis of the gradebook, which homework was most predictive of overall score

We can use Pearson's correlation using the `corr` function

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

```
           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
```

```
ans1<-apply(mask, 1,grade)
ans1
```

```
 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
```

```
cor(mask$hw1,ans1)
```

```
[1] 0.4250204
```

```
cor(mask$hw2,ans1)
```

```
[1] 0.176778
```

We can try to apply the function with cor, to do this we have to include the arguments for cor in the ... section

```
apply(mask,2,cor, y=ans)
```

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

So hw 5 is the most predictive