Class 6: R Functions

Katelyn Wei (PID: A16682595)

Starting Vectors

```
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)
```

Q1. Writing a Function grade()

The mean() function gives the average:

```
mean(student1)
```

```
[1] 98.75
```

We'd like to drop the lowest score. `min()` gives the lowest score, and `which.min()` tells you which position the score is in:

```
min(student1)
```

```
[1] 90
```

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

```
[1] 8
```

Putting a minus sign in front of a position number removes it from the lineup:

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

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

We can then calculate student 1's average, dropping the lowest score!

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

```
[1] 100
```

Student 2 has an NA. Plugging it into the code we used for student 2 doesn't give what we want because of it:

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

```
[1] NA
```

The problem is in the mean, not which.min:

```
which.min(student2)
```

```
[1] 8
```

```
mean(student2)
```

```
[1] NA
```

`na.rm` drops NA values if set to TRUE. Let's try it:

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

```
[1] 91
```

Inputting student 3 with the code for student 2 gives a really high mean because `na.rm` stripped all the NA's. But just using the default mean() also gives an NA(which we don't want!):

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

```
[1] 90
```

```r
mean(student3)
```

```
[1] NA
```

typing out `student1`, `student2` is getting tiring so let's just set everything to x. It'll also let us override things without affecting the original dataset.

```r
x <- student2
x
```

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

We want to make NA = 0. A quick google search gives you the `is.na()` function to do so(you can also use ChatGPT and Claude):

```r
x
```

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

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

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

Logicals index vectors. We can use that to access the NAs in x and assign it to 0:

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

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

Combining it with earlier code gives a string of functions for what we wanted it to do!

```
# Set NA values to zero
x[is.na(x)] <- 0
# Drop lowest score to calculate mean
mean(x[-which.min(x)])
```

```
[1] 91
```

Testing it on student 1 and student 3 also works:

```
x <- student1
x[is.na(x)] <- 0
mean(x[-which.min(x)])
```

```
[1] 100
```

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

```
[1] 12.85714
```

Now to turn it into the `grade()` function:

```
grade <- function(x) {
  # Set NA values to zero
  x[is.na(x)] <- 0
  # Drop lowest score to calculate mean
  mean(x[-which.min(x)])
}
```

Use this function(don't forget to run the code that makes grade a function!):

```
grade(student1)
```

```
[1] 100
```

Now we need to read the gradebook:

```
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
```

`apply(X, MARGIN, FUN)` allows you to apply a function across a dataset, or array. **X** is the dataset, **margin** is how the matrix will be read (by row = 1, by column = 2, row and column =c(1,2)), and **fun** is the function you want to apply.

```
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. Who's the Top Scoring Student?

Based on the `ans` output student 18 is the highest scoring student. You can also use the `which.max` function to spit it out for you:

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

```
student-18
        18
```

Q3. Which Homework was the Toughest?

We can calculate this by using the mean and apply functions. Since NA's set to 0 would seriously skew the mean, I've opted to set `na.rm = TRUE` to strip them from the calculation.

```
apply(gradebook, 2, mean, na.rm = TRUE)
```

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

Using the `which.min` function will then pinpoint the toughest homework:

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

```
hw3
  3
```

Q4. Which Homework was Predictive of Overall Score?

Gradebook still has NAs so let's make another vector that has them set to zero:

```
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
```

Now we can use the `apply()` and `cor()` functions on mask to find the correlation between homework and overall score(stored in the vector `ans`):

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

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

All the correlation coefficients are positive. Since homework 5 has the largest value, it is the most predictive of overall score:

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

```
hw5
  5
```